Kapitola 13

Práce s textovými soubory a řetězci

Po dokončení této kapitoly budete schopni:

- Zobrazit textový soubor pomocí objektu textového pole, funkce `LineInput` a třídy `StreamReader`
- Používat obor názvů `My`, což je prvek „rychlého přístupu" v rámci prostředí Visual Studia 2008
- Ukládat poznámky v textovém souboru pomocí funkce `PrintLine` a ovládacího prvku `SaveFileDialog`
- Používat postupy práce s proměnnými k porovnávání, spojování a řazení řetězců

Správa elektronických dokumentů je v mnoha moderních společnostech důležitým aspektem a jazyk Microsoft Visual Basic 2008 nabízí mechanismy pro práci s různými typy dokumentů a zpracování informací v dokumentech. Základním typem dokumentu je textový soubor, který je tvořen neformátovanými slovy a odstavci, písmeny, čísly a různými znaky a symboly.

V této kapitole se naučíte pracovat s informacemi uloženými v textových souborech. Dozvíte se, jak otevřít textový soubor a zobrazit jeho obsah v objektu textového pole, a vytvoříte nový textový soubor na disku. Podrobněji se také seznámíte se správou řetězců a použijete metody v třídách `String` a `StreamReader` rozhraní Microsoft .NET Framework ke spojení, seřazení a zobrazení slov, řádků a celých textových souborů.

Zobrazení textových souborů pomocí objektu textového pole

Nejjednodušším způsobem zobrazení textového souboru v programu je použití objektu textového pole. Jak jste se již naučili, objekty textových polí můžete vytvářet v jakékoli velikosti. Pokud se obsah textového souboru nevejde do textového pole, je možné přidat posuvníky, aby si uživatel mohl prohlédnout celý soubor. Aby jazyk Visual Basic načetl obsah textového souboru do textového pole, musíte použít čtyři funkce. Přehled těchto funkcí nabízí následující tabulka a práci s nimi si vyzkoušíte v prvním cvičení této kapitoly. Jak již bylo uvedeno dříve, některé z těchto funkcí nahrazují starší klíčová slova jazyka Visual Basic.

Funkce	Popis
`FileOpen`	Otevře textový soubor pro vstup nebo výstup.
`LineInput`	Přečte řádek z textového souboru.
`EOF`	Ověří, jedná-li se o konec textového souboru.
`FileClose`	Zavře textový soubor.

Otevření textového souboru pro vstup

Textový soubor se skládá z jednoho či více řádků čísel, slov nebo znaků. Textové soubory se liší od *dokumentů* a *webových stránek*, které obsahují formátovací kódy. *Spustitelné soubory* se odlišují tím, že obsahují instrukce pro operační systém. Typické textové soubory v systému jsou v Průzkumníkovi Windows označeny jako „Textové dokumenty", případně mají příponu *.txt*, *.ini*, *.log* nebo *.inf*. Protože textové soubory obsahují pouze běžné, rozpoznatelné znaky, můžete je snadno zobrazit prostřednictvím objektů textových polí.

Pomocí ovládacího prvku `OpenFileDialog`, který vyzve uživatele k zadání cesty k souboru, umožníte uživateli otevřít, v programu libovolný textový soubor. Tento ovládací prvek obsahuje vlastnost `Filter`, která určuje typ zobrazovaných souborů, metodu `ShowDialog`, jež zobrazuje dialogové okno Otevřít a vlastnost `FileName`, která vrátí cestu zadanou uživatelem. Ovládací prvek `OpenFileDialog` soubor neotevírá, ale pouze získává cestu.

Funkce FileOpen

Když od uživatele získáte cestu, otevřete soubor v programu pomocí funkce `FileOpen`.

Zkrácená syntaxe funkce `FileOpen` vypadá takto:

```
FileOpen(čísloSouboru, cesta, režim)
```

Úplný seznam argumentů naleznete v nápovědě k Visual Studiu. Nejdůležitější jsou tyto:

- `čísloSouboru` je celé číslo od 1 do 255,
- `cesta` je platná cesta v operačním systému Microsoft Windows,
- `režim` je konstanta (přesněji výčtový typ), která určuje, jak bude soubor použit. (V této kapitole použijete režimy `OpenMode.Input` a `OpenMode.Output`.)

Číslo souboru je se souborem spojeno při jeho otevření. Toto číslo souboru pak použijete v kódu vždy, když se potřebujete odkázat na otevřený soubor.

Typická funkce `FileOpen` využívající objekt `OpenFileDialog` vypadá takto:

```
FileOpen(1, OpenFileDialog1.FileName, OpenMode.Input)
```

Vlastnost `OpenFileDialog1.FileName` představuje cestu, `OpenMode.Input` je režim a 1 je číslo souboru.

Tip: Textové soubory, které se otevírají pomocí této syntaxe, se nazývají soubory se sekvenčním přístupem, protože s jejich obsahem musíte pracovat v sekvenčním pořadí. Naproti tomu můžete k informacím v databázových souborech přistupovat v jakémkoli pořadí. (Databázím se věnuje kapitola 18.)

Následující cvičení ukazuje, jak můžete použít ovládací prvek `OpenFileDialog` a funkci `FileOpen` k otevření textového souboru. Cvičení také představuje způsob použití funkcí `LineInput` a `EOF` pro zobrazení obsahu textového souboru v textovém poli a jak použít funkci `FileClose` k zavření souboru. (Více informací o používání ovládacích prvků na kartě Dialogs v okně Toolbox k vytváření standardních dialogových oken naleznete v kapitole 4.)

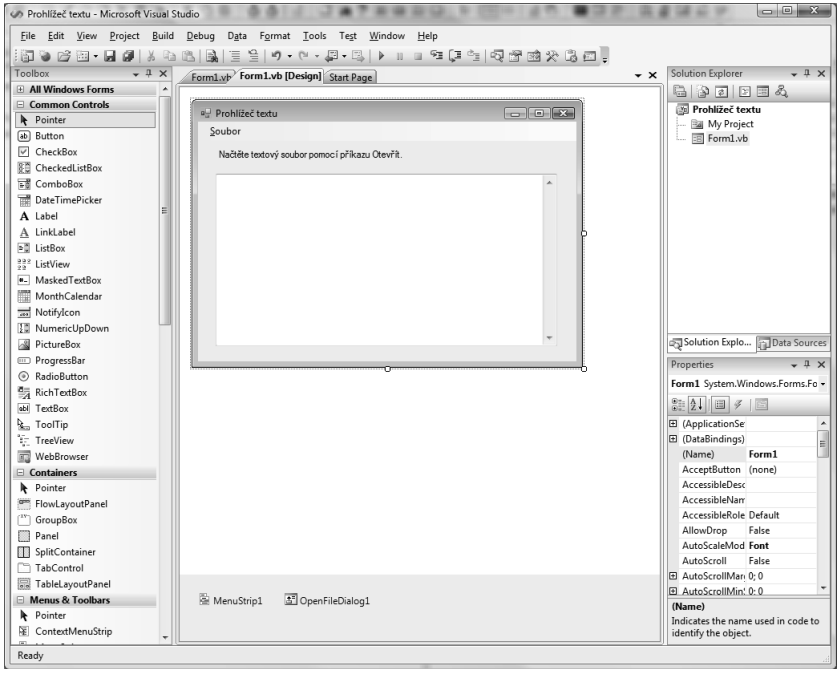
Spuštění programu Prohlížeč textu

1. Spusťte Microsoft Visual Studio a otevřete projekt Prohlížeč textu, který naleznete ve složce *c:\vb08sbs\kap13\Prohlížeč textu*.

Projekt se otevře ve vývojovém prostředí.

2. Nevidíte-li formulář projektu, zobrazte jej nyní.

Otevře se formulář Prohlížeč textu:



Formulář obsahuje velké textové pole s posuvníky. Je zde také objekt nabídky, který umísťuje příkazy Otevřít, Zavřít a Konec do nabídky Soubor, dále objekt dialogového okna pro zadání cesty k souboru a popisek s instrukcemi. Přehled nastavení vlastností naleznete v následující tabulce. (Všimněte si zvláště nastavení textového pole.)

Objekt	Vlastnost	Nastavení
txtPoznamka	Enabled	False
	Multiline	True
	Name	**txtPoznamka**
	ScrollBars	**Both**
ZavritToolStripMenuItem	Enabled	False
lblPoznamka	Text	**„Načtěte textový soubor pomocí příkazu Otevřít."**
	Name	**lblPoznamka**
Form1	Text	**„Prohlížeč textu"**

3. Na panelu nástrojů Standard klepněte na tlačítko Start Debugging.

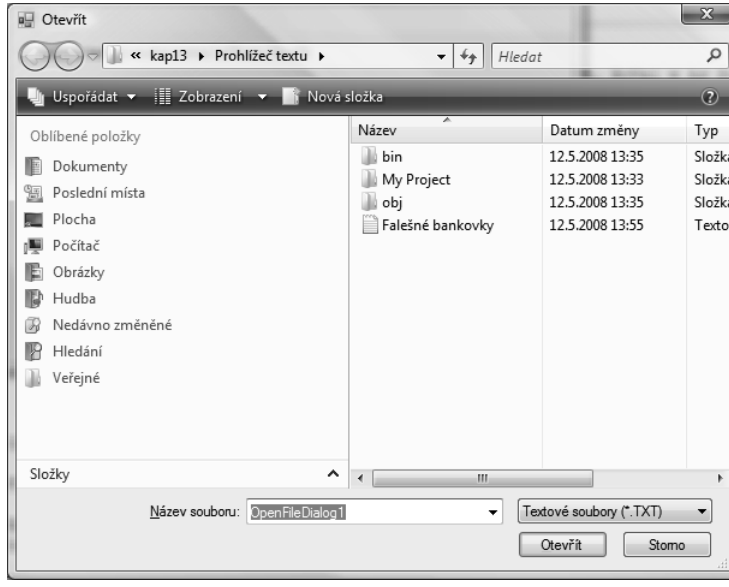
Program Prohlížeč textu se spustí.

4. V nabídce Soubor programu Prohlížeč textu zvolte příkaz Otevřít.

   Zobrazí se dialogové okno Otevřít.

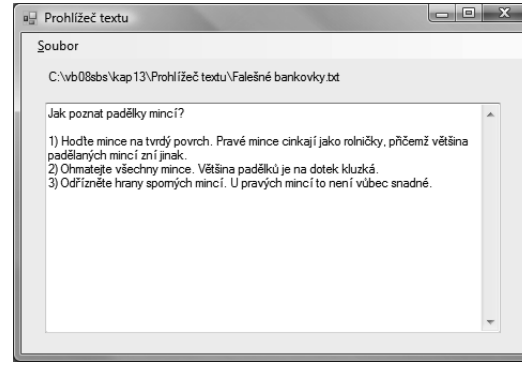
5. Otevřete složku *c:\vb08sbs\kap13\Prohlížeč textu*.

   Obsah složky Prohlížeč textu vidíte na následujícím obrázku:



6. Poklepejte na název souboru *Falešné bankovky.txt*.

   V textovém poli se zobrazí soubor *Falešné bankovky*, obsahující článek z roku 1951, který upozorňuje na nebezpečí padělaných bankovek ve Spojených státech (viz obrázek).



7. K prohlédnutí celého dokumentu použijte posuvníky.

8. Po prohlédnutí dokumentu zvolte v nabídce `Soubor` příkaz `Zavřít` a poté klepněte na příkaz `Konec` pro ukončení programu.

   Program se zastaví a vrátíte se do vývojového prostředí.

Nyní se zaměříme na dvě důležité procedury událostí v programu.

Kód programu Prohlížeč textu

1. V nabídce Soubor formuláře Prohlížeč textu poklepejte na příkaz Otevřít.

   V okně Code Editoru se zobrazí procedura události `OtevritToolStripMenuItem_Click`.

2. V případě potřeby zvětšete okno Code Editoru pro zobrazení větší části kódu programu.

Procedura události `OtevřítToolStripMenuItem_Click` obsahuje tento kód:

```
Dim CelýText As String = "", ŘádekTextu As String = ""
OpenFileDialog1.Filter = "Textové dokumenty (*.TXT)|*.TXT"
OpenFileDialog1.ShowDialog() 'zobraz dialogové okno Otevřít
If OpenFileDialog1.FileName <> "" Then
   Try 'otevři soubor a pomocí obsluhy chyb ošetři jakoukoli chybu
      FileOpen(1, OpenFileDialog1.FileName, OpenMode.Input)
        Do Until EOF(1) 'načti řádky ze souboru            ŘádekTextu =
LineInput(1)
         ' přidej všechny řádky do proměnné CelýText
         CelýText = CelýText & ŘádekTextu & vbCrLf
      Loop
      lblPoznámka.Text = OpenFileDialog1.FileName 'aktualizuj popisek
      txtPoznámka.Text = CelýText 'zobraz soubor
      txtPoznámka.Enabled = True 'povol kurzor
      ZavřítToolStripMenuItem.Enabled = True 'povol příkaz Zavřít
      OtevřítToolStripMenuItem.Enabled = False 'zakaž příkaz Otevřít
   Catch
      MsgBox("Chyba při otevírání souboru.")
   Finally
      FileClose(1) 'zavři soubor
   End Try
End If
```

Tato procedura události provádí následující akce:

- Deklaruje proměnné a přiřazuje hodnotu vlastnosti `Filter` objektu dialogového okna pro otevření souboru.
- Požádá uživatele o zadání cesty prostřednictvím objektu `OpenFileDialog1`.
- Zachytí chyby pomocí bloku kódu `Try...Catch`.
- Pomocí funkce `FileOpen` otevře určený soubor pro čtení.
- Použije funkci `LineInput` pro postupné zkopírování jednotlivých řádků ze souboru do řetězce s názvem `CelýText`.
- Kopíruje řádky, dokud nedosáhne do konce souboru (`EOF`) nebo pokud již v řetězci není žádné místo. Řetězec `CelýText` má prostor pro velmi velký soubor, pokud však dojde během procesu kopírování k chybě, klauzule `Catch` zobrazí chybu.
- Zobrazí řetězec `CelýText` v textovém poli a povolí posuvníky a textový kurzor.
- Aktualizuje příkazy v nabídce Soubor a nakonec soubor uzavře pomocí funkce `FileClose`.

Prohlédněte si, jak fungují příkazy v proceduře události `OtevřítToolStripMenuItem_Click` – zejména pak funkce `FileOpen`, `LineInput`, `EOF` a `FileClose`. Jestliže dojde k chybě, obsluha chyby v proceduře zobrazí zprávu a zruší proces načítání.

Tip: Pro zobrazení informací o příkazech a funkcích označte příslušné klíčové slovo a stiskněte klávesu F1 pro zobrazení odpovídající dokumentace ve Visual Studiu.

3. Zobrazte proceduru události `ZavřítToolStripMenuItem_Click`, která se spouští po klepnutí na příkaz Zavřít v nabídce.

   Procedura vypadá takto:

   ```
   txtPoznámka.Text = ""
   'vymaž textové pole
   lblPoznámka.Text = "Načtěte textový soubor pomocí příkazu Otevřít."
   ZavřítToolStripMenuItem.Enabled = False 'zakaž příkaz Zavřít
   OtevřítToolStripMenuItem.Enabled = True 'povol příkaz Otevřít
   ```

   Procedura vymaže textové pole, aktualizuje popisek `lblPoznámka`, zakáže příkaz Zavřít a povolí příkaz Otevřít.

Tento jednoduchý program nyní můžete použít jako šablonu pro pokročilejší programy, které pracují s textovými soubory. V další části se naučíte, jak do textového pole napsat vlastní text a jak jej uložit do souboru na disku.

Otevírání textových souborů pomocí třídy StreamReader a objektu My.Computer.FileSystem

Vedle příkazů jazyka Visual Basic, které otevírají a zavírají textové soubory, existují i dva další postupy pro otevírání textových souborů v programu vytvořeném v prostředí Visual Studia: třída `StreamReader` a obor názvů `My`. Protože v těchto postupech vystupují objekty rozhraní .NET Frameworku, které jsou dostupné ve všech programovacích jazycích Visual Studia, upřednostňuji je před funkcemi, jež jsou pouze doménou jazyka Visual Basic. Společnost Microsoft však z estetických důvodů i kvůli zpětné kompatibilitě zachovala více mechanismů operací se soubory, takže konečná volba je jen na vás.

Třída StreamReader

Třída `StreamReader` rozhraní .NET Framework vám umožňuje otevírat a zobrazovat textové soubory v programech. Tento postup použijeme v této knize několikrát při práci s textovými soubory (například v kapitole 16). Pro snazší použití třídy `StreamReader` přidáte na začátek kódu následující příkaz `Imports` (více o příkazu `Imports` najdete v kapitole 5):

```
Imports System.IO
```

Textový soubor pak můžete zobrazit v textovém poli pomocí následujícího kódu. (V tomto příkladu se otevře soubor *Falešné bankovky.txt* a kód předpokládá ve formuláři přítomnost objektu `TextBox1`.)

```
Dim DataKZobrazení As StreamReader
DataKZobrazení = New StreamReader _
   ("C:\vb08sbs\kap13\Prohlížeč textu\Falešné bankovky.txt")
TextBox1.Text = DataKZobrazení.ReadToEnd
DataKZobrazení.Close()
TextBox1.Select(0, 0)
```

Třída `StreamReader` z rozhraní .NET Framework nabízí alternativu k otevření textového souboru pomocí funkce `FileOpen` jazyka Visual Basic. Ve výše uvedené ukázce třídy `StreamReader` jsme deklarovali proměnnou s názvem `DataKZobrazení` typu `StreamReader`,

která bude uchovávat obsah textového souboru, a poté je definována platná cesta k souboru, jenž se má otevřít. Metoda `ReadToEnd` načte do proměnné `DataKZobrazení` veškerý text ze souboru v aktuálním umístění, který poté přiřadíme k vlastnosti `Text` objektu textového pole. Poslední příkazy zavřou textový soubor a použijí metodu `Select` k odebrání výběru z textového pole.

Obor názvů My

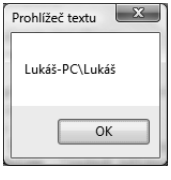
Druhou možností otevírání textových souborů v programu je užitečný prvek jazyka Visual Basic, který využívá obor názvů `My`. Tento obor názvů nabízí rychlý přístup, který má zjednodušovat přístup k rozhraní .NET Framework pro provádění běžných úkolů, jako je práce s formuláři, prohlížení hostitelského počítače a jeho systému souborů, zobrazování informací o aktuální aplikaci nebo jejím uživateli a pro přístup k webovým službám. Většina z těchto možností byla dříve k dispozici prostřednictvím rozhraní .NET Framework Base Class Library, ale kvůli její složitosti bylo pro mnoho programátorů obtížné najít a používat požadované funkce. Obor názvů `My` byl přidán do Visual Studia 2005, aby usnadnil práci programátorům.

Obor názvů `My` je rozdělen do několika kategorií, jak ukazuje následující tabulka.

Objekt	Popis
My.Application	Informace související s aktuální aplikací, včetně názvu, adresáře a čísla verze.
My.Computer	Informace o hardwaru, softwaru a souborech umístěných ve vašem počítači. Objekt `My.Computer` obsahuje i objekt `My.Computer.FileSystem`, který můžete použít k otevření textových a datových souborů v systému.
My.Forms	Informace o formulářích v aktuálním projektu Visual Studia. Kapitola 16 ukazuje použití objektu `My.Forms` pro přepínání mezi formuláři za běhu programu.
My.Resources	Informace o prostředcích vaší aplikace (pouze ke čtení). Umožňuje dynamické získávání prostředků pro vaši aplikaci.
My.Settings	Informace o nastavení vaší aplikace. Umožňuje dynamicky ukládat a získávat nastavení vlastností a dalších údajů pro vaši aplikaci.
My.User	Informace o aktuálním uživateli aktivním v objektu `My.Computer`.
My.WebServices	Informace o webových službách aktivních v objektu `My.Computer` a mechanismus pro přístup k novým webovým službám.

Obor názvů `My` skutečně nabízí rychlý způsob přístupu a můžete s ním pracovat prostřednictvím nástroje Microsoft IntelliSense v okně Code Editoru. Například pro použití okna se zprávou k zobrazení názvu místního počítače, za kterým následuje jméno aktuálního uživatele, napište:

```
MsgBox(My.User.Name)
```



Zobrazí se podobný výstup jako na obrázku vpravo.

Objekt `My.Computer` může zobrazovat mnoho kategorií informací o počítači a jeho souborech. Například následující příkaz zobrazí aktuální systémový čas (místní datum a čas) spravovaný počítačem:

```
MsgBox(My.Computer.Clock.LocalTime)
```

Objekt `My.Computer.FileSystem` můžete použít spolu s metodou `ReadAllText` k otevření textového souboru a zobrazení jeho obsahu v rámci textového pole. Níže uvedenou posloupnost příkazů můžete použít v případě, že máte ve formuláři objekt textového pole s názvem `txtPoznámka` (jako v poslední ukázce programu) a pro získání názvu textového souboru od uživatele chcete využít objekt dialogového okna s názvem `OpenFileDialog1`:

```
Dim CelýText As String =
OpenFileDialog1.Filter = "Textové dokumenty (*.TXT)|*.TXT"
OpenFileDialog1.ShowDialog() 'zobraz dialogové okno Otevřít
If OpenFileDialog1.FileName <> "" Then
   CelýText = _
      My.Computer.FileSystem.ReadAllText(OpenFileDialog1.FileName)
   txtPoznámka.Text = Veskery Text 'zobraz soubor
End If
```

Metoda `ReadAllText` zkopíruje celý obsah daného textového souboru do řetězcové proměnné nebo objektu (v tomto případě se jedná o řetězcovou proměnnou `CelýText`), takže co se týče výkonu a doby vytváření kódu, je metoda `ReadAllText` rychlejší než čtení souboru řádek po řádku pomocí funkce `LineInput`.

Díky své rychlosti je obor názvů `My` vhodný pro mnoho úkolů při programování. Je důležité o tomto prvku vědět a znát možnosti jeho použití. Obor názvů My je v našem případě zvlášť efektivní, protože čteme celý textový soubor. Funkce `LineInput` a třída `StreamReader` nabízejí více možností než tato implementace oboru názvů `My`, mezi něž patří zejména schopnost zpracovávat soubory po jednotlivých řádcích (což je nezbytné pro řazení a rozdělování, jak brzy uvidíte). Měli byste tedy umět pracovat se všemi třemi metodami pro otevírání textových souborů, s nimiž jste se seznámili v této kapitole.

Vytvoření nového textového souboru

Pro vytvoření nového textového souboru pomocí jazyka Visual Basic můžete využít jedné z mnoha funkcí a klíčových slov uvedených v posledním příkladu. Vytváření nových souborů na pevném disku a ukládání dat do těchto souborů se vám bude hodit, pokud se chystáte generovat vlastní hlášení nebo protokoly, ukládat důležité výpočty nebo hodnoty, případně pokud hodláte vytvořit textový editor. Následuje přehled kroků, na jejichž základě budete v programu postupovat:

1. Získejte vstup od uživatele nebo proveďte matematické výpočty, případně obojí.
2. Přiřaďte výsledky k jedné či více proměnným. Mohli byste například přiřadit obsah textového pole k řetězcové proměnné s názvem `VstupProSoubor`.
3. Pomocí ovládacího prvku `SaveFileDialog` požádejte uživatele o cestu k souboru. K zobrazení dialogového okna použijete metodu `ShowDialog`.
4. Cestu získanou v dialogovém okně použijte k otevření souboru pro zápis.
5. Použijte funkci `PrintLine` k uložení jedné či více hodnot do otevřeného souboru.
6. Po dokončení zavřete soubor pomocí funkce `FileClose`.

   Následující cvičení ukazuje, jak lze použít ovládací prvky `TextBox` a `SaveFileDialog` k vytvoření jednoduchého nástroje pro ukládání poznámek. Program používá funkci `FileOpen` k otevření souboru, funkci `PrintLine` k uložení dat a funkci `FileClose` k uza-

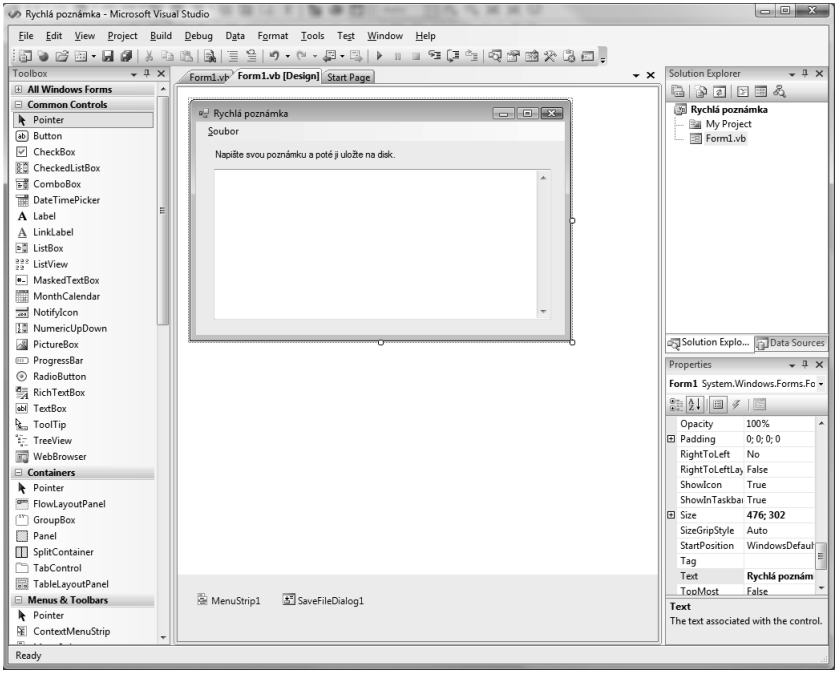
vření souboru. Tento program můžete použít k zápisu poznámek doma či v kanceláři a poté k nim připojit aktuální datum a čas.

Spuštění programu Rychlá poznámka

1. V nabídce File zvolte příkaz Close Project.
2. Otevřete projekt Rychlá poznámka ze složky *c:\vb08sbs\kap13\Rychlá poznámka*.

   Projekt se otevře ve vývojovém prostředí.
3. Nevidíte-li formulář projektu, zobrazte jej nyní.

   Otevře se formulář Rychlá poznámka, který vidíte na následujícím obrázku. Podobá se formuláři v programu Prohlížeč textu. V tomto případě je však ovládací prvek `OpenFileDialog` nahrazen ovládacím prvkem `SaveFileDialog`. Nabídka Soubor obsahuje příkazy Uložit jako, Vložit datum a Konec.



V projektu jsou nastaveny následující vlastnosti:

Objekt	Vlastnost	Nastavení
`txtPoznámka`	`Multiline`	`True`
	`Name`	`txtPoznámka`
	`ScrollBars`	**Vertical**
`lblPoznámka`	`Text`	**„Napište svou poznámku a uložte ji na disk."**
`Form1`	`Text`	**„Rychlá poznámka"**

4. Klepněte na tlačítko Start Debugging.

5. Do textového pole napište následující text nebo nějakou vlastní poznámku:

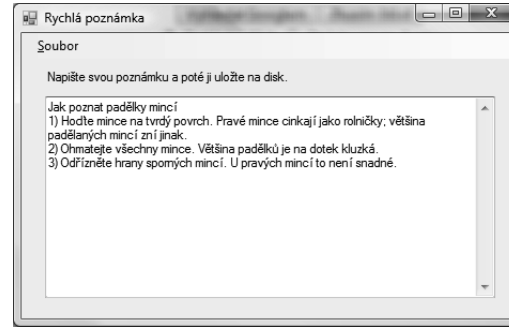
   Jak poznat padělky mincí

   1) Hoďte mince na tvrdý povrch. Pravé mince cinkají jako rolničky; většina padělaných mincí zní jinak.

   2) Ohmatejte všechny mince. Většina padělků je na dotek kluzká.

   3) Odřízněte hrany sporných mincí. U pravých mincí to není snadné.

   Po dokončení bude vaše obrazovka vypadat zhruba jako obrázek vpravo.



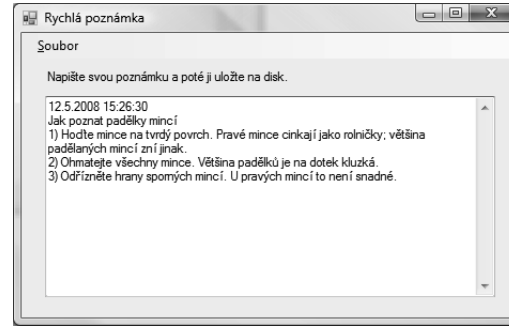
Tip: Pro vložení textu ze schránky Windows do textového pole stiskněte kombinaci kláves Ctrl+V nebo Shift+Insert. Pro zkopírování textu z textového pole do schránky Windows označte text a poté stiskněte kombinaci kláves Ctrl+C.

Nyní si vyzkoušíte použití příkazů v nabídce Soubor.

6. V nabídce Soubor zvolte nejprve příkaz Vložit datum.

   Aktuální datum a čas se zobrazí jako první řádek v textovém poli, jak ukazuje tento obrázek.

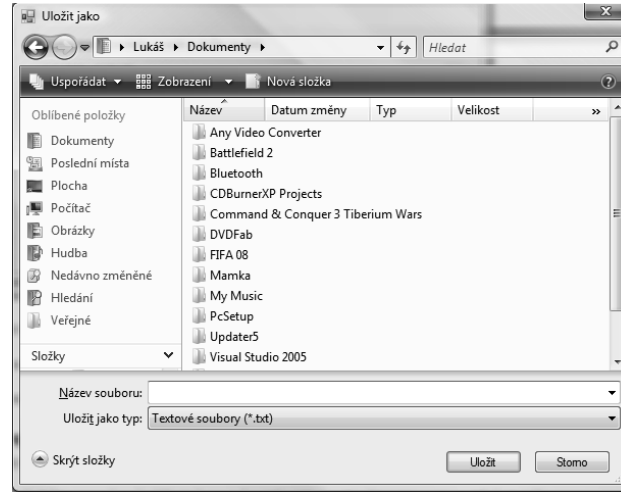
   Příkaz Vložit datum nabízí možnost vložit do souboru aktuální časové razítko, což je vhodné v případě, že vytváříte deník nebo zápisník.



7. V nabídce Soubor zvolte příkaz Uložit jako.

   Program zobrazí dialogové okno Uložit jako se všemi očekávanými funkcemi. Výchozí typ souboru je nastaven na *.txt*. Vaše obrazovka bude nyní vypadat přibližně jako na tomto obrázku:

8. Poté v dialogovém okně Uložit jako otevřete složku *c:\vb08sbs\kap13*



Rychlá poznámka. Do pole Název souboru napište *Falešné mince.txt* a klepněte na tlačítko Uložit.

Text dokumentu se uloží do nového textového souboru *Falešné mince.txt*.

9. V nabídce Soubor zvolte příkaz Konec.

Program se ukončí a vrátíte se do vývojového prostředí.

Nyní si projdeme jednotlivé procedury událostí v programu.

Kód programu Rychlá poznámka

1. V nabídce Soubor formuláře Rychlá poznámka poklepejte na příkaz Vložit datum.

V okně Code Editoru se zobrazí procedura události `VložitDatumToolStripMenuItem_Click`. Uvidíte následující kód programu:

```
txtPoznámka.Text = My.Computer.Clock.LocalTime & _
    vbCrLf & txtPoznámka.Text
txtPoznámka.Select(1, 0) 'odstraň výběr
```

Tato procedura události přidá do textového pole aktuální datum a čas, přičemž spojí řetězce aktuálního data (vygenerovaného objektem `My.Computer.Clock` a vlastností `LocalTime`), znaku pro návrat kurzoru (přidaného konstantou `vbCrLf`) a vlastnosti `Text`. Podobně byste mohli postupovat pro přidání jen aktuálního data (pomocí třídy `DateString`) nebo jakýchkoli jiných informací do textu v textovém poli.

2. Prohlédněte si, jak příkazy spojení fungují, a poté se zaměřte na proceduru události `UložitJakoToolStripMenuItem_Click`.

Uvidíte tento kód programu:

```
SaveFileDialog1.Filter = "Textové soubory (*.txt)|*.txt"
SaveFileDialog1.ShowDialog()
If SaveFileDialog1.FileName <> "" Then
    FileOpen(1, SaveFileDialog1.FileName, OpenMode.Output)
    PrintLine(1, txtPoznámka.Text) 'zkopíruj text na disk
    FileClose(1)
End If
```

Tento blok příkazů používá objekt dialogového okna pro uložení souboru k zobrazení dialogového okna Uložit jako, ověřuje, zda uživatel vybral soubor, otevře soubor pro zápis jako soubor číslo 1, zapíše hodnotu z vlastnosti `txtPoznámka.Text` na disk pomocí funkce `PrintLine` a poté textový soubor zavře. Všimněte si zvláště příkazu,

```
PrintLine(1, txtPoznámka.Text) 'zkopíruj text na disk
```

který přiřazuje celý obsah textového pole do otevřeného souboru. Funkce `PrintLine` se podobá příkazům `Print` a `Print #` ve starších verzích jazyka Visual Basic; směruje výstup do určeného souboru (a ne na obrazovku či tiskárnu). Zde je důležité poznamenat, že celý soubor je uložen ve vlastnosti `txtPoznámka.Text`.

3. Prohlédněte si funkce `FileOpen`, `PrintLine` a `FileClose` a potom pomocí příkazu Konec v nabídce Soubor zavřete program.

Dokončili jste práci s programem Rychlá poznámka.

Zpracování textových řetězců

Jak jste se naučili v předchozích příkladech, můžete pomocí ovládacího prvku `TextBox` a několika vhodně zvolených programových příkazů rychle otevírat, upravovat a ukládat textové soubory. Visual Basic nabízí také mnoho výkonných příkazů a funkcí určených výlučně pro zpracování textových prvků v programech. V této části se naučíte, jak extrahovat informace z textového řetězce, zkopírovat seznam řetězců do pole a jak seřadit seznam řetězců.

Při práci s textovými prvky je velmi důležitá možnost seřazení seznamu řetězců. Základní principy řazení jsou jednoduché. Navrhnete seznam položek, které chcete seřadit, a poté porovnáváte jednotlivé položky, dokud seznam není seřazen ve vzestupném nebo sestupném abecedním pořadí.

Ve Visual Basicu porovnáváte položky pomocí stejných relačních operátorů, které používáte při porovnávání číselných hodnot. Složitější částí (která někdy vyvolává rozsáhlé diskuse mezi počítačovými odborníky) je konkrétní algoritmus řazení, jejž použijete k porovnání prvků v seznamu. V této kapitole nechceme rozebírat výhody a nevýhody různých algoritmů řazení. (Předmětem sporu je obvykle rychlost, která je podstatná při řazení několika tisíců položek.) Zaměříme se raději na to, jak se řadí řetězce základním způsobem. Zároveň si osvojíte postupy nutné k řazení vlastních textových polí, seznamů, souborů a databází.

Třída String a užitečné metody

Dosud jste nejčastěji prováděli spojování řetězců pomocí operátoru `&`. Například následující programový příkaz spojí tři textové řetězcové výrazy a přiřadí výsledek „Přijíždí k nám cirkus!“ k řetězcové proměnné `Slogan`:

```
Dim Slogan As String
Slogan = "Přijíždí" & " k nám " & "cirkus!"
```

Řetězce můžete spojovat a zpracovávat také prostřednictvím metod ve třídě `String` rozhraní .NET Framework. Například metoda `String.Concat` umožňuje stejné spojení řetězců:

```
Dim Slogan As String
Slogan = String.Concat("Přijíždí", " k nám ", "cirkus!")
```

Visual Basic 2008 nabízí dvě metody spojování řetězců a mnoho dalších úkolů souvisejících s prací s řetězci: můžete používat operátory a funkce ze starších verzí jazyka Visual Basic (`Mid`, `UCase`, `LCase` atd.) nebo použít novější metody z rozhraní .NET Framework (`Substring`, `ToUpper`, `ToLower` atd.). Neexistuje žádné pravidlo pro volbu jednoho z postupů, ačkoli starší metody existují hlavně kvůli kompatibilitě. (Microsoft podporuje obě metody a doufá, že naláká programátory, aby se vlastním tempem seznámili s novými funkcemi.) Ve zbývající části této kapitoly se seznámíme s novějšími funkcemi pro práci s řetězci ze třídy `String` rozhraní .NET Framework. Můžete však používat kteroukoli metodu nebo je i vzájemně kombinovat.

Následující tabulka uvádí některé metody, které se vyskytnou v dalších cvičeních, a jejich ekvivalenty v programovacím jazyku Visual Basic. Čtvrtý sloupec tabulky nabízí ukázku kódu pro metody ve třídě `String` rozhraní .NET Framework.

Metoda .NET Frameworku	Funkce jazyka Visual Basic	Popis	Příklad v .NET Frameworku

`ToUpper`	`UCase`	**Změní písmena v řetězci na velké písmo.**	`Dim Jméno, NovéJméno As String`
			`Jméno = "Anna"`
			`NovéJméno = Jméno.ToUpper`
			`'NovéJméno = "ANNA"`

`ToLower`	`LCase`	**Změní písmena v řetězci na malé písmo.**	`Dim Jméno, NovéJméno As String`
			`Jméno = "Anna"`
			`NovéJméno = Jméno.ToLower`
			`'NovéJméno = "anna"`

`Length`	`Len`	**Určuje počet znaků v řetězci.**	`Dim Řeka As String`
			`Dim Délka As Short`
			`22Řeka = "Morava"`
			`Délka = Řeka.Length`
			`'Délka = 11`

`Substring`	`Mid`	**Vrátí fixní počet znaků v řetězci od určeného počátečního bodu. (Poznámka: První prvek v řetězci má index 0.)**	`Dim 2Slp, Střed As String`
			`Slp = "První Druhý Třetí"`
			`Střed = Slp.SubString(6, 5)`
			`'Střed = "Druhý"`

`IndexOf`	`InStr`	**Vyhledá počáteční index jednoho řetězce v rámci delšího řetězce.**	`Dim Jméno As String`
			`Dim PočátečníBod As Short`
			`Jméno = "Jakub"`
			`8PočátečníBod = Jméno.IndexOf("h")`
			`'PočátečníBod = 4`

`Trim`	`Trim`	**Odstraní mezery před a za řetězcem.**	`Dim Mezery, BezMezer As String`
			`Mezery = " Ahoj "`
			`BezMezer = Mezery.Trim`
			`'BezMezer = "Ahoj"`

`Remove`		**Odstraní znaky uprostřed řetězce.**	`Dim Řetězec, OpravenýŘetězec As String`
			`Řetězec = "Ahoj333 tam!"`
			`OpravenýŘetězec = Řetězec.Remove(5, 3)`
			`'OpravenýŘetězec = "Ahoj tam!"`

`Insert`		**Přidá znaky doprostřed řetězce.**	`Dim Starý, Nový As String`
			`Starý = "Zdravím Jano"`
			`Nový = Starý.Insert(8, "tě ")`
			`'Nový = "Zdravím tě Jano"`

`StrComp`		**Porovná řetězce, přičemž ignoruje rozdíl ve velikosti písmen.**	`Dim ř1 As String = "Fotbal"`
			`Dim ř2 As String = "FOTBAL"`
			`Dim Shoda As Short`
			`Shoda = StrComp(ř1, ř2, _`
			`CompareMethod.Text) 'Shoda = 0 [řetězce se shodují]`

Řazení textu

Aby mohl jazyk Visual Basic při řazení porovnat jednotlivé znaky, musí převést každý znak na číslo pomocí *sady znaků ASCII* (označované také jako *sada znaků ANSI*). ASCII je zkratka pro American Standard Code for Information Interchange. Každý ze základních symbolů, které můžete zobrazit ve svém počítači, má jiný kód ASCII. K těmto kódům patří základní sada znaků „z klávesnice" (kódy 32 až 127) a speciální „řídicí" znaky, jako je tabulátor, řádkování a návrat kurzoru (kódy 0 až 31). Například malé písmeno „a" odpovídá kódu ASCII 97 a velké písmeno „A" má kód ASCII 65. Následkem toho považuje Visual Basic tyto dva znaky při řazení nebo provádění jiných porovnávání za zcela odlišné.

V osmdesátých letech rozšířila společnost IBM sadu znaků ASCII o kódy 128 až 255, které představovaly přízvuk, řecké a grafické znaky a také různé symboly. Sada ASCII a tyto další znaky i symboly jsou známé pod označením *rozšířená sada znaků IBM*.

Tip: Chcete-li si prohlédnout kódy v sadě znaků ASCII, hledejte „Chr, ChrW functions" v dokumentaci k Visual Studiu a v části See Also téměř na konci článku klepněte na odkaz ASCII Character Codes.

Pro začínající programátory je velmi důležité, aby se naučili kódy ze sady znaků ASCII, nejedná se však o jedinou sadu znaků. Jak se počítačový trh a aplikační software globálně rozšiřoval, objevil se nový standard pro zastoupení znaků s názvem *Unicode*. Unicode může obsahovat až 65 536 symbolů – což je dostatečný prostor pro tradiční symboly ze sady znaků ASCII a navíc pro většinu (psaných) mezinárodních jazyků a symbolů. Úřad pro standardy spravuje sadu znaků Unicode a přidává do ní pravidelně další symboly. Produkty Microsoft Windows Server 2003, Windows XP, Windows Vista a Visual Studio byly navrhovány pro správu sady znaků ASCII a Unicode. (Více informací o datových typech Unicode, ASCII a Visual Basic naleznete v kapitole 5.)

V další části se blíže seznámíte s využitím sady znaků ASCII při práci s řetězci. S tím, jak budou vaše aplikace postupně složitější a začnete plánovat globální rozšíření svého softwaru, budete se muset naučit více o sadě znaků Unicode a dalších mezinárodních nastaveních.

Práce s kódy ASCII

Pro určení kódu ASCII určitého písmena můžete v jazyce Visual Basic použít funkci `Asc`. Například následující programový příkaz přiřadí číslo 122 (kód ASCII pro malé písmeno „z") k celočíselné proměnné `AscKód` typu `Short`:

```
Dim AscKód As Short
AscKód = Asc("z")
```

Na druhou stranu můžete převést kód ASCII na písmeno pomocí funkce `Chr`. Například tento programový příkaz přiřadí písmeno „z" k proměnné `Písmeno`:

```
Dim Písmeno As Char
Písmeno = Chr(122)
```

Stejného výsledku byste dosáhli i pomocí proměnné `AscKód`, kterou jste právě deklarovali:

```
Písmeno = Chr(AscKód)
```

Jakým způsobem můžete porovnat jeden textový řetězec nebo kód ASCII s jiným? Jednoduše použijete jeden ze šesti operátorů porovnávání, které Visual Basic nabízí pro práci s textovými a číselnými prvky. Přehled těchto operátorů naleznete v následující tabulce.

Operátor	Význam
<>	Nerovná se
=	Rovná se
<	Méně než
>	Více než
<=	Méně než nebo se rovná
>=	Více než nebo se rovná

Nějaký znak je „větší než“ jiný znak v případě, že je vyšší jeho kód ASCII. Například hodnota ASCII písmena „B“ je vyšší než hodnota ASCII písmena „A“, takže výraz

```
"A" < "B"
```

je pravdivý (`True`) a výraz

```
"A" > "B"
```

je nepravdivý (`False`).

Při porovnávání dvou řetězců, které obsahují více než jeden znak, začíná Visual Basic s porovnáním prvního znaku v prvním řetězci s prvním znakem ve druhém řetězci a takto pokračuje po jednotlivých znacích v řetězcích, dokud nenalezne rozdíl. Například řetězce Milena a Michal se shodují až do třetího znaku („l“ a „c“). Protože hodnota ASCII písmena „l“ je vyšší než hodnota písmena „c“, výraz

```
"Milena" > "Michael"
```

je tedy pravdivý (`True`).

Jestliže nejsou mezi řetězci nalezeny žádné rozdíly, tyto řetězce se rovnají. Pokud se v řetězci shoduje několik znaků, jeden řetězec však pokračuje a druhý skončí, je delší řetězec větší než řetězec, který je kratší. Například výraz

```
"AAAAA" > "AAA"
```

je pravdivý (`True`).

Řazení řetězců v textovém poli

Následující cvičení ukazuje, jak můžete použít operátory porovnávání spolu s několika metodami a funkcemi k řazení řádků textu v textovém poli. Program je přepracovanou verzí nástroje Rychlá poznámka a nabízí příkaz Otevřít, který otevře existující soubor, a příkaz Zavřít pro zavření souboru. V nabídce Soubor naleznete také příkaz Seřadit text, jehož prostřednictvím můžete seřadit text zobrazený v textovém poli.

Protože se celý obsah textového pole ukládá do jednoho řetězce, musí program tento dlouhý řetězec nejdříve rozdělit do menších, samostatných řetězců. Tyto řetězce lze uspořádat pomocí procedury s názvem `ShellSort`, což je rutina řazení založená na algoritmu vytvořeném panem Donaldem Shellem v roce 1959. Abychom si tyto úkoly zjednodušili, byl vytvořen modul, který definuje dynamické pole řetězců pro uložení všech řádků v textovém poli. Procedura

`ShellSort` je umístěná v tomto modulu, takže ji lze volat z kterékoli procedury události v projektu. (Více informací o používání modulů naleznete v kapitole 10.) Ačkoli jste se v kapitole 11 naučili pracovat s výkonnou metodou `Array.Sort`, procedura `ShellSort` je flexibilnějším a přizpůsobivějším nástrojem. Vytvořením rutiny navíc získáte více zkušeností s textovými hodnotami – což je jeden z cílů této kapitoly.

Dalším zajímavým aspektem tohoto programu je rutina, která stanovuje počet řádků v objektu textového pole. Žádná z existujících funkcí jazyka Visual Basic nevypočítá tuto hodnotu automaticky. Mým záměrem bylo, aby byl program schopen uspořádat řádek po řádku v textovém poli s jakoukoli velikostí. Abych toho dosáhl, vytvořil jsem níže uvedený kód. Metoda `Substring` slouží k postupnému prohlédnutí jednotlivých písmen v objektu textového pole a funkce `Chr` pak vyhledá znak pro návrat kurzoru (v sadě znaků ASCII má kód 13) na konci každého řádku. (Všimněte si zvláště toho, že metoda `Substring` je použita jako součást vlastnosti `Text` objektu `txtPoznámka`. Třída `String` poskytuje tuto a mnoho dalších metod automaticky pro kterékoli vlastnosti nebo proměnné deklarované jako typ `String`.)

```
Dim řádek, aktualníŘádek, písmeno As String
Dim i, znakyVSouboru, početŘádků As Short
'urči počet řádků v objektu textového pole (txtPoznámka)
početŘádků = 0 'tato proměnná uchovává informaci o celkovém počtu řádků
znakyVSouboru = txtPoznámka.Text.Length 'celkový počet znaků
For i = 0 To znakyVSouboru - 1 'přesun o jeden znak
   písmeno = txtPoznámka.Text.Substring(i, 1) 'získej písmeno
   If písmeno = Chr(13) Then 'je-li nalezen znak pro návrat kurzoru
      početŘádků += 1 'přejdi na další řádek (přičti k celkovému počtu)
      i += 1 'přeskoč znak řádkování (v počítači obvykle následuje za cr)
   End If
Next i
```

Celkový počet řádků v textovém poli je přiřazen k celočíselné proměnné `početŘádků` typu `Short`. Tuto hodnotu později použijeme k nastavení velikosti pole, které bude uchovávat jednotlivé textové řetězce. Výsledné pole řetězců je pak předáno proceduře `Sub ShellSort` pro seřazení a procedura `ShellSort` vrátí pole řetězců v abecedním pořadí. Po seřazení lze pole řetězců jednoduše zkopírovat zpět do textového pole pomocí cyklu `For`.

Spuštění programu Řazení textu

1. Otevřete projekt Razeni textu ze složky *c:\vb08sbs\kap13\Řazení textu*.
2. Klepněte na tlačítko Start Debugging pro spuštění programu.
3. Do textového pole napište následující text (nebo jakýkoli vlastní text):

   Zebra

   Gorila

   Měsíc

   Banán

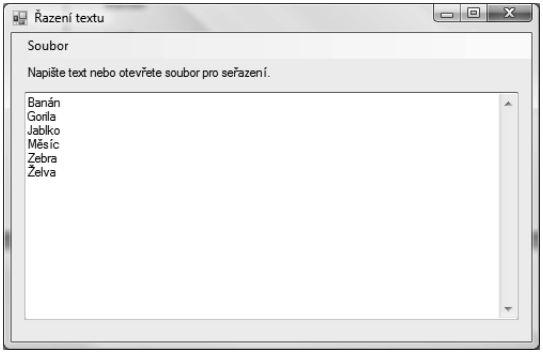
   Jablko

   Želva

Po slově „Želva“ (nebo po jiném posledním řádku) nezapomeňte stisknout klávesu Enter, aby mohl jazyk Visual Basic správně vypočítat počet řádků.

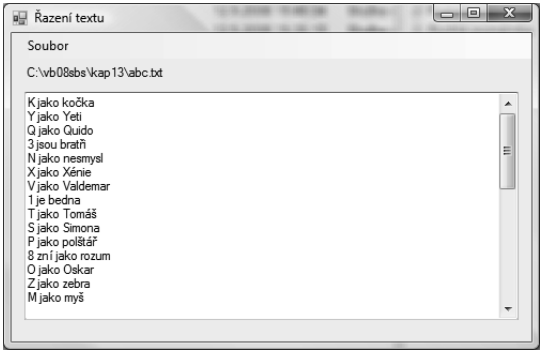
4. V nabídce Soubor zvolte příkaz Seřadit text.

   Napsaný text se abecedně seřadí a znovu se zobrazí v textovém poli (viz obrázek vpravo).



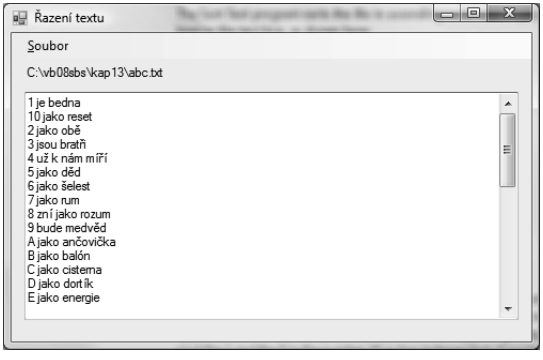
5. V nabídce Soubor zvolte příkaz Otevřít a otevřete soubor *abc.txt* ze složky *c:\vb08sbs\kap13* (viz prostřední obrázek).

   Soubor *abc.txt* obsahuje 36 řádků textu. Každý řádek začíná buď písmenem, nebo číslem od 1 do 10.



6. V nabídce Soubor zvolte příkaz Seřadit text pro uspořádání textu načteného ze souboru *abc.txt*.

   Program Řazení textu pak seřadí položky načtené ze souboru ve vzestupném pořadí a zobrazí seznam řádků v textovém poli, jak ukazuje spodní obrázek:



7. Prohlédněte si výsledek abecedního řazení.

Všimněte si, že ačkoli abecední část řazení proběhla bezchybně, vznikl zvláštní výsledek u jedné z číselných položek – řádek začínající číslem 10 se objevil jako druhý v seznamu a ne jako desátý. Jazyk Visual Basic přečetl 1 a 0 v čísle 10 jako dva nezávislé znaky, ne jako číslo. Protože jsme porovnávali kódy ASCII těchto řetězců zleva doprava, program vytvořil čistě abecední řazení. Pokud chcete v tomto programu seřadit pouze čísla, musíte zakázat textový vstup, upravit kód tak, aby se číselný vstup ukládal do číselných proměnných, a poté porovnat číselné proměnné místo řetězců.

O krok dál: Prohlídka kódu programu Řazení textu

V dalším cvičení budete pracovat s kódem programu Řazení textu, abyste se seznámili s několika dalšími nástroji a zopakovali si některé principy, jež jsme probírali v několika posledních kapitolách.

Program řazení textu

1. V nabídce Soubor programu Řazení textu klepněte na příkaz Konec pro ukončení programu.
2. Otevřete okno Code Editoru formuláře `Form1` a zobrazte kód procedury události `SeřaditTextToolStripMenuItem_Click`.

   S první rutinou v této proceduře jste se již setkali – pomocí metody `Substring`, která vyhledá kódy znaku pro návrat kurzoru, zjišťuje počet řádků v textovém poli. Zbývající část procedury události nastavuje velikost pole řetězců, kopíruje všechny řádky textu do tohoto pole, volá proceduru pro seřazení prvků v poli a zobrazí uspořádaný seznam v textovém poli.

   Celá procedura události `SeřaditTextToolStripMenuItem_Click` vypadá takto:

```
Dim řádek, aktuálníŘádek, písmeno As String
Dim i, znakyVSouboru, početŘádků As Short

'urči počet řádků v objektu textového pole (txtPoznámka)
početŘádků = 0 'tato proměnná uchovává informaci o celkovém počtu řádků
znakyVSouboru = txtPoznámka.Text.Length 'celkový počet znaků
For i = 0 To znakyVSouboru - 1 'posuň se o jeden znak
   písmeno = txtPoznámka.Text.Substring(i, 1) 'získej písmeno
   If písmeno = Chr(13) Then 'je-li nalezen znak pro návrat kurzoru
      početŘádků += 1 'přejdi na další řádek (přičti k celkovému počtu)
      i += 1 'přeskoč znak řádkování (v počítači obvykle následuje za cr)
   End If
Next i

'vytvoř pole pro uložení textu v textovém poli
ReDim poleRetezcu(početŘádků) 'nastav jeho správnou velikost
aktuálníŘádek = 1
řádek = "" 'použij řádek k vytvoření řádků po jednom znaku
For i = 0 To znakyVSouboru - 1 ' znovu projdi textem
   písmeno = txtPoznámka.Text.Substring(i, 1) 'získej písmeno
   If písmeno = Chr(13) Then 'v případě nalezení znaku pro návrat kurzoru
      aktuálníŘádek = aktuálníŘádek + 1 'zvyš počet řádků
      i += 1 'přeskoč znak řádkování
      řádek = "" 'vymaž řádek a přejdi k dalšímu
   Else
      řádek = řádek & písmeno 'přidej písmeno do řádku
      poleRetezcu(aktuálníŘádek) = řádek 'a vlož do pole
   End If
Next i

'seřaď prvky v poli
ShellSort(poleŘetězců, početŘádků)
```

```
'poté zobraz seřazené pole v textovém poli
txtPoznámka.Text = ""
aktuálníŘádek = 1
For i = 1 To početŘádků
   txtPoznámka.Text = txtPoznámka.Text & _
      poleŘetězců(aktuálníŘádek) & vbCrLf
   aktuálníŘádek += 1
Next i
txtPoznámka.Select(1, 0) 'odstraň výběr textu
```

Proměnná `poleŘetězců` byla deklarována v modulu (*Module1.vb*), který je také součástí tohoto projektu (kapitola 10). Pomocí příkazu `ReDim` (kapitola 11) se upraví velikost pole `poleŘetězců` pomocí proměnné `početŘádků`. Tento příkaz vytvoří pole, které obsahuje stejný počet prvků, jako má textové pole řádků (požadavek pro proceduru `ShellSort`). Pomocí cyklu `For` (kapitola 7) a proměnné `řádek` procházíte znovu textovým polem, vyhledáváte znaky pro návrat kurzoru a zkopírujete celý nalezený řádek do `poleŘetězců`. Když je text v poli, voláte proceduru `ShellSort` umístěnou v modulu *Module1.vb*, který jste vytvořili dříve v této kapitole.

3. V okně Code Editoru zobrazte kód pro modul *Module1.vb*.

Tento modul deklaruje veřejnou proměnnou `poleŘetězců` (kapitola 11) a poté definuje obsah procedury `ShellSort`. Procedura `ShellSort` používá příkaz `If` a operátor porovnávání `<=` (kapitoly 6, 8 a 10) k porovnání prvků pole a zamění všechny, které jsou mimo pořadí. Procedura vypadá takto:

```
Sub ShellSort(ByRef sort() As String, ByVal pocetPrvku As Short)
   Dim temp As String
   Dim i, j, rozsah As Short
   'Procedura ShellSort řadí prvky pole sort()
   'v sestupném pořadí a vrátí je volající
   'proceduře.

   rozsah = pocetPrvku \ 2
   Do While rozsah > 0
      For i = rozsah To pocetPrvku - 1
         For j = (i - rozsah + 1) To 1 Step - rozsah
         If sort(j) <= sort(j + rozsah) Then Exit For
            'vyměň prvky pole, které jsou mimo pořadí
            temp = sort(j)
            sort(j) = sort(j + rozsah)
            sort(j + rozsah) = temp
      Next j
   Next i
   rozsah = rozsah \ 2
   Loop
End Sub
```

Při řazení se neustále dělí hlavní seznam prvků do podseznamů, které jsou o polovinu menší. Řazení pak porovná začátek a konec podseznamů kvůli zjištění, zda jsou prvky mimo pořadí. Jsou-li prvky mimo pořadí, jsou zaměněny. Výsledkem je pole `sort()`, které je seřazeno abecedně ve vzestupném pořadí. Chcete-li změnit směr řazení, jednoduše obraťte operátor porovnávání (změňte <= na >=).

Zbývající procedury událostí ve formuláři `Form1` (`OtevřítToolStripMenuItem_Click`, `ZavřítToolStripMenuItem_Click`, `UložitJakoToolStripMenuItem_Click`, `VlozitDatum ToolStripMenuItem_Click` a `KonecToolStripMenuItem_Click`) se podobají procedurám, s nimiž jste se setkali v programech Prohlížeč textu a Rychlá poznámka. (Podrobnosti jsou vysvětlené dříve v této kapitole.)

4. V nabídce `F`ile zvolte příkaz Close Project.

Blahopřeji! Pokud jste se propracovali kapitolami 5 až 13, dokončili jste část o základech programování a jste připraveni zaměřit se na vytváření uživatelských rozhraní programů na profesionální úrovni. Získali jste mnoho zkušeností s programováním v jazyce Visual Basic a s používáním vývojového prostředí Visual Studia. Udělejte si krátkou přestávku, uvidíme se v části III!

Rychlý přehled kapitoly 13

Pro	učiňte následující

Otevření textového souboru	Použijte funkci `FileOpen`. Například:
	`FileOpen(1, OpenFileDialog1.FileName, _`
	`  OpenMode.Input)`

Získání řádku z textového souboru	Použijte funkci `LineInput`. Například: `Dim ŘádekTextu As String ŘádekTextu = LineInput(1)`

Zjištění konce souboru	Použijte funkci `EOF`. Například:
	`Dim ŘádekTextu, CelýText As String`
	`Do Until EOF(1)`
	`  ŘádekTextu = LineInput(1)`
	`  CelýText = CelýText & ŘádekTextu & VbCrLf`
	`Loop`

Zavření otevřeného souboru	Použijte funkci `FileClose`. Například: `FileClose(1)`

Zobrazení textového souboru za použití funkce LineInput	Funkci `LineInput` použijte pro zkopírování textu z otevřeného souboru do řetězcové proměnné a poté přiřaďte řetězcovou proměnnou k textovému poli. Například:
	`Dim CelýText, ŘádekTextu As String`
	`Do Until EOF(1)`
	`  ŘádekTextu = LineInput(1)`
	`  CelýText = CelýText & ŘádekTextu & VbCrLf`
	`Loop`
	`txtPoznámka.Text = CelýText 'zobrazit soubor`

Pro	učiňte následující

Zobrazení textového souboru pomocí třídy `StreamReaderIO`"	Přidejte do sekce deklarací ve formuláři příkaz "`Imports System.` a pak použijte třídu `StreamReader`. Například pro zobrazení souboru v textovém poli `TextBox1`:
	`Dim DataKZobrazení As StreamReader`
	`DataKZobrazení = New StreamReader _`
	`("C:\vb08sbs\kap13\Prohlížeč textu\" & _`
	`"Falešné bankovky.txt")`
	`TextBox1.Text = DataKZobrazení.ReadToEnd`
	`DataKZobrazení.Close()`
	`TextBox1.Select(0, 0)`

Zobrazení textového souboru pomocí oboru názvů `My`	Použijte objekt `My.Computer.FileSystem` a metodu `ReadAllText`. Předpokládejme například, že používáte také objekt dialogového okna pro otevření souboru s názvem `dos` a textové pole `txtPoznámka`:
	`Dim CelýText As String = ""`
	`dos.Filter = "Textové dokumenty (*.TXT)|*.TXT"`
	`dos.ShowDialog() 'zobrazit dialog Otevřít`
	`If dos.FileName <> "" Then`
	`CelýText = _`
	`My.Computer.FileSystem.ReadAllText(dos.FileName)`
	`txtPoznámka.Text = CelýText 'zobrazit soubor`
	`End If`

Zobrazení dialogového okna Otevřít	Přidejte do formuláře ovládací prvek `OpenFileDialog` a použijte metodu `ShowDialog` k otevření dialogu pro otevření souboru. Například: `OpenFileDialog1.ShowDialog()`

Vytvoření nového textového souboru	Použijte funkci `FileOpen`. Například:
	`FileOpen(1, SaveFileDialog1.FileName, _`
	`OpenMode.Output)`

Zobrazení dialogového okna Uložit jako	Přidejte do formuláře ovládací prvek `SaveFileDialog` a použijte metodu `ShowDialog` k uložení objektu. Například: `SaveFileDialog1.ShowDialog()`
Uložení textu do souboru	Použijte funkci `Print` nebo `PrintLine`. Například: `PrintLine(1, txtPoznámka.Text)`
Převedení textových znaků na kódy ASCII	Použijte funkci `Asc`. Například: `ASCII Dim Kod As Short Kod = Asc("A") 'Kód se rovná 65`

Převedení kódů ASCII na textové znaky	Použijte funkci `Chr`. Například:
	`Dim Písmeno As Char`
	`Písmeno = Chr(65) 'Písmeno rovná se "A"`

Extrahování znaků z řetězce	Použijte metodu `Substring` nebo funkci `Mid`. Například:
	`Dim Slp, Střed As String`
	`Cols = "První Druhý Třetí"`
	`Střed = Slp.SubString(6,5)`
	`'Střed = "Druhý"`