Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta – Katedra aplikované fyziky a techniky

Spolupráce mezi VB.NET A MS Office – metodická příručka

Diplomová práce

Vedoucí práce: Ing. Michal Šerý, Ph.D.

Autor: Bc. Václav Prošek

Anotace

Tato práce se zabývá postupy, jak pracovat s aplikacemi kancelářského balíku Microsoft Office z prostředí .NET Framework, prostřednictvím programovacího jazyka Visual Basic .NET. Těchto postupů je využito při vývoji aplikace pro kalibraci vah. Výstupem této aplikace je kalibrační list jako dokument textového editoru Microsoft Word a sešit tabulkového kalkulátoru Microsoft Excel porovnávající výpočty v aplikaci se vzorci aplikace Excel.

Abstract

This dissertation deals with processes how to work with applications of office package by Microsoft Office from the ambience of NET Framework through the programming language Visual Basic.NET. These processes are used to develop the application for calibration of balance. The result of this application is a calibration sheet as a document of Microsoft Word text editor and a workbook of spreadsheet Microsoft Excel comparing calculations in the application with formulas of spreadsheet Excel.

Prohlašuji, že svoji diplomovou práci jsem vypracoval/a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Touto formou děkuji svému vedoucímu práce Ing. Michalu Šerému, Ph.D., za cenné rady a připomínky při zpracování mé práce.

Obsah

1 Úvod

Microsoft Office patří v současné době mezi nejrozšířenější kancelářské aplikace. Proto je při vývoji vlastních aplikací častým požadavkem, aby výstup byl ve formátu některé aplikace z tohoto kancelářského balíku, případně aby naše aplikace dokázala soubory tohoto typu zpracovat.

Sada aplikací Microsoft Office poskytuje objektový model, který popisuje jednotlivé části, z nichž se sada i jednotlivé aplikace v ní skládají, jako objekty, a umožňuje přístup k jejich vlastnostem a metodám.

Je samozřejmě možné programovat přímo v integrovaném vývojovém prostředí Visual Basic for Application. To má však omezené možnosti a je proto vhodné spíše pro jednodušší projekty.

Naproti tomu .NET Framework představuje moderní prostředí pro programování aplikací a předpokládá se jeho další rozvoj. Vývojové prostředí Microsoft Visual Studio pak poskytuje výrazně vyšší komfort a efektivitu práce. Spolupráce s Microsoft Office mu umožňuje vytvářet poměrně jednoduchým způsobem efektní výstupy ve formátech podporovaných Microsoft Office, např. dokumenty, grafy, prezentace apod. To je důležité pro lidi, kteří se neživí jako programátoři, ale vývoj vlastních jednoduchých aplikací je pro ně často rychlejší a levnější než spolupráce s profesionálním programátorem. Proto bylo zvoleno vývojové prostředí Visual Studio Express, které poskytuje dostatek možností a přitom je i pro komerční použití zdarma.

Další výhodou je možnost využití některých funkcí, které Microsoft Office obsahuje, např. kontrola pravopisu nebo výpočetní funkce Excelu.

Nevýhodou tohoto přístupu je závislost na verzi Microsoft Office. Při instalaci nové verze nemusí aplikace pracovat správně a bude potřeba ji upravit. Většina problémů však vzniká, pokud je aplikace použita na počítači s nainstalovanou nižší verzí Microsoft Office, než pro kterou byla napsána.

Tato práce ukazuje postupy, jak pracovat s aplikacemi Microsoft Office z prostředí VB.NET. Těchto postupů je využito v příkladu konkrétního projektu aplikace pro kalibraci vah. Výstupem této aplikace je kalibrační list jako dokument Microsoft Word a sešit Microsoft Excel porovnávající výpočty v aplikaci se vzorci aplikace Excel.

Pro vývoj aplikace bylo použito prostředí Microsoft Visual Studio 2010 Express Edition a Kancelářský balík Microsoft Office 2010.

Pro pochopení následujícího textu je nezbytná alespoň základní znalost programování ve Visual Basic .NET a pokročilá znalost Microsoft Office.

Typografická konvence

Názvy objektů, metod, vlastností, proměnných, parametrů, výčtů konstant a klíčových slov, jsou v textu uvedeny neproporcionálním písmem, v případě barevného tisku odlišeny i barvou podle následujícího schématu: `Jmenný prostor`, `Objekt`, `Metoda`, `Vlastnost`, `Parametr`, `Výčet`, `Proměnná`, `Konstanta`, `Klíčové slovo`. Pro výpisy kódu je použito rovněž neproporcionální písmo menší velikosti v barvách odpovídajících standartnímu nastavení prostředí Visual Studio 2010 Express. Názvy položek menu a oken uživatelského rozhraní jsou uvedeny bezpatkovým písmem tučně: **Menu.**

1.1 Stručný popis .NET Framework

.Net Framework je softwarový rámec vyvinutý firmou Microsoft, který se skládá ze dvou hlavních částí:

– rozšiřitelné knihovny tříd, které poskytují uživatelské rozhraní, přístup k datům, připojení k databázím, šifrování, vývoj webových aplikací, matematické algoritmy a síťovou komunikaci. Tyto třídy mohou programátoři použít ve svých aplikacích,

– run-time prostředí Common Language Runtime (CLR), ve kterém aplikace napsané pro .NET Framework běží. (1) Toto prostředí poskytuje mimo jiné tyto služby:

- překlad z jazyka Microsoft Intermediate Language (MSIL) do nativního kódu procesoru,
- správu paměti alokované objekty,
- integraci s COM modelem (2 str. 22).

Vývoj aplikací pro .NET Framework může probíhat v kterémkoliv z podporovaných jazyků (jedním z nich je i Visual Basic .Net). Kompilátor konkrétního jazyka vygeneruje kód v jazyku MSIL, který CLR přeloží do nativního kódu procesoru modelem. (2 str. 23) Postupy v této práci uvedené lze tedy principiálně použít i pro jiné podporované jazyky.

O správu paměti, kterou alokují objekty v aplikaci, se stará Garbage Collector (GC). V .NET Framework nesledují reference na sebe samy objekty, jak je tomu v technologii COM, ale dělá to právě GC, který tak může odhalit i např. kruhové reference. Objekty, na které neexistují žádné reference, jsou pak z paměti uvolněny. (2 str. 63)

Aplikace vytvořené na platformě .NET Framework nejsou schopné pracovat přímo s objekty MS Office, protože jejich kód je uložen v COM typových knihovnách. Aby mohla aplikace .NET rozeznávat a zpracovávat datové typy, jež jsou uloženy v příslušné typové knihovně jisté aplikace Office, musí disponovat speciální datově-informační vrstvou. Tato vrstva je tvořena typovými metadaty, která mapují všechny datové typy COM typové knihovny zvolené aplikace Office. Typová metadata jsou pro každou z aplikací Office uložena v příslušném primárním sestavení vzájemné spolupráce PIA (Primary Interop Assembly) Office obsahuje všechna nezbytná primární sestavení vzájemné spolupráce, stačí je jenom zahrnout do instalace. Sestavení PIA jsou instalována do mezipaměti sestavení (GAC), odkud jsou přístupná „vnějšímu světu". To znamená, že jakákoliv řízená aplikace se může odkázat na požadované sestavení PIA, které jí nabídne vše potřebné pro nastartování procesu automatizace cílové aplikace Office. Jakmile jsou do projektu aplikace .NET zavedeny reference na sestavení PIA, technologie IntelliSense začne vývojářům nabízet seznamy dosažitelných typů z importovaných jmenných prostorů. (3)

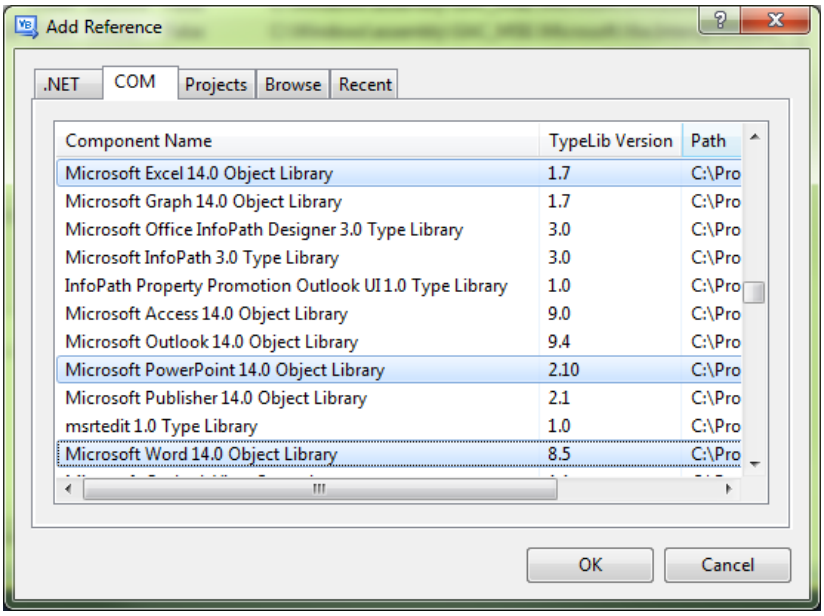
1.2 Všeobecné požadavky na systém

Následující text je otestován pro operační systém Microsoft Windows 7 Home Premium 64-bit CZ, vývojové prostředí Microsoft Visual Studio 2010 Express Edition s programovacím jazykem Microsoft Visual Basic 2010 a kancelářský balík Microsoft Office 2010 Professional. Pro jiné verze výše uvedeného softwaru nemusí dále uvedené postupy platit.

Abychom mohli pracovat s některou aplikací Microsoft Office, musíme mít nainstalovaný kancelářský balík Microsoft Office a do našeho projektu je nutno přidat reference na knihovnu této aplikace. (4) Učiníme tak v menu **Project** – **Add Reference** nebo v okně **Solution Explorer,** položka **My Project,** karta **References,** tlačítko **Add**.

V kartě **COM**, viz obrázek Obr. 1–1, vybereme příslušnou knihovnu Microsoft Word (Excel, PowerPoint, atd.) Object Library a klepneme na tlačítko **OK**. Pokud knihovna není v seznamu, byla sada Microsoft Office pravděpodobně nainstalována bez Primary Interop Assemblies for Microsoft Office a je nutné je nainstalovat, a to buď novou instalací Microsoft Office, nebo samostatně ze stránek společnosti Microsoft. PIA pro Microsoft Office 2010 je ke stažení na adrese:

www.microsoft.com/en-us/download/details.aspx?id=3508

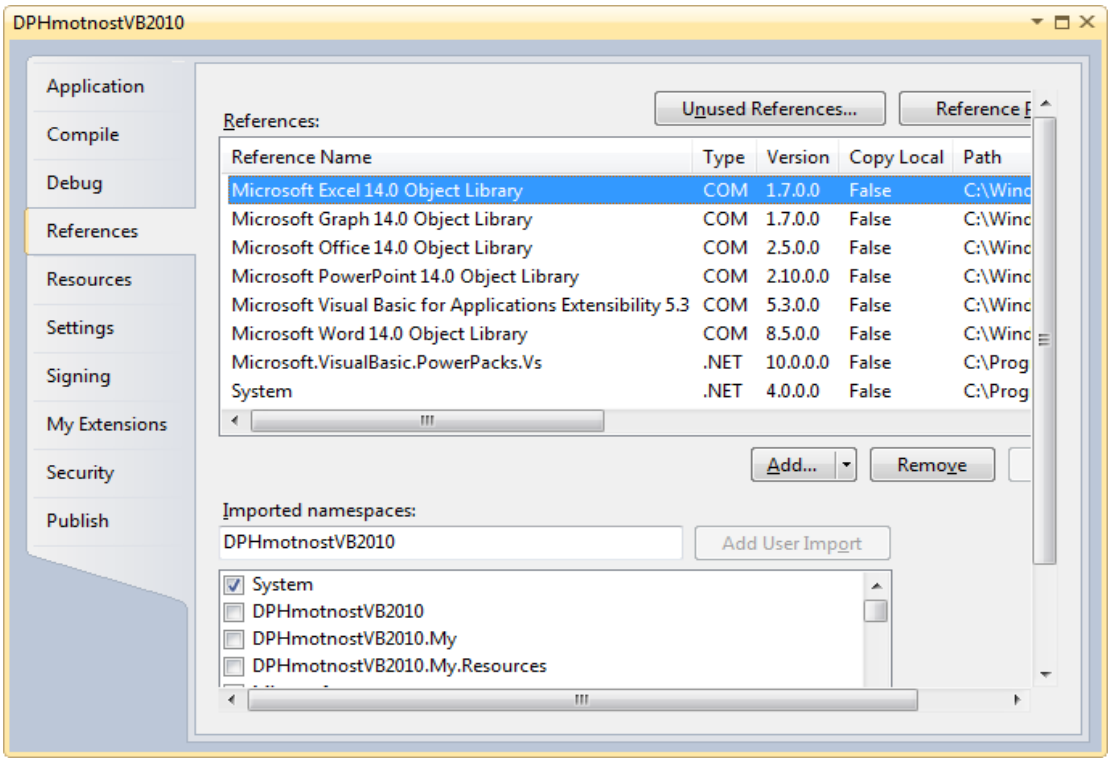


Obr. 1–1 Okno pro přidání referencí

Konfigurace, která nutná pro vývoj aplikací využívajících objektový model Microsoft Office, je i pro ostatní verze popsána na internetové adrese:

http://msdn.microsoft.com/en-us/library/bb398242(v=vs.100).aspx

Pro pohodlnější práci bez zbytečného vypisování celých názvů jmenných prostorů a kvůli kratšímu a přehlednějšímu zápisu je vhodné importovat využívané jmenné prostory. Můžeme tak učinit pomocí zaškrtnutí v seznamu **Imported namespaces** v položce **My Project,** viz obrázek Obr. 1–2, nebo přímo v kódu příkazem `Imports`.



Obr. 1–2 Okno s vlastnostmi projektu

V realizovaném praktickém projektu je třída `Vahy` rozdělena pomocí `Partial` do několika souborů podle aplikací se kterými se právě pracuje a příslušné jmenné prostory jsou importovány v záhlaví těchto souborů. Pokud bychom tyto jmenné prostory importovali zaškrtnutím v seznamu, budou tyto importy platit pro celý projekt a bude docházet ke konfliktům při použití objektů stejného jména z různých jmenných prostorů. Například objekt se jménem `Range` využívá Word i Excel. Ke stejnému efektu dojde i v tom případě, že importujeme příliš obecný jmenný prostoru `Microsoft.Office.Interop`.

V záhlaví souboru pro práci s aplikací Word je proto příkaz:

```
Imports Microsoft.Office.Interop.Word
```

a v souboru pro práci s aplikací Excel příkaz:

```
Imports Microsoft.Office.Interop.Excel
```

Abychom mohli, tam kde chceme, zdůraznit příslušnost objektu do určitého jmenného prostoru, pomůžeme si malým trikem a doplníme do souborů ještě příkazy:

```
Imports Word = Microsoft.Office.Interop.Word
```

popřípadě

```
Imports Excel = Microsoft.Office.Interop.Excel
```

Tyto substituce nám umožňují napsat příkaz dvěma způsoby[1]:

```
Dim WApp As Application
Dim WApp As Word.Application
```

Slovo `Word` zde není označení objektu, ale jde zkrácený zápis jmenného prostoru. V příkladech kódu uvedených v tomto textu je v zájmu názornosti často použit druhý způsob. Tyto příklady jsou ve velké většině napsány jako celé procedury a je možné je z (.pdf) verze této práce snadno zkopírovat do vývojového prostředí a vyzkoušet. Malý problém je pouze tam, kde text příkladu přesahuje přes stránku. V tom případě se přenese i číslo stránky nebo poznámka pod čarou, kterou je nutno vymazat.

1.3 Objektový model

Objektový model jednotlivých aplikací Microsoft Office obsahuje mnoho objektů, které jsou uspořádány v hierarchické struktuře. Na vrcholu této struktury je objekt `Application`. Většina objektů je sdružena do kolekcí, které obsahují všechny objekty daného typu v příslušné úrovni struktury. Některé části dokumentu však existují pouze jako kolekce. Například kolekce `Words` obsahuje všechna slova v dokumentu, ale objekt „`Word`“, představující slovo, neexistuje. Stejné je to i u kolekce `Characters` v aplikacích Excel a Word. Jména kolekcí jsou většinou odvozena z množného čísla názvu příslušného objektu (např. objekty typu `Document` patří do kolekce `Documents`). Počet jejich členů vrací vlastnost `Count`, a na jednotlivé členy se můžeme odkazovat pomocí indexu (u některých kolekcí i pomocí jména konkrétního objektu, vlastnost `Name`).

---

[1] Uvedený příklad platí pokud je v souboru importován pouze samotný jmenný prostor `Microsoft.Office.Interop.Word`, importujeme-li další jmenné prostory `Microsoft.Office.Interop.`..., musíme použít způsob uvedený na druhém řádku, protože kompilátor v tom případě neví do kterého jmenného prostoru objekt `Application` patří.

Příklad: máme dokument Word, který obsahuje dva oddíly (objekt `Section`), první s pěti a druhý s deseti odstavci (objekt `Paragraph`). Kolekce `Sections` objektu `Document` proto obsahuje dva objekty `Section` a kolekce `Paragraphs` obsahuje patnáct objektů `Paragraph`. Kolekce `Paragraphs` prvního oddílu obsahuje pět objektů `Paragraph` a kolekce druhého oddílu deset objektů. Zápis v kódu – viz výpis V 1-1.

V 1-1 Počty členů v kolekcích dokumentu

```
Dim WordApp As Word.Application
Dim poc As Integer
poc = WordApp.Documents(1).Sections.Count
'poc má hodnotu 2
poc = WordApp.Documents(1).Paragraphs.Count
'poc má hodnotu 15
poc = WordApp.Documents(1).Sections(1).Range.Paragraphs.Count
'poc má hodnotu 5
poc = WordApp.Documents(1).Sections(2).Range.Paragraphs.Count
'poc má hodnotu 10
```

1.4 Obecný postup pro práci s aplikacemi Microsoft Office

Při práci s objekty jednotlivých aplikací Microsoft Office (s dokumenty, sešity, prezentacemi atd.) je potřeba dodržet následující kroky:

1. Deklarace proměnné typu `Application` a vytvoření nové instance této proměnné viz – výpis V 1-2. Tím dojde ke spuštění příslušné aplikace na pozadí, o čemž se můžeme přesvědčit ve správci souborů.
2. Otevření již existujícího nebo nového základního dokumentu. Pro Word je to `Document`, pro Excel `Workbook` a pro PowerPoint `Presentation`.
3. Práce s příslušným dokumentem.
4. Uzavření dokumentu, ať už s jeho uložením nebo bez uložení. Uzavření dokumentu je nezbytné kvůli následnému korektnímu ukončení aplikace. Pokud bychom tak neučinili, zobrazí se okno s dotazem na uložení.
5. Ukončení aplikace.

V 1-2 Spuštění konkrétní aplikace Word, Excel a PowerPoint.

```
Dim AppWord As New Word.Application
Dim AppExcel As New Excel.Application
Dim AppPowPoint As New PowerPoint.Application
```

Krok č. 2 není vždy nutný. Např. pro práci s funkcemi aplikace Excel není třeba otvírat sešit, stačí spustit aplikaci a můžeme pracovat s třídou `WorksheetFunction`. Potom vynecháme i krok č. 3.

Stejně tak, jestliže chceme, aby s vytvořeným dokumentem uživatel dále pracoval, kroky č. 4. a č. 5 vynecháme a aplikaci naopak zviditelníme. Neúmyslné opomenutí kroku č. 5 má však za následek hromadění spuštěných procesů aplikací na pozadí a následkem toho vzrůstající požadavky na systémové prostředky. Zejména v průběhu ladění vlastních programů je vhodné čas od času zkontrolovat **Správce úloh** a ukončit přebytečné instance.

Pokud z nějakého důvodu, například pro ladění nebo pro zobrazení výsledného dokumentu, chceme, aby byl viditelný, přiřadíme vlastnosti `Visible` objektu `Application` hodnotu `True`.[2]

Pro zamezení nechtěné komunikace s uživatelem ve skrytém režimu je potřeba zamezit zobrazování dialogových oken. Musíme dávat pozor, zejména při zavírání a ukládání dokumentů.

[2] U aplikace PowerPoint hodnotu `MsoTriState.msoCTrue`

2 Práce s textovým editorem Microsoft Word

Aplikaci Microsoft Word lze použít k vytvoření nebo úpravě dokumentu, anebo využít některých jejích funkcí, např. ke kontrole pravopisu libovolného textu. Postupy níže uvedené ukazují, jak spustit a ukončit Microsoft Word, jak vytvořit dokument a naplnit ho zformátovaným textem, práci s tabulkami, obrázky apod.

Další možností je vytvořit dokument ručně, na vhodná místa vložit záložky a pomocí nějaké aplikace je naplnit. Jako příklad použití funkce aplikace Word je uveden postup, jak zkontrolovat pravopis v ovládacím prvku formuláře `RichTextBox`.

Jak už bylo výše uvedeno, objektový model aplikace Word obsahuje mnoho objektů, v příkladech uvedených v následujícím textu je popsáno, jak pracovat s několika z nich. Jsou to především objekt `Application` – představující vlastní aplikaci Word, a objekt `Document` – představující dokument aplikace Word. Dále pak objekty `Selection` a `Range`, představující nějakou oblast v dokumentu. Objekty `Paragraph` – odstavec, `Section` – oddíl, `Table` – tabulka, `Field` – pole, `HeadersFooters` – záhlaví/zápatí. A nakonec objekty pro práci s kontrolou pravopisu `ProofreadingErrors`, navrhovanými opravami `SpellingSuggestions` a se synonymy `SynonymInfo`.

Informace o objektech, jejich metodách a vlastnostech, uvedených v této kapitole, byly čerpány z online dokumentace pro `Microsoft.Office.Interop.Word` na internetových stránkách firmy Microsoft (5)

2.1 Spuštění a ukončení aplikace Word

Pokud v naší aplikaci vytvoříme novou instanci objektu `Word.Application` příkazem:

```
Dim WordApp As New Word.Application,
```

spustí se nový proces aplikace Microsoft Word. Tento proces je standardně spuštěn na pozadí. Uživatel jeho existenci nijak nezaznamená a nemá možnost s ním pracovat. Toto chování se liší od situace, kdy uživatel otvírá další dokumenty sám. Tyto jsou otevřeny v rámci jednoho procesu a jejich seznam je vidět v pásu karet **Zobrazení** skupina **Okno** příkaz **Přepnout Okna**[3]. Můžeme se o tom přesvědčit ve **Správci úloh**.

[3] v Microsoft® Office Word 2003 nabídka **Okno**

Pro ukončení aplikace Word použijeme metodu `Quit`, v jejíchž parametrech můžeme určit, zda uložíme změny v dokumentu. Potom je třeba přiřadit nepoužívané proměnné hodnotu `Nothing` kvůli jejímu uvolnění Garbage Collectorem z paměti.

V následujícím příkladu, viz výpis V 2-1, deklarujeme dvě proměnné typu `Word.Application`, čímž spustíme Word ve dvou nezávislých procesech, které následně ukončíme. Vytváření a ukončování jednotlivých procesů lze sledovat ve **Správci úloh**. Na začátku tohoto výpisu je uveden i import jmenného prostoru `Microsoft.Office.Interop.Word`, který v dalších výpisech uveden není, ale předpokládá se.

Proměnná `WordApp1` je deklarována na úrovni třídy mimo proceduru, kde vytváříme instanci objektu `Word.Application` pomocí `New`. Toho využijeme, pokud chceme v jedné proceduře Word spustit a pracovat s ním i v jiných procedurách. Nesmíme pak zapomenout na ukončení aplikace Word a přiřazení hodnoty `Nothing` do této proměnné[4].

V 2-1 Spuštění aplikace Word ve dvou nezávislých procesech a jejich ukončení.

```
Imports Word = Microsoft.Office.Interop.Word
Imports Microsoft.Office.Interop.Word

    'Deklarace proměnné WordApp1
Dim WordApp1 As Word.Application

Sub Word_RunQuit ()

    'Vytvoření instance objektu
    WordApp1 = New Word.Application

    'Současná deklarace a vytvoření instance objektu
    Dim WordApp2 As New Word.Application

    'Ukončení procesů aplikace Word
    WordApp2.Quit()
    WordApp2 = Nothing
    WordApp1.Quit()
    WordApp1 = Nothing
End Sub
```

[4]V případě verze Word 2010 jsem pozoroval, že k ukončení procesu stačí zavolání metody `Quit`. Přesto bych přiřazení hodnoty `Nothing` nevynechával.

2.2 Otevření a zavření dokumentu

Pro práci s dokumentem používáme objekt `Document`. Každý otevřený dokument je členem kolekce `Documents`, která obsahuje všechny otevřené dokumenty aktuální instance objektu `Word.Application`.

Existující dokument otevřeme metodou `Open` kolekce `Documents`, a pro vytvoření nového dokumentu použijeme metodu `Add` kolekce `Documents`.

S dokumentem můžeme potom pracovat, a to buď pomocí proměnné typu `Document`, které nový dokument přiřadíme nejlépe hned při jeho otevření, nebo pomocí kolekce `Documents`. Viz výpis V 2-2.

V 2-2 Otevření existujícího dokumentu aplikace Word

```
    'Spuštění aplikace Word
Dim WordApp As New Word.Application

    'Deklarace proměné typu Document
Dim Doc As Document

    'Otevření existujícího dokumentu a přiřazení do proměnné
Doc = WordApp.Documents.Open("C:\Priklad\ExistDokument.doc")

    'Přidání nového dokumentu a přiřazení do proměnné
Doc = WordApp.Documents.Add()
```

Na konkrétní dokument se odkazujeme buď pomocí jeho jména (vlastnost `Name`) nebo pomocí indexu. Pokud použijeme index, musíme si dát pozor, se kterým dokumentem pracujeme. Nové dokumenty jsou přidávány na začátek seznamu.

V příkladu viz výpis V 2-3 vytvoříme dva nové dokumenty, vypíšeme jejich jména v pořadí, v jakém jsou v kolekci `Documents`, a ukončíme Word. Můžeme si všimnout, že druhý vložený dokument je na prvním místě.

V 2-3 Vytvoření dvou nových dokumentů a výpis jejich názvů

```
Sub Doc_index()

        'Spuštění aplikace Word
    Dim WordApp As New Word.Application
    WordApp.Documents.Add()
    WordApp.Documents.Add()
    With WordApp.Documents
        Console.WriteLine("Jméno 1. položky: " & .Item(1).Name)
        Console.WriteLine("Jméno 2. položky: " & .Item(2).Name)
    End With

    'Ukončení aplikace Word
    WordApp.Quit()
```

```
    WordApp = Nothing
End Sub
```

Tento příklad vypíše:

```
Jméno 1. položky: Dokument2
Jméno 2. položky: Dokument1
```

Poznámka: Pokud je už otevřen nějaký dokument aplikace Word, mohou se jména nových dokumentů lišit od těch uvedených v příkladu, obrácené pořadí však zůstane zachováno.

Vzhledem k tomu, že jsme do dokumentů ve výpisu V 2-3 nic nezapisovali, ani je jinak neměnili, jsou při ukončení aplikace Word zavřeny bez vyvolání dialogového okna s dotazem na uložení.

Pro zavření dokumentu slouží metoda `Close` kolekce `Documents`. Parametr `SaveChanges` určuje, zda je dokument uzavřen bez uložení změn, nebo zda se objeví dialogové okno s možností uložení dokumentu, v tomto okně si může uživatel zvolit jméno a formát, ve kterém bude dokument uložen.

V dokumentu musí být také provedeny nějaké změny, jinak je zavřen bez ohledu na tento parametr; v případě, že jde o nový dokument, který nebyl předtím uložen, vyvolá zmíněné okno i pokud zvolíme hodnotu `WdSaveOptions.wdSaveChanges`.

Poznámka: Stisk tlačítka **Storno** v tomto dialogovém okně vyvolá výjimku, kterou je nutné ošetřit.

Jestliže nám záleží na tom, jak bude dokument uložen, musíme ho uložit sami. K tomu slouží metoda `SaveAs2`[5]. v jejích parametrech můžeme specifikovat i typ ukládaného dokumentu. To nám umožňuje například jednoduché vytváření souborů ve formátu (.pdf). Po uložení dokument zavřeme metodou `Close`, s parametrem `WdSaveOptions.wdDoNotSaveChanges`.

Metodu `Close` můžeme použít jak na jednotlivý dokument, viz výpis V 2-4, tak i na celou kolekci `Documents` – viz výpis V 2-5, kde každý dokument uložíme v jiném formátu a potom je zavřeme všechny najednou.

Metoda `Close` volaná bez parametrů uzavře dokument jako by měl parametr `SaveChanges` hodnotu `WdSaveOptions.wdPromptToSaveChanges`.

[5] Existuje i starší metoda `SaveAs` která se liší jen tím, že neobsahuje nepovinný parametr `CompatibilityMode`.

V 2-4 Zavření jednotlivého dokumentu s příklady parametrů

```
    'Vytvoření dokumentů
Dim WDoc1 As Word.Document = WordApp.Documents.Add()
Dim WDoc2 As Word.Document = WordApp.Documents.Add()
Dim WDoc3 As Word.Document = WordApp.Documents.Add()

    'Zavření dokumentů
WDoc1.Close(WdSaveOptions.wdDoNotSaveChanges)
WDoc2.Close(WdSaveOptions.wdPromptToSaveChanges)
WDoc3.Close(WdSaveOptions.wdSaveChanges)
```

V 2-5 Uložení a zavření všech dokumentů najednou po předchozím uložení

```
    'Vytvoření dokumentů
Dim WDoc1 As Word.Document = WordApp.Documents.Add()
Dim WDoc2 As Word.Document = WordApp.Documents.Add()
Dim WDoc3 As Word.Document = WordApp.Documents.Add()

    'Uložení dokumentů
WDoc1.SaveAs2("Test", FileFormat:=WdSaveFormat.wdFormatDocumentDefault)
WDoc2.SaveAs2("Test", FileFormat:=WdSaveFormat.wdFormatDocument97)
WDoc3.SaveAs2("Test", FileFormat:=WdSaveFormat.wdFormatPDF)

    'Zavření dokumentů
WordApp.Documents.Close(WdSaveOptions.wdDoNotSaveChanges)
```

2.3 Práce s obsahem dokumentu

S obsahem dokumentu můžeme pracovat prostřednictvím objektů `Document` nebo `Selection` a jejich dalších členských objektů, jako například odstavec (objekt `Paragraph`), záložka (objekt `Bookmark`), tabulka (objekt `Table`), atd. Objekt `Document` je členem kolekce `Documents`, která obsahuje všechny otevřené dokumenty v aktuální instanci aplikace Word. Objekt `Selection` představuje výběr v panelu nebo okně a týká se vždy aktivního dokumentu, zatímco objekt `Document` může představovat kterýkoliv otevřený dokument. (Pro práci s částí dokumentu pak používáme objekt `Range`.)

Prostřednictvím těchto dvou objektů můžeme přistupovat k jednotlivým částem dokumentu a dále je formátovat nebo číst či upravovat jejich obsah. Práce s těmito objekty jako jsou tabulky, odstavce, slova, obrázky, atd. je stejná, ať už patří do výběru představovaného objektem `Selection` nebo `Range`, nebo do objektu `Document`. Je jen potřeba sledovat, s čím právě pracujeme. Viz výpis V 2-6, kde proměnné `Pa` a `Pb` mohou, ale nemusí obsahovat stejné odstavce, záleží na rozsahu objektu `Selection`.

V 2-6 Přiřazení odstavce do proměnné

```
Dim Pa As Paragraph
Dim Pb As Paragraph
Pa = WordApp.ActiveDocument.Paragraphs(1)
Pb = WordApp.Selection.Paragraphs(1)
```

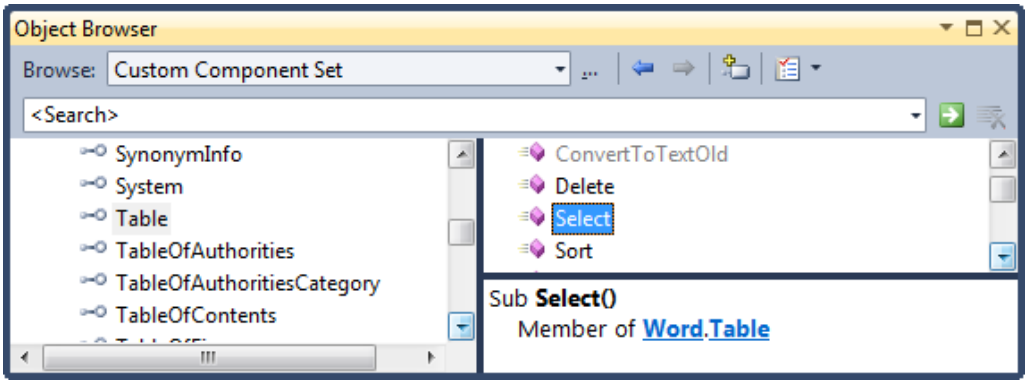
Pro formátování písma slouží objekt `Font` a pro formátování odstavců objekt `ParagraphFormat`, tyto objekty vrací vlastnost `Font` respektive `Format` objektů, které mohou obsahovat nějaký text. Například `Range`, `Paragraph`, `Selection` atd.

2.3.1 Výběr – objekt Selection

Objekt `Selection` je v jedné instanci aplikace Word vždy pouze jeden[6] a představuje výběr v panelu nebo okně. Může jít o celý dokument nebo o jeho část: odstavec, řádek, skupinu znaků, tabulku, záložku atd. V případě, že není vybrána žádná část dokumentu, představuje objekt `Selection` pozici kurzoru.

Objekt `Selection` vrací vlastnost `Selection` objektů: `Application`, `Pane` a `Windows`. Tato vlastnost je pouze pro čtení. Vlastnosti `Start` a `End` objektu `Selection` udávají aktuální pozici začátku a konce tohoto objektu v dokumentu. Pozice na začátku dokumentu má hodnotu 0. Pokud není nic vybráno a objekt `Selection` představuje pozici kurzoru, mají obě vlastnosti shodnou hodnotu.

Pro vložení obsahu do objektu `Selection`, tzn. pro vybrání nějaké části dokumentu, můžeme změnit hodnoty vlastností `Start` a `End`, nebo použijeme metodu `Select` příslušného objektu. Objekty, které tuto metodu implementují, si můžeme zobrazit, například pomocí prohlížeče objektů viz obrázek Obr. 2–1



Obr. 2–1 Okno prohlížeče objektů

[6] Pokud máme vytvořeno více instancí aplikace Word, pracujeme s více objekty `Selection`

Práce s dokumentem pomocí objektu `Selection` se podobá ručnímu psaní. Můžeme přesouvat kurzor na místo, kam chceme vložit text, vybírat části textu a formátovat je. Stejně tak tabulky, obrázky atd.

Pro vložení textu a konce odstavce můžeme použít metody `TypeText` respektive `TypeParagraph`. Objekt `Selection` se přitom přesouvá za vložené a kolabuje na velikost kurzoru.

Například ve výpisu V 2-7 vložíme do nového dokumentu dva odstavce a nastavíme jim řez písma na tučně.

Protože pracujeme s novým prázdným dokumentem, je objekt `Selection` nastaven na začátek dokumentu a je zkolabovaný (tzn. `WdApp.Selection.Start = WdApp.Selection.End = 0`), představuje tedy pozici kurzoru.

Pomocí metody `TypeText` vložíme do dokumentu text a vložením znaku konce odstavce metodou `TypeParagraph` vytvoříme první odstavec. Dále vložíme text druhého odstavce. Objekt `Selection` stále představuje pozici kurzoru na konci dokumentu. Nastavením vlastnosti `Start` objektu `Selection` na začátek dokumentu (hodnota 0) rozšíříme tento objekt na oba odstavce, a nastavením jeho vlastnosti `Font.Bold = True` ho zformátujeme.

V 2-7 Objekt Selection

```
Sub Doc_Selection()

    'Spuštění aplikace Word a vytvoření dokumentu
    Dim WordApp As New Word.Application
    Dim WDoc As Word.Document = WordApp.Documents.Add()
    WordApp.Visible = True
    With WordApp
        .Selection.TypeText("Toto je 1. odstavec")
        .Selection.Paragraphs.Add()
        .Selection.Start = 0
        .Selection.Font.Bold = True
        .Selection.Start = .Selection.End
        .Selection.TypeText("Toto je 2. odstavec")
        .Selection.TypeParagraph()
    End With

    'Ukončení aplikace Word – Pozor, vyvolá dotaz na uložení.
    WordApp.Quit()
    WordApp = Nothing
End Sub
```

2.3.2 Objekty Document a Range

Pokud pracujeme s dokumentem prostřednictvím objektu `Document`, používáme pro práci s jeho částí objekt `Range`.

Objekt `Range` představuje souvislou oblast v dokumentu, která je definována pozicí počátečního a koncového znaku. Objekt `Range` je nezávislý na objektu `Selection` a můžeme s ním pracovat, aniž by to na výběr mělo vliv. Objekt `Range` lze získat i prostřednictvím vlastnosti `Range` objektu `Selection`. V dokumentu může být takovýchto oblastí více. Pro objekt `Document` a kolekce `Shapes` (tvary, obrázky, apod.), `CanvasShapes` (tvary na kreslicím plátně) a `GroupShapes` (seskupené tvary) získáme objekt `Range` pomocí stejnojmenné metody `Range`. Například metoda `Range` objektu `Document` – vrací oblast zahrnující celý dokument. Metoda `Range` kolekce `Paragraphs` – vrací oblast zahrnující příslušný odstavec. Pokud napíšeme: `Document.Paragraphs(1).Range` – bude vrácena oblast zahrnující první odstavec dokumentu.

U kolekcí je povinný parametr `index` specifikující konkrétní objekt[7]. V případě objektu `Document` má metoda `Range` volitelné parametry `Start` a `End`. Pokud je nepoužijeme, je obsahem objektu `Range` celý dokument mimo poznámek pod čarou a záhlaví/zápatí. Pozici a rozsah objektu `Range` lze změnit editací vlastností `Start` a `End`.

Objekt `Document` samozřejmě má mnoho dalších vlastností, které vracejí objekty pro práci s: vzhledem stránky (objekt `PageSetup`), styly (objekt `Style`), záhlavím/zápatím (objekt `HeaderFoter`), atd. Některé z nich jsou popsány dále.

2.3.3 Odstavce, věty, slova, písmena

Odstavce, věty, slova a písmena, tyto části textu, kromě odstavců nemají vlastní objekty, ale existují pro ně kolekce, `Sentences`, `Words` a `Characters`. Můžeme tak například pomocí vlastnosti `Count` kolekce `Sentences` zjistit počet vět v dokumentu. Členy těchto kolekcí jsou objekty `Range` a můžeme se na ně odkazovat pomocí indexu.

Zde pokládám za nutné upozornit na kolekci `Characters`, protože je na rozdíl od ostatních číslovaná od nuly.

---

[7] Samozřejmě pouze pokud chceme pracovat s některým z objektů kolekce, pro práci s celou kolekcí index nepoužíváme.

Každý tento objekt Range má vlastnosti Sentences, Words a Characters. Není tedy chyba napsat takovouto konstrukci, která zjišťuje počet vět ve slově, i když to logicky nedává smysl a proměnná PocVet bude mít hodnotu nula:

```
PocVet = WDoc.Paragraphs(1).Range.Words(2).Characters(2).Sentences.Count
```

Odstavec představuje objekt Paragraph který není typu Range a pro odkaz na věty, slova nebo písmena musíme tedy použít vlastnost Range.

Pokud tedy chceme mít v dokumentu WDoc druhé slovo prvního odstavce tučné, použijeme příkaz:

```
WDoc.Paragraphs(1).Range.Words(2).Font.Bold = True
```

Je však třeba počítat s tím, že Word považuje za slovo i například čárku nebo jiný nepísmenný respektive nečíselný znak oddělený mezerou. Ve větě „Les, který hořel celý týden, hasilo pět hasičských sborů“ je slovo „který“ až třetí.

2.3.4 Záložky

Záložku představuje objekt Bookmark. Pomocí záložek můžeme vkládat data do předem vytvořených dokumentů aplikace Word na určitá místa. Může se jednat např. o formuláře ve firmě. Do takového dokumentu vložíme předem definované záložky a potom ho vyplníme daty z naší aplikace. Ve výpisu V 2-8 jsou jména záložek obsažených v dokumentu vložena do pole. V praktické aplikaci pak můžeme nalezená jména porovnat se seznamem v naší aplikaci a do záložek, které v dokumentu existují, vložit text.

V 2-8 Záložky

```
Sub Zálozky()

   'Spuštění aplikace Word a otevření dokumentu obsahujícího záložky
   Dim WordApp As New Word.Application
   WordApp.Visible = True
   Dim WDoc As Word.Document
   WDoc = WordApp.Documents.Open("C:\pokus.docx")

   'vložení názvů záložek do pole
   Dim poczalozek As Integer
   poczalozek = WDoc.Bookmarks.Count
   Dim polezalozek(poczalozek) As String

   For i = 1 To poczalozek
      polezalozek(i) = WDoc.Bookmarks(i).Name
   Next
```

```
    'Ukončení aplikace
    WordApp.Quit()
    WordApp = Nothing
End Sub
```

2.3.5 Tabulky

Tabulku představuje objekt `Table`. Každá tabulka patří do nějaké kolekce `Tables`, která obsahuje všechny tabulky v příslušném dokumentu, výběru nebo rozsahu.

Tabulku můžeme vložit prostřednictvím výběru (objekt `Selection`) metodou `Add` – viz výpis V 2-9. Parametr `Range` získáme z objektu `Selection` jako hodnotu vlastnosti `Range`. Pokud objekt `Selection` představuje kurzor, pak vložení tabulky na něj nemá žádný vliv, hodnota jeho vlastností `Start` a `End` se nemění a kurzor je v první buňce tabulky. Představuje-li nějaký výběr, je tento nahrazen tabulkou a objekt `Selection` zkolabuje na kurzor, který se opět nachází v první buňce tabulky.

Pro vložení tabulky jinam než do aktuálního výběru je vhodnější objekt `Document` – viz výpis V 2-10, kde pomocí objektu `Selection` vytvoříme dokument se třemi odstavci a pro umístění tabulku do druhého využijeme objekt `Range` .

V 2-9 Vložení tabulky do dokumentu prostřednictvím objektu `Selection`

```
Sub TabulkaSelection()

    'Spuštění aplikace Word a vytvoření dokumentu
    WordApp = New Word.Application
    WordApp.Visible = True
    Dim DocKL As New Document
    DocKL = WordApp.Documents.Add()

    WordApp.Selection.TypeText("Text před tabulkou")
    WordApp.Selection.TypeParagraph()       'odstavec
    WordApp.Selection.TypeText("Zde je tabulka (za chvilku)")
    WordApp.Selection.Start = WordApp.Selection.End - 12 'rozšíření výběru

    'Vložení tabulky
    WordApp.Selection.Tables.Add(WordApp.Selection.Range, 2, 2)

    'Ukončení aplikace Word – Pozor, vyvolá dotaz na uložení.
    WordApp.Quit()
    WordApp = Nothing
End Sub
```

V 2-10 Vložení tabulky prostřednictvím objektu `Range`

```
Sub Tabulka()

    'Spuštění aplikace Word a vytvoření dokumentu
    WordApp = New Word.Application
    WordApp.Visible = True
    Dim DocKL As New Document
    DocKL = WordApp.Documents.Add()

     'Vložení textu a odstavců
    WordApp.Selection.TypeText("Text před tabulkou")
    WordApp.Selection.TypeParagraph()       '1. odstavec
    WordApp.Selection.TypeParagraph()       '2. odstavec zde bude tabulka
    WordApp.Selection.TypeText("Text za tabulkou")

        'Vložení tabulky
    Dim Misto As Range = DocKL.Paragraphs(2).Range
    DocKL.Tables.Add(Range:=Misto, NumRows:=3, NumColumns:=4)

     'Ukončení aplikace Word – Pozor, vyvolá dotaz na uložení.
    WordApp.Quit()
    WordApp = Nothing
End Sub
```

Na jednotlivé buňky tabulky se odkazujeme prostřednictvím metody `Cell(radek, sloupec)` s parametry `radek` a `sloupec`, která vrací objekt `Cell`. Viz výpis V 2-11, kde vytvoříme tabulku se třemi řádky a pěti sloupci, které ve druhém řádku očíslujeme. Řádky i sloupce tabulky jsou číslovány od jedničky.

Pro úpravu textu v buňce musíme použít vlastnost `Range` objektu `Cell`, která nám vrátí objekt `Range` jehož vlastnost `Text` můžeme upravit. Objekt `Cell` vlastnost `Text` nemá.

Buňky tabulky můžeme i slučovat. Slouží k tomu metoda `Merge`. Na sloučené buňky se potom odkazujeme jako na první buňku sloučené oblasti. Viz výpis V 2-11, kde sloučíme buňky v prvním řádku tabulky a vložíme do něj nadpis.

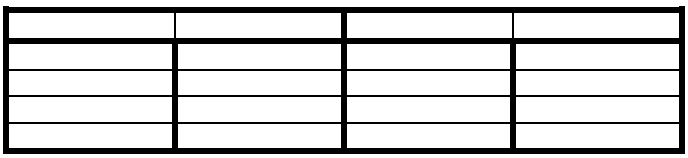
V 2-11 Slučování buněk v tabulce

```
Sub Tabulka3x5()

    'Spuštění aplikace Word a vytvoření dokumentu
    WordApp = New Word.Application
    Dim DocKL As New Document
    WordApp.Visible = True
    DocKL = WordApp.Documents.Add()
    WordApp.Selection.TypeParagraph() '1. odstavec
    WordApp.Selection.TypeParagraph() '2. odstavec
```

```
    'Vložení tabulky
    Dim Tab1 As Table
    Dim Misto As Range = DocKL.Paragraphs(2).Range
    Tab1 = DocKL.Tables.Add(Range:=Misto, NumRows:=3, NumColumns:=5)
    For i = 1 To 5
        Tab1.cell(2, i).range.text = i
    Next

        'Sloučení buněk prvního řádku
    Tab1.Cell(1, 1).Merge(MergeTo:=Tab1.Cell(1, 5))

    Tab1.cell(1, 1).Range.text = "Tabulka 1"

     'Ukončení aplikace Word – Pozor, vyvolá dotaz na uložení.
    WordApp.Quit()
    WordApp = Nothing
End Sub
```

Tabulky můžeme samozřejmě formátovat, měnit barvu buněk, tloušťku čar ohraničení atd. Můžeme to dělat jak pro jednotlivé buňky, tak pro celou tabulku.

Pro práci s okraji slouží objekt `Border`, který patří do kolekce `Borders`. Tuto kolekci vrací stejnojmenná vlastnost `Borders` objektů `Table` nebo `Range`. Počet členů kolekce je pevně daný a závisí na typu objektu, s jehož ohraničeními pracujeme. Na jednotlivé členy odkazujeme prostřednictvím indexu. Tento index může nabývat jedné z hodnot výčtu `WdBorderType`. Protože tyto hodnoty jsou vlastně konstanty typu `Integer`, lze tam, kde upravujeme stejnou vlastnost více objektů `Border`, použít cyklus `For`. Například formátování všech vnějších (případně i vnitřních) ohraničení tabulky nebo její části, jak je ukázáno ve výpisu V 2-12, ve kterém vytvoříme tabulku podle obrázku Obr. 2–2. Zde musím upozornit na drobný problém. Hodnotu indexu `WdBorderType.wdBorderVertical` lze použít pouze pro objekt `Table`, tedy pro celou tabulku. Pro objekt `Range` představující nějakou část tabulky, i kdyby zahrnoval celou tabulku, nefunguje. Vlastnost `HasVertical` vrací `False`. Vertikální vnitřní tlusté ohraničení v tabulce je tedy vytvořeno postupně. Pro rychlejší práci se všemi vnitřními a vnějšími ohraničeními můžeme použít i vlastnosti kolekce `Borders` `Inside…` respektive `Outside…`, ale i pro tyto vlastnosti platí výše uvedené upozornění.



Obr. 2–2 Příklad tabulky

V 2-12 Vytvoření tabulky s různě tlustými čarami.

```
Private Sub TabulkaCary()

    'Spuštění aplikace Word a vytvoření dokumentu
    WordApp = New Word.Application
    WordApp.Visible = True
    Dim DocKL As New Document
    DocKL = WordApp.Documents.Add()

    'vložení tabulky
    Dim Tab1 As Table
    Tab1 = DocKL.Tables.Add(Range:=DocKL.Paragraphs(1).Range, NumRows:=5, NumColumns:=4)

    'Nastavení ohraničení všech buněk na tabulky na plnou čáru šířky 0,5b
    For i = -1 To -6 Step -1
        Tab1.Borders(i).LineStyle = WdLineStyle.wdLineStyleSingle
        Tab1.Borders(i).LineWidth = WdLineWidth.wdLineWidth050pt
    Next

    'Nastavení ohraničení prvního řádku na plnou čáru šířky 2,25b
    Dim rRozsah As Range
    rRozsah = Tab1.Rows(1).Range
    For i = -1 To -4 Step -1
        rRozsah.Borders(i).LineWidth = WdLineWidth.wdLineWidth225pt
    Next

    'Nastavení pravého ohraničení druhé buňky prvního řádku na plnou čáru šířky 2,25b
    rRozsah.SetRange(Tab1.Cell(1, 1).Range.Start, Tab1.Cell(1, 2).Range.End)
    rRozsah.Borders(WdBorderType.wdBorderRight).LineWidth = WdLineWidth.wdLineWidth225pt

    'Vložení oblasti od duhého řádku včetně do konce tabulky do proměnné rRozsah
    s ůmyslem nastvit tlusté vertikální čáry
    rRozsah.SetRange(Tab1.Cell(2, 1).Range.Start, Tab1.Cell(5, 4).Range.End)

    'POZOR tento řádek nefunguje pro objekt Range, je tedy označen jako komentář
    'rRozsah.Borders(WdBorderType.wdBorderVertical).LineWidth = WdLine-
     Width.wdLineWidth225pt

    'Je nutné to udělat postupně po sloupcích
    For s = 1 To Tab1.Columns.Count
        rRozsah.SetRange(  Tab1.Cell(2, s).Range.Start,
                           Tab1.Cell(5, s).Range.End)
        For i = -1 To -4 Step -1
            rRozsah.Borders(i).LineWidth = WdLineWidth.wdLineWidth225pt
        Next
    Next
```

```
    'Ukončení aplikace Word – Pozor, vyvolá dotaz na uložení
    WordApp.Quit()
    WordApp = Nothing
End Sub
```

2.3.6 Záhlaví a zápatí

Se záhlavím nebo zápatím v dokumentu pracujeme pomocí kolekce `HeadersFooters`, která obsahuje objekty `HeaderFoter`. Objekt `HeaderFoter` představuje jednotlivé záhlaví nebo zápatí. Tento objekt vrací vlastnosti `Headers(parametr)` a `Footers(parametr)` objektu `Section`.

Každý oddíl v dokumentu (objekt `Section`) obsahuje vlastní kolekci `HeadersFooters`. Propojení na předchozí oddíly zajišťuje vlastnost `LinkToPrevious`, která může nabývat hodnot `True` – záhlaví/zápatí je stejné jako předchozí, nebo `False` – záhlaví/zápatí se liší od předchozího. Objekt `HeaderFoter` vrací také stejnojmenná vlastnost objektu `Selection`, pokud se tento v záhlaví nebo zápatí nachází.

Zda jde o záhlaví nebo zápatí, můžeme určit pomocí vlastnosti `IsHeader`. Protože existují tři druhy záhlaví/zápatí, může být parametrem vlastností `Headers` a `Footers` jedna z následujících konstant obsažených ve výčtu `WdHeaderFooterIndex`.

`wdHeaderFooterPrimary` – normální záhlaví/zápatí

`wdHeaderFooterFirstPage` – záhlaví/zápatí první stránky

`wdHeaderFooterEvenPages` – záhlaví/zápatí sudých stránek.

Pokud vložíme nějaký text do záhlaví/zápatí první stránky bude v dokumentu viditelný, až po nastavení vlastnosti `DifferentFirstPageHeaderFooter` objektu `PageSetup`. Stejně tak je tomu popřípadě záhlaví/zápatí sudých stránek, kde nastavujeme vlastnost `OddAndEvenPagesHeaderFooter`. Vlastnost `Count` kolekce `HeadersFooters` vždy vrací hodnotu 3, i když nejsou zobrazeny všechny druhy záhlaví/zápatí.

Ve výpisu V 2-13 vytvoříme čtyřstránkový dokument s odlišným zápatím pro první i sudé stánky:

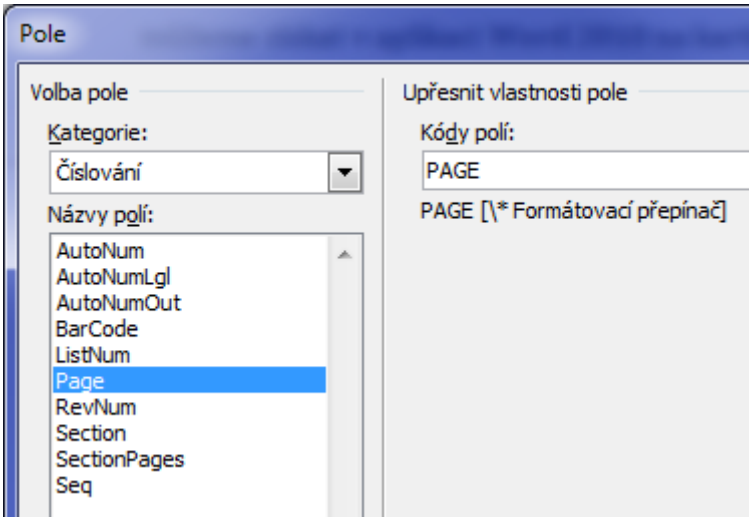
V 2-13 Vložení zápatí

```
Sub VlozZapati()

    'Spuštění aplikace Word a vytvoření dokumentu
    Dim WordApp As New Word.Application
    WordApp.Visible = True
```

```
    Dim WDoc As Word.Document = WordApp.Documents.Add()
    Dim r As Range

        'Vložíme 4 stránky
    For i = 1 To 4
        WDoc.Range.InsertAfter(i & ". strana")
        WDoc.Range.InsertParagraphAfter()
    Next
    For i = 1 To 5 Step 2 'Zalomení stránky se počítá jako odstavec
        r = WDoc.Paragraphs(i).Range
        r.Start = r.End
        r.InsertBreak()
    Next

        'Nastavíme vlastnosti PageSetup pro první a sudé stránky
    WDoc.PageSetup.DifferentFirstPageHeaderFooter = CInt(True)
    WDoc.PageSetup.OddAndEvenPagesHeaderFooter = CInt(True)

    'Vložíme texty do zápatí
    With WDoc.Sections(1).Footers
        .Item(wdHeaderFooterFirstPage).Range.Text = "Zápatí 1. strana"
        .Item(wdHeaderFooterEvenPages).Range.Text = "Zápatí sudá strana"
        .Item(wdHeaderFooterPrimary).Range.Text = "Zápatí normálné"
    End With

    'Ukončení aplikace Word – Pozor, vyvolá dotaz na uložení
    WordApp.Quit()
    WordApp = Nothing
End Sub
```

2.3.7 Vkládání polí

Protože do záhlaví nebo zápatí často vkládáme pole s číslem stránky, zařazuji popis práce s poli hned za text o záhlaví/zápatí. Pole jsou reprezentována objektem `Field`, který je členem kolekce `Fields`. Tato kolekce obsahuje všechna pole ve výběru (objekt `Selection`), oblasti (objekt `Range`) nebo v dokumentu (objekt `Document`). Pro vložení pole metodou `Add` kolekce `Fields` máme dvě možnosti: buď pro parametr `Type` použijeme hodnotu `WdFieldType.wdFieldEmpty` a konkrétní pole specifikuje v parametru `Text` (hodnoty tohoto parametru můžeme získat v aplikaci Word 2010 na kartě **Vložení** –> **Rychlé části** –> **Pole** viz Obr. 2–3), nebo použijeme přímo příslušnou hodnotu parametru `Type`, kterou zjistíme v prohlížeči objektů (**Object Browser**).



Obr. 2–3 Okno pro vkládání polí

V příkladu ve výpisu V 2-14 deklarujeme pomocnou proměnnou `Oblast` typu `Range`, kterou použijeme pro určení místa, kde bude pole. Potom vložíme prvním pole s číslem aktuální stránky a druhým způsobem pole s počtem stránek v oddílu.

V 2-14 Vložení pole

```
Sub VlozPole()

    'Spuštění aplikace Word a vytvoření dokumentu
    Dim WordApp As New Word.Application
    WordApp.Visible = True
    Dim WDoc As Word.Document = WordApp.Documents.Add()
    Dim Oblast As Range

    'Vložíme text do zápatí
    Oblast = WDoc.Sections(1).Footers(WdHeaderFooterIndex.wdHeaderFooterPrimary).Range
    Oblast.Text = "Strana č. "

    'Objekt Range necháme zkolabovat, aby nebyl přepsán, a vložíme pole s číslem strany
    Oblast.Collapse(WdCollapseDirection.wdCollapseEnd)
    Oblast.Fields.Add( Range:=Oblast,
                       Type:=WdFieldType.wdFieldEmpty,
                       Text:="Page ", PreserveFormatting:=True)

    'Na konec oblasti zápatí vložíme text a pole s počtem stran oddílu
    Oblast = WDoc.Sections(1).Footers(WdHeaderFooterIndex.wdHeaderFooterPrimary).Range
    Oblast.Collapse(WdCollapseDirection.wdCollapseEnd)
    Oblast.Text = ", Počet stran "
    Oblast.Collapse(WdCollapseDirection.wdCollapseEnd)
    Oblast.Fields.Add(Range:=Oblast, Type:=WdFieldType.wdFieldSectionPages)

    'Ukončení aplikace Word – Pozor, vyvolá dotaz na uložení
    WordApp.Quit()
    WordApp = Nothing
End Sub
```

2.3.8 Obrázky

Obrázky jsou v tomto případě myšleny i automatické tvary, objekty WordArt atd. Ty mohou být umístěny buď v textové vrstvě, nebo ve výkresové vrstvě dokumentu. Obrázek v textové vrstvě dokumentu je představován objektem `InlineShape` a obrázek ve výkresové vrstvě dokumentu je představován objektem `Shape`.

Pro práci s objekty `Shape` v určité části dokumentu je třeba použít vlastnost `ShapeRange`, která vrací kolekci `ShapeRange` obsahující všechny objekty `Shape` v dané části dokumentu. I když by se na první pohled mohlo zdát, že objekt ve výkresové vrstvě dokumentu nepatří například do odstavce (objekt `Paragraph`), opak je pravdou. Každý takový objekt má kotvu (objekt `Anchor`), a právě pozice kotvy určuje příslušnost objektu `Shape` do kolekce `ShapeRange` příslušné části dokumentu, například odstavce, oddílu, záhlaví atd. Pokud je to zapotřebí, lze objekty `Shape` a `InlineShape` mezi sebou převádět pomocí metod `ConvertToInlineShape` a `ConvertToShape`.

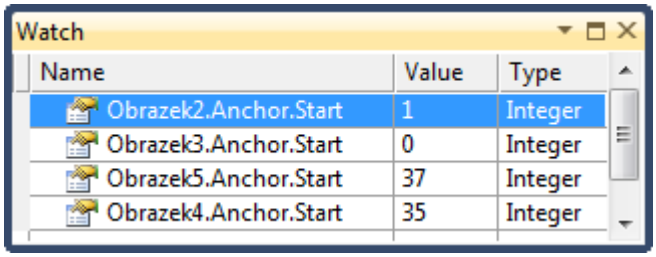
Pro vložení různých „druhů“ obrázků používáme různé metody `Add...`. Pro obrázek ze souboru je to metoda `AddPicture`, pro tvary `AddShape` pro čáru `AddLine` atd.

Pro objekty `Shape` a `InlineShape` se tyto metody přes stejný název liší a žádný z objektů neobsahuje všechny. Například obrazce (metoda `AddShape`) lze vložit pouze do výkresové vrstvy dokument a naopak vodorovnou čáru pouze do textové vrstvy (metoda `AddHorizontalLineStandard`). Obecně metody objektu `Shape` obsahují navíc parametry pro určení pozice, rozměrů a parametr `Anchor` pro umístění kotvy. U metod objektu `InlineShape` parametr `Range` určuje přímo místo kam bude obrázek vložen. Pokud není parametr `Range` uveden, je obrázek vložen na pozici kurzoru; v případě, že uveden je a označuje nějakou oblast, obrázek nahradí obsah této oblasti. Pokud není uveden parametr `Anchor`, je pozice kotvy zvolena podle souřadnic vkládaného obrázku. Bohužel nemusí to platit vždy. Ve výpisu V 2-15 vkládáme obrázek představovaný proměnou `Obrazek1` do textové vrstvy a obrázky představované proměnnými `Obrazek1 až 4`, do výkresové vrstvy. Kotvy obrázků `Obrazek1` a `Obrazek2` vkládaných na první stranu se pro oba vloží do odstavce číslo 1, i když podle jejich parametru `Anchor` by se měly vložit do odstavců číslo 3 a 4. Zatímco kotvy obrázků `Obrazek3` a `Obrazek4` vkládaných na stranu druhou se vloží správně. Pozici kotvy každého objektu můžeme zjistit z vlastností `Start` a `End` objektu `An-`

chor. Na obrázku Obr. 2–4 jsou uvedeny vlastnosti Start jednotlivých obrázků a jejich hodnoty těsně před skončením procedury Obrazky z výpisu V 2-15.

V 2-15 Vkládání obrázků do dokumenu

```
Sub Obrazky()

    'Spuštění aplikace Word a vytvoření dokumentu
    Dim WordApp As New Word.Application
    WordApp.Visible = True
    Dim WDoc As Word.Document = WordApp.Documents.Add()
    'Vložíme obrázek do textu a upravíme jeho rozměry
    WordApp.Selection.TypeText("Zde je obrázek v textu ")
    Dim CestakObrazku As String = "d:\obrazek.png"
    Dim Obrazek1 As InlineShape
    Obrazek1 = WordApp.Selection.InlineShapes.AddPicture(FileName:=CestakObrazku)
    Obrazek1.Width = 50 :  Obrazek1.Height = 50

    'Vložíme 5 odstavců, konec stránky a zase 5 odstavců
    For i = 1 To 5
        WordApp.Selection.TypeParagraph()
    Next
    WordApp.Selection.InsertBreak()
    For i = 1 To 5
        WordApp.Selection.TypeParagraph()
    Next
    Dim p As Integer = WDoc.Paragraphs.Count
    Dim Obrazek2, Obrazek3, Obrazek4, Obrazek5 As Shape

        'Vložíme obrázky do výkresové vrstvy s různou pozicí kotvy
    Dim Oblast As Range

    'Obrázky na první straně
    Oblast = WDoc.Paragraphs(3).Range
    Obrazek2 = WDoc.Shapes.AddPicture(CestakObrazku, Left:=150, Anchor:=Oblast)

    Oblast = WDoc.Paragraphs(4).Range
    Obrazek3 = WDoc.Shapes.AddPicture(CestakObrazku, Left:=150, Anchor:=Oblast)

    'Obrázky na druhé straně
    Oblast = WDoc.Paragraphs(9).Range
    Obrazek4 = WDoc.Shapes.AddPicture(CestakObrazku, Left:=150, Anchor:=Oblast)

    Oblast = WDoc.Paragraphs(10).Range
    Obrazek5 = WDoc.Shapes.AddPicture(CestakObrazku, Left:=150, Anchor:=Oblast)

    WordApp.Quit()
    WordApp = Nothing
End Sub
```



Obr. 2–4 Okno se zobrazením proměnných

2.3.9 Kontrola pravopisu, opravy a synonyma

2.3.9.1 Kontrola pravopisu

Pokud v naší aplikaci někam píšeme texty delší než několik slov, může být užitečné zkontrolovat jejich pravopis. Zde uvádím postup pro kontrolu pravopisu s využitím aplikace Word. Kontrolu pravopisu můžeme využít pro jakýkoliv text. Uvedený příklad, viz výpis V 2-16, zkontroluje obsah ovládacího prvku typu `RichTextBox` a chybná slova obarví na červeno. Tento příklad předpokládá existenci `RichTextBoxu` jménem `rtxVysledek` s nějakým textem.

Ve výpisu V 2-16 nejprve deklarujeme proměnou s názvem `ChybnaSlova` typu `ProofreadingErrors,` což je kolekce slov s pravopisnou nebo gramatickou chybou. Tuto kolekci naplníme pomocí vlastnosti `SpellingErrors` objektu `Range`, který obsahuje kontrolovaný text. (4) Pro získání objektu `Range` si musíme vytvořit dokument. Pro gramatické chyby bychom použili vlastnost `GrammaticalErrors`. Máme tak vlastně seznam slov, se kterým můžeme dále pracovat. Cyklem `For` pak projdeme celou kolekci `ChybnaSlova` a každé slovo najdeme v `RichTextBoxu` a obarvíme na červeno.

V 2-16 Kontrola Pravopisu

```
    'Spuštění aplikace Word a vytvoření dokumentu
Sub KontolaPravopisu()
    Dim WordApp As New Word.Application
    WordApp.Documents.Add()

    'Vytvoříme proměnnou typu Range do které vložíme text, který budeme kontrolovat
    Dim DRange As Range = WordApp.ActiveDocument.Range
    DRange.Text = rtxVysledek.Text

    'Vytvoříme proměnnou, vložíme do ní chybná slova, najdeme je v textboxu a obarvíme
    Dim ChybnaSlova As ProofreadingErrors
    ChybnaSlova = DRange.SpellingErrors
    Dim poz As Integer
    For cslova = 1 To ChybnaSlova.Count
```

```
        poz = rtxVysledek.Find(str:=ChybnaSlova.Item(cslova).Text)
        Dim dSlova As Integer = ChybnaSlova.Item(cslova).Text.Length
        rtxVysledek.Select(start:=poz, length:=dSlova)
        rtxVysledek.SelectionColor = Color.Red
        Me.Refresh()'Překreslí formulář pro zviditelnění přebarvování slov při krokování
    Next

    'Ukončení aplikace Word – Pozor, vyvolá dotaz na uložení
    WordApp.Quit()
    WordApp = Nothing
End Sub
```

2.3.9.2 Opravy

Chceme-li využít funkci navrhování oprav, deklarujeme si proměnnou `NavrzeneOpravy` typu `SpellingSuggestions`, kterou pomocí metody `GetSpellingSuggestions` naplníme návrhy na opravu pro konkrétní slovo. Viz výpis V 2-17, kde navržená slova, pokud nějaká jsou, z proměnné `NavrzeneOpravy` vypíšeme do `ListBoxu` `LsbOpravy`. (4)

V 2-17 Seznam navržených oprav

```
Sub KontolaPravopisuOprava(ByVal ChSlovo As String)

    'Spuštění aplikace Word a vytvoření dokumentu
    Dim WordApp As New Word.Application
    WordApp.Documents.Add()

    'Vygeneruje seznam navrhovaných náhrad
    Dim NavrzeneOpravy As SpellingSuggestions
    NavrzeneOpravy = WordApp.GetSpellingSuggestions(ChSlovo)
    If NavrzeneOpravy.Count > 0 Then 'Pokud tento seznam není prázdný
        Dim copravy As Integer
        For copravy = 1 To NavrzeneOpravy.Count 'Naplní ListBox
            lsbOpravy.Items.Add(NavrzeneOpravy.Item(copravy).Name)
        Next
    End If

    'Ukončení aplikace Word – Pozor, vyvolá dotaz na uložení
    WordApp.Quit()
    WordApp = Nothing
End Sub
```

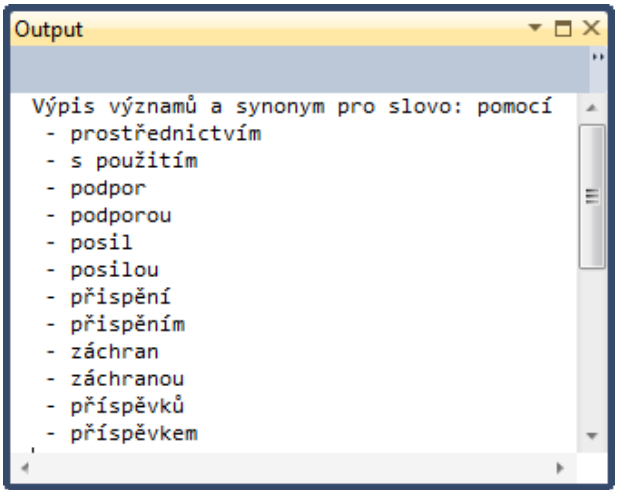
2.3.9.3 Synonyma

Aplikace Word zahrnuje i tezaurus, který nabízí seznam synonym a antonym. Pro práci s tezaurem obsahuje Word objekt `SynonymInfo`, který vrací stejnojmenná vlastnost objektu `Application`. Parametry této vlastnosti jsou: slovo, ke kterému hledáme významy, a jazyk, pro který je hledáme. Objekt `SynonymInfo` pomocí vlast-

nosti `MeaningList` vrací seznam možných významů a ke každému z nich potom prostřednictvím vlastností `SynonymList` a `AntononymList` výčet synonym a antonym. Ve výpisu V 2-18 – je předvedena práce se synonymy, která jsou vypsána do okna **Output**, a na obrázku Obr. 2–5je uveden takový výpis pro slovo „pomocí“.

V 2-18 Výpis synonym

```
Sub Synonyma(ByVal Slovo As String)

    'Spuštění aplikace Word
    Dim WordApp As New Word.Application
    Dim Syn As SynonymInfo

    Syn = WordApp.SynonymInfo(Slovo, WdLanguageID.wdCzech)
    Console.WriteLine("Výpis významů a synonym pro slovo: " & Syn.Word)

    Dim Vyznamy As Object
    Vyznamy = Syn.MeaningList

    For i = 1 To Syn.MeaningCount
        Dim Syno As Object
        Syno = Syn.SynonymList(Syn.MeaningList(i))
        For Each s In Syno
            Console.WriteLine(" - " & s)
        Next
    Next

    'Ukončení aplikace Word
    WordApp.Quit()
    WordApp = Nothing
End Sub
```



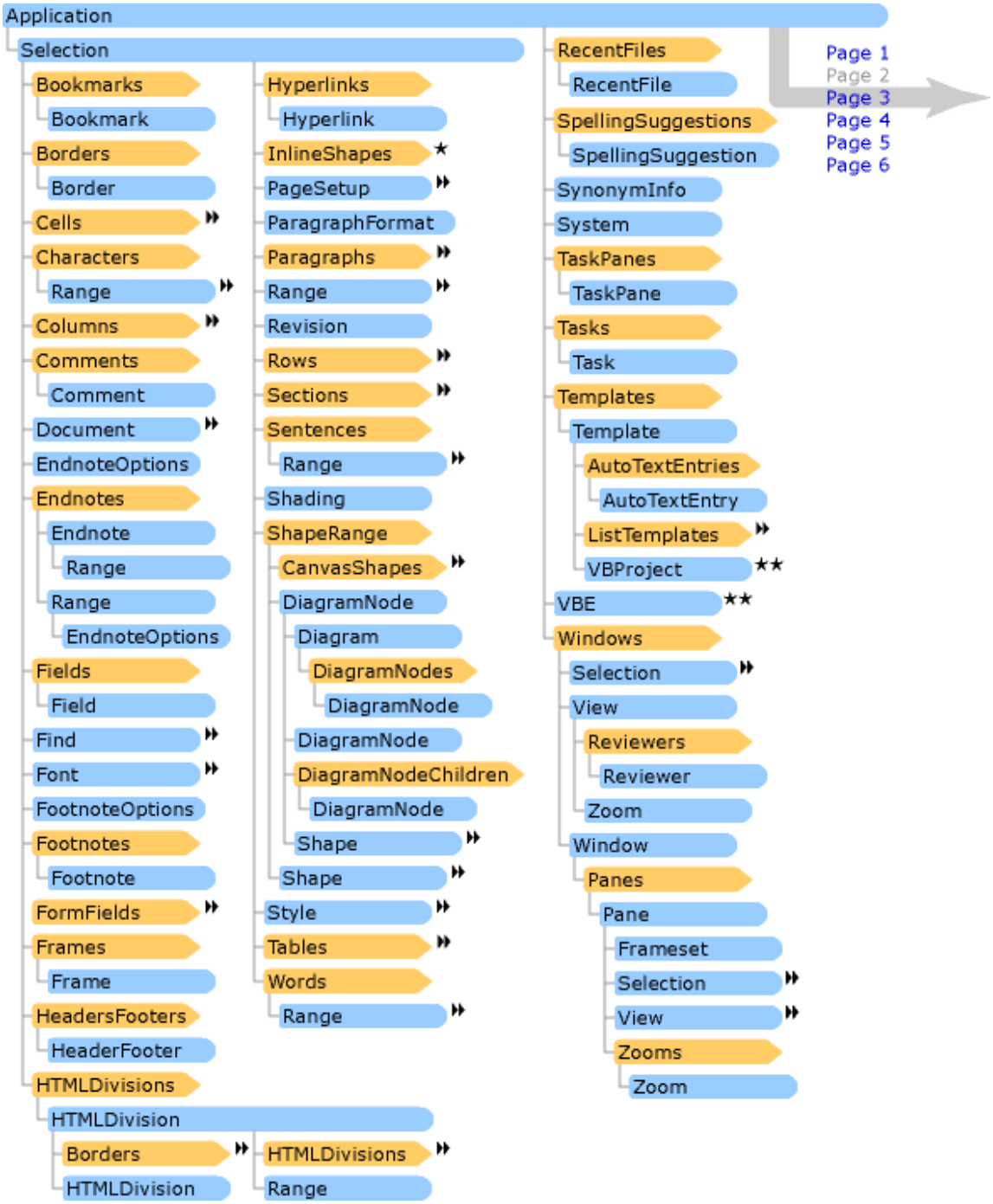
Obr. 2–5 Okno Output s výpisem synonym

2.4 Grafické znázornění objektového modelu Microsoft Word

Toto znázornění pochází z internetových stránek společnosti Microsoft (6) a uvádím je zde, protože dovoluje učinit si názornou představu o hierarchii objektů aplikace Microsoft Word. Vzhledem k tomu, že zobrazuje verzi XP, nemusí přesně odpovídat novějším verzím a některé objekty mohou chybět, ale stejně názorný novější se mi nalézt nepodařilo. Symbol **>>** v obrázku naznačuje, že objekt je více rozveden na jiném místě

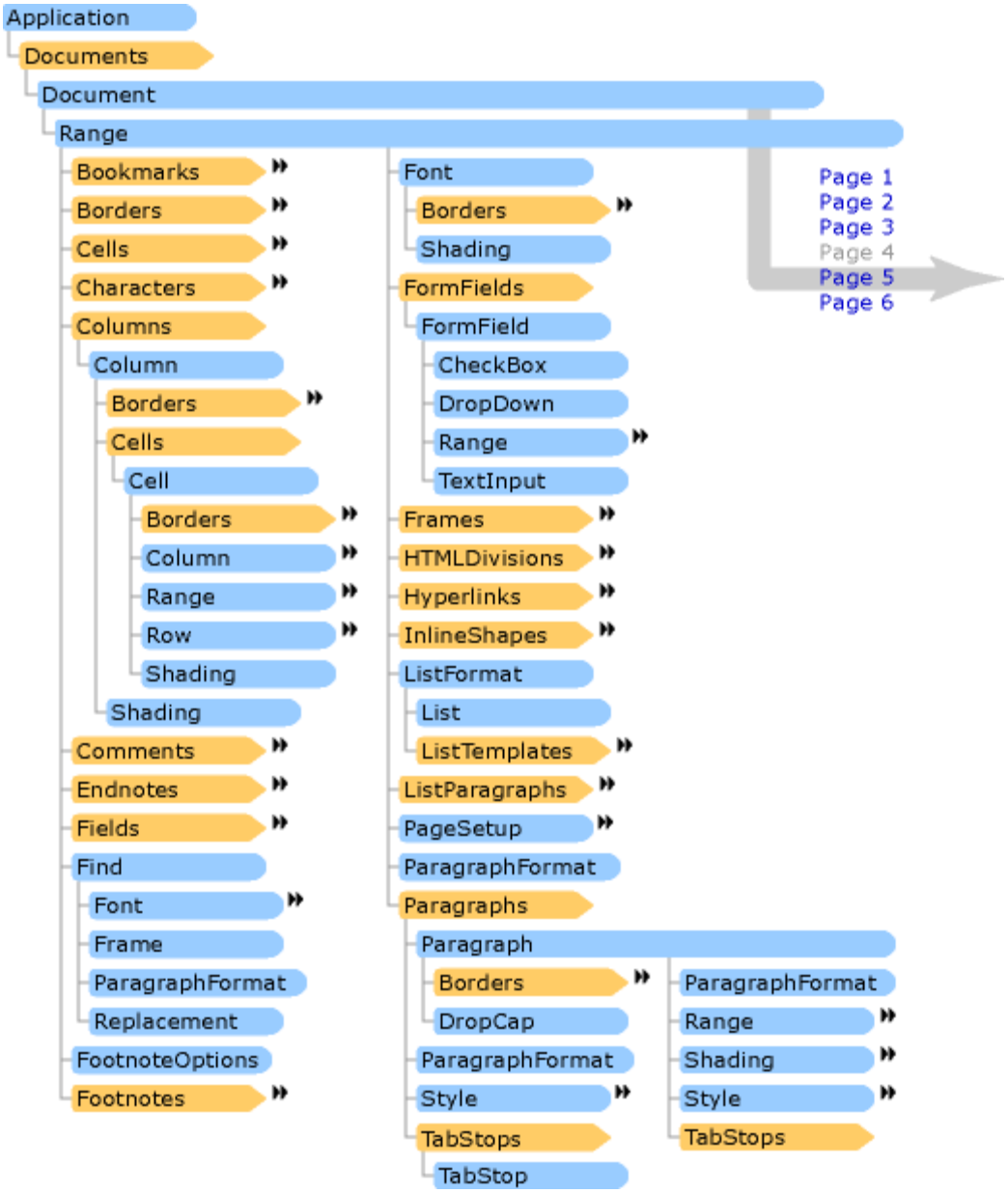
Grafické znázornění objektového modelu Microsoft Word strana 1

Grafické znázornění objektového modelu Microsoft Word strana 2

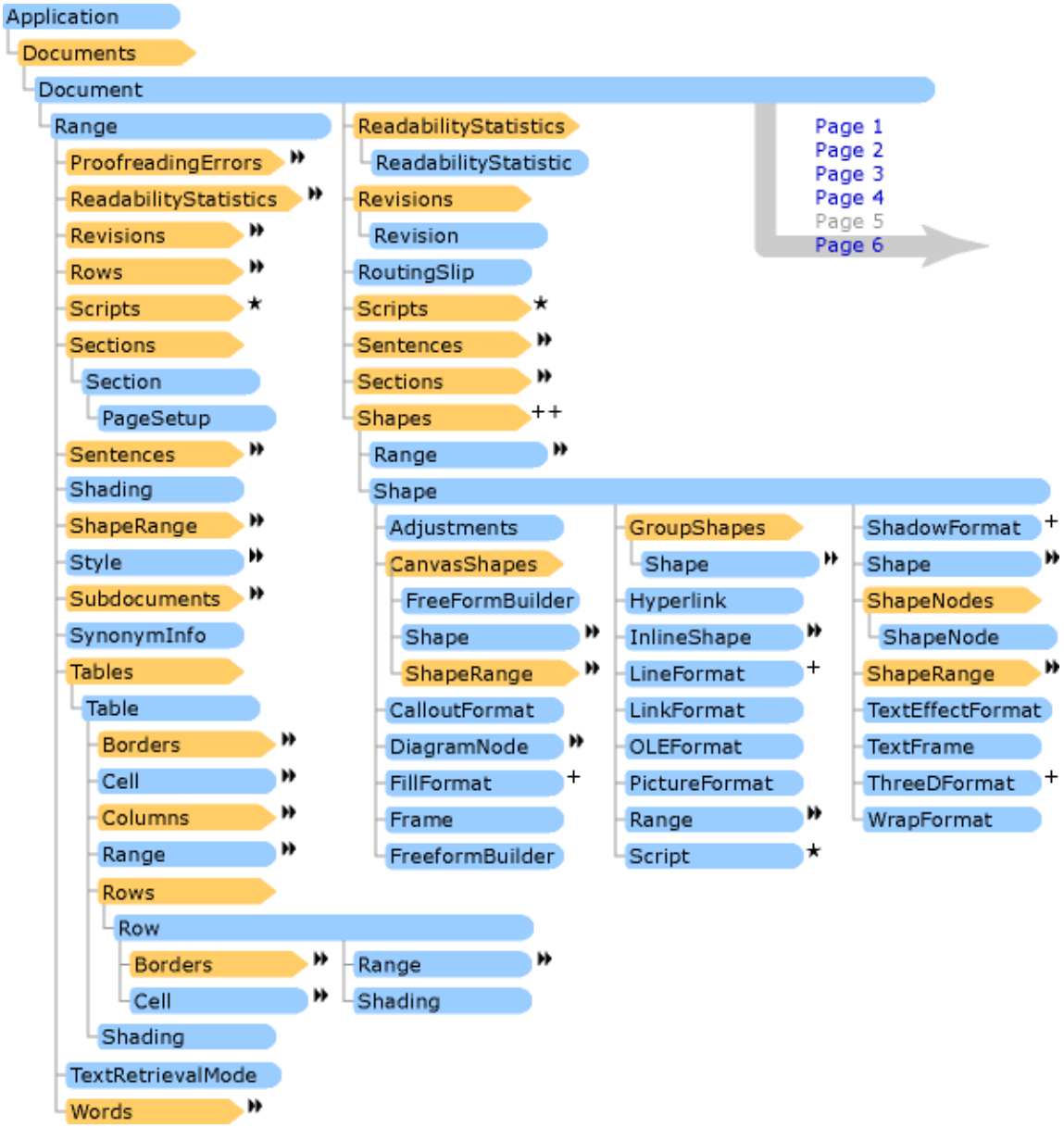


Grafické znázornění objektového modelu Microsoft Word strana 3

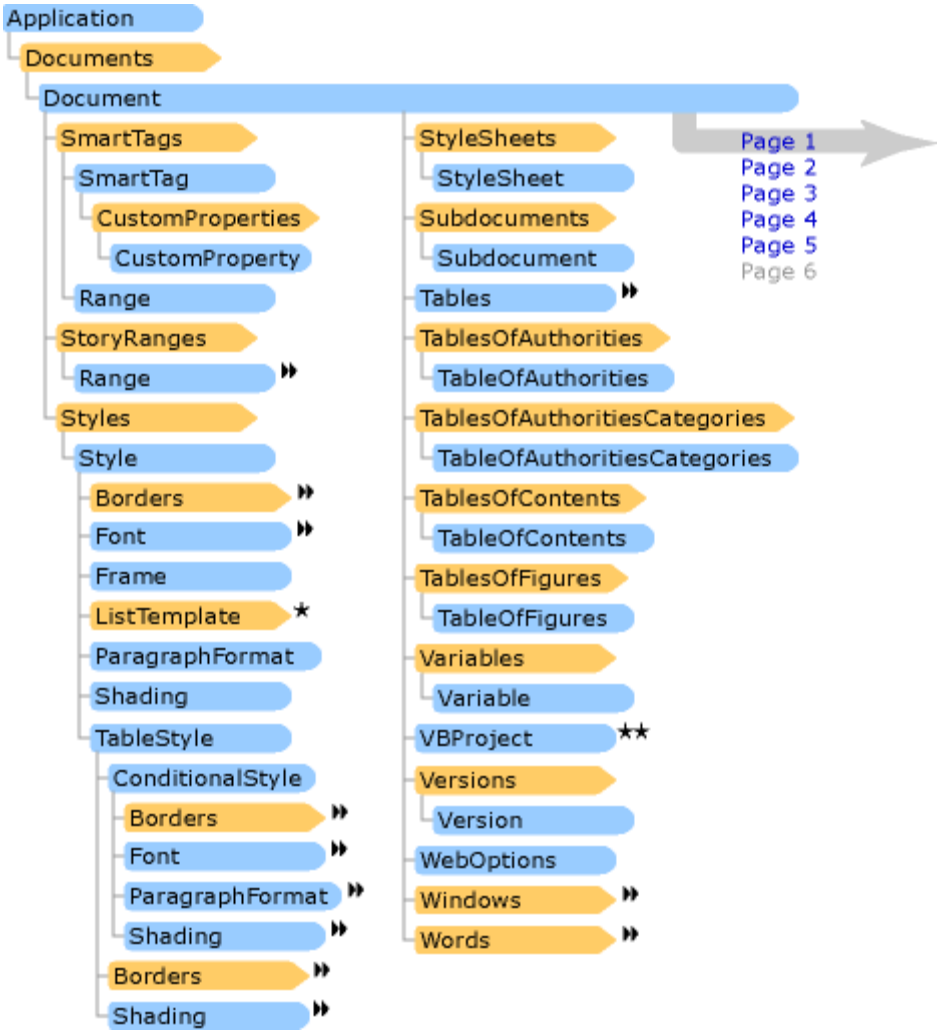
Grafické znázornění objektového modelu Microsoft Word strana 4



Grafické znázornění objektového modelu Microsoft Word strana 5



Grafické znázornění objektového modelu Microsoft Word strana 6



3 Práce s tabulkovým kalkulátorem Microsoft Excel

Pomocí aplikace Microsoft Excel můžeme vytvořit, případně otevřít sešit této aplikace, anebo využít obsažených funkcí pro výpočty, které použijeme v naší aplikaci.

Informace o objektech, jejich metodách a vlastnostech uvedených v této kapitole, byly čerpány z online dokumentace pro `Microsoft.Office.Interop.Excel` na internetových stránkách firmy Microsoft (7)

3.1 Spuštění a ukončení aplikace Microsoft Excel

Podobně jako u aplikace Word dojde ke spuštění aplikace Excel vytvořením nové instance objektu `Excel.Application`. Nový proces aplikace Excel je spuštěn na pozadí. Pro ukončení aplikace Excel použijeme metodu `Quit`, v jejíchž parametrech můžeme určit, zda uložíme změny v dokumentu. Potom je třeba přiřadit nepoužívané proměnné hodnotu `Nothing` kvůli jejímu uvolnění Garbage collectorem z paměti. Na rozdíl od aplikace Word je to nutné, jinak k ukončení procesu dojde až po zavření aplikace, ze které jsme Excel spustili. To platí pro OS W7 a verzi Office 2010; v jiných verzích se toto chování může lišit, a je třeba ho otestovat.

Poznámka: Ukončení procesu EXCEL.EXE může chvíli trvat, protože Garbage Collector provádí uvolňování nepoužívaných objektů z paměti podle vlastní logiky. Okamžité uvolnění paměti je možné vyvolat metodou `GC.Collect`. – viz výpis V 3-1. V tomto výpisu je uveden i import jmenného prostoru Microsoft.Office.Interop.Excel, který v dalších výpisech uveden není, ale je v nich předpokládán.

V 3-1 Spuštění a ukončení aplikace Excel

```
Imports Excel = Microsoft.Office.Interop.Excel
Imports Microsoft.Office.Interop.Excel

Sub Excel_RunQuit()

    'Spustíme Excel
    Dim ExcelApp As New Excel.Application
    ExcelApp.Visible = True

    'Ukončíme Excel
    ExcelApp.Quit()
    ExcelApp = Nothing
    GC.Collect()
End Sub
```

3.2 Práce se sešitem a listy

Chceme-li pracovat s listem tabulkového kalkulátoru Excel, musíme vytvořit nový sešit s nejméně jedním listem. Každý sešit je členem kolekce `Workbooks`. K jednotlivým sešitům této kolekce můžeme přistupovat prostřednictvím jména (vlastnost `Name`) nebo prostřednictvím indexu. Pro vytvoření nového sešitu použijeme metodu `Add` této kolekce. Nyní máme sešit, který obsahuje několik listů. Počet listů, které obsahuje nový sešit, je standardně nastaven na tři. To můžeme změnit úpravou hodnoty vlastnosti `SheetsInNewWorkbook`. Tato změna však bude platit i pro nové sešity vytvořené uživatelem. Je proto vhodné ji nastavit na původní hodnotu. Viz výpis V 3-2, kde hodnotu vlastnosti `SheetsInNewWorkbook` uschováme v pomocné proměnné, kterou změníme na jedničku, vytvoříme sešit s jedním listem a znovu nastavíme původní hodnotu.

V 3-2 Otevření a zavření sešitu

```
Sub SesitOpenClose()
    Dim ExcelApp As New Excel.Application 'spustíme Excel
    ExcelApp.Visible = True

    Dim OldPocetListu As Integer = ExcelApp.SheetsInNewWorkbook
    ExcelApp.SheetsInNewWorkbook = 1 'počet listů v novém sešitě
    Dim Sesit As Excel.Workbook
    Sesit = ExcelApp.Workbooks.Add() 'vložíme nový sešit
    ExcelApp.SheetsInNewWorkbook = OldPocetListu

    ExcelApp.Quit() 'ukončíme Excel
    ExcelApp = Nothing
End Sub
```

Jednotlivý list je představován objektem `Worksheet` a patří do kolekce `Worksheets`. Tuto kolekci vrací vlastnost `Worksheets` objektu `Workbook`. Vlastnost `Worksheets` má i objekt `Application`. Zde se jedná kolekci listů právě aktivního sešitu. Listy můžeme samozřejmě i přidávat pomocí metody `Add`. Objekt `Workbook` má ještě vlastnost `Sheets`, vracející kolekci `Sheets`. Tato kolekce obsahuje kromě listů (objekt `Worksheet`) i grafy umístěné na samostatném listu (objekt `Chart`). I kolekce `Sheets` má metodu `Add`. O tom, jestli bude přidán list nebo graf, rozhoduje hodnota. Pro list má parametr `Type` hodnotu `XlSheetType.xlWorksheet` a pro graf hodnotu `XlSheetType.xlChart`. Parametr `Type:= XlSheetType.xlChart` můžeme zapsat i do metody `Add` objektu `Worksheet`, ale dojde k chybě za běhu programu. Existují ještě další typy listů, např. list maker, ale ty existují pouze z důvodu zpět-

né kompatibility souborů ze starších verzí tabulkového kalkulátoru Excel a jejich další používání není ze strany společnosti Microsoft doporučováno. Proto se o nich nebudu nadále zmiňovat.

3.2.1 Práce s buňkami a oblastmi buněk

Pro práci s buňkami, což je hlavní funkce tabulkového kalkulátoru Excel, používáme objekt `Range`. Tento objekt může obsahovat jednu buňku, nějakou pravoúhlou oblast sousedních buněk nebo i více takovýchto oblastí. Objekt `Range` získáme například pomocí vlastnosti `Range`, která má jeden povinný a jeden volitelný argument. Ty jsou v dokumentaci Microsoftu označeny jako `Cell1` a `Cell2`, což není úplně přesné, protože hodnotou každého argumentu může být jak jedna buňka, tak i celá oblast buněk. Objekt `Range` vrácený touto vlastností pak zahrnuje oblast, jejíž levý horní roh je na místě buňky v argumentu nebo argumentech, která je nejvíce vlevo nahoře, a pravý dolní na místě buňky, která je nejvíce vpravo dole. Bez ohledu na pořadí buněk v argumentech – viz výpis V 3-3, kde je pro označení buněk použit standardní zápis aplikace Excel (čísla označují řádky a písmena sloupce).

Další vlastností, která vrací objekt `Range`, je vlastnost `Cells` s argumenty řádek a sloupec. Objekt `Range` vrácený touto vlastností obsahuje jednu buňku, lze ho tedy použít jako argument vlastnosti `Range` viz výpis V 3-3. Tento způsob je nepochybně výhodnější, než skládat řetězec pro standardní zápis, protože označení pro sloupce s vyšším pořadovým číslem než 26 je tvořeno složením až tří písmen. Je třeba však mít na paměti, že v tomto případě musíme použít oba argumenty. Pokud by měla být obsahem objektu `Range` jedna buňka, budou oba shodné. Viz oblast `R1` ve výpisu V 3-3.

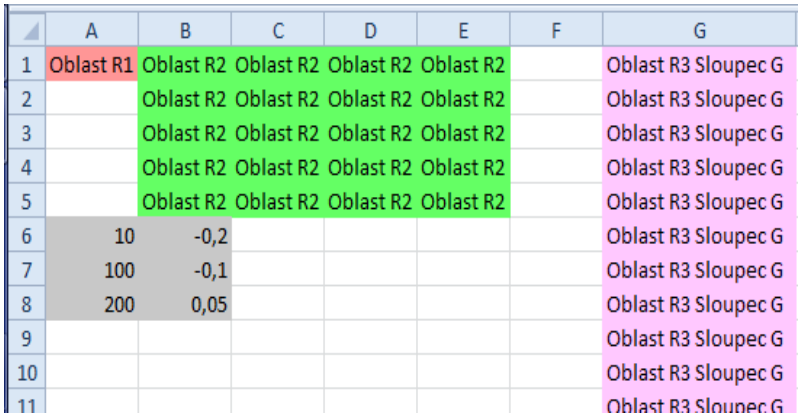
Poznámka: Argumentem vlastnosti `Range` musí být objekt `Range` [například `List.Cells(1, 1)`], nebo řetězec pro standardní zápis. Způsob uvedený v (4 str. 479) „range1 = Worksheet.Range(4, 8)“ v Microsoft Excel 2010 nefunguje.

Pro čtení, vložení a změnu hodnoty objektu `Range` se používá vlastnost `Value`, která je typu `Object`. Přiřadíme-li nějaké proměnné typu `Object` hodnotu z vlastnosti `Value`, bude obsahovat číslo, řetězec, nebo odpovídající pole, a to podle rozsahu objektu `Range`, který jsme použili. Naopak vložíme-li do této vlastnosti objekt typu `Object` obsahující jednu hodnotu, budou ji obsahovat všechny buňky, které objekt `Range` zahrnuje. Ve výpisu V 3-3 jsou uvedeny příklady různých způsobů vložení oblastí do objektů `Range`. Proměnné stejného jména obsahují stále stejnou oblast, jak se

můžeme přesvědčit, pokud budeme proceduru krokovat. Jednotlivé oblasti budou pouze měnit barvu. Do oblasti R4, která je menší než vkládané pole, bude vloženo pouze tolik prvků, kolik odpovídá velikosti oblasti. Výsledek je zobrazen na obrázku Obr. 3–1.

V 3-3 Nastavení oblasti pomocí standardního zápisu Excel

```
Sub NastaveniOblasti()

    'Spustíme Excel a vložíme nový sešit
    Dim ExcelApp As New Excel.Application
    Dim Sesit As Excel.Workbook
    Dim List As Excel.Worksheet
    ExcelApp.Visible = True
    Sesit = ExcelApp.Workbooks.Add()
    List = Sesit.Worksheets(1)
    Dim R1, R2, R3, R4 As Range
With List

    'Výběr jedné buňky
       R1 =.Range("A1")                      :R1.Interior.Color = RGB(0, 255, 0)
       R1 =.Range(.Cells(1, 1),.Cells(1, 1))  :R1.Interior.Color = RGB(255, 150, 150)
       R1.Value = "Oblast R1"

    'Výběr oblasti buněk – všechny zápisy označují stejnou oblast
       R2 =.Range("B1:E5")                    :R2.Interior.Color = RGB(0, 255, 0)
       R2 =.Range("B1", "E5")                 :R2.Interior.Color = RGB(255, 0, 0)
       R2 =.Range("E5", "B1")                 :R2.Interior.Color = RGB(0, 255, 0)
       R2 =.Range("B1:C2", "C3:E5")           :R2.Interior.Color = RGB(255, 0, 0)
       R2 =.Range(.Cells(1, 2),.Cells(5, 5))  :R2.Interior.Color = RGB(100, 255, 100)
       R2.Value = "Oblast R2"

    'Výběr sloupce
       R3 =.Range("G:G")                      :R3.Interior.Color = RGB(150, 100, 255)
       R3 =.Columns(7)                        :R3.Interior.Color = RGB(255, 200, 255)
       R3.Value = "Oblast R3 Sloupec G"

    'Vložení hodnot pole do oblasti – vloží se jen část odpovídající rozsahu oblasti
       R4 =.Range("A6", "B8")                 :R4.Interior.Color = RGB(200, 200, 200)
       Dim pole(,) As Double = { {10, -0.2, 0.03},
                                 {100, -0.1, 0.05},
                                 {200, 0.05, 0.07},
                                 {500, 0.1, 0.08},
                                 {700, 0.15, 0.08},
                                 {1000, 0.3, 0.09}}
       R4.Value = pole
End With

    'Ukončíme Excel
    ExcelApp.Quit()
    ExcelApp = Nothing
End Sub
```



	A	B	C	D	E	F	G
1	Oblast R1	Oblast R2	Oblast R2	Oblast R2	Oblast R2		Oblast R3 Sloupec G
2		Oblast R2	Oblast R2	Oblast R2	Oblast R2		Oblast R3 Sloupec G
3		Oblast R2	Oblast R2	Oblast R2	Oblast R2		Oblast R3 Sloupec G
4		Oblast R2	Oblast R2	Oblast R2	Oblast R2		Oblast R3 Sloupec G
5		Oblast R2	Oblast R2	Oblast R2	Oblast R2		Oblast R3 Sloupec G
6	10	-0,2					Oblast R3 Sloupec G
7	100	-0,1					Oblast R3 Sloupec G
8	200	0,05					Oblast R3 Sloupec G
9							Oblast R3 Sloupec G
10							Oblast R3 Sloupec G
11							Oblast R3 Sloupec G

Obr. 3–1 Příklady oblastí

3.2.1.1 Vkládání vzorců

Potřebujeme-li vložit do buňky nějaký vzorec, stačí do ní vložit řetězec začínající znakem „=“ podobně jako při ručním vkládání vzorců přímo v pracovním listu tabulkového kalkulátoru Excel. Odkazy na buňky sestavíme ve formátu R#C#, kde „R“ – znamená řádek, „C“ – sloupec a znak „#“ – číslo řádku nebo sloupce. Pokud číslo neuvedeme, odkazujeme na aktuální řádek nebo sloupec. Můžeme použít i relativní odkazy tak, že za znaky „C“ nebo „R“ napíšeme číslo v hranaté závorce. Vkládáme-li funkce, musíme použít jejich anglické názvy. U desetinných čísel pak musíme použít jako oddělovač desetinných míst tečku. V tabulce T 1 jsou uvedeny příklady zápisu textových řetězců vkládaných do buňky, a výsledek, který je pak zobrazen. Ve výpisu V 3-4 je postup zápisu vzorců demonstrován na příkladu výpočtu směrodatné odchylky dvojím způsobem, jednou pomocí pomocných vzorců, podruhé pomocí funkce. Zápis vkládaných řetězců musí vypadat přesně tak jak je zde uveden, pozor je nutno dát zejména na nadbytečné mezery. V případě chybného zápisu Excel funkci nerozezná a dojde k chybě za běhu programu.

T 1 Příklady řetězců s odkazy, vzorci a funkcemi

Řetězec vložený do buňky "B2"	V buňce "B2" (R2C2) je zobrazeno
"=R1C1"	hodnota z buňky "A1"
"=RC1"	hodnota z buňky "A2"
"=RC[-1]"	hodnota z buňky "A2"
"=R1C1* R1C2"	
nebo **"=R1C1* R1C"**	součin buněk "A1"a "B1"
nebo **"=R[-1]C[-1]* R[-1]C[0]"**	
"=AVERAGE(R1C1:R5C1)"	průměr buněk "A1" až "A5"
"=(SUMSQ(R1C1:R2C1))^0.5"	odmocnina[$(A1)^2 + (A2)^2$]

V 3-4 Vkládání vzorců do buněk

```
Private Sub ZapisVypOpak()

    'Spustíme Excel a vložíme nový sešit
    Dim ExcelApp As New Excel.Application
    Dim Sesit As Excel.Workbook
    Dim List As Excel.Worksheet
    ExcelApp.Visible = True
    Sesit = ExcelApp.Workbooks.Add()
    List = Sesit.Worksheets(1)
    Dim R As Integer = 1
    Dim S As Integer = 1

    'Naplníme pole
    Dim pole() As Double = {5.23, 5.21, 5.22, 5.22, 5.22, 5.21, 5.2, 5.21, 5.23, 5.2}
    Dim N As Integer = pole.GetLength(0) 'počet hodnot

    'Napíšeme popisy sloupců
    List.Cells(R, S) = "Výpočet směrodatné odchylky"
    R += 1
    List.Cells(R, S) = "Měření č."
    List.Cells(R, S) = "Hodnota"

    'Pro každou položku pole vložíme její číslo, hodnotu a vzorec pro pomocné výpočty,
    kde využijeme relativní i absolutní adresy buněk
    For i = 1 To N
        List.Cells(R + i, S) = i.ToString
        List.Cells(R + i, S + 1) = pole(i - 1)
        List.Cells(R + i, S + 2) = "=(RC[-1]-R" & R + N + 1 & "C" & S + 1 & ")^2"
    Next
    R += 11
    List.Cells(R, S) = "Průměr"
```

```
    'Vložíme vzorce pro výpočet průměru, směrodatné odchylky a sumy a komentáře s popisem
    List.Cells(R, S + 1) = "=AVERAGE(R[-10]C:R[-1]C)"
    R += 1
    List.Cells(R, S) = "s (směrodatná odchylka výběr)"
    List.Cells(R, S + 1) = "=STDEV.S(R[-11]C:R[-2]C)"
    List.Cells(R, S + 1).AddComment()
    List.Cells(R, S + 1).Comment.Text("Výpočet pomocí funkce Excelu")

    List.Cells(R, S + 2) = "=((SUM(R[-11]C:R[-2]C))/(10-1))^0.5"
    List.Cells(R, S + 2).AddComment()
    List.Cells(R, S + 2).Comment.Text("Výpočet pomocí vlastního vzorce")

        'Ukončíme Excel
    ExcelApp.Quit()
    ExcelApp = Nothing
End Sub
```

3.2.1.2 Formátování buněk

Pro formátování ohraničení buněk nebo oblastí buněk používáme kolekci `Borders` a její členské objekty `Border`. Pro práci s výplní buněk pak objekt `Interior` a pro formátování písma objekt `Font`. Tyto objekty vracejí stejnojmenné vlastnosti objektu `Range`. Další užitečnou vlastností je vlastnost `NumberFormat` pro nastavení formátu čísla. Jejím parametrem je textový řetězec ve stejném tvaru v jakém ho známe z ručního nastavování formátu čísla. Použití těchto vlastností je díky technologii IntelliSense intuitivní. Jednotlivé znaky v buňce můžeme formátovat prostřednictvím objektu `Characters` a vytvořit tak například indexy. Příklad formátování je uveden ve výpisu V 3-5, kde vložíme do jedné buňky text s indexem, do druhé číslo se dvěma desetinnými místy a jednotkou, a obě buňky orámujeme a obarvíme.

V 3-5 Formátování buňky

```
Sub FormatBunky()

    'Spustíme Excel a vytvoříme sešit
    Dim ExcelApp As New Excel.Application
    ExcelApp.Visible = True
    Dim pSesit As Excel.Workbook
    pSesit = ExcelApp.Workbooks.Add()
    Dim pList As Excel.Worksheet
    pList = pSesit.Worksheets(1)
    Dim rOblast As Range

    'Vložíme text a druhý a třetí znak zformátujeme jako dolní index
    pList.Cells(1, 1) = "tEt"
    pList.Cells(1, 1).characters(2, 2).font.subscript = True
```

```
    'Vložíme text a nastavíme formát čísla
    Dim formstrSt_C As String = "0.## °C"
    pList.Cells(1, 2) = 100.323
    pList.Cells(1, 2).NumberFormat = formstrSt_C
    rOblast = pList.Range(pList.Cells(1, 1), pList.Cells(1, 2))

    'Orámujeme celou oblast plnou střední čárou a obarvíme ji na zeleno
    For i = XlBordersIndex.xlEdgeLeft To XlBordersIndex.xlEdgeRight
        rOblast.Borders(i).LineStyle = XlLineStyle.xlContinuous
        rOblast.Borders(i).Weight = XlBorderWeight.xlMedium
    Next
    rOblast.Interior.Color = XlRgbColor.rgbLime

    'Ukončíme Excel
    ExcelApp.Quit()
    ExcelApp = Nothing
End Sub
```

3.2.2 Práce s grafy

V aplikaci Excel můžeme mít graf na dvou místech, a to jako samostatný list a jako objekt na klasickém listu. Práce s vlastním grafem je v obou případech stejná, rozdíl je v jejich vkládání do sešitu. Vlastní graf reprezentuje objekt `Chart`; v případě, že chceme mít graf na samostatném listu, použijeme metodu `Add` kolekce `Sheets` s parametrem `Type:=XlSheetType.xlChart`. Připomínám, že kolekce `Sheets` obsahuje jak pracovní listy, tak listy grafů. Graf na pracovním listu je umístěn v kontejneru `ChartObject` , který umožnuje například určení rozměrů a pozice grafu, jeho kopírování a mazání, atd. Všechny objekty `ChartObject` jsou členy kolekce `ChartObjects` . Metoda `Add`, kterou přidáváme graf do listu, má čtyři parametry udávající rozměry a pozici grafu. Ať už jsme vložili graf kamkoli, je to stejný objekt `Chart` a podle toho s ním můžeme pracovat. Ve výpisu V 3-6, je to demonstrováno tak, že jsou vytvořeny dva grafy, jeden na samostatném listu, druhý v klasickém pracovním listu a oba jsou upraveny stejnou procedurou. Výsledkem je sešit, který obsahuje dva identické grafy.

Ve výpisu V 3-6, je uveden příklad vytvoření grafu naměřených odchylek v několika bodech. První procedura, která je v něm uvedena s názvem `VlozitGrafy`, spustí Excel, vytvoří sešit, naplní oblast na listu1 daty z pole ve svém parametru, vytvoří dva grafy a zavolá druhou proceduru `UpravyGraf`. První graf, představovaný proměnnou `ListGrafu` typu `Chart`, je vložen do sešitu jako samostatný list grafu, druhý, představovaný proměnnou `ObjektGraf` typu `ChartObject`, je vložen na

list1. Procedura UpravyGraf potom provede formátování a úpravy grafu, který je jí předán v parametru.

Ve druhé proceduře ve výpisu V 3-6 jsou na grafu provedeny tyto úpravy:

– nastavení typu grafu na bodový,

– vložení dat z pole,

– vložení dalších dvou datových řad pro zobrazení dovolených tolerancí,

– vložení názvu grafu,

– popsání obou os,

– upravení legendy.

Pro každý typ vstupních dat se hodí jiný typ grafu. Ten nastavíme pomocí vlastnosti ChartType do které můžeme přiřazovat hodnoty z výčtu XlChartType, v našem případě pro bodový graf s rovnými spojnicemi je to hodnota xlXYScatterLines.

Při plnění grafu daty můžeme buď vložit odkaz na oblast na některém listu, nebo vložit přímo hodnoty. Odkaz zachovává propojení a při změně dat v odkazované oblasti je graf aktualizován. Ve výpisu V 3-6 je pro vložení hodnot ze vstupního pole použita metoda SetSourceData. Pomocí této metody lze vložit pouze odkaz – výhodou je, že tak můžeme vytvořit několik datových řad najednou.

Pokud chceme mít nad vkládáním dat větší kontrolu, vkládáme přímo jednotlivé datové řady. Ty jsou představovány objekty Series, sdruženými do kolekce SeriesCollection. K přidání další řady můžeme použít metodu Add, v jejíchž parametrech určíme, z jaké oblasti data vkládáme – zda jsou uspořádána do řad nebo sloupců, jestli první buňky sloupců nebo řad obsahují název datové řady, atd. V této metodě je parametr Source typu Object a lze mu přiřadit objekt Range s oblastí na listu, nebo pole hodnot. Další možností je metoda NewSeries, která vrací objekt Series. Data, která bude nová datová řada obsahovat – její název, formát zobrazení a další její parametry – musíme potom nastavit ve vlastnostech objektu Series. Pro nastavení hodnot osy **Y** slouží vlastnost Values a pro nastavení hodnot osy **X** vlastnost XValues. Ve výpisu V 3-6 jsme vložili další dvě datové řady a jako jejich hodnoty nastavili kladné a záporné hodnoty tolerance. Formát vykreslované čáry měníme pomocí vlastnosti Formát.Line[8], vracející objekt LineFormát, který má vlastnosti

[8] Posloupnost je ve skutečnosti následující: vlastnost Formát ⟹ objekt ChartFormat ⟹ vlastnost Line ⟹ objekt LineFormát.

umožňující upravit čáru podle našich představ. Název datové řady, který se zobrazuje v legendě, vkládáme do vlastnosti `Name`. Protože pro popis čar zobrazujících tolerance nám stačí jedna položka v legendě, je ve výpisu ta přebytečná odstraněna. S legendou pracujeme přes objekt `Legend`, který vrací stejnojmenná vlastnost objektu `Chart`. Jednotlivé položky legendy sdružené v kolekci `LegendEntrys` jsou objekty `LegendEntry`. Ve výpisu V 3-6 používáme metodu `Delete` objektu `LegendEntry` k jeho odstranění z legendy.

Poznámka: Posloupnost je ve skutečnosti následující: vlastnost Formát ⟹ objekt `ChartFormat` ⟹ vlastnost `Line` ⟹ objekt `LineFormát`.

Každý graf by měl mít název. K nastavování obsahu a vzhledu názvu grafu slouží objekt `ChartTitle`, k jehož získání slouží vlastnost `ChartTitle` objektu `Chart`. Dříve než tuto vlastnost použijeme, musíme vlastnosti `HastTitle` přiřadit hodnotu `True`. To lze udělat i metodu `SetElement`, v níž volbou konkrétní hodnoty výčtu `MsoChartElementType` můžeme určit i některé vlastnosti názvu. Tato metoda existuje až od verze Excel 2007 a ve výpisu V 3-6 je použita pro vložení popisů os.

Jestliže v grafu název být může, popisy osy v něm být musí. S osami v grafu pracujeme prostřednictvím objektu `Axis`, který nám vrací metoda `Axes` objektu `Chart`. V parametrech této metody určíme, o kterou osu se jedná, případně zda je to osa hlavní nebo vedlejší. Ve výpisu V 3-6 můžeme vidět vložení popisů os X a Y metodou `SetElement`, v jejímž parametru jsme přímo určili, že osa Y bude mít popis vodorovně.

V 3-6 Vytvoření grafu

```
    'Vytvoření pole s daty vkládanými do grafu
Dim pole(,) As Double = {{10, -0.15},
                         {100, -0.1},
                         {200, 0.05},
                         {500, 0.08},
                         {700, 0.15},
                         {1000, 0.3}}

Sub VloziGrafy(ByVal pole(,) As Double)

    'Spustíme Excel a vložíme nový sešit
    Dim ExcelApp As New Excel.Application
    ExcelApp.Visible = True
    Dim pSesit As Excel.Workbook
    pSesit = ExcelApp.Workbooks.Add()
    Dim pList As Excel.Worksheet
    Dim ListGrafu As Chart
```

```
    Dim ObjektGraf As ChartObject
    Dim rOblast As Excel.Range

    'Vyplníme tabulku s daty
    pList = pSesit.Worksheets(1)
    pList.Cells(1, 1) = "Hodnoty oddchylky pro jednotlivé zátěže "
    pList.Cells(2, 1) = "Jm. hmotnost (g)"
    pList.Cells(2, 2) = "Odchylka(mg)"
    Dim pocM As Integer = pole.GetUpperBound(0) + 1 'počet měření
    For i = 1 To pocM
       For j = 1 To 2
          pList.Cells(i + 2, j) = pole(i - 1, j - 1)
       Next
    Next

    'Do proměnné rOblast vložíme oblast s daty pro graf
    rOblast = pList.Range(pList.Cells(3, 1), pList.Cells(3 + pocM, 3))

       'Vytvoříme jeden graf na samostatném a jeden v pracovním listu
    ListGrafu = pSesit.Sheets.Add(Type:=XlSheetType.xlChart)
    ObjektGraf = pList.ChartObjects.add(20, 150, 500, 400)

    'Oba grafy upravíme a vložíme do nich data
    UpravyGraf(ListGrafu, rOblast, pole)
    UpravyGraf(ObjektGraf.Chart, rOblast, pole)

    'Ukončíme Excel
    ExcelApp.Quit() 'ukončíme Excel
    ExcelApp = Nothing
    End Sub

    'Procedura, která vloží data do grafu a zformátuje ho
Sub UpravyGraf(ByRef Graf As Chart, ByVal rOblast As Excel.Range, ByVal pole(,) As
Double)

    'Zvolíme typ grafu a vložíme data
    Graf.ChartType = XlChartType.xlXYScatterLines
    Graf.SetSourceData(rOblast)

    'Vložíme název grafu
    Graf.HasTitle = True
    Graf.ChartTitle.Caption = "Hodnoty oddchylky pro jednotlivé zátěže "

    'Popíšeme osy a datovou řadu
    Graf.SetElement(Microsoft.Office.Core.MsoChartElementType.msoElementPrimaryCateg
                    oryAxisTitleAdjacentToAxis)
    Graf.Axes(XlAxisType.xlCategory).AxisTitle.Text = "Jmenovité hodnoty zatížení
    (g)"
    Graf.SetElement(Microsoft.Office.Core.MsoChartElementType.msoElementPrimaryValue
                    AxisTitleHorizontal)
    Graf.Axes(XlAxisType.xlValue).AxisTitle.Text = "Odchylky (mg)"
    Dim Rada As Series = Graf.SeriesCollection(1)
```

```
    Rada.Name = "Odchylky"

    'Vložíme řady pro zobrazení tolerancí
    Dim Xhodnoty(1), Hodnoty(1) As Double
    Dim tolerance As Double = 0.1
    For i = -1 To 1 Step 2
        Xhodnoty(0) = pole(0, 0)
        Xhodnoty(1) = pole(pole.GetUpperBound(0), 0)
        Hodnoty(0) = tolerance * i
        Hodnoty(1) = tolerance * i

    'Naplníme pole pro x-ové a y-ové hodnoty řad
        Rada = Graf.SeriesCollection.NewSeries()
        With Rada
            .XValues = Xhodnoty
            .Values = Hodnoty
            .MarkerStyle = XlMarkerStyle.xlMarkerStyleNone
        End With
        With Rada.Format.Line
            .Visible = True
            .ForeColor.RGB = RGB(255, 0, 0)
            .BackColor.RGB = RGB(255, 0, 0)
            .Weight = 2
        End With
    Next

    'Pojmenujeme poslední řadu a smažeme přebytečné položky legendy
    Rada.Name = "Tolerance"
    For i = 2 To Graf.Legend.LegendEntries.count - 1
        Graf.Legend.LegendEntries(2).delete()
    Next
End Sub
```

3.3 Využití funkcí Microsoft Excel pro výpočty

Pro použití funkcí tabulkového kalkulátoru Excel stačí spustit aplikaci vytvořením instance objektu typu `Excel.Application`, a poté už můžeme prostřednictvím vlastnosti `WorksheetFunction` a metod objektu `WorksheetFunction` využívat požadovanou funkci. Viz výpis V 3-7, kde je uveden příklad výpočtu směrodatné odchylky. Názvy metod odpovídají anglickým názvům funkcí v aplikaci Excel a jejich seznam si můžeme zobrazit v prohlížeči objektů nebo v dokumentaci (7). Jako oddělovač desetinných míst musíme použít tečku. U metod jejichž argumenty jsou typu `Object` můžeme jako hodnotu použít pole. Jestliže má některý argument typu `Object` hodnotu `Nothing`, je s ním počítáno, jako by měl hodnotu 0. Proměnné `d1` a `d2` ve výpisu V 3-8 budou tedy shodné.

V 3-7 Výpočet směrodatné odchylky pomocí funkce stDev

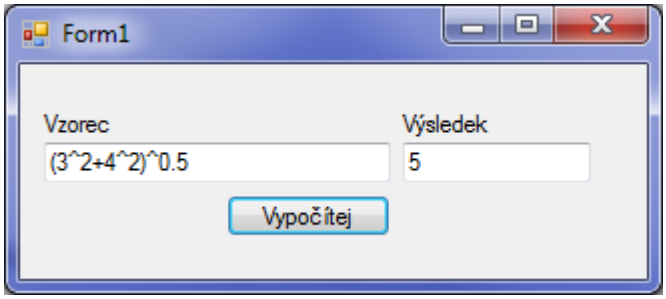
```
Sub SOdchylkaExcel()

    'Spustíme Excel a vytvoříme pole hodnot
    Dim ExcApp As New Excel.Application
    Dim poleHodnot() As Double = {1.1, 1.2, 1.1, 1.3, 1.2, 1.1, 1.1}
    Dim d As Double

    'použijeme funkci StDev
    d = ExcApp.WorksheetFunction.StDev(poleHodnot)

    ExcApp.Quit()
    ExcApp = Nothing
End Sub
```

V 3-8 Výpočet průměru pomocí funkce Average

```
Sub PrumerExcel()

    'Spustíme Excel a vytvoříme pole hodnot
    Dim ExcApp As New Excel.Application
    Dim poleHodnot() As Double = {2, 1, 0}
    Dim d1, d2 As Double

    d1 = ExcApp.WorksheetFunction.Average(poleHodnot)     'd1 = 1
    d2 = ExcApp.WorksheetFunction.Average(2, 1, Nothing)  'd2 = 1

    ExcApp.Quit()
    ExcApp = Nothing
End Sub
```

Dalším možným využitím funkce tabulkového kalkulátoru Excel je metoda `Evaluate` která vrací výsledek matematického výrazu v jejím argumentu, což nám umožnuje vložit do naší aplikace například výkonnou kalkulačku.

Pro vyzkoušení této funkce si vytvoříme formulář s jedním ovládacím prvkem typu `Button` pojmenovaným `btnVypocitej` a dvěma `TextBoxy` pojmenovanými `txtVzorec` a `txtVysledek` – viz obrázek Obr. 3–2. Událostní proceduru `btnVypocitej_Click` tlačítka `btnVypocitej` upravíme podle výpisu V 3-9. Do `TextBoxu` jménem `txtVzorec` můžeme po spuštění formuláře psát výrazy a po kliknutí na tlačítko `btnVypocitej` se v `TextBoxu` jménem `txtVysledek` zobrazí jejich hodnota.



Obr. 3–2 Jednoduchá kalkulačka

V 3-9 Použití metody `Evaluate`

```
Sub btnVypocitej_Click(  ByVal sender As System.Object,
                         ByVal e As System.EventArgs) Handles btnVypocitej.Click

    'Spustíme Excel
    Dim ExcelApp As New Application
    Dim vzorec As String

    Vzorec = txtVzorec.Text
    txtVysledek.Text = ExcelApp.Evaluate(vzorec).ToString

    'Ukončíme Excel
    ExcelApp.Quit()
    ExcelApp = Nothing
End Sub
```

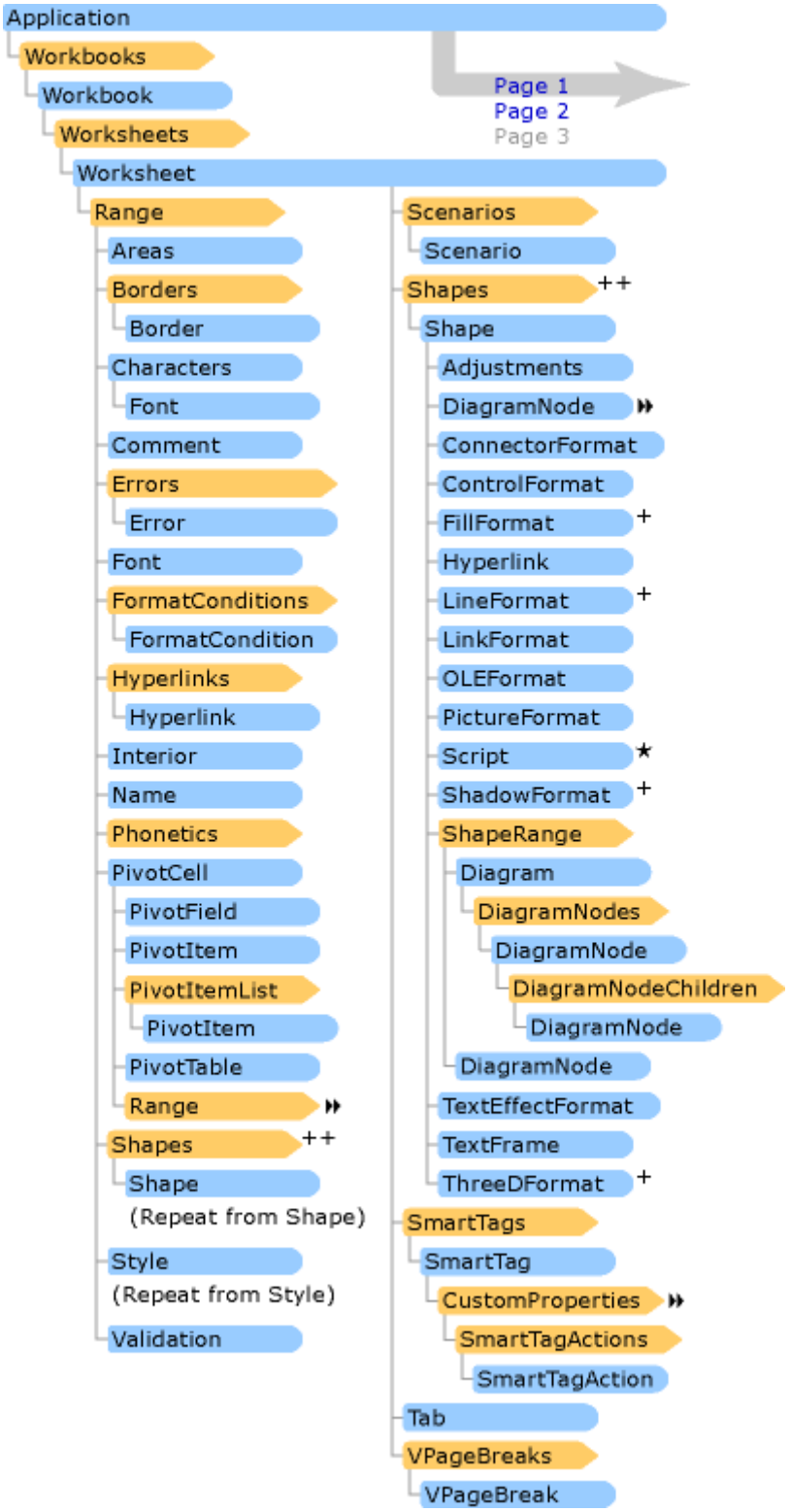
3.4 Grafické znázornění objektového modelu Microsoft Excel

Toto znázornění pochází z internetových stránek společnosti Microsoft (8) a uvádím je zde, protože dovoluje učinit si názornou představu o hierarchii objektů aplikace Excel. Vzhledem k tomu, že zobrazuje verzi XP, nemusí přesně odpovídat novějším verzím a některé objekty mohou chybět, ale stejně názorný novější se mi nalézt nepodařilo. Symbol >> v obrázku naznačuje, že objekt je více rozveden na jiném místě

Grafické znázornění objektového modelu Microsoft Excel strana 1

Grafické znázornění objektového modelu Microsoft Excel strana 2

Grafické znázornění objektového modelu Microsoft Excel strana 3



4 Práce s aplikací Microsoft PowerPoint

Pomocí aplikace Microsoft PowerPoint můžeme vytvořit a případně i otevřít prezentaci této aplikace. Protože se domnívám, že automatizované vytváření prezentací není příliš časté, je v následujícím textu uveden jen stručný přehled.

Informace o objektech, jejich metodách a vlastnostech, uvedených v této kapitole, byly čerpány z dokumentace pro `Microsoft.Office.Interop.PowerPoint` na internetových stránkách firmy Microsoft (9)

4.1 Spuštění a ukončení aplikace Microsoft PowerPoint

Stejně jako je tomu v ostatních aplikacích, i Microsoft PowerPoint spustíme vytvořením nové instanci objektu `PowerPoint.Application`. Pokud se pokusíme vytvořit další, je do nové proměnné vložen dříve vytvořený objekt. PowerPoint běží stále v jednom procesu a obě proměnné obsahují stejný objekt. Na to je potřeba myslet při ukončování aplikace metodou `Quit`. Při použití této metody dojde k uzavření všech otevřených prezentací, a to včetně těch otevřených uživatelem. Lze tomu zabránit sledováním počtu otevřených prezentací, a pokud je jich víc, než kolik jsme jich otevřeli my, tak metodu `Quit` nevolat.

4.2 Vytvoření a otevřen prezentace

Pro vytvoření nové prezentace slouží metoda `Add` kolekce `Presentations`. Tato kolekce obsahuje všechny otevřené prezentace představované objekty `Presentation`; znovu připomínám, že včetně těch vytvořených uživatelem. Pokud chceme prezentaci založit na nějaké šabloně, otevřeme soubor se šablonou pomocí metody `Open` s parametrem `Untitled:=MsoTriState.msoCTrue.`

K uložení a následnému uzavření prezentace slouží metody `SaveAs` (případně `Save`) a `Close`.

4.3 Vložení snímku

Prezentace se skládá z jednotlivých snímků, které jsou představovány objekty `Slide` seskupenými do kolekce `Slides`. Pro přidání nového snímku slouží metoda `AddSlide` (metoda `Add` funguje také, ale v dokumentaci je u ní napsáno, že je vyhrazena pro interní použití). V parametru této metody lze specifikovat rozložení vkládaného snímku.

4.4 Práce s objekty na snímku

Objekty na snímku, jako jsou textová pole, obrázky, tabulky atd., jsou reprezentovány objekty `Shape`; o jaký typ objektu jde, se dozvíme z vlastnosti `Type`, která může obsahovat jednu z hodnot výčtu `MsoShapeType`. Objekty `Shape` sdružuje kolekce `Shapes`. Pro přidání objektu do této kolekce slouží metody `Add...` Pro textové pole je to metoda `AddTextbox`, pro obrázek `AddPicture`, pro tabulku `AddTable` atp. Pro práci s jednotlivými objekty je vhodné využívat proměnné, protože při použití motivu může dojít ke změně pořadí objektů v kolekci `Shapes` a přístup pomocí indexu potom vede k chybám.

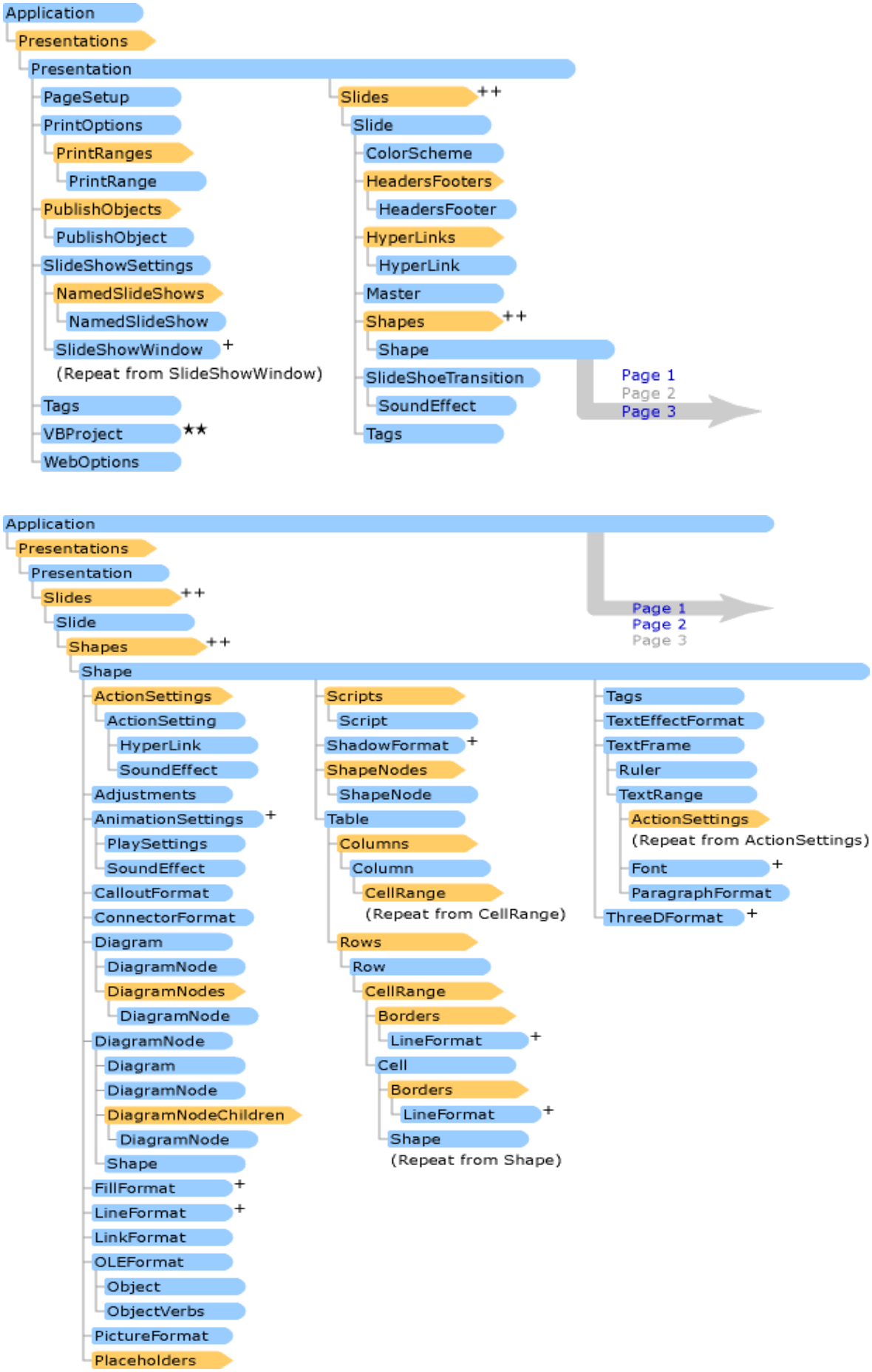
V zájmu jednotného vzhledu vytvořených prezentací se používají motivy a šablony. Pro nastavení motivu se používá metoda `ApplyTheme`, jejímž parametrem je řetězec s názvem motivu nebo s cestou k souboru s motivem (.thmx), případně i souboru se šablonou (.potx) nebo prezentací (.pptx) jejichž motiv bude převzat.

4.5 Grafické znázornění objektového modelu MS PowerPoint

Toto znázornění pochází z internetových stránek společnosti Microsoft (10) a uvádím je zde, protože dovoluje učinit si názornou představu o hierarchii objektů aplikace Microsoft Word. Vzhledem k tomu, že zobrazuje verzi XP, nemusí přesně odpovídat novějším verzím a některé objekty mohou chybět, ale stejně názorný novější se mi nalézt nepodařilo.

Grafické znázornění objektového modelu MS PowerPoint strana 1

Grafické znázornění objektového modelu MS PowerPoint strany 2 a 3



5 Příklady možných problémů a jejich řešení – HOWTO

Potřebuji:

Vytvořit výstup z aplikace ve formátu (.doc), (.docx) nebo (.pdf)

Řešení	Potřebné postupy
• Spustit Word z aplikace	kapitola 2.1
• Vytvořit nový dokument	kapitola 2.2
• Naplnit dokument textem	kapitola 2.3
• Dokument uložit nebo zobrazit a nechat otevřený	kapitola 2.2
• Ukončit nebo nechat spuštěný Word	kapitola 2.1

Plnit existují dokument daty

Řešení	Potřebné postupy
• Vložit záložky do dokumentu	Ruční vložení: Karta **Vložení** – **Odkazy**
• Spustit Word z aplikace	kapitola 2.1
• Vložit data	kapitola 2.3.4

Zkontrolovat pravopis nějakého textu

Řešení	Potřebné postupy
• Spustit Word z aplikace	kapitola 2.1
• Vytvořit nový dokument	kapitola 2.2
• Vložit text do objektu Range	kapitola 2.3.3
• Zkontrolovat pravopis	kapitola 2.3.9
• Ukončit Word	kapitola 2.1

Nezávisle zkontrolovat výsledky výpočtů

Řešení	Potřebné postupy
• Spustit Excel z aplikace	kapitola 3.1
• Vytvořit nový sešit	kapitola 3.2
• Vložit do listu vzorce a výsledky výpočtů pro porovnání	kapitola 3.2.1
• Sešit uložit nebo zobrazit a nechat otevřený	kapitola 3.2
• Ukončit nebo nechat spuštěný Excel	kapitola 3.1

Načíst data do vlastní aplikace z listu (.xlsx)

Řešení	Potřebné postupy
• Spustit Excel z aplikace	kapitola 3.1
• Otevřít sešit	kapitola 3.2
• Načíst data z buněk	kapitola 3.2.1
• Ukončit Excel	kapitola 3.1

6 Popis projektu

Praktický projekt realizovaný v rámci této práce řeší problém vytvoření kalibračního listu a výpočtu nejistot pro kalibrace vah podle metodiky vydané Českým kalibračním sdružením s názvem Revize kalibračních postupů pro váhy s neautomatickou činností. Důvodem pro jeho realizaci je nespokojenost s firmou dodávající software do naší metrologické laboratoře. SW používaný v laboratoři obsahuje různé moduly pro jednotlivé veličiny. Tyto moduly v ideálním případě umožnují po vložení naměřených hodnot spočítat nejistoty a vytvořit kalibrační listy pro jednotlivá měřidla ukládat je do databáze. Bohužel dokončení modulu pro hmotnost se stále odkládá a vhledem k finančním požadavkům dodavatelské firmy a poměrně nízkému počtu kalibrovaných vah nelze očekávat, že by se situace v brzké době zlepšila. Navíc zkušenosti z ostatních modulů ukazují, že pokud vyvstane potřeba upravit nebo změnit výpočty nejistot, jde o časově a i finančně náročný proces.

Kalibrace vah v současné době probíhá tak, že naměřená data jsou vložena do sešitu tabulkového kalkulátoru Excel se zadanými vzorci pro výpočet výsledů jednotlivých zkoušek a tabulky s těmito výsledky jsou pak zkopírovány do textového okna v modulu Hmotnost a ten pak vytvoří kalibrační list. Tento postup je však zdlouhavý, přináší rizika chybného přepisu naměřených hodnot a použití neaktuálních dat etalonů a ani estetická úprava kalibračního listu není úplně nejlepší.

Protože modul hmotnost umožnuje i vkládání dokumentů ve formátu (.docx) jako kalibračních listů, rozhodl jsem se vyvinout aplikaci, která zjednoduší zápis kalibračních dat a jejím výstupem bude kalibrační list právě ve formátu (.docx). Nezanedbatelnou výhodou je i to, že případné opravy nebo změny mohou být rychlé a nezávislé na přidělených finančních prostředcích.

Kalibrační postup vyžaduje provedení tří typů zkoušek a stanovuje způsob výpočtu nejistot pro jednotlivé body, ve kterých je váha kalibrována. Jsou to: zkouška opakovatelnosti, excentricity a správnosti. Zkoušky opakovatelnosti a excentricity se provádí každá v jednom bodě a zkouška správnosti pak v minimálně šesti bodech, pokud možno rovnoměrně rozložených v celém rozsahu váhy. Pro výpočet nejistoty kalibrace jednotlivých bodů je nutné uvažovat a tedy i zaznamenat teplotu a vlhkost okolí, nejistoty a chyby etalonů použitých v jednotlivých bodech a parametry kalibrované váhy.

Aplikace dokáže z prvotních dat vložených do ní uživatelem vytvořit kalibrační list ve formátu (.docx) včetně výpočtu nejistot. Vložená prvotní data včetně jmenovitých hodnot, chyb a nejistot použitých etalonů je možné uložit do XML souboru a tento soubor následně načíst. Na obsah textového okna, do kterého se píše výsledek kalibrace, je využita kontrola pravopisu aplikace Microsoft Word s možností nahradit nalezená chybná slova navrženými opravami.

Protože se počítá s nasazením této aplikace v akreditované metrologické laboratoři, poskytuje aplikace výstup ve formátu (.xls), ve kterém je možné nejistoty spočítané aplikací porovnat s hodnotami vypočítanými pomocí vzorců Excelu. Každý software použitý v takovéto laboratoři musí být validovaný, což v podstatě znamená, že musí existovat důkazy, že daný software zpracovává vstupní data správně a pokaždé stejně. Protože Český institut pro akreditaci uznává Microsoft Excel za validovaný software, může tento dokument sloužit k validaci a následné pravidelné verifikaci této aplikace.

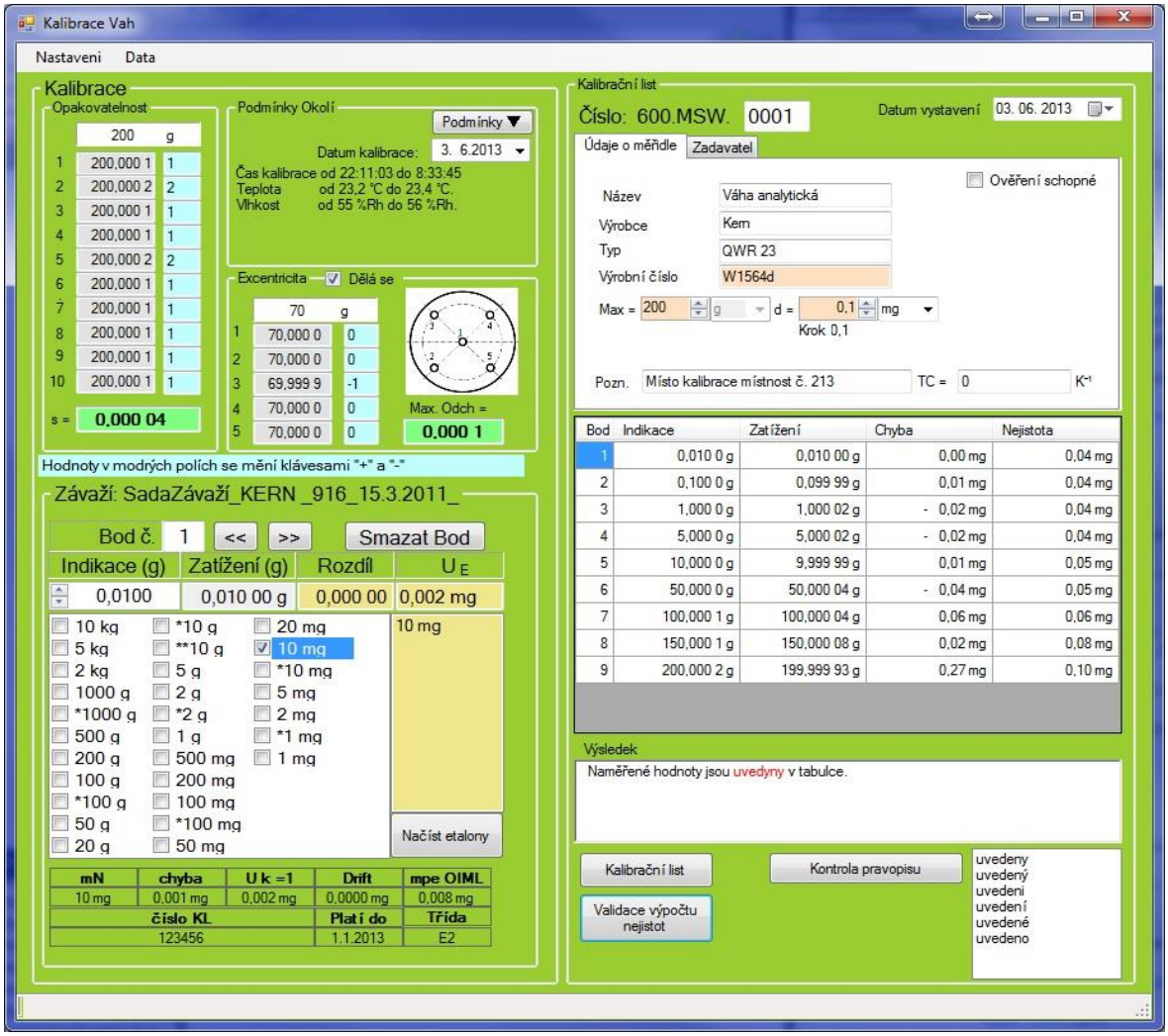
Pro vývoj aplikace bylo použito prostředí Visual Studio 2010 Expres. Kód uživatelského rozhraní je tedy vygenerován tímto prostředím. Hlavní třída `Vahy` je z důvodu lepší orientace rozdělena do několika souborů pomocí klíčového slova `Partial`. Pro událostní procedury uživatelského rozhraní a procedury s nimi bezprostředně související je to soubor Vahy.vb. Pro procedury pracující s aplikacemi Microsoft Word Microsoft Excel jsou to soubory VahyWord.vb a VahyExcel.vb. Pomocné procedury použité na více místech kódu jsou v souboru VahyVlProc.vb. V záhlaví těchto souborů jsou importované jmenné prostory použité v příslušném souboru, což zkracuje a zpřehledňuje zápis kódu. Jmenné prostory `Systém` a `Microsoft.VisualBasic` používané ve většině kódu jsou importovány ve vlastnostech projektu, karta **References**. Proměnné deklarované na úrovni třídy `Váhy` mají prefix „aaV_“ a konstanty prefix „C_“ kvůli přednostnímu zobrazení v nabídce IntelliSense. V modulu `VClass`, umístěném v souboru VClass.vb, jsou definovány třídy pro ukládání dat a výpočty pro jednotlivé zkoušky. Popis těchto tříd je uveden v tabulce. Dále jsou v tomto modulu definovány pomocné struktury a výčty.

T 2 Seznam tříd

Název třídy	Popis funkce
HmVaha	Ukládá parametry váhy
HmSadaZavazi	Ukládá parametry použité sady závaží
HmOpakovatelnost	Uchovává zadané hodnoty opakovatelnost a počítá směrodatnou odchylku
HmExcentricita	Uchovává zadané hodnoty excentricity a počítá maximální chybu
HmSprávnostVse	Uchovává pole bodů s hodnotami správnosti + podmínky při kalibraci
HmSpravnostBod	Uchovává naměřené a vypočítané hodnoty správnosti jednoho bodu měření, použité etalony v tomto bodě a počítá nejistoty
HmNejistotaBodu	Počítá nejistoty
HmPodminky	Uchovává podmínky okolí při kalibraci

Okno uživatelského rozhraní, viz obrázek Obr. 6–1,je rozděleno na dvě části. Levá část slouží k zadávání naměřených hodnot, podmínek měření a použitých etalonů, a pravá část k vložení a zobrazení dat, která budou na kalibračním listu. Při zahájení kalibrace je nejprve nutné zadat výrobní číslo, velikost maximálního zatížení a hodnotu dílku váhy. Teprve potom jsou zpřístupněna pole pro zadávání kalibračních dat. Data pro výpočet opakovatelnosti a excentricity se zadávají tak, že do bílých polí příslušných panelů vložíme jmenovitou hodnotu zatížení, kterou potvrdíme klávesou **Enter**. Ve světle modrých polích potom pomocí kláves „**+**“ a „**–**“ nastavíme hodnotu, která řádově odpovídá posledním digitům displeje váhy, přičemž krok odpovídá hodnotě dílku váhy. V šedivých polích je pak zobrazen součet této hodnoty a zadané jmenovité hodnoty zatížení, což odpovídá indikaci váhy. Protože při vkládání dat nemusí uživatel spustit zrak z monitoru, snižuje se tak možnost zadání chybné hodnoty. Při zadávání hodnot pro jednotlivé zátěže vybírá uživatel použitá závaží ze zaškrtávacího seznamu, a stiskem pravého tlačítka myši vloží součet jmenovitých hodnot vybraných závaží do pole indikace. Hodnotu v tomto poli potom může upravit pomocí šipek tak, aby odpovídala skutečné indikaci váhy. V dalších textových polích je zobrazena skutečná hodnota zatížení, rozdíl mezi zatížením a indikací, a nejistota etalonů. Přechod na další bod se provádí pomocí tlačítek se šipkami umístěnými vedle čísla bodu. Zpra-

cované body jsou přehledně zobrazeny v tabulce umístěné v pravé části okna aplikace.



Obr. 6–1 Okno uživatelského rozhraní aplikace

Pro načtení sady etalonových závaží slouží tlačítko **Načíst etalony**, které vyvolá dialogové okno pro výběr souboru s daty etalonů. Tento soubor je v textovém formátu, a jeho první řádek obsahuje identifikaci sady závaží (výrobní číslo, třída přesnosti, výrobce) a data o její kalibraci (číslo kalibračního listu, datum a doba platnosti kalibrace). V dalších řádcích je pro každé závaží uvedena jeho: jmenovitá hodnota (v gramech), odchylka, nejistota kalibrace, drift[9] (vše v miligramech) a zda má značku[10]. Jednotlivé etalony sady jsou zobrazeny v zaškrtávacím seznamu. V textových polích pod tímto seznamem jsou pak vypsány podrobnosti o aktuálně vybraném etalonu, včetně data platnosti kalibrace a maximální povolené chyby. V menu **Nastavení** lze

[9] Drift je změna hodnoty od poslední kalibrace.

[10] Stejné jmenovité hodnoty v sadě jsou odlišeny jedním nebo více důlky.

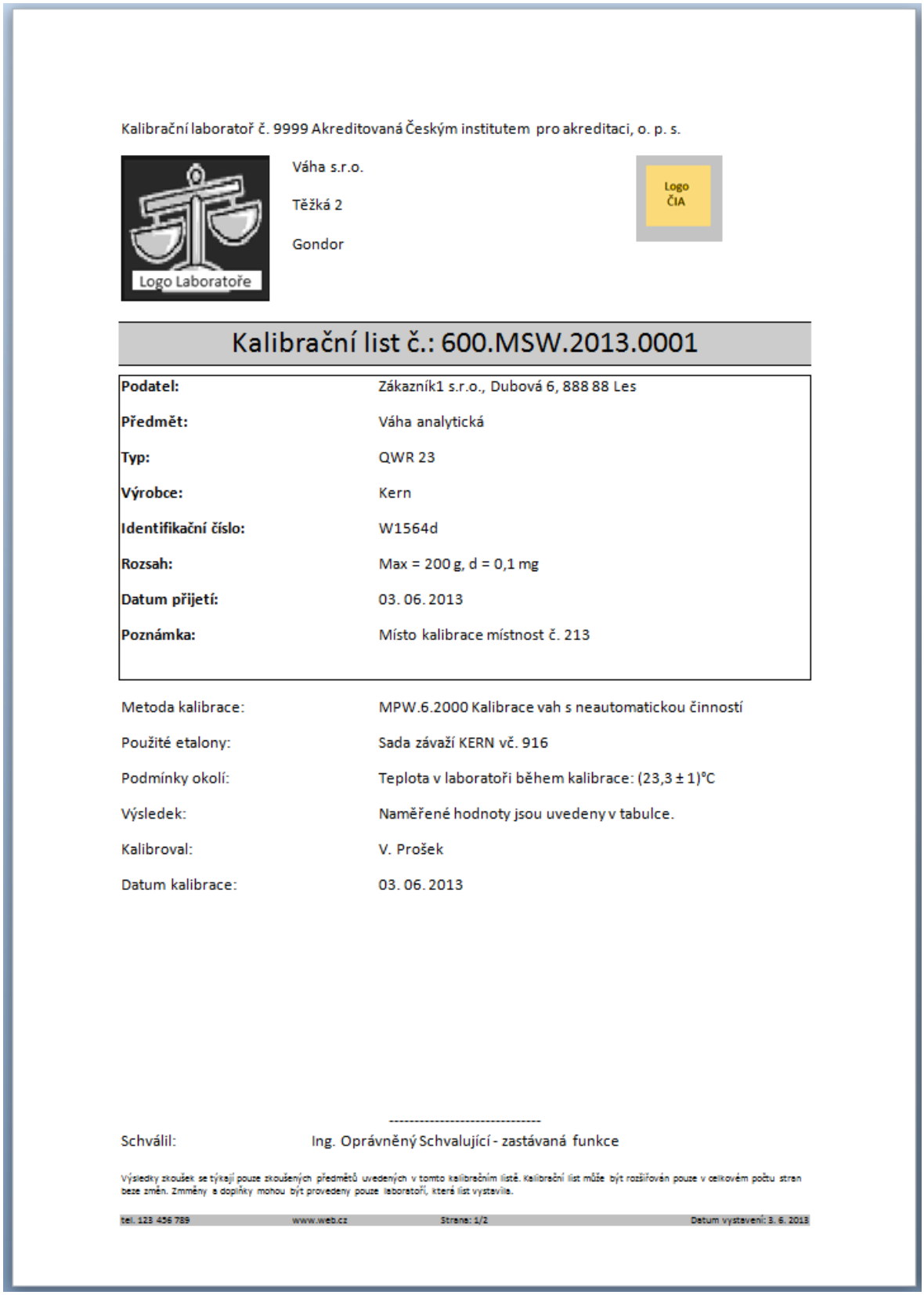
zadat cesty k souborům s použitými obrázky, adresu laboratoře a texty vkládané do kalibračního listu. Text s výsledkem kalibrace se píše do textboxu **Výsledek**, na obrázku Obr. 6–1 je vidět červeně zdůrazněné slovo s chybou a ve vedlejším textboxu seznam navržených oprav. Dokončené nebo i rozpracované kalibrace lze uložit a později znovu načíst. Kalibrace se ukládají včetně sady použitých závaží do souboru ve formátu (.xml). Tlačítky **Kalibrační list** a **Validace výpočtu nejistot** vytvoříme kalibrační list ve formátu (.docx), respektive pracovní list ve formátu (.xls) s porovnáním výpočtu nejistot.

Výsledný kalibrační list je na obrázcích Obr. 6–3 a Obr. 6–4. Jeho záhlaví je naformátováno pomocí tabulky, ve které jsou umístěny obrázky s logem laboratoře, případně akreditačního orgánu, a adresa laboratoře. Další odstavce s číslem kalibračního listu a údaji o váze jsou zvýrazněny ohraničením. Hodnoty naměřené a vypočtené při kalibraci, jsou uvedeny v tabulkách na druhé straně kalibračního listu. Zápatí je pro obě strany stejné, a obsahuje mimo data vystavení, telefonního čísla a internetové adresy laboratoře, i jedno pole s číslem stránky a druhé s celkovým počtem stránek. Po kliknutí na tlačítko **Kalibrační list**, je dokument vytvořen na pozadí a uživatel se kliknutím v zobrazeném dialogovém okně může rozhodnout, zda ho chce uložit pod automaticky vygenerovaným názvem nebo nechat otevřený pro případné úpravy.

V sešitu vytvořeném po kliknutí na tlačítko **Validace výpočtu nejistot,** jsou do modře zvýrazněných buněk (s titulkem Vz. Excel.) vloženy vypočítané hodnoty nejistoty a jejích jednotlivých složek. Do zeleně zvýrazněných buněk (s titulkem Hod. SW) jsou vloženy vzorce pro výpočet odpovídajících složek nejistoty. Na obrázku Obr. 6–2 je část listu se zobrazenými vzorci. V případě, že jsou výpočty správně, tak se hodnoty odpovídajících sousedních buněk shodují. Na obrázku Obr. 6–5 je červeně vyznačena oblast, kde se hodnoty liší a v aplikaci bude třeba najít a odstranit chybu. Parametry váhy a podmínky okolí, jsou ve žlutých polích. Naměřená data ze zkoušek opakovatelnosti a excentricity jsou v hnědých respektive oranžových polích. Sešit s porovnáním výpočtu nejistot zůstává zobrazený a otevřený, a je na uživateli, aby ho uložil nebo zavřel.

	A	B	C
28	**Nejistoty**		
29	Nejistoty společné pro		
30	u_{opak}	=$G$15/(10)^0,5	0,000013333333333776
31	u_{exc}(rel)	=(($E$4)/(2*$E$3*(6)^0,5))	2,91605921759902E-07
32	u_{d0}	=$B$5/(2*(3)^0,5)	0,0000288675134594813
33	u_{di}	=$B$4/(2*(3)^0,5)	0,0000288675134594813

Obr. 6–2 Část listu se zobrazenými vzorci



Kalibrační laboratoř č. 9999 Akreditovaná Českým institutem pro akreditaci, o. p. s.

Logo Laboratoře

Váha s.r.o.

Těžká 2

Gondor

Logo ČIA

Kalibrační list č.: 600.MSW.2013.0001

Podatel:	Zákazník1 s.r.o., Dubová 6, 888 88 Les
Předmět:	Váha analytická
Typ:	QWR 23
Výrobce:	Kern
Identifikační číslo:	W1564d
Rozsah:	Max = 200 g, d = 0,1 mg
Datum přijetí:	03. 06. 2013
Poznámka:	Místo kalibrace místnost č. 213

Metoda kalibrace:	MPW.6.2000 Kalibrace vah s neautomatickou činností
Použité etalony:	Sada závaží KERN vč. 916
Podmínky okolí:	Teplota v laboratoři během kalibrace: (23,3 ± 1)°C
Výsledek:	Naměřené hodnoty jsou uvedeny v tabulce.
Kalibroval:	V. Prošek
Datum kalibrace:	03. 06. 2013

Schválil: Ing. Oprávněný Schvalující - zastávaná funkce

Výsledky zkoušek se týkají pouze zkoušených předmětů uvedených v tomto kalibračním listě. Kalibrační list může být rozšiřován pouze v celkovém počtu stran beze změn. Změny a doplňky mohou být provedeny pouze laboratoří, která list vystavila.

tel. 123 456 789 www.web.cz Strana: 1/2 Datum vystavení: 3. 6. 2013

Obr. 6–3 Vytvořený kalibrační list strana 1

Naměřené a vypočtené hodnoty

Opakovatelnost:

Opakovatelnost při zatížení 200 g byla stanovena na: 0,04 mg

Excentricita:

Uložení zátěže	Zatížení (g)	Sektor	Indikace (g)
	70	1	70,000 0
2 3		2	70,000 0
1		3	69,999 9
4 5		4	70,000 0
		5	70,000 0

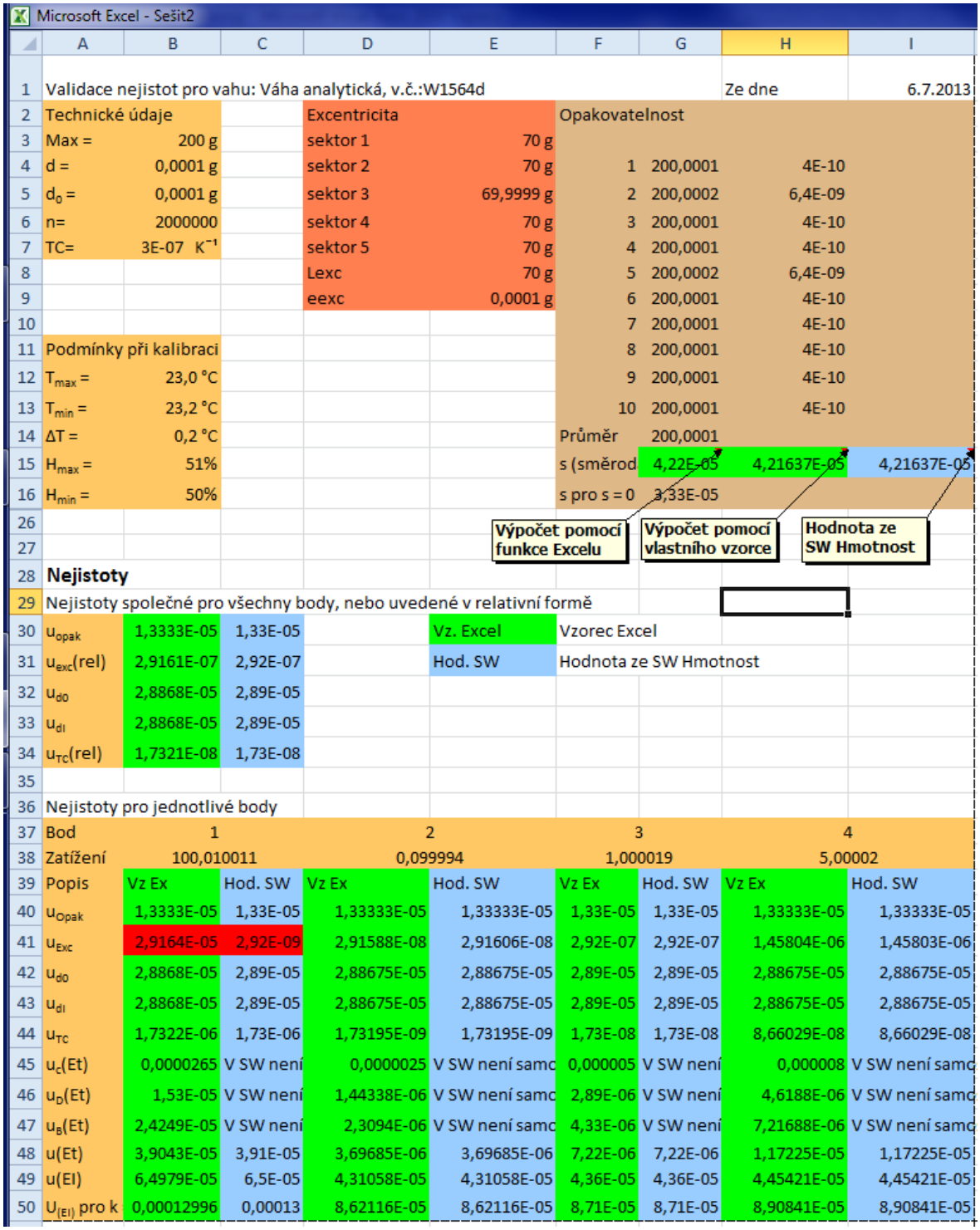
Správnost:

Zatížení (g)	Indikace (g)	Odchylka (mg)	Nejistota (mg)
0,010 00	0,010 0	0,00	0,09
0,099 99	0,100 0	0,00	0,09
1,000 02	1,000 0	0,00	0,09
5,000 02	5,000 0	0,00	0,09
9,999 99	10,000 0	0,00	0,09
50,000 04	50,000 0	0,00	0,10
100,000 04	100,000 1	0,00	0,12
150,000 08	150,000 1	0,00	0,16
199,999 93	200,000 2	0,00	0,20

Uvedená rozšířená nejistota měření je součinem standardní nejistoty měření a koeficientu rozšíření k = 2, což pro normální rozdělení odpovídá pravděpodobnosti pokrytí cca 95%. Standardní nejistota byla určena v souladu s dokumentem EA 4/02. Nejistota nezahrnuje časovou stabilitu kalibrovaného přístroje."

Výsledky zkoušek se týkají pouze zkoušených předmětů uvedených v tomto kalibračním listě. Kalibrační list může být rozšiřován pouze v celkovém počtu stran beze změn. Změny a doplňky mohou být provedeny pouze laboratoří, které list vystavila.

Obr. 6–4 Vytvořený kalibrační list strana 2



Microsoft Excel - Sešit2

	A	B	C	D	E	F	G	H	I
1	Validace nejistot pro vahu: Váha analytická, v.č.:W1564d							Ze dne	6.7.2013
2	Technické údaje			Excentricita		Opakovatelnost			
3	Max =	200 g		sektor 1	70 g				
4	d =	0,0001 g		sektor 2	70 g	1	200,0001	4E-10	
5	d_0 =	0,0001 g		sektor 3	69,9999 g	2	200,0002	6,4E-09	
6	n=	2000000		sektor 4	70 g	3	200,0001	4E-10	
7	TC=	3E-07 K^{-1}		sektor 5	70 g	4	200,0001	4E-10	
8				Lexc	70 g	5	200,0002	6,4E-09	
9				eexc	0,0001 g	6	200,0001	4E-10	
10						7	200,0001	4E-10	
11	Podmínky při kalibraci					8	200,0001	4E-10	
12	T_{max} =	23,0 °C				9	200,0001	4E-10	
13	T_{min} =	23,2 °C				10	200,0001	4E-10	
14	ΔT =	0,2 °C				Průměr	200,0001		
15	H_{max} =	51%				s (směrod	4,22E-05	4,21637E-05	4,21637E-05
16	H_{min} =	50%				s pro s = 0	3,33E-05		
26									
27									
28	**Nejistoty**								
29	Nejistoty společné pro všechny body, nebo uvedené v relativní formě								
30	u_{opak}	1,3333E-05	1,33E-05		Vz. Excel	Vzorec Excel			
31	u_{exc}(rel)	2,9161E-07	2,92E-07		Hod. SW	Hodnota ze SW Hmotnost			
32	u_{d0}	2,8868E-05	2,89E-05						
33	u_{dl}	2,8868E-05	2,89E-05						
34	u_{TC}(rel)	1,7321E-08	1,73E-08						
35									
36	Nejistoty pro jednotlivé body								
37	Bod	1		2		3		4	
38	Zatížení	100,010011		0,099994		1,000019		5,00002	
39	Popis	Vz Ex	Hod. SW	Vz Ex	Hod. SW	Vz Ex	Hod. SW	Vz Ex	Hod. SW
40	u_{Opak}	1,3333E-05	1,33E-05	1,33333E-05	1,33333E-05	1,33E-05	1,33E-05	1,33333E-05	1,33333E-05
41	u_{Exc}	2,9164E-05	2,92E-09	2,91588E-08	2,91606E-08	2,92E-07	2,92E-07	1,45804E-06	1,45803E-06
42	u_{d0}	2,8868E-05	2,89E-05	2,88675E-05	2,88675E-05	2,89E-05	2,89E-05	2,88675E-05	2,88675E-05
43	u_{dl}	2,8868E-05	2,89E-05	2,88675E-05	2,88675E-05	2,89E-05	2,89E-05	2,88675E-05	2,88675E-05
44	u_{TC}	1,7322E-06	1,73E-06	1,73195E-09	1,73195E-09	1,73E-08	1,73E-08	8,66029E-08	8,66029E-08
45	u_c(Et)	0,0000265	V SW není	0,0000025	V SW není samc	0,000005	V SW není	0,000008	V SW není samc
46	u_D(Et)	1,53E-05	V SW není	1,44338E-06	V SW není samc	2,89E-06	V SW není	4,6188E-06	V SW není samc
47	u_B(Et)	2,4249E-05	V SW není	2,3094E-06	V SW není samc	4,33E-06	V SW není	7,21688E-06	V SW není samc
48	u(Et)	3,9043E-05	3,91E-05	3,69685E-06	3,69685E-06	7,22E-06	7,22E-06	1,17225E-05	1,17225E-05
49	u(El)	6,4979E-05	6,5E-05	4,31058E-05	4,31058E-05	4,36E-05	4,36E-05	4,45421E-05	4,45421E-05
50	$U_{(El)}$ pro k	0,00012996	0,00013	8,62116E-05	8,62116E-05	8,71E-05	8,71E-05	8,90841E-05	8,90841E-05

Výpočet pomocí funkce Excelu

Výpočet pomocí vlastního vzorce

Hodnota ze SW Hmotnost

Obr. 6–5 Vytvořený pracovní list

7 Zhodnocení a závěr

Vzhledem k rozsahu objektového modelu Microsoft Office nelze popsat všechny funkce, které obsahuje, ale v této práci jde zde především o předvedení principů, jak jednotlivé aplikace fungují a jak využít možností, které poskytují. Pokud požadujeme nějakou funkci, která zde není popsána, můžeme ji dohledat v dokumentaci.

V případě tabulkového kalkulátoru Excel považuji za zajímavé jeho využití pro porovnání výpočtů, které provedla námi vyvinutá aplikace, s výpočty provedenými touto aplikací. Protože Excel je některými akreditačními orgány (např. ČIA, www.cai.cz) uznáván jako validovaný software, lze do aplikace jednoduše přidat funkci pro generování protokolu o validaci. Tento protokol, který vlastně dokazuje, že naše aplikace počítá správně, je pak možné vystavit snadno a rychle. Tato funkce může být užitečná i při vlastním vývoji aplikace, protože přispívá k snazšímu odhalení chyb, což se mi potvrdilo i při realizaci praktického projektu.

Využití PowerPointu může být diskutabilní, protože prezentaci vytvoříme většinou jen jednou a pak ji případně upravujeme a dolaďujeme, ale není důvod dělat ji automaticky.

Pokud chceme nějakou činnost zautomatizovat a nevíme přesně jak, můžeme si pomoci tím, že požadovanou činnost provedeme ručně se zapnutým záznamem makra. Pokud si zaznamenané makro potom prostudujeme, nalezneme v něm konkrétní objekty, metody, vlastnosti a konstanty, se kterými lze danou činnost provést. Zejména konstant je mnoho a takto nalezneme tu správnou rychleji než experimentováním. Bohužel nefunguje to vždy a záznamník maker nezaznamenává vše. Pak nezbývá než studovat dokumentaci.

Význam spolupráce VB.NET a Microsoft Office spatřuji především v tom, že umožňuje jednoduše vytvářet kvalitní výstupy v široce rozšířeném formátu s nízkými náklady, protože Microsoft Office patří k nejrozšířenějším kancelářským balíkům a Visual Studio 2010 Expres je k dispozici zdarma i ke komerčním účelům.

8 Seznam výpisů a obrázků

Výpisy:

Obrázky:

9 Použitá literatura

1. .NET Framework. WIKIPEDIA. *Wikipedia* [online]. 29. 6..2013 [cit. 2013-06-30]. Dostupné z: http://en.wikipedia.org/wiki/.NET_Framework

2. KAČMÁŘ, Dalibor. *Programujeme: NET aplikace ve Visual Studiu .NET*. 1. vyd. Praha: Computer Press, 2001, 335 s. ISBN 80-722-6569-5.

3. Automatizace Microsoft Office 2007. In: HANÁK, Ján. *PC World* [online]. 01.12.08 [cit. 2013-06-30]. Dostupné z: http://pcworld.cz/archiv/automatizace-microsoft-office-2007-17622

4. PETROUTSOS, Evangelos. *Myslíme v jazyku Visual Basic .Net 1. díl: knihovna programátora*. 1. vyd. Praha: Grada Publishing, 2003, 675 s. ISBN 80-247-0371-8.

5. Microsoft.Office.Interop.Word Namespace. MICROSOFT. *Office for developers* [online]. © 2013 [cit. 2013-06-26]. Dostupné z: http://msdn.microsoft.com/en-us/library/office/microsoft.office.interop.word(v=office.14).aspx

6. Microsoft Word Object Model. MICROSOFT. *Office for developers* [online]. © 2013 [cit. 2013-06-30]. Dostupné z: http://msdn.microsoft.com/en-us/library/office/aa165321(v=office.10).aspx

7. Microsoft.Office.Interop.Excel Namespace. MICROSOFT. *Office for developers* [online]. © 2013 [cit. 2013-06-30]. Dostupné z: http://msdn.microsoft.com/en-us/library/office/microsoft.office.interop.excel(v=office.14).aspx

8. Microsoft Excel Object Model. MICROSOFT. *Office for developers* [online]. © 2013 [cit. 2013-06-30]. Dostupné z: http://msdn.microsoft.com/en-us/library/office/aa141044(v=office.10).aspx

9. Microsoft.Office.Interop.PowerPoint Namespace. MICROSOFT. *Office for developers* [online]. © 2013 [cit. 2013-06-30]. Dostupné z: http://msdn.microsoft.com/en-us/library/office/microsoft.office.interop.powerpoint(v=office.14).aspx

10. Microsoft PowerPoint Object Model. MICROSOFT. *Office for developers* [online]. © 2013 [cit. 2013-06-30]. Dostupné z: http://msdn.microsoft.com/en-us/library/office/aa141155(v%20=office.10).aspx