Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta – Katedra fyziky

Použití maker v MS Excel a v OpenOffice.org Calc

Bakalářská práce

Vedoucí práce: Ing. Michal Šerý

Autor: Daniel Sobol

Anotace

Bakalářská práce čtenáře seznamuje s vytvářením a použitím maker zejména v MS Excel a částečně v OpenOffice.org Calc. Rámcově porovnává možnosti programovacích jazyků Visual Basic for Application a OpenOffice.org Basic. Její součástí jsou mnou vytvořená ukázková makra, jež ilustrují možnosti případného použití v praxi.

Abstract

The bachelor thesis acquaints the revers with the creation and the use of macros, especially in MS Excel and partly in OpenOffice.org Calc. Generally compares the options of programming languages Visual Basic for Application and OpenOffice.org Basic. Its part are the sample macros created by me, which illustrate the options of possible using in practice.

Prohlášení

Prohlašuji že předloženou práci jsem vypracoval samostatně a pouze s použitím uvedené (citované) literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne 30. 4. 2010 ……

Daniel Sobol

Poděkování

Touto formou děkuji svému vedoucímu Ing. Šerému, za cenné rady a připomínky při zpracování této bakalářské práce.

Obsah

1 Úvod

Některé programy již řadu let nabízejí možnosti sebe sama rozšířit, doplnit či zautomatizovat pomocí vestavěného makrojazyka. Mezi těmito makrojazyky významné místo zaujímají jazyky vycházející z Visual Basicu. To je případ také kancelářských balíků MS Office a OpenOffice.org.

Předkládaná bakalářská práce si klade za cíl o dotčených možnostech pojednat, popsat programovací jazyk a zásady programování v rámci MS Excel a částečně v OpenOffice.org Calc. Předložená práce má čtenáře uvést do problematiky vytváření maker v těchto programech.

V první části je obecný popis maker, způsoby jejich vytváření v závislosti na zásadách pro jejich vytváření a způsoby práce s editorem VBA a porovnáním možností programování v MS Excel a v OpenOffice.org Calc .

V další části se čtenář seznámí se způsoby vytváření vlastních funkcí, jejich zařazování do kategorií v seznamu a také s přidáváním stručných popisků k vytvořené funkci a způsoby používání uživatelských dialogů.

Poslední část obsahuje několik maker, na nichž demonstruji poznatky z předchozích částí.

2 Zásady programování ve Visual Basic for Application

2.1 Použití maker

Při denním používání nejrůznějších operací v Excelu musíme často opakovat některé činnosti i operace vícekrát. Tento postup je velmi zdlouhavý a také časově náročný. Pro zjednodušení těchto činností a úsporu času mělo klíčový význam vyvinutí prostředků pro psaní tzv. maker. „**Makro** je posloupnost příkazů, spuštěná jako celek, jejímž cílem je automatizovat některé pracovní postupy, vyloučit tak opakované operace a snížit počet případných chyb při zpracování dat.“[1] Makra se využívají v mnoha případech. Nejvíce však při:

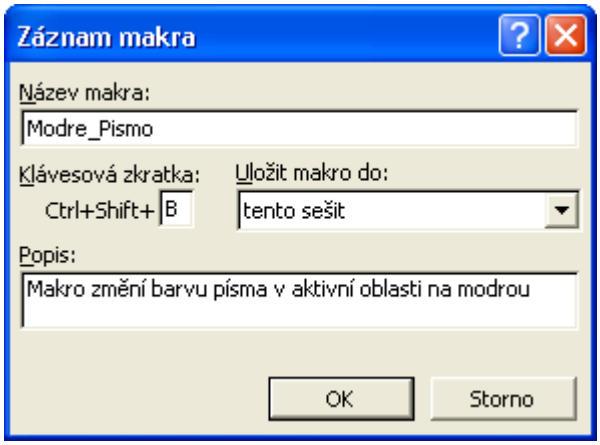
- zpracování dat z jedné a více tabulek do výsledné tabulky
- vytváření grafů
- vkládání funkcí
- formátování stylů buněk
- vytváření vlastních aplikací

2.2 Vytváření maker

Podle rozsahu a složitosti požadavků na makro lze volit mezi třemi způsoby vytvoření:

- **nahrání makra pomocí záznamníku**
  (příkaz pro záznam se nachází na panelu nástrojů v položce Nástroje → Makro → Záznam nového makra). V úvodním dialogovém okně (obr. 1) zvolíme název makra, místo, kam bude makro uloženo a dále můžeme přiřadit klávesovou zkratku a napsat stručný popisek jako např. k čemu makro slouží, kdy bylo vytvořeno etc.

[1] [1] str. 14



Obr. 1 Úvodní dialogové okno záznamníku maker

Po zaktivování záznamníku se začnou nahrávat veškeré úkony až do ukončení záznamu uživatelem. Při nahrávání můžeme zvolit buď standardní absolutní odkaz, nebo můžeme zapnout relativní odkaz. Pokud užijeme absolutní odkaz, např. při výběru buňky (nebo oblasti buněk), pak po spuštění makra se vybere táž buňka (oblast buněk). V případě relativního odkazu se buňka (oblast buněk) vybírá vzhledem k aktuálně vybrané buňce. Zaznamenávání se neukončí automaticky, proto je třeba jej zastavit manuálně. Jinak budou až do ukončení Excelu nahrávány i ty činnosti, které nemají být součástí makra. Zmíněný způsob vytvoření makra je použitelný tehdy, když:

1) se jedná o jednoduché makro, např. orámování či nastavení formátů buněk v aktuální oblasti.

2) se jedná o jednoduché makro, které nahrajeme a použijeme jako součást složitějšího (ručně psaného) makra.

3) použijeme-li záznamníku jako nástroje výuky.

   - pro začátečníky při prohlížení kódu vygenerovaného záznamníkem maker [3]
   - při rozpoznávání objektů [1]
   - při zjišťování anglických názvů funkcí (tj., při používání funkcí v Excelu jsou jejich názvy v češtině, kdežto při psaní makra v editoru

VBA se názvy funkcí zadávají v angličtině. Pokud programátor nezná anglický název funkce, kterou chce použít, nahraje si makro s dotyčnou funkcí a po přečtení zdrojového kódu zjistí její anglický název).

4) chceme-li vyřešit přiřazení klávesové zkratky některé funkci (např. vložení aktuálního času do vybrané buňky).

- **ruční napsání makra**

  V editoru VBA lze celé makro napsat ručně, k tomu je potřeba znát daný programovací jazyk.

- **nahrání makra pomocí záznamníku a následná úprava v editoru Visual Basic**

  Ve složitějších případech nemusí být nahrané makro použitelné, proto je třeba je upravit, např. vložením cyklů, rozhodovacích příkazů, přiřazováním proměnných, přidáním dialogových oken, smazáním nepotřebných kroků (odstranění nadbytečných příkazů). Tyto části neumí záznamník generovat. [2]

2.3 Ukládání maker

Makra se vždy ukládají do sešitu do tzv. **modulů** (pokud je v sešitu uloženo makro, musí sešit obsahovat minimálně jeden modul, ale může jich v něm být více). Viditelná jsou ovšem pouze při otevřeném editoru VBA. Podle způsobu použití lze makro uložit do **aktivního sešitu** (volba Tento sešit). Aby bylo makro dostupné, musí být sešit otevřen. [3]

Kromě tohoto sešitu lze použít také tzv. **osobní sešit maker,** který nese název **Personal.xls.** Ten je Excelem vytvořen teprve v okamžiku, kdy je makro nahráno. Již nahrané makro je potom přístupné pro všechny sešity. Adresář, ve kterém je tento soubor umístěn, je různý podle toho, s jakou verzí Excelu pracujeme. [1]

Makro lze uložit také do **nového sešitu,** který se vytvoří při nahrávání. Udaný postup se dá použít například v případě, když chceme makro kopírovat pro jiné uživatele, aniž bychom museli hledat sešit Personal.xls. V něm navíc mohou být i jiná makra, která nechceme poskytovat dále.

2.4 Spouštění maker

V Excelu je možné spustit makra několika způsoby [1]:

- z dialogového okna Makro (umístěno: Nástroje → Makro → Makra; či klávesová zkratka Alt F8), načež se objeví okno se seznamem všech dostupných maker.

- přiřazenou klávesovou zkratkou (makru lze přiřadit klávesovou zkratku v podobě Ctrl + písmeno nebo případně Ctrl + Shift + písmeno). Tento způsob se nabízí při zaznamenávání makra. Použít se dá i jiná kombinace kláves, ovšem tu je nutno naprogramovat v editoru VBA.

- přiřazením makra nějakému tlačítku (tlačítko se dá graficky upravit). Zmíněný způsob je vhodný zejména pro makro uložené v osobním sešitu maker, protože tlačítko zůstává na liště při každém dalším spuštění Excelu.

- umístěním tlačítka přímo do listu (můžeme navolit různé umístění a velikost tlačítka kdekoli na listu). Využívá se v rámci jednoho konkrétního sešitu:

  - umístěním tlačítka na formuláři
  - přímo z editoru VBA (při úpravě makra v editoruVBA můžeme toto makro spustit příkazem **Run → Run Macro** nebo klávesou **F5**. Kurzor musí být kdekoliv ve zdrojovém kódu.)
  - voláním z jiného makra
  - příkazem nabídky
  - přiřazením k události

Nevýhodou je, že po proběhnutí makra v Excelu nelze vrátit ani samotné makro, ani všechny předchozí kroky (na rozdíl od Wordu) .

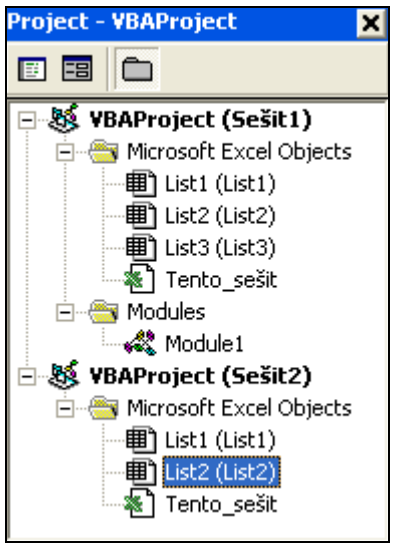
2.5 Editor jazyka Visual Basic for Application (VBA)

Makro je tvořeno zdrojovým kódem, který se dá zobrazit pouze v editoru Visual Basic. Editor VBA je sice samostatnou aplikací, ale jeho spuštění je závislé na tom, zda je či není aktivní Excel. Tj. nemůžeme jej spustit samostatně.

Nedílnými součástmi Editoru VBA jsou **panely** a **okna**. Stejně jako u jiných programů i zde je Panel nabídek. Obsahuje různé příkazy pro práci s makrem. Některým příkazům jsou přiřazeny klávesové zkratky.

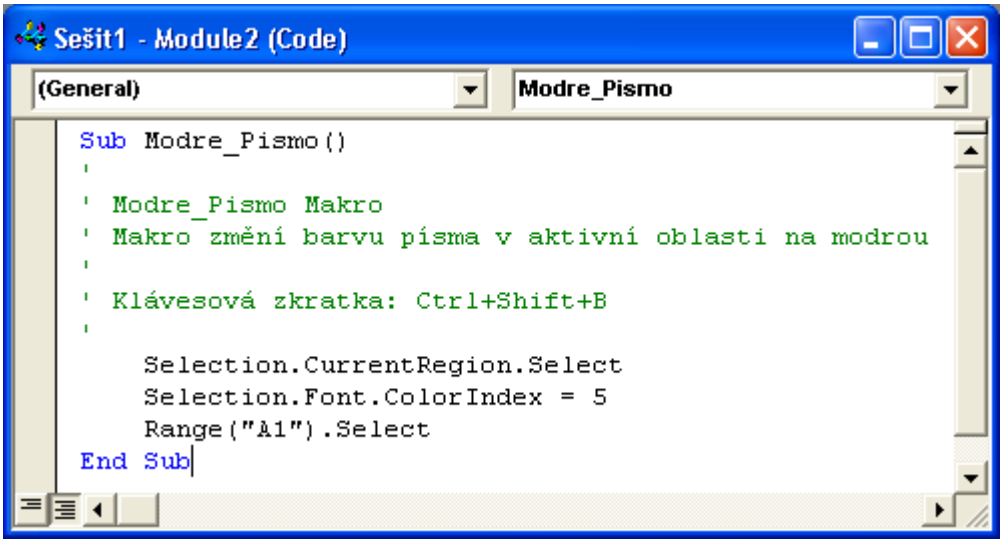
Editor dále obsahuje Panely nástrojů. Je jich celkem šest, přičemž nejužívanější je panel **Standard**, který je vždy při prvním spuštění zapnutý. Ostatní panely můžeme vypínat nebo zapínat, přidat na horní lištu nebo je z ní odtrhnout a také posouvat po ploše. Vybrané příkazy, které jsou obsaženy v Panelu nabídek, jsou na Panelu nástrojů zobrazeny v podobě ikon. Ikony je samozřejmě možné jak přidávat, tak ubírat. Jeden ze šesti panelů („vlastní“) umožňuje uživateli nadefinování vlastních příkazů.

Nezastupitelná jsou ve VBA **okna**. Prostřednictvím Okna průzkumníka projektu (Project Explorer) (obr. 2) můžeme zobrazovat stromové diagramy, získávat přehled o otevřených sešitech (jejich listech) a o modulech (např. kolik má sešit modulů).



Obr. 2 Okno Project Explorer

V okně **Code** (obr. 3) (zpravidla největším okně, někdy nazývaném jako okno modulu) je zobrazen **zdrojový kód**. Kód je možné v okně psát, kopírovat jeho části nebo jej mazat, přesouvat jeho části jinam. Každá součást sešitu má přiřazené okno kódu.



Obr. 3 Okno kódu (Code)

K ladění makra slouží okno `Immediate` a okno `Watches`. (viz dále v textu)

2.6 Zdrojový kód

Skládá se z hlavičky makra, sledů příkazů, které jsou ve formě textu a většinou postupují po řádcích. Tyto řádky mohou být libovolně dlouhé. Kvůli lepší přehlednosti je dobré rozdělit delší příkazy do dvou a více řádků a to pomocí mezery a podtržítka.

```
ActiveSheet.QueryTables.Add(Connection:="TEXT;" & _
      Jmeno_Vstupniho_Souboru, Destination:= _
      Range("A1")).TextFileStartRow = 5
```

Naopak krátké příkazy se mohou spojovat do jedné řádky pomocí dvojtečky. Abychom předešli nepřehlednosti, užíváme tento způsob raději jen výjimečně.

```
For K = 1 To Index: Cells(1, K) = DP_Polozky(K): Next K
```

Části textu zdrojového kódu jsou pro lepší orientaci v kódu barevně odlišeny. Snadněji tak rozeznáme chyby v příkazech (červeně) a komentáře (zeleně). Programovací jazyk obsahuje tzv. klíčová slova, která nelze použít v názvech jiných částí kódu. Tato slova

mají modrou barvu. Barvy různých částí kódu jsou v editoru nastaveny standardně, dají se však upravit dle vlastních potřeb.

2.7 Komentáře

Tvoří nedílnou součást zdrojového kódu. Jsou užitečné v několika ohledech. Jednak umožňují lepší orientaci ve struktuře makra samotnému programátorovi i dalším (lze si poznamenat to, k čemu makro, příp. procedura či funkce slouží, co vykonává příkaz nebo blok příkazů, eventuálně cyklus, a rovněž oddělení částí makra libovolnými grafickými znaky, jako jsou např. tečka, pomlčka, podtržítko). Může sloužit také k vyřazení momentálně nevyužívané části zdrojového kódu.

Na začátku každého komentáře je **apostrof**. Komentář může stát sám o sobě, nebo může být uveden za nějakým příkazem. Místo apostrofu lze užít slovo **Rem**. V současnosti se tento způsob příliš nedoporučuje (využívá se zejména kvůli kompatibilitě starších zdrojových kódů s VBA). K jeho nevýhodám patří především to, že jej nemůžeme umístit za nějaký příkaz. Musí být vždy prvním slovem v řádku. Velmi dobrými pomocníky jsou příkazy **Comment Block** a **Uncomment Block**, jimiž můžeme vyřadit/aktivovat větší část zdrojového kódu najednou. Předejdeme tím mnohačetnému umísťování apostrofů.

Při psaní zdrojového kódu je výhodné odsazování řádků. Výrazně se jím totiž zvyšuje **přehlednost kódu**. K tomuto účelu slouží nejlépe tabulátor. Uplatnění uvedeného postupu se hodí nejvíce u vnořených cyklů nebo u rozhodovacích příkazů.

```
Do
      'příkazy
      If Podmínka Then
            'příkazy
            For Promenna1 = hodnota To Promenna2
                  'příkazy
            Next Promenna1
             'příkazy
      End If
      'příkazy
Loop
```

2.8 Hlavička makra

Je tvořena dvěma řádky, první se skládá ze slova `Sub`, názvu makra a dvou závorek, ve kterých může být uveden 1 či více parametrů, jde – li o proceduru nebo funkci (u procedur a funkcí je princip hlavičky stejný). Název makra musí **vždy** začínat písmenem a v jeho názvu mohou být pouze písmena, číslice a podtržítka. V názvu makra nesmějí být interpunkční znaky, jako např. mezery, tečky, čárky, pomlčky. Název je dobré volit tak, aby alespoň částečně vystihoval to, k čemu makro slouží. Makro je vždy ukončeno příkazem `End Sub`.

2.9 Procedury a funkce

Když se v makru několikrát opakuje stejný blok příkazů, je užitečné vytvořit z něj **podprogram**, který se vyvolá jedním příkazem za běhu makra nebo jiného podprogramu. Podprogram tvoří uzavřenou jednotku. Je to sled příkazů vykonávaný jako celek.

Podprogramem může být kupříkladu **procedura**. Ta provede jeden příkaz nebo sérii příkazů a skončí. Hlavička procedury se řídí stejným principem jako u makra. Začíná slovem `Sub` a končí `End Sub`.

Rozdíl makra a procedury spočívá v tom, že makro je procedura bez parametrů, kterou je možno nahrát a dá se spustit z libovolného místa Excelu i z dialogového okna Makro.

Každé makro je procedura, ale každá procedura nemusí být makrem. Pojmy makro a procedura mezi sebou často splývají.

[1]

Ručně psané procedury mohou přebírat parametry. Ty jsou důležité proto, aby byly procedury použitelné i pro proměnné, které jsou v makru obsaženy. Parametry jsou umístěny v hlavičce v závorkách za názvem procedury.

```
Sub název_procedury (parametry)
    'sled příkazů
End Sub
```

Dále jím může být **funkce.** Ta, narozdíl od procedury, navíc vrací určitou hodnotu, kterou je možno přiřadit k nějaké proměnné a lze ji dále zpracovat. Funkce někdy bývá označována jako **funkční procedura**. Hlavička funkce vypadá následovně: počíná se slovem `Function` , končí `End Funtion`.

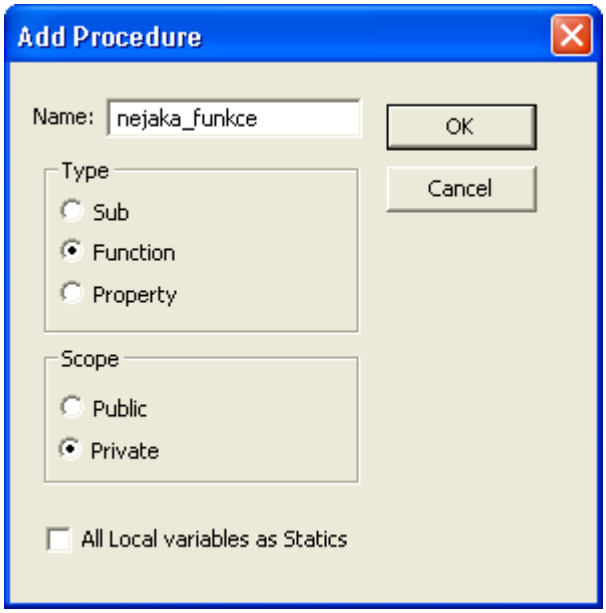
```
Function název_funkce (parametry)
     'sled příkazů, výpočtů
End Function
```

Funkce se dají vyvolat nejen makrem, ale i v Excelu ze seznamu funkcí. Před slovy `Sub` či `Function` může být slovo `Public` či `Private`. Tato slova určují rozsah platnosti a viditelnost.

Slovo `Public` znamená, že procedura či funkce je přístupná pro všechny moduly a pokud nemá parametry, je viditelná v seznamu maker. Je to tedy to samé, jako by tam toto slovo nebylo. Umístění slova `Public` programátorovi jasně signalizuje, že jde o proceduru.

Slovo `Private` znamená to, že podprogram je přístupný pouze v rámci jednoho modulu, a to v tom, ve kterém je, a není viditelný v seznamu maker.

Hlavička podprogramu, kromě toho, že se dá napsat ručně, se dá také vytvořit příkazem **insert -> procedure** (obr. 4), kde se napíše název podprogramu, vybere se, zda jde o proceduru či funkci a její rozsah a viditelnost (`Public, Private`).

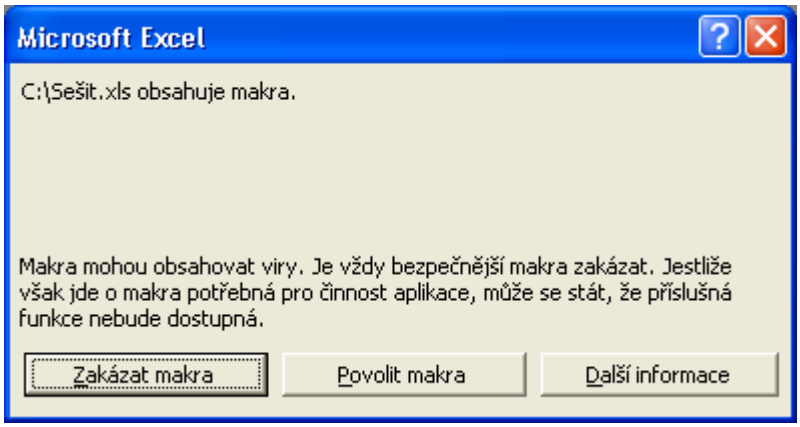


Obr. 4 Vložení hlavičky podprogramu

V podprogramech se mohou vyskytnout chyby, které se dají ošetřit. To znamená, že podprogram by měl obsahovat tzv. **chybové rutiny,** např. test na existenci souboru.

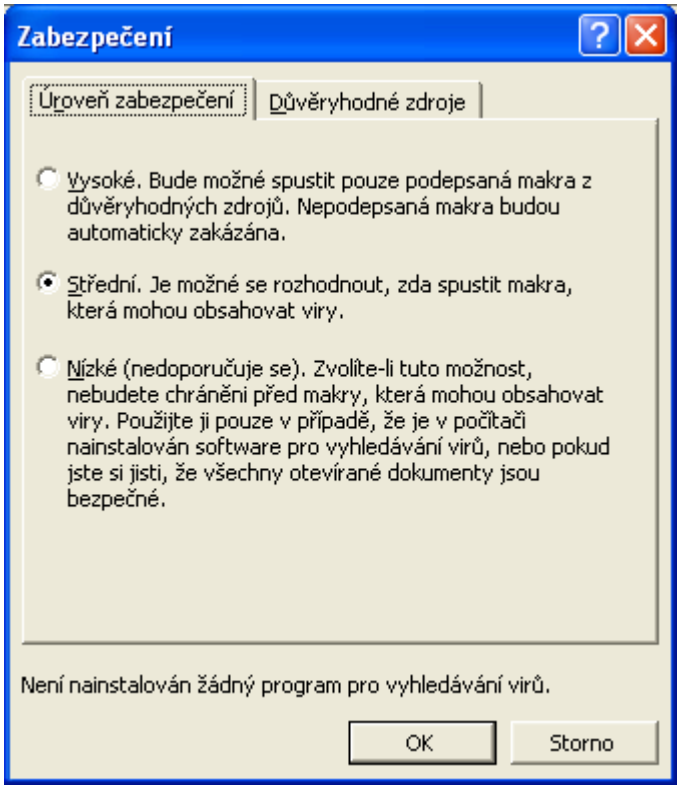
2.10 Zabezpečení maker

Při spouštění makra, jehož zdroj není známý a ověřený, může dojít k zavirování počítače, případně až k poškození systému. Proto jako ochrana slouží ruční povolení spuštění makra (obr. 5). Toto ruční povolení je v Excelu nastaveno **standardně**.



Obr. 5 Dialog pro povolení či zákaz maker při spuštění sešitu s makry

Samozřejmě existují i jiné možnosti ochrany, jež je možno nastavit v okně ***nástroje – makro – zabezpečení*** (obr. 6).



Obr. 6 Dialog pro nastavení stupně zabezpečení

- v prvním případě (***vysoké zabezpečení***) se v okně (panelu) Důvěryhodné zdroje dají nastavit cesty k adresářům bez virů/ důvěryhodné, ověřené
- v druhém případě (***střední zabezpečení***) se jedná o již zmiňované standardní nastavení. Tento způsob je bezpečnější než třetí (***nízké***) proto, že upozorní na přítomnost maker. Třetí způsob je záhodno využít v případě, pokud jsme si naprosto jistí původem a bezpečností makra, nebo pokud jde o naše vlastní makra.

V Excelu 2007 je standardně nastaven formát `.xlsx`, který nepodporuje makra. Při ukládání sešitu, který obsahuje makro, máme na výběr ze tří možností [3]:

- Sešit aplikace Excel s podporou maker (`.xlsm`)
- Binární sešit aplikace Excel (`.xlsb`)
- Sešit aplikace Excel 97- 2003 (`.xls`)

2.11 Objekty, metody, vlastnosti

Excel obsahuje položky, které nazýváme **objekty**. Tyto objekty vytvářejí hierarchii, na jejímž vrcholu je aplikace Excel. Z toho vyplývá, že aplikace Excel je sama objektem. Každý objekt může obsahovat další objekty. Např. Sešit (`Workbook`) obsahuje Listy (`Worksheets`), ty zase obsahují Buňky či Oblasti buněk (`Range`). Veškeré objekty Excelu se dají použít při vytváření maker, jedná se o tzv. **objektové programování**.

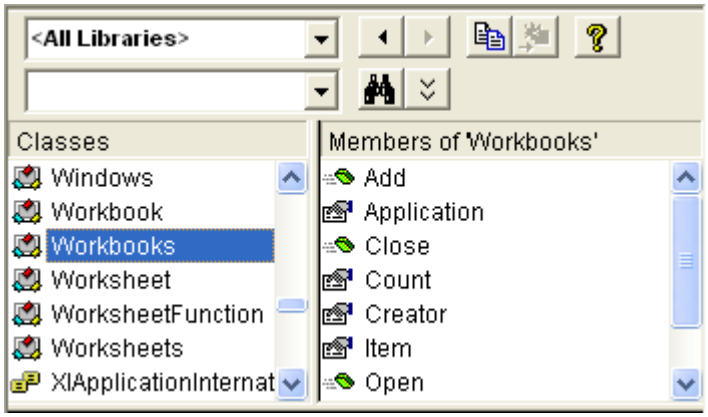
Jednotlivé objekty mají své **metody**. Jsou to činnosti, které objekt provede sám. Kupříkladu Sešit lze otevřít, zavřít, kopírovat, smazat .

Kromě metod mají objekty také své **vlastnosti**. Ty popisují vzhled nebo stav objektu. V případě objektu `Range` k nim například patří kromě jiných druh písma, barva, velikost, šířka sloupce, ohraničení.

Objekty klasifikujeme v Excelu do více než 100 tříd. Pro každou třídu je specifická určitá sada vlastností a metod. V rámci paměti počítače hovoříme o konkrétních objektech dané třídy spíše jako o instancích.

Skupiny objektů stejných tříd vytvářejí **kolekce**. Ty jsou jako takové také objekty. S celou kolekcí můžeme pracovat buď jako s objektem, nebo jako s jednotlivými objekty kolekce. “[2]

K prohlížení objektů slouží prohlížeč **Object Browser** (obr. 7). Pomocí tohoto prohlížeče pozorujeme jejich metody a vlastnosti. Můžeme to vidět na následujícím obrázku (č. 7).



Obr. 7 Prohlížeč Object Browser

[2] [2] str. 119

2.12 Datové typy

Proměnné jsou pozice v paměti počítače, které se během programu mohou měnit. Slouží k ukládání hodnot. Podle toho, jakou hodnotu ukládáme, má proměnná svůj datový typ. Tyto typy mohou být:

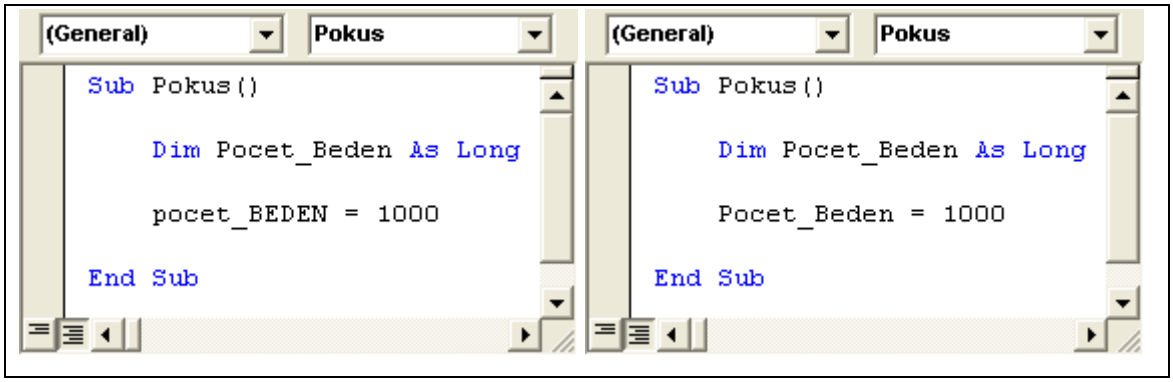
- Řetězcové - string
- Celočíslové - byte, integer, long,
- Reálné - single, double, currency, decimal
- Logické - boolean
- Datové - date
- Objektové - object
- Univerzální - variant
- Pole, Výčet - array
- Definované uživatelem

Název proměnné, stejně jako název procedury, musí začínat písmenem a nesmí začínat číslem. Nikde v názvu nesmějí být, kromě jiných, přítomny tyto znaky: , . ! # & % $, mezery a znaky s diakritikou.

Názvy proměnných mohou mít délku až 254 znaků. Z praktických důvodů používáme raději názvy kratší.

Název proměnné nesmí být stejný jako klíčové slovo. Pokud bychom toto slovo užili, editor by automaticky nahlásil chybu. Ovšem nastane i situace, kdy klíčové slovo může být vhodné pro název proměnné, např. slovo `Next`.Uvedený případ se týká pouze těch programátorů, kteří píší zdrojový kód v anglickém jazyce, protože VBA je též v angličtině (tudíž i veškerá klíčová slova). Programátor užívající český jazyk se problémům vyhne tehdy, pokud místo slova `Next` prostě použije české `Dalsi`.

VBA nerozlišuje velká a malá písmena. Vždy je lepší psát názvy tak, aby obsahovaly **velká písmena**. Napomůžeme tím jednak čitelnosti, jednak přehlednosti názvů proměnných. Pokud napíšeme proměnnou kdekoliv ve zdrojovém kódu velkým či malým písmenem, potom se velká či malá písmena automaticky upraví podle toho, jak jsme proměnnou deklarovali (obr. 8).



Obr. 8 Ukázka změny písma v názvu proměnné

Při užití tohoto způsobu snadněji rozpoznáme chyby v názvech proměnných (například ty, při kterých nedojde k žádoucím změnám písmen).

Správně zvolený název zcela jasně a zřetelně vystihuje to, k čemu se proměnná používá. Např. `Pocet_Sloupcu`, `Prumer_Trubky`, `Cyklus_Probehl`

Názvy proměnných je vhodné volit tak, aby bylo okamžitě zřejmé, ke kterému datovému typu náleží. Podle typu proměnné lze zvolit některou ze standardních předpon. Ty jsou tvořeny malými písmeny, např. `str_Jmeno_Psa` (typ string), `lCisloDomu` (typ long). Způsob užívání předpon je pro nás výhodný, neboť okamžitě poznáme, s jakou proměnnou budeme pracovat. Přehled předpon, které označují datové typy, je v tabulce č. 1.

Datový typ	Předpona
Boolean	b
Integer	i
Long	l
Single	c
Double	d
Currency	c
Date/Time	dt
String	str
Object	obj
Variant	v
Uživatelsky definovaný	u

Tab. 1 Doporučené předpony pro datové typy “[3]

[3] [2] str. 147

2.13 Deklarace proměnných

Jedná se o operaci, kdy určíme jméno, typ a platnost proměnné ještě před jejím použitím ve zdrojovém kódu. Proměnné se deklarují nejčastěji příkazy `Dim` a dále `Static`, `Private` nebo `Public`. O jejich použití více v dalším textu.

Deklarace proměnné vypadá takto:

```
Dim název_proměnné As typ_proměnné
```

Např:

```
Dim Pocet_Cestujicich As Byte
Dim I As Long
Dim J As Long
```

Nebo můžeme více proměnných deklarovat na jednom řádku:

```
Dim Pocet_Cestujicich As Byte, I As Long, J As Long
```

Když k proměnné nepřiřadíme datový typ:

```
Dim Pocet_Cestujicich
```

tehdy se k proměnné přiřadí typ Variant. Bude to tedy ten samý případ, jako:

```
Dim Pocet_Cestujicich As Variant
```

Při použití tohoto způsobu deklarace:

```
Dim I, J As Long
```

bude proměnné I také přiřazen typ Variant. Proměnná s typem Variant se mění podle hodnoty, kterou jí přiřadíme za průběhu makra. Tzn., že jí můžeme například přiřadit nejdříve číselnou hodnotu a poté řetězec.

Podle starého způsobu můžeme proměnnou deklarovat přidáním znaku za jméno proměnné (! # & % $ @).

Např:

```
Dim  Druh_peciva$
```

Tento způsob deklarace se ale nedoporučuje. Přehled znaků je v tabulce č. 2

Datový typ	Znak pro deklaraci
Inreger	%
Long	&
Single	!
Double	#
Currency	@
String	$

Tab. 2 Přehled znaků pro deklaraci datových typů "[4]

2.14 Rozsah platnosti a životnost proměnných

Použití proměnných je limitováno rozsahem jejich platnosti. Ne každý podprogram je schopen s nimi pracovat. Z tohoto hlediska rozlišujeme 3 druhy proměnných:

- **lokální proměnné**

  Jsou použitelné pouze v konkrétním makru (v proceduře, ve funkci) ohraničeném příkazy `Sub` a `End Sub` (`Function` a `End Function`). Deklarují se příkazy `Dim` či `Static`. Slovo `Static` značí druh statické proměnné, která si uchovává svoji hodnotu i po skončení procedury. Editor VBA umožňuje to, že lokální proměnná se nemusí deklarovat. V tom případě je této proměnné automaticky přiřazen typ **Variant**, což může usnadnit práci. Existují ale minimálně dva důvody, proč se mu vyhnout. Prvním je skutečnost, že typ Variant zabírá díky své univerzálnosti více operační paměti, což také zpomaluje běh makra. Je tedy lepší zvolit datový typ podle rozsahu a druhu, které proměnné přísluší. Druhý důvod je ten, že tento způsob někdy bývá zdrojem chyb.

[4] [2] str. 145

Typickým projevem je, že při chybném názvu proměnné se hodnota, kterou chceme dané proměnné přiřadit, přiřadí do chybného názvu proměnné. Uvedená proměnná se vytvoří jako nová s typem Variant. Správné hodnotě nebyla tato proměnná přiřazena a proto program nebude pracovat správně. Zmíněná chyba může nastat i tehdy, když proměnné deklarujeme. Probíranému problému se můžeme vyhnout tehdy, jestliže na začátku modulu napíšeme příkaz `Option Explicit`. Tím pádem jsme nuceni každou proměnnou deklarovat, protože pokud se pokoušíme spustit makro s nedeklarovanou proměnnou (proměnnou s chybným názvem), tak editor hlásí chybu. V editoru se dá také nastavit to, aby se příkaz `Option Explicit` objevil na začátku každého nově vytvořeného modulu.

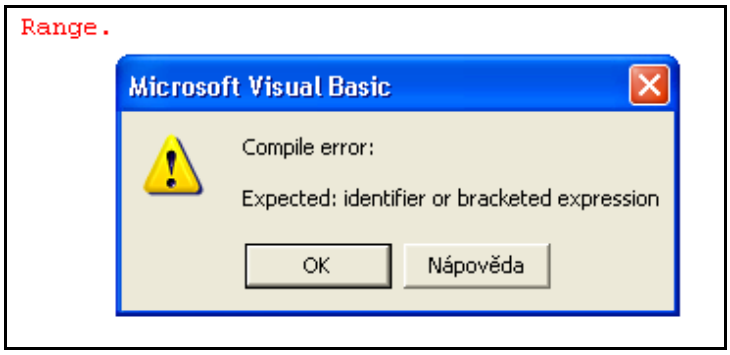
- **modulové proměnné**
  Musejí být vždy deklarovány a mohou být používány pro celý modul. Deklarují se příkazy `Dim` či `Private`.

- **veřejné (globální) proměnné**
  Jsou k použití ve všech modulech, tj. v celém sešitu. Deklarujeme je příkazem `Public`.
  [1]

2.15 Chyby ve zdrojovém kódu

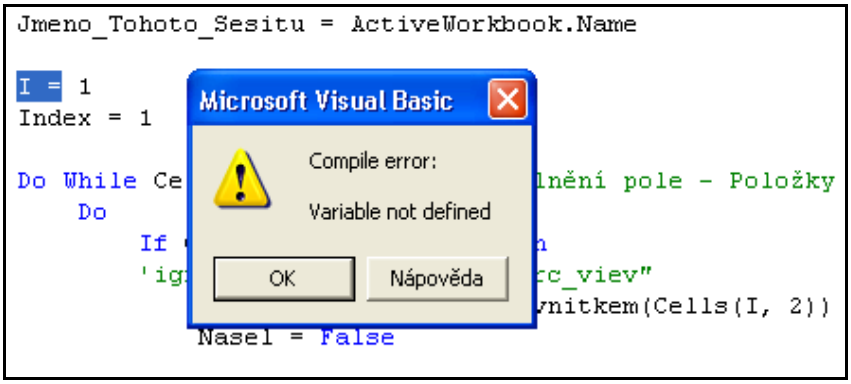
Při vytváření maker (zejména těch složitějších) se téměř žádný programátor nemůže beze zbytku vyhnout všem chybám. Ovšem editor je umí odhalovat:

- při chybně napsaném (či neúplně napsaném) příkazu editor hlásí chybu okamžitě a to zčervenáním řádku a zobrazením dialogového okna, v němž je popsán druh chyby (obr. 9).



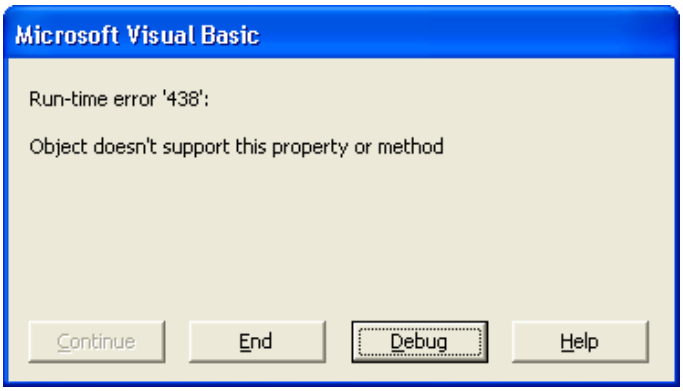
Obr. 9 Okamžité hlášení chyby při nepřidělení vlastnosti k objektu Range

- při pokusu o spuštění makra (obr. 10)



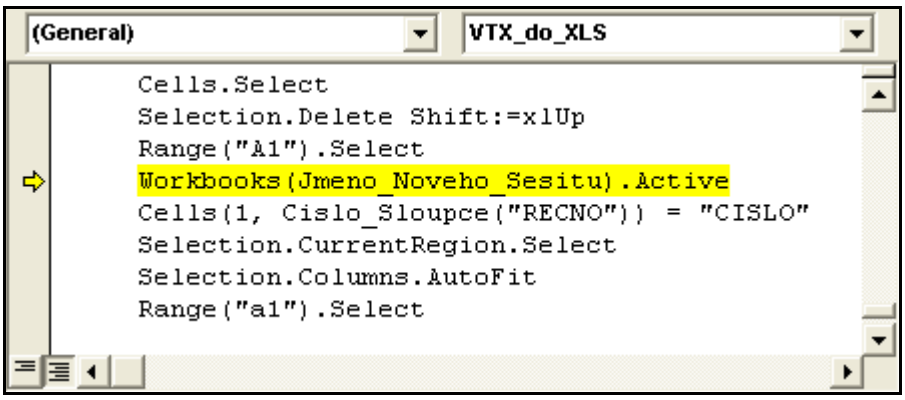
Obr. 10 Hlášení chyby u nedeklarované proměnné při pokusu o spuštění makra

- při běhu makra (opět zobrazení dialogového okna s popisem chyby – např. u nesprávně zadaných parametrů, nebo u špatně přidělených vlastností či metod objektům), jako na obr. 12, kde místo chybně zadané metody - slovo `Active`, má být správně slovo `Activate`.



Obr. 11 Hlášení chyby pro špatně zadanou metodu

V případě nahlášení chyby máme možnost makro buď ukončit, nebo si necháme zobrazit chybný řádek (obr. 12). Ten můžeme případně hned opravit a makro nechat pokračovat.



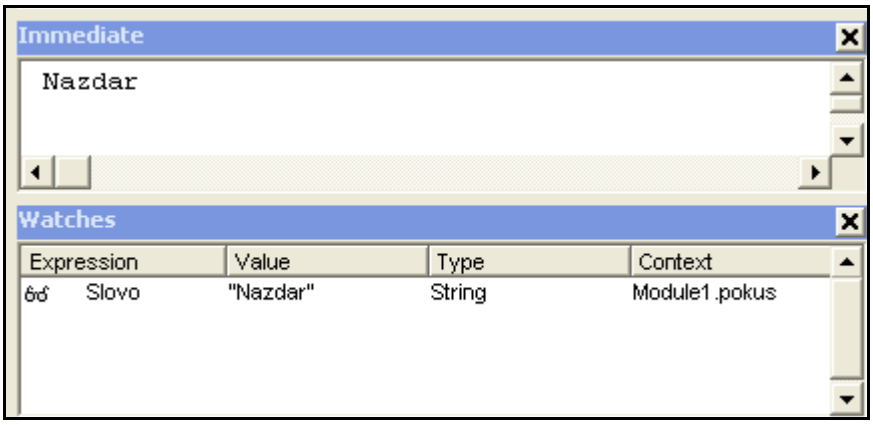
Obr. 12 Následné zobrazení chybného příkazu

2.16 Ladění maker

Kromě výše zmíněných chyb existují v rámci zdrojového kódu také takové, které editor VBA odhalit neumí. V tom případě makro proběhne normálně, ale výsledky budou chybné. Proto je potřeba chyby odhalovat manuálně. Lze to učinit několika způsoby. Jedním z nich je tzv. **krokování**. Jedná se o sledování průběhu makra po jednotlivých příkazech. Krokovat můžeme příkazem **Step Into** (F8). Dojde k tomu, že krokování probíhá v celém makru a také ve všech procedurách volaných tímto makrem. Pokud chceme aktivní proceduru ukončit bez krokování, použijeme **Step Out** (Ctrl + Shift + F8). Metoda **Step Over** (Shift + F8) chápe proceduru jako příkaz a neprochází ji po krocích. [3]

Během krokování můžeme sledovat hodnoty proměnných v okně **Watches**. Ty se do něj nechají uvést před i v průběhu krokování. Hodnotu proměnné můžeme dále zjistit tak, že na ni najedeme kurzorem. Potom se pod proměnnou objeví okénko s aktuální hodnotou proměnné.

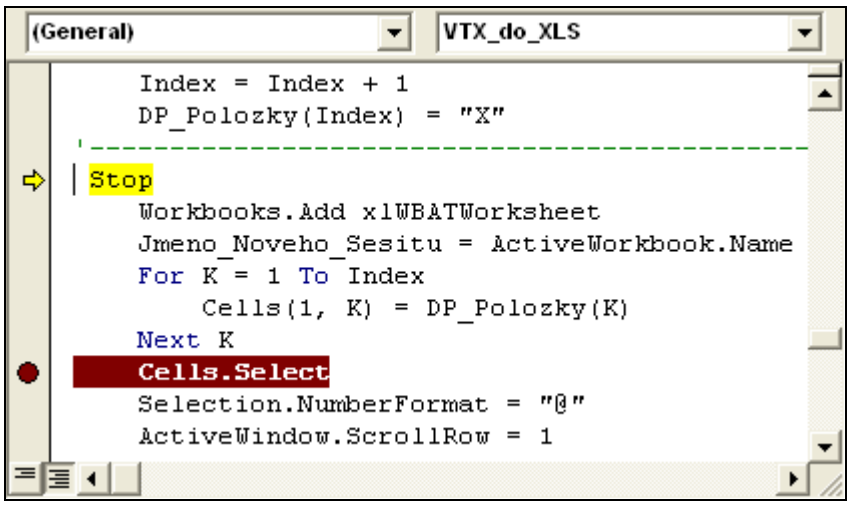
Známe i možnost sledování proměnné pomocí okna **Quick Watch** a okna **Immediate**. U naposledy zmíněného okna napíšeme do zdrojového kódu příkaz `Debug.Print` a za něj do závorky uvedeme název proměnné, např. `Debug.Print(I)`. Po vykonání tohoto příkazu se v okně Immediate zobrazí hodnota proměnné.



Obr. 13 Okna `Immediate` a `Watches`

Pokud chceme zjistit hodnotu proměnných na určitých zvolených místech ve zdrojovém kódu, můžeme na těchto místech běh pozastavit a přečíst si hodnoty proměnných, případně také sledovat chování makra (stejně jako u krokování). Pozastavení dosáhneme těmito způsoby [3]:

- **Run To Cursor** (průběh makra se zastaví na pozici kurzoru)
- napsáním příkazu `Stop` do zdrojového kódu (obr. 14)
- umístěním zarážky pomocí příkazu **Toggle Breakpoint** (F9) (obr. 14)
- umístěním podmíněné zarážky (makro se zastaví, má-li proměnná zadánu určitou hodnotu)



Obr. 14 Zastavení běhu makra příkazem `Stop` a pomocí zarážky

2.17 Pomocníci při psaní zdrojového kódu

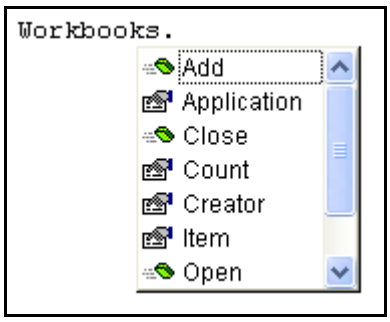
Podstatnou součástí Editoru VBA jsou pomocníci. O některých z nich jsme se zmínili již dříve (změna písmen v názvech proměnných, barevné rozlišení ve zdrojovém kódu).

VBA kromě toho, že mění malá či velká písmena v názvu proměnné způsobem, jenž je uveden v deklaraci, také automaticky změní název klíčového slova nebo u názvu objektu na tvar s prvním velkým písmenem a s dalšími malými. Poznáme tak, zda je klíčové slovo napsáno správně. Např.:

`while`, nebo `wHIlE` se změní na `While`

Editor VBA je schopen tzv. **automatického dokončování**. Např. pokud napíšeme příkaz `Sub` a název makra, napíše se na konec hlavičky automaticky `End Sub`.

Kvůli časové úspoře a pro správnost znění příkazu využíváme automatické nabídky možností při psaní příkazů. Například když deklarujeme proměnné, editor nám automaticky nabídne seznam všech datových typů. Při práci s objekty se automaticky zobrazí seznam všech vlastností a metod k danému objektu (obr. 15).

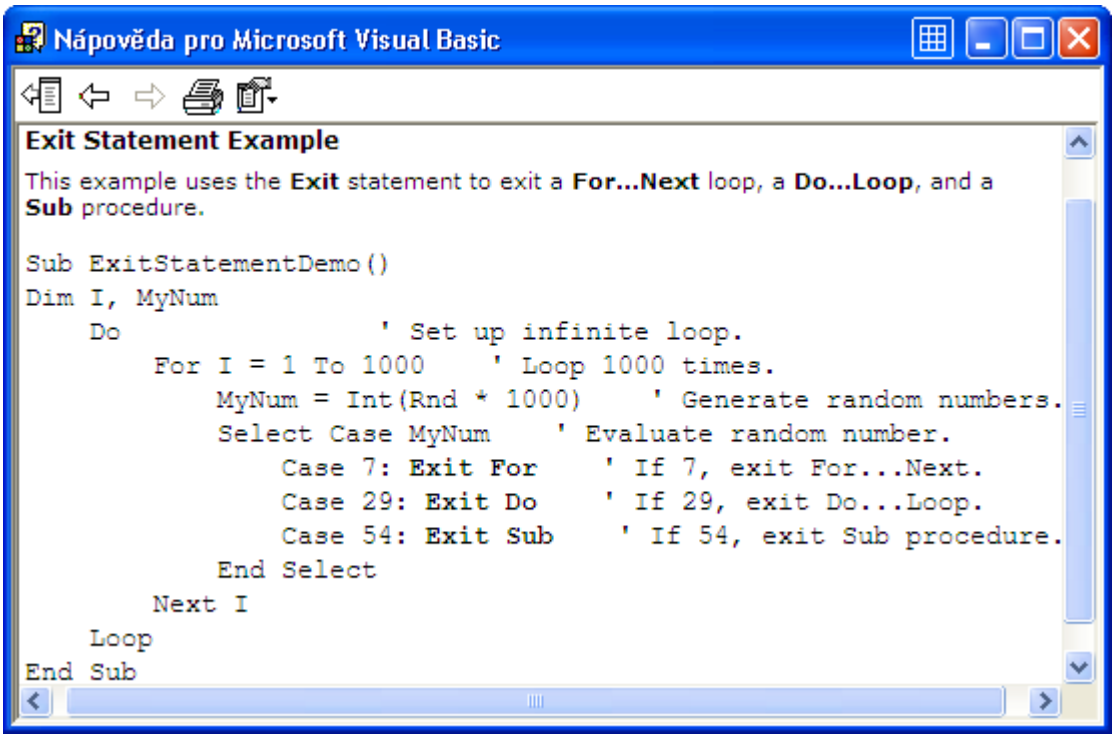


Obr. 15 Nabídka metod k objektu `Workbooks`

2.18 Nápověda

Jedná se o nástroj, který nám rychle zprostředkuje informace o tom, jak psát zdrojový kód (např. práce s proměnnými, psaní rozhodovacích příkazů, cyklů, práce s objekty atd.).

Pakliže chceme zjistit informace o klíčovém slovu, posuneme kurzor na klíčové slovo a stiskneme F1. Následně se objeví požadovaný popis nebo jiné údaje. Nápověda obsahuje také několik ukázek, které lze zkopírovat do schránky a použít je ve zdrojovém kódu. Na obr. 16 vidíme příklad použití nápovědy pro klíčové slovo `Exit`.



Obr. 16 Názorná ukázka nápovědy (způsoby použití příkazu `Exit`)

3 Porovnání možností programování v MS Excel a OpenOffice.org Calc

3.1 Co je OpenOffice.org a kde se využívá

OpenOffice.org (dále OOo) je možné považovat za alternativu MS Office. Jedná se o balík kancelářských aplikací srovnatelné kvality. Licence je přístupná zdarma, i proto jej můžeme používat na více počítačích v jakémkoli prostředí. [4]

Uplatnění OOo je velmi široké. Setkat se s ním můžeme např. v kancelářích, ve školách, ve firmách, v našem regionu jej aktuálně využívá např. Jihočeská vědecká knihovna v Českých Budějovicích. Vzhledem k jeho dostupnosti může být ideální také pro domácnosti a studenty.

Rozlišujeme 5 základních aplikací OOo, které pracují s textovými, grafickými nebo tabulkovými dokumenty. Jeho součástmi jsou [5]:

- **OOo Writer** - textový editor. Zpracovávání textových dokumentů, vkládání a úprava obrázků do těchto dokumentů, tvorba HTML stránek. Obdoba MS Word. Dokument Writeru má příponu **odt.**
- **OOo Calc**- tabulkový procesor. Zpracovává tabulky a grafy, provádí výpočty, analyzuje informace. Obdoba MS Excel. Sešit má příponu **ods**.
- **OOo Impress** - program pro vytváření a prohlížení prezentací při školeních, poradách, přednáškách či pro webové stránky. Obdoba MS Power Point. Přípona **odp**.
- **OOo Draw** - program pro kreslení a editování obrázků, blokových schémat, log. Přípona **odg**.
- **OOo Basse** – program pro zpracovávání databází. Přípona **odb**.

Noví uživatelé OOo jsou ve výhodě oproti těm, kteří k němu přecházejí od MS Office. Oba balíky dokáží naplnit stejné cíle, ovšem k některým se dojde různými způsoby. Toto tvrzení může podpořit např. způsob vytváření maker. O možných úskalích pojednáme v následujících kapitolách. [5]

3.2 Rozdíly ve vytváření a programování maker

Základní rozdíl při tvorbě maker spočívá v možnosti užít různý počet programovacích jazyků.

Excel používá VBA for Application, popř. Microsoft Script Edior. OOo nabízejí k programování maker hlavně OpenOffice.org Basic (dále OOoB) (dříve označovaný též jako Star Basic). Kromě toho mohou používat také další jazyky: Python, BeanShell, JavaScript.

Z uvedeného vyplývá, že oba programy pracují s modifikacemi jednoho jediného jazyka, kterým je Basic.

3.3 Rozdíly v záznamu, ukládání a spouštění maker

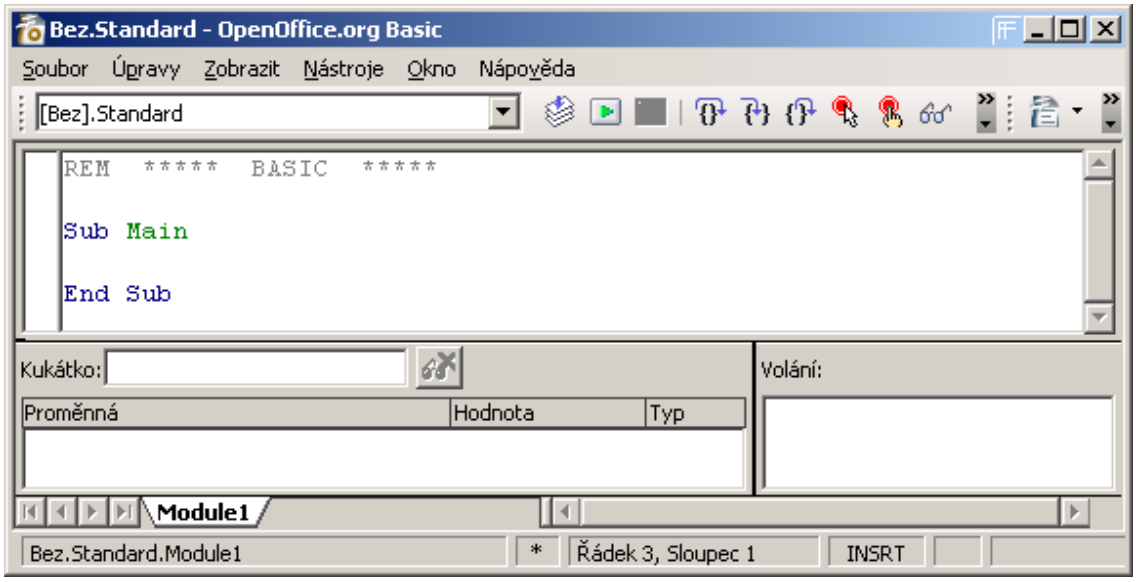
Při nahrávání makra v OOo určíme název makra a místo, kam bude uloženo, při ukončení nahrávání. Naproti tomu Excel nabízí tyto úkony ještě před samotným nahráváním makra a navíc poskytne přiřazení klávesové zkratky k nahrávanému makru.

Při ukládání maker v MS Excel jsou moduly obsaženy přímo v sešitu. Oproti tomu v Oppen Office.org jsou moduly ještě navíc uloženy v knihovnách, které jsou automaticky pojmenovány Standard (1, 2, 3 atd.). V Calcu můžeme pojmenování knihoven i modulů libovolně měnit.

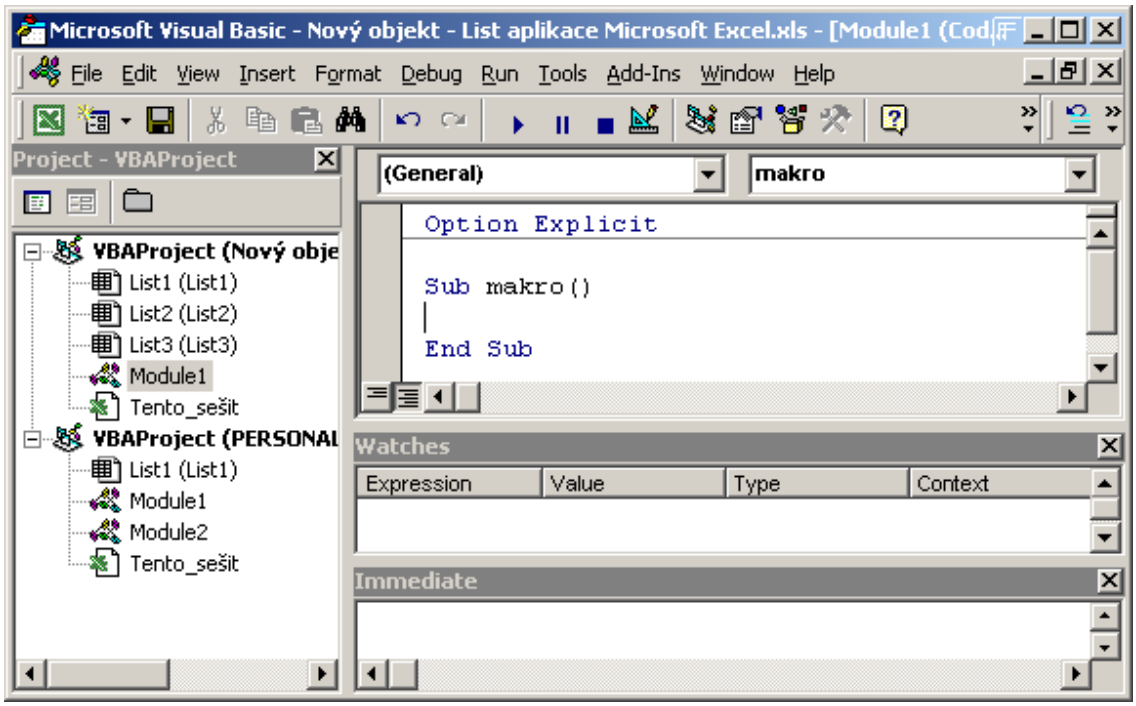
Spouštění maker v Calcu probíhá obdobně jako v MS Excel.

3.4 Rozdíly v editorech OOoB a VBA

Zmíněné editory se na první pohled liší umístěním panelů, ikon, nástrojů aj., což demonstrují obrázky 17 (OOoB) a 18 (VBA). Na obrázku 17 nalézáme například modul v dolní části pod oknem kódu. Má podobu záložky a v případě, kdy chceme zobrazit jiný modul, klikneme na jinou záložku. Oproti tomu v editoru VBA je modul umístěn v okně VBA Project v podobě stromové struktury. Chceme-li v OOoB sledovat hodnoty proměnných, použijeme k tomu **Kukátko**. Vlevo od Kukátka je umístěno okno Volání, které zobrazuje název momentálně spuštěného makra. Ve VBA plní funkci obdobnou Kukátku okno **Watches**.



Obr. 17 Editor OpenOffice.org Basic



Obr. 18 Editor VBA

Výbornou vlastností OOoB je to, že je, stejně jako celý OOo kompletně i s nápovědou, ve více jazykových verzích (cca 30) včetně češtiny [4]. Naproti tomu editor VBA for Application výhodu překladu do češtiny nenabízí. Výjimku tvoří starší verze z roku 1997 [1].

OOoB nepodporuje automatické dokončování, tzn. že například u klíčových slov se počáteční písmena automaticky neopravují na velká. Dále neupravuje velká a malá písmena v názvech proměnných podle předchozí deklarace. Také při psaní procedury se hlavička sama neukončí.

V tomto editoru pozorujeme méně možností při ohlašování chyb. Při nedokončeném příkazu a následným kliknutím do jiného místa ve zdrojovém kódu nedochází (jako ve VBA) k objevení dialogového okna a ani ke zčervenání nedokončeného příkazu. Jestliže příkaz nedokončíme záměrně (např. proto, že do něho chceme kopírovat část zdrojového kódu), je pro nás výhodou, že se okno obsahující hlášení chyby neobjeví. Tím pádem nemusíme ztrácet čas jeho odstraňováním. Při upravování textu v OOoB je nevýhodou to, že nemůžeme např. pro kopírování, vkládání či vyjímání textu použít pravého tlačítka myši, ale musíme postupovat přes hlavní menu, nebo klávesovými zkratkami.

Chceme-li ve VBA spustit makro přímo v editoru (ať už chceme krokovat či spustit celé makro), spustí se nám makro, ve kterém je umístěn kurzor. Naproti tomu v OOoB se nám spustí vždy první makro v modulu.

3.5 Rozdíly ve zdrojovém kódu OOoB a VBA

OOo jsou s kompatibilní s MS Office. OOo dokáže číst soubory MS Office. Názorně to lze popsat takto: OOo Calc umí číst sešity MS Excel (soubory xls). OOo Writer dokáže přečíst dokument MS Word (soubory doc). OOo Impresse umí číst prezentace programu MS Power Point (soubory pps, ppt.).

Přestože je OOo Calc schopný číst excelovské pracovní sešity, jeho kompatibilita nefunguje pro makra vytvořená ve VBA, ačkoli mají oba programovací jazyky společnou syntaxi. Důvod, proč makra vytvořená ve VBA nepracují v OOoB, je způsoben rozdíly v objektových modelech pro Excel a Calc. Objekty, metody a vlastnosti užívají odlišná jména a odpovídající chování je někdy mírně odlišné.“[5]

Hlavní rozdíl mezi oběma objektovými modely spočívá v tom, že excelovský model nevyužívá všechny rysy, které vytvářejí objektově orientované programové prostředí. V některých publikacích Microsoftu je objektový model pro jeho produkty,

[5] [6] Překlad str. 2

takové jako Excel, označován za „jako-objektový“. Ve skutečném objektově orientovaném programovacím modelu existuje koncept převzetí. Tento koncept dovoluje, aby definice a realizace jednoho objektu byla založena na definici a realizaci jiného objektu. Jako-objektový model Microsoftu nepodporuje převzetí. Abychom ilustrovali převzetí, uvažujme následující příklad. Existuje objekt nazývaný „shape“ (tvar) s metodou nazvanou „move (přesun)“, která pohybuje „shape“ dokola na displeji. Ve skutečném objektově orientovaném programovacím prostředí nový objekt nazvaný „circle“ (kruh), který je typu shape, může být uveden následujícím způsobem. Místo toho, aby byl circle nucen použít svou vlastní metodu move pro pohyb dokola na displeji, circle objekt převezme metodu move od objektu shape.“[6]

3.6 Přenos maker z VBA do OOoB

Jestliže je makro vytvořeno bez odkazů na objekty (buňky, listy), nebo pokud jsou vstupní údaje zadávány např. funkcí InputBox, pak makro funguje jak v MS Excel, tak v OOo Calc. Ukázkové makro č. 1 je možno zkopírovat bez jakékoliv úpravy z VBA do OOoB. Toto makro nám přepočítává suroviny pro přípravu ovocných vín podle toho, jaké zadáme celkové množství vína, dané např. velikostí demižonu. Druh a Množství se zadávají funkcí InputBox a pomocí funkce MsgBox je zobrazen výsledek. Modul obsahuje 16 procedur, které představují druh vína. Každá z těchto procedur obsahuje hodnoty potřebných ingrediencí na 1 litr. Podle zvoleného druhu je volána jedna z těchto procedur a poté dochází k přepočtu surovin na zadané celkové množství. Přesto, že makro 1 po zkopírování z VBA do OOoB funguje, je tomu tak pouze navenek. Některé příkazy totiž fungují jinak, nebo vůbec ne. Např. v seznamu maker se objeví všechny procedury, přestože jsou deklarovány jako privátní. Tzn. že slovo Private je ignorováno. Také ošetření chyby ve funkci Mnozstvi je v OOoB zcela zbytečné, neboť při zadání chybné hodnoty množství (např. když místo čísla zadáme řetězec), nedochází v OOoB vlivem chyby k přerušení makra, ale funkce InputBox vrací nulu.

[6] [6] Překlad str. 2

```
Option Explicit
'_____________________________________________________________________________
Dim A As Single, Stava As Single, Cukr As Single
Dim Cukr1 As Single, Voda As Single, Ovoce As Single
Dim Z As String, Druh As String
'_____________________________________________________________________________

Private Sub Rybiz()
    Druh = "R I B Í Z O V É"
    Stava = 0.35
    Cukr = 0.225     'mnozstvi surovin na 1 litr
    Voda = 5 / 10
End Sub
'_____________________________________________________________________________

Private Sub Angrest()
    Druh = "A N G R E Š T O V É"
    Stava = 5 / 10
    Cukr = 0.225
    Voda = 0.35
End Sub
'_____________________________________________________________________________

Private Sub Tresne()
    Druh = "T Ř E Š Ň O V É"
    Stava = 7 / 10
    Cukr = 0.175
    Voda = 0.2
End Sub
'_____________________________________________________________________________

Private Sub Visne()
    Druh = "V I Š Ň O V É"
    Stava = 4 / 10
    Cukr = 0.225
    Voda = 0.45
End Sub
'_____________________________________________________________________________

Private Sub Boruvky()
    Druh = "B O R Ů V K O V É"
    Stava = 6 / 10
    Cukr = 2 / 10
    Voda = 3 / 10
    Ovoce = 0.75
    Cukr1 = 0.225
End Sub
'_____________________________________________________________________________

Private Sub Jablka()
    Druh = "J A B L E Č N É"
    Stava = 8 / 10
    Cukr = 0.175
    Voda = 0.1
End Sub
'_____________________________________________________________________________

Private Sub Hrusky()
    Druh = "H R U Š K O V É"
    Stava = 8 / 10
    Cukr = 0.175
    Voda = 0.1
End Sub
'_____________________________________________________________________________

Private Sub Ostruziny()
```

```
    Druh = "O S T R U Ž I N O V É"
    Stava = 0.55
    Cukr = 0.225
    Voda = 0.3
    Ovoce = 0.7
    Cukr1 = 0.25
End Sub
'_____________________________________________________________________________

Private Sub Maliny()
    Druh = "M A L I N O V É"
    Stava = 0.55
    Cukr = 0.225
    Voda = 0.3
    Ovoce = 0.35
    Cukr1 = 0.25
End Sub
'_____________________________________________________________________________

Private Sub Sipky()
    Druh = "Š Í P K O V É"
    Ovoce = 0.3
    Cukr = 0.25
End Sub
'_____________________________________________________________________________

Private Sub Trnky()
    Druh = "T R N K O V É"
    Ovoce = 0.35
    Cukr = 0.25
End Sub
'_____________________________________________________________________________

Private Sub Jerabiny()
    Druh = "J E Ř A B I N O V É"
    Ovoce = 0.3
    Cukr = 0.25
End Sub
'_____________________________________________________________________________

Private Sub Sussipky()
    Druh = "Z E   S U Š E N Ý C H   Š Í P K Ů"
    Ovoce = 0.1
    Cukr = 0.25
End Sub
'_____________________________________________________________________________

Private Sub Susovoce()
    Druh = "Z E   S U Š E N É H O   O V O C E"
    Ovoce = 0.15
    Cukr = 0.225
End Sub
'_____________________________________________________________________________

Private Sub Povidla()
    Druh = "Z   P O V I D E L"
    Ovoce = 0.1
    Cukr = 0.2
End Sub
'_____________________________________________________________________________

Private Sub Chleba()
    Druh = "Z   C H L E B O V Ý C H   K Ů R E K"
    Ovoce = 0.06
    Cukr = 0.25
End Sub
'_____________________________________________________________________________
```

```
Private Function Mnozstvi() 'As Single()
Dim Pom As Single
    On Error Resume Next
    Pom = InputBox("Zadej množství vína v litrech:")
    If Err <> 0 Or Pom <= 0 Then
        Pom = -1
    End If
    Mnozstvi = Pom
End Function
'_____________________________________________________________________________

Sub Priprava_vina1()
'program prepocita mnozstvi ovoce, nebo ovocne stavy, cukru a vody
'podle zdaného množství vína, nabo velikosti nádoby
Dim Kvasinky As Single, Sul As Single

Kvasinky = 0.1
Sul = 0.1
'---------------menu, výzva k volbě druhu-----------------------
Z = InputBox("rybízové - R, angreštové - A, třešňové - T, " _
 & "višnové - V, borůvkové - B,  jablečné - J, hruškové - H," _
 & " ostružinové - O,  malinové - M, šípkové - S," _
 & " trnkové - N, jeřabinové - I,  ze sušených šípků - U," _
 & " ze sušeného ovoce - C, z povidel - P," _
 & "  z chlebových kůrek - L", "Zadej druh vína:")
Z = UCase(Z)
Select Case Z                   'podle druhu získá skokem
    Case "R": Rybiz             'do procedury ingredience na 1l
    Case "A": Angrest
    Case "T": Tresne
    Case "V": Visne
    Case "B": Boruvky
    Case "J": Jablka
    Case "H": Hrusky
    Case "O": Ostruziny
    Case "M": Maliny
    Case "S": Sipky
    Case "N": Trnky
    Case "I": Jerabiny
    Case "U": Sussipky
    Case "C": Susovoce
    Case "P": Povidla
    Case "L": Chleba
    Case Else
        MsgBox "Druh byl zvolen chybně, prgram bude ukončen."
        Exit Sub
End Select

A = Mnozstvi
If A = -1 Then
    MsgBox "Množství bylo zadáno chybně, prgram bude ukončen."
    Exit Sub
End If

Stava = Stava * A 'přepočet na zadané množství
Cukr = Cukr * A
Cukr1 = Cukr1 * A
Voda = Voda * A
Ovoce = Ovoce * A
Kvasinky = Kvasinky * A
Sul = A * Sul
'-------------zobrazení výsledku v informačním okně-----------
Select Case Z
Case "B", "O", "M"
    MsgBox "Šťáva: " & Stava & " l, Cukr: " & Cukr & _
    " Kg, Voda: " & Voda & " l, kvasinky: " & Kvasinky & _
    " bal., zivna sul: " & Sul & " bal." & _
    "  nebo  " & _
```

```
    "Ovoce: " & Ovoce & " Kg, Cukr: " & Cukr1 & _
    " Kg, Voda: doplhit, kvasinky: " & Kvasinky & _
    " bal., zivna sul: " & Sul & " bal.", , Druh & _
    "  na  " & A & " l"
Case "R", "A", "T", "V", "J", "H"
    MsgBox "Šťáva: " & Stava & " l, Cukr: " & Cukr & _
    " Kg, Voda: " & Voda & " l, kvasinky: " & Kvasinky & _
    " bal., zivna sul: " & Sul & " bal.", , Druh & _
    "  na  " & A & " l"
Case "S", "N", "I", "U", "C", "P", "L"
    MsgBox "Ovoce: " & Ovoce & " Kg, Cukr: " & Cukr & _
    " Kg, Voda: doplhit, kvasinky: " & Kvasinky & _
    " bal., zivna sul: " & Sul & " bal.", , Druh & _
    "  na  " & A & " l"
End Select

End Sub
```

Ve většině případů se s makrem č. 1 nesetkáme. Jeho existence v rámci jiné aplikace by byla zbytečná, neboť může být vytvořeno jako samostatný program. Téměř všechna makra ale používají odkazy na objekty, protože zpracovávají například tabulky. Kvůli tomu, že objekty jsou různé, nelze makra mezi sebou přenášet. V případě, kdy chceme makra přenést, musíme zdrojový kód upravit. Významným pomocníkem by nám mohl být internetový převodník, který je k nalezení na http://www.business-spreadsheets.com/vba2oo.asp. Převody ovšem bohužel nebývají zcela dostatečné a musíme je dokončit ručně.

Jako názorná ukázka rozdílných způsobů zápisu některých příkazů nám poslouží následující příklady [6, 7]:

aktivace buňky:

OOoB

```
ThisComponent.getCurrentController.select(ThisComponent. _
getCurrentController.getActiveSheet.getCellRangeByName("E10")
```

VBA

```
Range("E10").select
```

OOoB

```
ThisComponent.getCurrentController.select(ThisComponent. _
getCurrentController.getActiveSheet.getCellByPosition(1,4))
```

VBA

```
Cells(5,2).select
```

zápis do buňky:

OOoB

```
ThisComponent.getCurrentController.getActiveSheet. _
GetCellbyposition(1,9).value=100
```

VBA

```
Cells(10,2)=100
```

čtení z buňky:

OOoB

```
slovo = ThisComponent.getCurrentController.getActiveSheet. _
GetCellbyposition(1,9).string
```

VBA

```
slovo = Cells(10,2)
```

Nyní uvedeme 2 makra (č. 2 a č. 3) pro porovnání. Jedno z nich je vytvořené ve VBA a druhé v OOoB. Obě plní tentýž úkol, ale na rozdíl od makra č. 1 nejsou spolu kompatibilní. Důvodem je to, že používají odkazy na objekty. Budeme sledovat to, jak makra zapisují do buněk celá náhodná čísla. Pomocí vstupních oken InputBox zadáváme dolní mez, horní mez a počet generovaných čísel. Čísla se zapisují do sloupce od první buňky. Jestliže uživatel zadá více čísel, než je největší možný počet řádků, začnou se čísla zapisovat do dalšího sloupce znovu od prvního řádku. Pokud je počet generovaných čísel větší než je celkový počet buněk, tak makro zmenší počet čísel na největší možný počet čísel. Při pohledu na oba zdrojové kódy je patrný nejen rozdílný vzhled (barva písma, malá a velká písmena), ale i různý způsob zápisu generovaného čísla do buňky. V modulu makra č. 3 se navíc v horní části nachází vyřazená procedura (`ladeni`), pomocí které můžeme během editace volat různé procedury/funkce nacházející se v tomtéž modulu, chceme-li je např. krokovat. Důvod je ten, že editor OOoB nám spouští vždy první proceduru v modulu, ne však tu, ve které se nachází kurzor.

Makro č. 2 Generátor náhodných čísel (VBA)

```
Option Explicit
'_____________________________________________________________________
```

```
Private Function Nahodne_Cele_Cislo _
 (Dolni_mez As Long, Horni_mez As Long) As Long
    Nahodne_Cele_Cislo = _
     Int((Horni_mez - 1) - Dolni_mez + 1) _
     * Rnd + Dolni_mez
End Function
'_____________________________________________________________________

Sub Generator_NC()

Dim Hor_M As Long, Dol_M As Long, A As Long
Dim Pocet As Long, I As Long, J As Byte, K As Long
Const Max = 16777216 'počet všech buněk v listu

Dol_M = InputBox("Zadej dolní mez")
Hor_M = InputBox("Zadej horní mez")
Pocet = InputBox("Zadej počet generovaných čísel max. " & Max)
If Pocet > Max Then
    Pocet = Max
End If
J = 0
I = 1
For K = 1 To Pocet
    If I = 65537 Then
        I = 1
        J = J + 1
    End If
    A = Nahodne_Cele_Cislo(Dol_M, Hor_M)

    'zápis hodnoty do buňky
    Cells(I, J + 1).Value = A
    I = I + 1
Next K

End Sub
```

Makro č. 3 Generátor náhodných čísel (OOoB)

```
REM  *****  BASIC  *****
option explicit
'_____________________________________________________________________

'private sub ladeni
'dim a as long
       'Generator_NC
       'a = nahodne_cele_cislo(1,100)
'end sub
'_____________________________________________________________________

private function Nahodne_Cele_Cislo _
 (Horni_mez as long, Dolni_mez as long) as long

      nahodne_cele_cislo = _
       int((horni_mez - 1) - dolni_mez + 1) _
       * rnd + dolni_mez
end function
'_____________________________________________________________________

sub Generator_NC

dim Hor_M as long, Dol_M as long, A as long
dim pocet as long, I As Long, J As byte, K as long
dim dokument, stav, list, bunka
const Max = 16777216 'počet všech buněk v listu
```

```
dol_m = inputbox ("Zadej dolní mez")
hor_m = inputbox ("Zadej horní mez")
pocet = inputbox ("Zadej počet generovaných čísel max. " & max )
if pocet > max then
    pocet = max
end if

'získání objektu list [7]
dokument = ThisComponent
stav = dokument.getCurrentController
list = stav.getActiveSheet

j = 0
i = 0
for k = 1 to pocet
    if i = 65536 then
        i = 0
        j = j + 1
    end if
    a = nahodne_cele_cislo(dol_m,hor_m)

    'získání objektu buňka [7]
    bunka = list.getCellbyposition(j,i)

    'zápis do buňky [7]
    bunka.value = a

       'způsob zápisu bez použití proměnných  dokument, stav,
        'list, bunka
       'ThisComponent.getCurrentController.getActiveSheet. _
         'getCellbyposition(j,i).value = a

       i = i + 1
    next k
end sub
```

3.7 Závěr

Z komparace výše zmíněných rozdílů vyplývá, že MS Excel má více pomůcek v editoru VBA (např. odhalování a opravy chyb, automatické dokončování), jeho zdrojový kód je jednodušší a tudíž je práce s tímto editorem pohodlnější. Je tedy nasnadě, že je jeho užívání výhodnější i pro začátečníky. V případě, že programátor dobře ovládá jazyk OOoB, může v něm úspěšně programovat jako ve VBA. Co se týče programování, poskytují oba editory přibližně stejné možnosti.

4 VYTVÁŘENÍ VLASTNÍCH FUNKCÍ A JEJICH VKLÁDÁNÍ DO SEZNAMU

4.1 Úvod

Microsoft Office Excel nabízí více než 300 předdefinovaných funkcí [2], které např. při vytváření tabulek pokryjí potřeby většiny uživatelů. Přesto ale může nastat situace, kdy nemáme při naší práci k dispozici vhodnou funkci. Proto je dobré naučit se vyvinout a využívat vlastní uživatelské funkce. Jejich vytvořením můžeme dosáhnout velkého zjednodušení práce a zkrácení vzorců v tabulkách Excelu. V porovnání s vestavěnými funkcemi budou ty naše o něco pomalejší, ovšem jedná se řádově o několik vteřin a to bude pro nás zanedbatelná položka.

Vlastní funkce se (stejně jako makra) vytvářejí v jazyku VBA. Výstupem proběhnuvší funkce není provedená akce, nýbrž hodnota. Např. nelze pomocí funkce provádět činnosti s objekty (tj. formátování buněk, otevírání souborů atp.). Pokus o napsání takovéto funkce končí neúspěchem, neboť funkce vždy vrací hodnotu. [2]

4.2 Provádění funkce

Funkce lze, narozdíl od procedur, provádět pouze dvěma způsoby, a to voláním funkce v makru, podprogramu nebo použitím funkce ve vzorci pracovního listu.

V případě prvním voláme funkci stejně jako každou jinou vestavěnou funkci. Postupujeme tak, že proměnné se přiřadí funkce, která provede např. nějaký výpočet s jinou proměnnou (jež je předávána funkci jako parametr, pokud jej funkce vyžaduje).

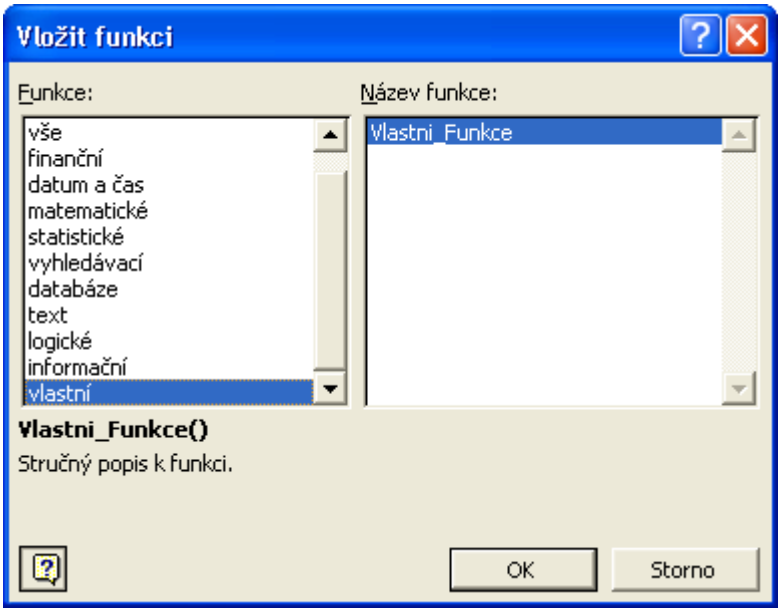
```
Sub Volani_Funkce()

 Dim Promenna1 As String
 Dim Promenna2 As Long

    'volání funkce
    Promenna1 = Funkce(Promenna2)

End Sub
```

V druhém případě se při práci v Excelu funkce používá vkládáním do vzorců ze seznamu funkcí nebo její název napíšeme do buňky za rovnítko (stejně jako u psaní vzorce). Pokud je funkce uložena v jiném sešitě, musíme před název funkce uvést jméno sešitu. Podmínkou je, že sešit musí být otevřen.

Významnou pomůckou je tzv. **doplněk**. Tvoří jej samostatný sešit, do něhož jsou uloženy uživatelské funkce. Tento sešit bude dostupný při každém dalším spuštění Excelu a nebude nutné zadávat název souboru, v němž je funkce. [3]

4.3 Deklarace funkce

Funkce se deklaruje takto: `Function` *`Název_funkce`* `End Function`. Před slovo `Function` můžeme podle toho, jak ji chceme použít, napsat jedno z následujících slov: `Private, Public, Static`. Výraz `Public` značí, že funkce je přístupná ve všech modulech a že se název (na rozdíl od procedury) neobjeví v seznamu maker, nýbrž v seznamu funkcí pod položkou Vlastní.



Obr. 19 Dialog pro vložení funkce do tabulky

Funkci můžeme použít pro práci se sešity v Excelu, stejně jako pro vestavěné funkce. Tím pádem je možné vkládat ji do vzorců. Pojem `Public Function` je v hlavičce plně zaměnitelný s pojmem `Function`.

Slovo `Private` způsobí, že funkce se dá volat pouze v rámci makra (nebo procedurou či jinou funkcí). A to navíc jen v jednom modulu. Funkce se tedy neobjeví v seznamu funkcí Excelu.

Slovo `Static` znamená, že hodnoty proměnných deklarované v dané funkci se uchovávají i po skončení této funkce.

Pro názvy funkcí platí ta samá pravidla, jako pro názvy procedur a proměnných. Pokud chceme funkci používat ve vzorci na pracovním listu, nesmí být její název zaměnitelný s označením buňky (např. A5). Také je lépe nepoužívat názvy, které jsou totožné s názvy vestavěných funkcí. Excel by totiž přednostně použil právě vestavěné funkce.

Funkci můžeme přiřadit datový typ. Pokud to neučiníme, potom jí bude automaticky přiřazen typ Variant.

Důležitou součástí funkce je minimálně jedno přiřazení výsledné hodnoty do názvu funkce, protože jinak by funkce nevracela žádnou hodnotu.

```
Function nazev_funkce As Double
    ' výpočty
    nazev_funkce = hodnota
End Function
```

Funkce mohou být bez parametrů, a to např. takové, které vracejí aktuální datum a čas, název uživatele, čísla řádku a sloupce aktivní buňky a mnohé další.

```
Function Jmeno_Aktivního_Sesitu() As String
    Jmeno_Aktivniho_Sesitu = ActiveWorkbook.Name
End Function
```

Nebo mají parametry, a to povinné, nebo nepovinné. Počet povinných může být stanoven pevně a to na 1 až 60 [2]. Povinné musejí být funkci předány vždy, nepovinné nikoli.

Existují funkce, které mají pouze povinné, pouze nepovinné a nebo oba typy parametrů.

4.4 Ladění funkcí

Ladění funkcí se provádí stejně jako ladění maker. Rozdíl spočívá v tom, že funkci nelze spustit samostatně z editoru VBA (příkazem **Run Macro**). Jestliže chceme funkci při hledání chyb spustit se zarážkami, musíme nejprve vytvořit jednoduché makro, které funkci vyvolá.

4.5 Zařazení funkce do kategorie

Uživatelské funkce v Excelu jsou zařazeny do kategorií dle tabulky č. 3.

Číslo kategorie	Název kategorie
0	vše (pouze v seznamu všech funkcí)
1	finanční
2	datum a čas
3	matematické
4	statistické
5	vyhledávací
6	databáze
7	text
8	logické
9	informační
10	příkazy
11	přizpůsobení
12	ovládání maker
13	Externí DDE
14	vlastní
15	inženýrská analýza

Tab. 3 Kategorie funkcí v Excelu "[7]

Pokud chceme vlastní funkci zařadit do některé z těchto kategorií, nemůžeme použít žádný přímý způsob, protože ho Excel nenabízí. Máme ale možnost danou funkci zařadit pomocí makra, které musí obsahovat příkaz, jenž žádanou akci provede:

Application.MacroOptions macro:="*název zařazované funkce*", _ Category:=*číslo kategorie*

[7] [2] str. 217

K tomu, aby funkce zůstala ve vybrané kategorii, úplně dostačuje jednorázové provedení příkazu.

[2]

4.6 Přidání popisu funkce

Přidání můžeme docílit dvěma způsoby. Jednak pomocí dialogového okna Makro. V něm se sice neobjeví seznam funkcí (pouze procedur), ale pokud napíšeme do políčka Název makra jméno funkce, objeví se dialogové okno Možnosti makra. Do něho následně napíšeme popis funkce.

Dále popis přidáváme (podobně jako u zařazování funkcí do kategorie) pomocí procedury, která má tento příkaz:

```
Application.MacroOptions Macro := "název funkce", _
Description := "stručný popisek"
```

[2]

Obě makra mohou mít následující podobu:

Makro č. 3 Přiřazení funkce a makro č. 4 Přidání popisku k funkci

```
' proměnné pro obě makra
Private Nazev_Funkce As String, Odpoved As String
'_____________________________________________________________________

Sub Prirad_Funkci()
 Dim Cislo_Kategorie As Byte
 Dim arKategorie

 arKategorie = Array("vše", "finanční", "datum a čas", _
    "matematické", "statistické", "vyhledávací", "databáze" _
    , "text", "logické", "informační", "příkazy", _
    "přizpůsobení", "ovládání maker", "externí DDE", _
    "vlastní", "inženýrská analýza")
 Const Kategorie As String = "0-bez zařazení (vše)," & _
    " 1-finanční, 2-datum a čas, 3-matematické," & _
    " 4-statistické, 5-vyhledávací, 6-databáze, 7-text," & _
    " 8-logické, 9-informační, 10-příkazy, 11-přizpůsobení," & _
    " 12-ovládání maker, 13-externí DDE, 14-vlastní," & _
    " 15-inženýrská analýza"

    Do
        On Error Resume Next
        Nazev_Funkce = InputBox("Zadej název funkce")
        Cislo_Kategorie = InputBox("Zadej číslo kategorie: " _
            & Kategorie)

        ' příkaz pro zařazení funkce
        Application.MacroOptions Macro:=Nazev_Funkce, _
            Category:=Cislo_Kategorie
```

```
        If Err <> 0 Then ' kontrola správného názvu
            Odpoved = MsgBox("Byl zadán chybný název " & _
                "funkce, nebo chybné číslo kategorie," & _
                " funkce nebyla zařazena.", vbRetryCancel)
            If Odpoved = "2" Then
                Exit Do
            End If
        Else
            MsgBox ("Funkce byla zařazena do kategorie " & _
                UCase(arKategorie(Cislo_Kategorie)))
            Exit Do
        End If
    Loop
End Sub
'_____________________________________________________________________

Sub Popisek_K_Funkci()
 Dim Popisek As String
    Do
       On Error Resume Next
        Nazev_Funkce = InputBox("Zadej název funkce")

        ' kontrola správného názvu
        Application.MacroOptions Macro:=Nazev_Funkce
        If Err <> 0 Then
            Odpoved = MsgBox("Tato funkce neexistuje", _
                vbRetryCancel)

            If Odpoved = "2" Then
                Exit Sub
            End If
        Else
            Popisek = InputBox("Přidej stručný popis k funkci")

            If Popisek = "" Then
                Odpoved = MsgBox("Popisek nebyl zadán -" & _
                " pokračovat?", vbOKCancel)

                If Odpoved = "2" Then
                    Exit Sub
                End If
            End If
        End If
    Loop Until Err = 0

    ' příkaz, který přidá popisek k funkci
    Application.MacroOptions Macro:=Nazev_Funkce, _
        Description:=Popisek
    MsgBox ("Zadaný popisek byl přidán k funkci " _
    & Nazev_Funkce)
End Sub
```

Při častém vytváření vlastních funkcí a jejich zařazování je výhodné nastavit modul s těmito makry tak, aby makra byla přístupná při každém spuštění Excelu, např. zkopírováním modulu do souboru *personal.xls*.

4.7 Názorná ukázka vlastních funkcí

Funkci č. 1 se jako parametr předává textová hodnota, ze které tato funkce odstraní diakritiku z českých slov.

Druhá funkce (č. 2) vrací celé náhodné číslo. Jeho hodnota se v buňce zachovává i po přepočítání listu. Funkci se předávají 2 parametry - dolní a horní mez.

Funkce č. 1 Odstranění diakritiky

```
Public Function Bez_diakr(Vst_text As String) As String
 Dim Vyst_text As String, Znak As String
 Dim I As Long, K As Byte, Pocet_znaku As Long
 Dim Zn_s_diakr, Zn_bez_diakr, Pocet_cyklu As Byte

 Pocet_cyklu = 0
 Vyst_text = ""
 Zn_s_diakr = Array("á", "é", "ě", "í", "ó", "ú", "ů", _
                    "č", "ď", "ň", "ř", "š", "ť", "ž")
 Zn_bez_diakr = Array("a", "e", "e", "i", "o", "u", "u", _
                      "c", "d", "n", "r", "s", "t", "z")
 Pocet_znaku = Len(Vst_text)
 For I = 1 To Pocet_znaku
     Znak = Mid(Vst_text, I, 1)
     Do
         For K = 0 To 13
             If Pocet_cyklu = 1 Then
                 If Znak = UCase(Zn_s_diakr(K)) Then
                     Znak = UCase(Zn_bez_diakr(K))
                     Exit Do
                 End If
             Else
                 If Znak = Zn_s_diakr(K) Then
                     Znak = Zn_bez_diakr(K)
                     Exit Do
                 End If
             End If
         Next K
         Pocet_cyklu = Pocet_cyklu + 1
     Loop Until Pocet_cyklu = 2
     Pocet_cyklu = 0
     Vyst_text = Vyst_text & Znak
 Next I
 Bez_diakr = Vyst_text
End Function
```

Funkce č. 2 Náhodné celé číslo

```
Function Nahodne_Cele_Cislo _
  (Dolni_mez As Long, Horni_mez As Long) As Long
    Nahodne_Cele_Cislo = _
     Int((Horni_mez - 1) - Dolni_mez + 1) _
     * Rnd + Dolni_mez
End Function
```

5 Formuláře

5.1 Využití vestavěných dialogů

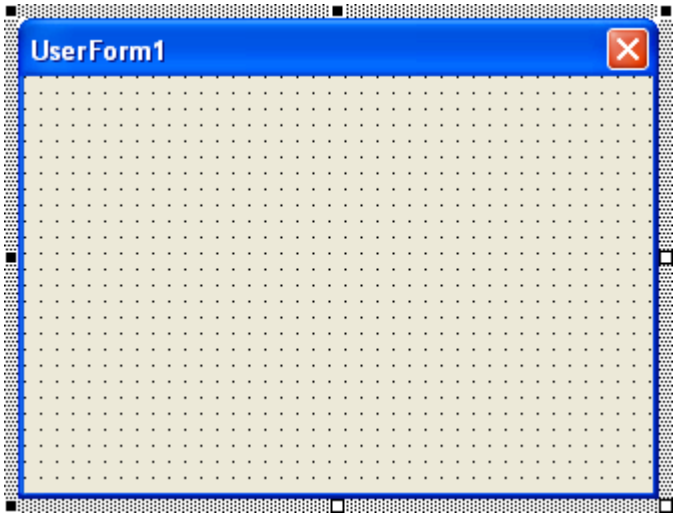
VBA nabízí několik **vestavěných funkcí**. Díky nim uživatel komunikuje s makrem pomocí vestavěných dialogových oken. Jednou z nich je například funkce **MsgBox**. Tato funkce může uživatele informovat (o chybě, o špatně zadané hodnotě atd.), může ho také vyzvat k rozhodnutí, zda operaci opakovat, přeskočit či přerušit (nabízí varianty „Ano“, „Ne“, a jiné). Druhou funkcí je funkce **InputBox**. Pomocí ní je uživatel požádán a vyzván k tomu, aby zapsal hodnotu, kterou makro vyžaduje k dalšímu průběhu. Třetí funkce je **GetOpenFilename**. Ta vyzývá uživatele k tomu, aby nalezl soubor a to tak, že bude procházet adresářovou strukturu. Zmíněnou operaci lze vykonat také použitím okna **InputBox**, kde je ovšem potřeba název souboru i s jeho cestou zapsat ručně. Úskalí tohoto procesu tkví v tom, že při zápisu může dojít k chybě. Pro uložení souboru slouží funkce **GetSaveAsFilename**. Některé funkce včetně dialogů lze získat voláním z Windows API. API je množina prvků, které můžeme použít ve VBA, ačkoli nejsou jeho přímou součástí. Jedním z prvků API je dialogové okno pro výběr adresáře. [2]

5.2 Vlastní formuláře

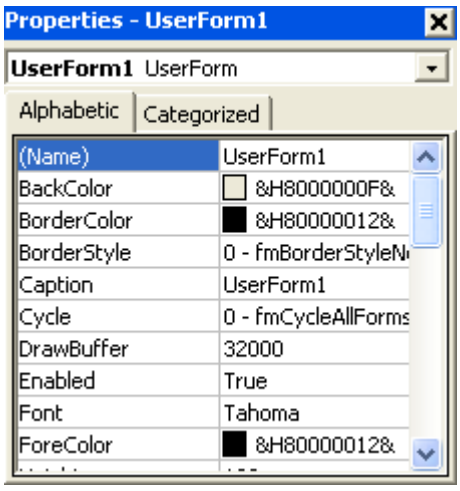
Kromě varianty použít vestavěné dialogy se nám nabízí možnost vytvořit si okna vlastní, která nazýváme **formuláře** (případně uživatelské dialogy, uživatelská dialogová okna, vlastní dialogová okna, uživatelské formuláře) [8]. Vytvoření vlastního okna s sebou nese řadu výhod. Zejména jde o lepší ovládání makra, dále zkrácení zdrojového kódu a také alternativu navolení si dialogového okna podle vlastních potřeb uživatele.

Formulář vytvoříme následujícím způsobem. V editoru VBA vložíme nový formulář příkazem **Insert → UserForm**. Poté se objeví prázdné okno (obr. 20), jehož velikost a tvar lze pozměňovat pomocí myši. Jeho další vlastnosti, jako jsou například barva, název, zobrazování obrázků, nebo to, zda je modální či nemodální (tj. jestli umožňuje kliknout myší do listu a případně do něho něco doplnit), můžeme nastavit

v okně **Properties** (obr. 21). Jiný způsob změny vlastností a metod je pomocí příkazů ve zdrojovém kódu makra nebo zdrojového formuláře.



Obr. 20 Prázdné okno formuláře



Obr. 21 Okno Properties

Má-li být formulář funkční, je třeba na jeho plochu přidat tlačítka, tzv. **ovládací prvky**. Máme několik možností. Jednou z nich je využití základních typů ovládacích prvků, jež jsou součástí MS Excel. Další možností je získání tlačítek z jiných programů, nainstalovaných ve Windows. V úvahu připadá také jejich získání z internetových stránek. Obvykle bývají obsažena v knihovnách dll nebo ocx. [8]

Mezi základní ovládací prvky náležejí:

- **Příkazové tlačítko (CommandButton)** – slouží k provedení nějaké akce, často užívané k ukončení formuláře jako tlačítko „OK“ nebo „Storno“.
- **Popisek (Label)** – na formuláři se zobrazí nějaký text, který může sloužit jako informace o nějakém ovládacím prvku.
- **Textové pole (TextBox)** – do něho zapisujeme hodnoty.
- **Zaškrtávací políčko (CheckBox) –** plní potřebu nabízet možnost volby („vybráno“, „nevybráno“).
- **Přepínač (OptionButton) –** nabízí výběr z několika možností.
- **Seznam (Listbox) -** umožňuje výběr z několika možností. Nabídka je zobrazena v jednom okně a lze zvolit více variant najednou.
- **Rozvírací seznam (ComboBox) –** obdobná funkce jako Listbox. Seznam nabídky se objeví až po rozbalení.
- **Rámeček (Framme) –** plní estetickou funkci a dále vymezuje skupiny prvků.
- **Přepínací tlačítko (ToggleButton) -** funkčně stejné jako zaškrtávací políčko.
- **Číselník (SpinButton) –** má podobu dvou šipek, jejichž prostřednictvím se přidávají (ubírají) hodnoty v textovém poli, se kterým se číselník propojuje.
- **Posuvník (ScrollBar) –** stejně jako u číselníku slouží k přidávání a ubrání hodnot. Má navíc posuvného jezdce, který umožňuje rychlejší změnu větších rozsahů hodnot.
- **RefEdit –** nabízí možnost výběru oblasti buněk na pracovním listu.
- **Vícenásobná stránka (MultiPage) –** složena ze dvou i více karet. Díky nim lze tlačítka skládat podle různých kategorií a také přidat na formulář větší počet tlačítek.
- **Karty (TabStrip) –** tento ovládací prvek je vzhledově stejný jako Vícenásobná stránka. Lze provádět různé akce s jedním ovládacím prvkem, který je ve skutečnosti na formuláři a na kartě je fiktivně.
- **Obraz (Image) –** slouží k vložení obrázku na formulář.

Přidání tlačítek na formulář uskutečníme tak, že z vyvolaného okna Toolbox (obr. 22) přetáhneme ovládací prvek na plochu formuláře. Zároveň můžeme měnit jeho vlastnosti, podobně jako u samotného formuláře. Je vhodné upravit názvy ovládacích prvků podle účelů, k nimž slouží a přidáním předpony před jejich názvy podle druhu

ovládacích prvků. Volba předpon je odvislá od určitých konvencí, které je záhodno dodržovat zvláště kvůli **obecné srozumitelnosti kódu**. Toto tvrzení ilustruje tabulka číslo 4.

Zkratka typu ovládacího prvku	**Ovládací prvek**
frm	formulář
txt	Textové pole (text box)
cmd	Příkazové tlačítko (command button)
lbl	Popisek (label)
lst	Seznam (list)
cbo	Rozevírací seznam (combo box)
chk	Zaškrtávací políčko (check box)
opt	Přepínač (option button)
fra	Rámeček (frame)
pic	Obraz (picture)

Tab. 4. Pravidla pro názvy ovládacích prvků "[8]

Obr. 22 Základní ovládací prvky na panelu Toolbox

Každý ovládací prvek má svůj zdrojový kód, který funguje jako procedura. Ta se spustí tím, že ve spuštěném formuláři aktivujeme ovládací prvek. Do procedury ovládacího prvku lze vkládat příkazy, které probíhají stejným způsobem jako v makru. Např. ve zdrojovém kódu procedury ovládacího prvku, jehož úkolem je zavření spuštěného formuláře, je příkaz `Unload Me`. Formulář se spustí z makra příkazem *`název_formuláře`*`.Show.`

[8] [8] str. 85

5.3 Ovládací prvky na pracovním listu

Ovládací prvky můžeme vložit také na pracovní list přímo do sešitu Excel, a to buď z panelu Ovládací prvky nebo z panelu Formuláře. Při nastavování prvků z obou těchto panelů používáme různé způsoby.

Jestliže je nastavujeme z panelu Ovládací prvky, postupujeme obdobně, jako při jejich aplikování na formuláři. Tzn., že vlastnosti se mění také v okně Properties a pracuje se se zdrojovým kódem.

Prvky z panelu Formuláře jsou výhodnější v tom případě, kdy chceme propojit ovládací prvek s určitou buňkou (například můžeme vložit na pracovní list číselník a propojit ho pomocí položky **Formát ovládacího prvku** s buňkou A10 a měnit v ní číselnou hodnotu). [8]

Další způsob využití je ten, že můžeme ovládacímu prvku z panelu Formuláře přiřadit makro.

Zmíněné postupy značně šetří programátorův čas proto, že nemusí vytvářet formuláře v editoru VBA. Zároveň lze využitím propojení maker, formulářů a vkládáním ovládacích prvků přímo na pracovní list vytvářet celá uživatelská rozhraní.

6 Další ukázková makra

6.1 Úvod

Užitečnost maker se prokáže téměř v každém oboru lidské činnosti, ať už jde o administrativu, technické obory nebo domácnost. V následujícím textu si ukážeme funkční využití maker v praxi.

Všechna vytvořená makra jsou k nahlédnutí na CD, přiloženém k této práci. Jsou uložena v sešitech (xls nebo ods) a číslování jejich názvů odpovídá číslům maker v textu práce. Některé ze sešitů zároveň obsahují vstupní data, jiným přísluší vstupní data v adresářích očíslovaných shodně se soubory s makry.

6.2 Makro č. 6 Výpočet planimetrážního listu

Úkolem tohoto makra je spočítat planimetrážní list, který obsahuje pětimístný kód BPEJ doplněný o tečky (5.29.11), počet lokalit se stejným kódem a jejich celkovou výměru. Vstupní tabulka musí obsahovat 2 sloupce, jeden s pětimístnými kódy (B5) a druhý s výměrami ploch (AREA), které náležejí těmto kódům. Ostatní sloupce jsou v tabulce navíc.

Makro nejdříve vyhledává sloupce B5 a AREA a pokud je nenajde, bude ohlášena chyba a makro bude ukončeno. Když je vše v pořádku, jsou vytvořeny pomocné sloupce (BPEJ, PL, VYMERA) a probíhá cyklus, který do nich zapisuje údaje. V případě, že následující kód je stejný, přičítá výměru k předcházejícímu a načte počet lokalit o 1. Podle množství jedinečných kódů má buď 1 nebo 2 stránky a jsou do ní nakopírována data z pomocných sloupců. Obě tabulky jsou na přiloženém CD v adresáři MAKRA\04-Pl_list-tiskopisy“[9]. Pro otevření těchto souborů je v makru nastavena cesta na D:\. V případě, že žádaný soubor není nalezen, makro vyzve uživatele, aby jej nalezl. Po zkopírování údajů jsou pomocné sloupce vymazány. Pokud uživatel soubor nenalezne, makro bude ukončeno a pomocné sloupce vymazány nebudou.

Vzhledem k tomu, že toto makro je mé první, nachází se v jeho zdrojovém kódu několik zásadových chyb jako odsazování řádků jednou mezerou, používání zastaralého

[9] [9] str. 40

slova `REM` pro komentáře a nesprávná deklarace některých proměnných, které jsou chybou zápisu deklarovány jako `Variant`, namísto `String`, či `Long`. Původní záměr tyto nedostatky opravit jsem opustil a zdrojový kód jsem nakonec ponechal s těmito chybami pro názornou ukázku nesprávného zápisu. Makro zpracovává pokaždé jiný soubor se stejným typem dat, proto je vhodné uložit ho do osobního sešitu maker (personal.xls).

Makro č. 6 Planimetrážní list

```
Option Explicit
'_____________________________________________________________________________

Private Function Otevren_sesit(Soubor As String) As Boolean
 Rem funkce se pokusi otevrit soubor s prazdnymi tabulkami
 Rem a podle uspesneho ci neuspesneho pokusu vraci Tr./Fa.

On Error Resume Next
 Workbooks.Add Soubor
 If Err = 0 Then
  Otevren_sesit = True
 ElseIf Err = 1004 Then
  Otevren_sesit = False
 End If
End Function
'_____________________________________________________________________________

Sub pl_list1()

' Klávesová zkratka: Ctrl+p
Dim I, M As Integer
Dim A, Dalsi, Predchozi, B, Sloupec As String
Dim Soubor, NazevSoub, Jmeno_Databaze As String
Dim Pocet_Lok, Pocet_bpejek, Pocet_bpejek_2, K, J, L, N As Byte
Dim Vymera As Double

Range("a1").Select
I = 0

'----------------------kontrola tabulky-----------------------
Rem vyhledani sloupce B5 (J)
J = 0
Do Until Selection.Offset(I, J + 1).Value = "" Or _
 Selection.Offset(I, J).Value = "B5" Or _
 Selection.Offset(I, J).Value = "b5"
  J = J + 1
Loop

Rem pokud chybí B5, aplikace je ukončena
If Selection.Offset(I, J + 1).Value = "" Then
 MsgBox "Chybí sloupec B5, nebo není otevřen správný soubor" _
  , vbCritical
 Exit Sub
End If

Rem serazeni vzestupne podle B5
Selection.Sort Key1:=ActiveCell.Offset(0, J).Range("a1"), _
 Order1:=xlAscending, Header:=xlGuess, OrderCustom:=1, _
 MatchCase:=False, Orientation:=xlTopToBottom

Rem vyhledani sloupce s vymerou (area)(N)
N = 0
```

```
Do Until Selection.Offset(I, N + 1).Value = "" Or _
 Selection.Offset(I, N).Value = "AREA"
  N = N + 1
Loop

Rem pokud chybí AREA, aplikace je ukončena
If Selection.Offset(I, N + 1).Value = "" _
 And Selection.Offset(I, N).Value <> "AREA" Then
  MsgBox "Chybí sloupec s výměrami AREA, " _
   & "nebo není otevřen správný soubor", vbCritical
  Exit Sub
End If

Rem vyhledani prazdneho sloupce (konec tabulky) (L)
L = 0
Do Until Selection.Offset(I, L).Value = ""
 L = L + 1
Loop

Rem vytvoreni pomocnych sloupcu
Selection.Offset(I, L).Value = "BPEJ"
Selection.Offset(I, L + 1).Value = "PL"
Selection.Offset(I, L + 2).Value = "VYMERA"

'----------------vypocet planimetrazniho listu-----------------
K = 0
Do While Selection.Offset(I + 1, J).Value <> ""
 I = I + 1                              'preskoci na dalsi radek
 A = Selection.Offset(I, J).Value
 Dalsi = Selection.Offset(I + 1, J).Value
 Vymera = Selection.Offset(I, N).Value

 If Len(A) <> 2 Then 'vynechava 2 mistne kody 23, 29, 35, 99 aj.
  Pocet_Lok = 1
  K = K + 1

  Do While A = Dalsi  'opakuje, dokud dalsi kod je stejny
   Pocet_Lok = Pocet_Lok + 1
   I = I + 1                   'zvisi pocet lokalit o 1
   A = Selection.Offset(I, J).Value
   Dalsi = Selection.Offset(I + 1, J).Value
   'pricteni vymery
   Vymera = Vymera + Selection.Offset(I, N).Value
  Loop

  'zapis 5 mistneho kodu doplneneho o tecky (2901 -> 0.29.11)
  If Len(A) = 1 Then 'kdyz chybi kod (0)
   Selection.Offset(K, L).Value = "0.00.00"
  ElseIf Len(A) = 5 Then 'normalni kod
   Selection.Offset(K, L).Value = Mid(A, 1, 1) & "." & _
    Mid(A, 2, 2) & "." & Mid(A, 4, 5)
  ElseIf Len(A) = 4 Then 'nulty KR
   Selection.Offset(K, L).Value = "0." & Mid(A, 1, 2) & _
    "." & Mid(A, 3, 4)
  ElseIf Len(A) = 3 Then 'nulty KR, HPJ 01 az 09
   Selection.Offset(K, L).Value = "0.0" & Mid(A, 1, 1) & _
    "." & Mid(A, 2, 3)
  End If

 Rem zapis poctu lokalit a vymery
 Selection.Offset(K, L + 1).Value = Pocet_Lok
 Selection.Offset(K, L + 2).Value = Vymera
 End If
Loop

Pocet_bpejek = K
Rem Pocet_bpejek_2 = K
```

```
Rem ------kopie dat z pomocnych sloupcu do prazdne tabulky------

Rem ..............otevreni prazdne tabulky z c:\..........
Jmeno_Databaze = ActiveWorkbook.Name
K = 1
ChDir "D:\"
 If Pocet_bpejek < 50 Then
  Soubor = "Planim_list_1_stranka.xls"
 Else 'když se data nevejdou na 1 stránku, otevře se soubor
  Soubor = "Planim_list_2_stranky.xls"     'se 2 strankami
 End If

If Not Otevren_sesit((Soubor)) Then
  Soubor = Application.GetOpenFilename("xls,*.xls", , _
    "Najdi soubor " & Soubor)

  Rem vyzvání uživatele, aby soubor našel neotevřený soubor
  If Soubor = "False" Then
   MsgBox "Hledaný soubor nebyl nalezen, " & _
    "aplikace bude ukončena", vbCritical
   Exit Sub
  Else
   Workbooks.Add Soubor
  End If
End If
Soubor = ActiveWorkbook.Name

Rem ......................kopie dat.........................
Range("c6").Select
M = 0
Pocet_bpejek = 0

Do While Windows(Jmeno_Databaze).Selection.Offset(K, L).Value _
 <> ""

 'BPEJ
 Selection.Offset(M, 0).Value = _
  Windows(Jmeno_Databaze).Selection.Offset(K, L).Value

 'pocet lokalit
 Selection.Offset(M, 2).Value = _
  Windows(Jmeno_Databaze).Selection.Offset(K, L + 1).Value

 'vymera
 Selection.Offset(M, 17).Value = _
 Windows(Jmeno_Databaze).Selection.Offset(K, L + 2).Value

Rem.................smazani pom. tabulky.....................
 Windows(Jmeno_Databaze).Selection.Offset(K, L).Value = ""
 Windows(Jmeno_Databaze).Selection.Offset(K, L + 1).Value = ""
 Windows(Jmeno_Databaze).Selection.Offset(K, L + 2).Value = ""

 M = M + 1
 K = K + 1
 Pocet_bpejek = Pocet_bpejek + 1

 'kdyz se data nevejdouna stranku,
 'preskoci o 2 radky na dalsi stranku
 If Pocet_bpejek = 49 Then
  M = M + 2
  Pocet_bpejek = 0 'pro případné zozšíření na 3. stránky
 End If
Loop

Rem.............smazani pom. tabulky dokonceni..............
 Windows(Jmeno_Databaze).Selection.Offset(0, L).Value = ""
 Windows(Jmeno_Databaze).Selection.Offset(0, L + 1).Value = ""
 Windows(Jmeno_Databaze).Selection.Offset(0, L + 2).Value = ""
```

```
Workbooks(Jmeno_Databaze).Activate

Rem seřazení zpět podle čísla
Selection.Sort Key1:=Range("A1"), Order1:=xlAscending, _
 Header:=xlGuess, OrderCustom:=1, MatchCase:=False, _
 Orientation:=xlTopToBottom

Workbooks(Soubor).Activate

End Sub
```

6.3 Makro č. 7 Setřídění kartogramů zrnitosti

Další ukázkové makro je podobné předchozímu. Jeho úkolem je připravit vstupní databázový soubor mapy kartogramů zrnitosti, skeletovitosti a zamokření ke kontrole chyb. Ta probíhá pomocí tabulky, kterou makro vygeneruje. Jejím obsahem musí být pro daný účel 2 sloupce, a to sloučený kód kartogramu (KARTOGRAM) a počet lokalit. Třetí sloupec s výměrou má pouze informační charakter, stejně tak jako poslední řádek s celkovým počtem lokalit a s celkovou výměrou.

Makro nejdříve ve vstupní tabulce doplní prázdný sloupec „KARTO“ o zkopírované nebo sloučené kódy z vedlejších sloupců. Např. pokud je ve sloupci s ornicí (ORNICE) kód JH a ve sloupci s podorničím (PODOR) kód JV, tak do sloupce s kódem kartogramu (KARTO) je zapsán kód JH-JV, příp. může ještě být doplněn o poznámku, která se může nacházet ve sloupci „OR_POZN“ nebo „PDOR_POZN“, např. PH-H(J+P). Pokud se pod položkou „ORNICE“ nenachází kód, je do položky „KARTO“ zapsán kód ostatní plochy 99. Pro hledání sloupců s příslušnými položkami je na rozdíl od makra č. 6 vytvořena funkce `Cislo_Sloupce`, která vrací číslo hledaného sloupce potřebné pro průběhy cyklů. V další části je otevřen nový sešit, ve kterém je vytvořena nová tabulka, do které se zapisují přímo (bez pomocných sloupců ve zdrojové tabulce) údaje, které jsou získány stejným způsobem jako při výpočtu planimetrážního listu v předchozím makru. I toto makro je vhodné zpřístupnit při každém spuštění Excelu.

Makro č. 7 Setřídění kartogramů

```
Option Explicit
'_____________________________________________________________________

Private Function Cislo_Sloupce(Text As String)
'funkce vrátí pořad. číslo hledaného sloupce

Dim Sloupec As Integer
```

```
Sloupec = 1
Do While Not IsEmpty(Cells(1, Sloupec + 1).Value) And _
 UCase(Cells(1, Sloupec)) <> (Text)
    Sloupec = Sloupec + 1
Loop
Cislo_Sloupce = Sloupec

'když v tabulce není hledaný text,
 'funkce vrátí odkaz na prázdnou buňku
If IsEmpty(Cells(1, Sloupec + 1).Value) And _
 UCase(Cells(1, Sloupec)) <> (Text) Then
    Cislo_Sloupce = Cislo_Sloupce + 1
End If

End Function
'_____________________________________________________________________________

Sub Setrideni_kartogramu()

Dim Kar_J As Integer, Cis_J As Integer, Are_J As Integer
Dim Pocet_Lok As Integer, Pocet_Lok_Celkem As Integer
Dim I As Integer, J As Integer, K As Integer, L As Integer
Dim Kartogram As String, A As String, Dalsi As String
Dim Predchozi As String, Databaze As String
Dim Vymera As Double, Vymera_Celkem As Double
Dim Probehl As Boolean, Novy_Sesit As String

Kar_J = 1
I = 1
J = 1
Cis_J = 1

Kar_J = Cislo_Sloupce("KARTO") 'vyhledání sloupce KARTO
Cis_J = Cislo_Sloupce("CISLO") 'vyhledání sloupce CISLO

'-------------sloučení sloupců do jednoho - KARTO --------------
Kartogram = ""
Do While Cells(I + 1, Cis_J) <> ""
    Kartogram = ""
    Probehl = False
    J = Cislo_Sloupce("ORNICE")
    If Cells(I + 1, J) <> "" Then
        Kartogram = Cells(I + 1, J)
        J = Cislo_Sloupce("OR_POZN")
        If Cells(I + 1, J) <> "" Then
            Kartogram = Kartogram & "(" & Cells(I + 1, J) & ")"
        End If
        J = Cislo_Sloupce("PODOR")
        If Cells(I + 1, J) <> "" Then
            Kartogram = Kartogram & "-" & Cells(I + 1, J)
            J = Cislo_Sloupce("PODOR_POZN")
            If Cells(I + 1, J) <> "" Then
                Kartogram = Kartogram & "(" & _
                 Cells(I + 1, J) & ")"
            End If
        End If
        Probehl = True
    End If
    If Not Probehl Then ' nebo If kartogram = ""
        J = Cislo_Sloupce("OST_PL")
        If Cells(I + 1, J) <> "" Then
            Kartogram = Cells(I + 1, J) & "d"
            Probehl = True
        End If
    End If
    If Not Probehl Then
        J = Cislo_Sloupce("JINE_PT")
```

```
        If Cells(I + 1, J) <> "" Then
            Kartogram = Cells(I + 1, J)
        End If
    End If
    Cells(I + 1, Kar_J) = Kartogram
    I = I + 1
Loop
'-------------------------------------------------------------

Databaze = ActiveWorkbook.Name
K = 1
L = 1
Workbooks.Add xlWBATWorksheet
Novy_Sesit = ActiveWorkbook.Name

'---------vytvoření nových sloupců a nastavení formárů----------
With Cells(1, 1)
    Cells(1, 1) = "KARTOGRAM"
    .Columns.AutoFit
    .Borders(xlEdgeBottom).LineStyle = xlDouble
End With
With Cells(1, 2)
    Cells(1, 2) = "POCET_LOK"
    .Columns.AutoFit
    .Borders(xlEdgeBottom).LineStyle = xlDouble
End With
With Cells(1, 3)
    Cells(1, 3) = "VYMERA"
    .Columns.AutoFit
    .Borders(xlEdgeBottom).LineStyle = xlDouble
End With
Range("c:c").Select
Selection.NumberFormat = "0.0" 'formát sloupce "VYMERA"
Range("a1").Select             'na 1 desetiné místo
'-------------------------------------------------------------
Workbooks(Databaze).Activate

'serazeni vzestupne podle "KARTO"
Selection.Sort Key1:=Cells(1, Kar_J), Order1:=xlAscending, _
 Header:=xlGuess, OrderCustom:=1, MatchCase:=False, _
 Orientation:=xlTopToBottom
Are_J = Cislo_Sloupce("AREA")
I = 1
Do While Cells(I + 1, Kar_J) <> ""
    I = I + 1
    A = Cells(I, Kar_J)
    Dalsi = Cells(I + 1, Kar_J)
    Vymera = Cells(I, Are_J)
    Pocet_Lok = 1
    K = K + 1
    Do While A = Dalsi  'opakuje, dokud je další köd stejný
        Pocet_Lok = Pocet_Lok + 1 'zvýší počet lokalit o 1
        I = I + 1
        A = Cells(I, Kar_J)
        Dalsi = Cells(I + 1, Kar_J)
        Vymera = Vymera + Cells(I, Are_J) 'přičte výměru
    Loop

    'zápis kódu kartogramu
    Windows(Novy_Sesit).Selection.Offset(K - 1, 0) = A
    'zápis počtu lokalit
    Windows(Novy_Sesit).Selection.Offset(K - 1, 1) = Pocet_Lok
    'zápis výměry
    Windows(Novy_Sesit).Selection.Offset(K - 1, 2) = Vymera
    Pocet_Lok_Celkem = Pocet_Lok_Celkem + Pocet_Lok
    Vymera_Celkem = Vymera_Celkem + Vymera
Loop
```

```
'seřazení vzestupně opět podle "CISLO"
Selection.Sort Key1:=Cells(1, Cis_J), Order1:=xlAscending, _
 Header:=xlGuess, OrderCustom:=1, MatchCase:=False, _
 Orientation:=xlTopToBottom


Workbooks(Novy_Sesit).Activate

Cells(K, 1).Borders(xlEdgeBottom).LineStyle = xlDouble
Cells(K, 2).Borders(xlEdgeBottom).LineStyle = xlDouble

'dvojitá čára - konec tabulky
Cells(K, 3).Borders(xlEdgeBottom).LineStyle = xlDouble

Cells(K + 1, 1) = "celkem"
Cells(K + 1, 2) = Pocet_Lok_Celkem
Cells(K + 1, 3) = Vymera_Celkem

End Sub
```

6.4 Makro č. 8 Převod dat z mapového souboru vtx do sešitu xls

Následující makro převede body získané v terénu např. pomocí GPS přístroje i s dalšími zapsanými informacemi a souřadnicemi, které jsou uloženy v textovém souboru s příponou vtx do sešitu MS Excel. Získaný sešit je možno dále zpracovávat. Vstupní soubor nemá informace zapsané ve sloupcích, ale po řádcích v určitých blocích, kde položka může a nemusí být uvedena podle toho, zda do ní byla zapsána hodnota či nikoliv. Každý blok vždy končí řádkem se zápisem souřadnice (&L P 637475.73 1053864.59).

V prvním kroku makro importuje vstupní soubor vtx. Tato část je nahrána pomocí záznamníku maker. V další části dojde k analýze importovaných dat, kde jsou zjištěny všechny názvy položek a ty jsou zapsány do proměnné typu pole. Poté je vytvořena hlavička tabulky v nově otevřeném sešitu. Následně makro prochází importovaná data po jednotlivých řádcích a získané informace zapisuje do nového sešitu.

Makro č. 8 Převod dat z vtx do xls

```
Option Explicit
'_____________________________________________________________________

Private Function Text_pred_rovnitkem(Text As String)
 Dim Slovo
    Slovo = Split(Text, "=", 2)
    Text_pred_rovnitkem = Slovo(0)

End Function
'_____________________________________________________________________
```

```
Private Function Text_za_rovnitkem _
    (Text As String, Radek As Long)
 Dim Slovo
 Dim Sloupec As Byte
    Slovo = Split(Text, "=", 2)
    Text_za_rovnitkem = Slovo(1)
    Sloupec = 2
    Do While Cells(Radek, Sloupec + 1).Value <> Empty
        Text_za_rovnitkem = Text_za_rovnitkem & " " & _
            Cells(Radek, Sloupec + 1)
        Sloupec = Sloupec + 1
    Loop
End Function
'_____________________________________________________________________________

Private Function Cislo_Sloupce(Text As String)
'funkce vrátí pořad. číslo hledaného sloupce

 Dim Sloupec As Byte

    Sloupec = 1
    Do While Not IsEmpty(Cells(1, Sloupec + 1).Value) And _
            (Cells(1, Sloupec)) <> (Text)
        Sloupec = Sloupec + 1
    Loop
    Cislo_Sloupce = Sloupec
End Function
'_____________________________________________________________________________

Sub VTX_do_XLS()
 Dim I  As Long, K As Byte, M As Long
 Dim Obsah_Bunky As String
 Dim Polozka As String, Hodnota As String
 Dim Jmeno_Tohoto_Sesitu, Jmeno_Noveho_Sesitu As String
 Dim Pocet_Indexu As Byte, Index As Byte, Pocet_Polozek As Byte
 Dim DP_Polozky(1 To 250)
 Dim Nasel As Boolean
 Dim Jmeno_Vstupniho_Souboru As String
'---------------------------------------------------------------

    Jmeno_Vstupniho_Souboru = Application.GetOpenFilename _
     (",*.vtx", , "Zadej vstupní soubor VTX ")

    If Jmeno_Vstupniho_Souboru = "False" Then
        MsgBox ("Vstupní soubor nebyl zadán aplikace bude " _
         & "ukončena"), vbCritical
        Exit Sub
    End If

'---import zadaného vstupního souboru VTX nahráno záznamníkem---
    With ActiveSheet.QueryTables.Add(Connection:="TEXT;" & _
        Jmeno_Vstupniho_Souboru, Destination:=Range("A1"))
        '.Name = "sondy"
        .FieldNames = True
        .RowNumbers = False
        .FillAdjacentFormulas = False
        .PreserveFormatting = True
        .RefreshOnFileOpen = False
        .RefreshStyle = xlInsertDeleteCells
        .SavePassword = False
        .SaveData = True
        .AdjustColumnWidth = True
        .RefreshPeriod = 0
        .TextFilePromptOnRefresh = False
        .TextFilePlatform = xlWindows
        .TextFileStartRow = 6
        .TextFileParseType = xlDelimited
        .TextFileTextQualifier = xlTextQualifierDoubleQuote
```

```
        .TextFileConsecutiveDelimiter = True
        .TextFileTabDelimiter = False
        .TextFileSemicolonDelimiter = False
        .TextFileCommaDelimiter = False
        .TextFileSpaceDelimiter = True
        .TextFileColumnDataTypes = Array(2, 1, 1, 1)
        .Refresh BackgroundQuery:=False
    End With
'-----------------------------------------------------------------

    Jmeno_Tohoto_Sesitu = ActiveWorkbook.Name

    I = 1
    Index = 1

    Do While Cells(I, 1) <> "&K"  'naplnění pole - Položky
        Do
            'ignoruje první položku "arc_viev"
            If Cells(I, 1) <> "&O" Then
                Polozka = Text_pred_rovnitkem(Cells(I, 2))
                Nasel = False

                'cyklus projde pole Polozky a zjistí
                 'přítomnost aktuálního názvu položky
                For K = 1 To Index
                    If DP_Polozky(K) = Polozka Then
                        Nasel = True
                        Exit For
                    End If
                Next K

                If Not Nasel Then
                    DP_Polozky(Index) = Polozka
                    Index = Index + 1
                End If
            End If

            I = I + 1
        Loop Until Cells(I, 1) = "&L"

        I = I + 1
    Loop

    DP_Polozky(Index) = "Y"
    Index = Index + 1
    DP_Polozky(Index) = "X"
'-----------------------------------------------------------------
    Workbooks.Add xlWBATWorksheet
    Jmeno_Noveho_Sesitu = ActiveWorkbook.Name

    'vytvoření nových sloupců se získanými názvy položek
    For K = 1 To Index
        Cells(1, K) = DP_Polozky(K)
    Next K

    Cells.Select
    Selection.NumberFormat = "@"
    ActiveWindow.ScrollRow = 1
    Range("A1").Select
    Workbooks(Jmeno_Tohoto_Sesitu).Activate
    I = 1
    M = 2

    Application.ScreenUpdating = False

    Do While Cells(I, 1) <> "&K"
        Do 'získání hodnot a uložení do nového sešitu
```

```
            'ignoruje první položku "arc_viev"
            If Cells(I, 1) <> "&O" Then
                Hodnota = Text_za_rovnitkem(Cells(I, 2), (I))
                Polozka = Text_pred_rovnitkem(Cells(I, 2))
                Workbooks(Jmeno_Noveho_Sesitu).Activate
                Cells(M, Cislo_Sloupce(Polozka)) = Hodnota
                Workbooks(Jmeno_Tohoto_Sesitu).Activate
            End If

            I = I + 1
        Loop Until Cells(I, 1) = "&L"

        'zkopírování souřadnic X, Y
        Hodnota = Cells(I, 3)
        Workbooks(Jmeno_Noveho_Sesitu).Activate
        Cells(M, Cislo_Sloupce("Y")) = Hodnota
        Workbooks(Jmeno_Tohoto_Sesitu).Activate
        Hodnota = Cells(I, 4)
        Workbooks(Jmeno_Noveho_Sesitu).Activate
        Cells(M, Cislo_Sloupce("X")) = Hodnota
        Workbooks(Jmeno_Tohoto_Sesitu).Activate

        I = I + 1
        M = M + 1
    Loop

    Cells.Select
    Selection.Delete Shift:=xlUp
    Range("A1").Select
    Workbooks(Jmeno_Noveho_Sesitu).Activate
    Cells(1, Cislo_Sloupce("RECNO")) = "CISLO"
    Selection.CurrentRegion.Select
    Selection.Columns.AutoFit
    Range("a1").Select
End Sub
```

6.5 Makro č. 9 Příprava ovocných vín 2

Další předkládané makro provádí totéž, co makro č. 1. Na rozdíl od prvního makra je zde pro zadání vstupních dat (druh vína a množství) použito formuláře, kde druh vína se vybírá pomocí přepínače (OptionButton) a množství pomocí číselníku. Procedury jednotlivých druhů s hodnotami ingrediencí jsou umístěny v kódu formuláře. Pro přepočet surovin je zde volána funkce `Ingred_prepoc`, které se jako parametr předává množství zadané uživatelem a návratová hodnota funkce je pole proměnných, které obsahuje přepočítané suroviny. Ty jsou následně zapsány do sešitu MS Excel pomocí procedury `Zapis1` nebo `Zapis2`, případně obou, podle toho jaký druh je zvolen. Toto makro je, stejně jako makro č. 1, vhodné spouštět tlačítkem umístěným v sešitu.

Makro č. 9 Příprava vín 2 – kód modulu

```
Option Explicit
```

```
'_____________________________________________________________________________
Public A As Single, Stava As Single, Cukr As Single
Public Cukr1 As Single, Voda As Single, Ovoce As Single
Public Druh As String, OK As Boolean
Dim Kvasinky As Single, Sul As Single
Dim Ingr, Tab1, Tab2, J As Byte
'_____________________________________________________________________________

Sub Priprava_vina2()
'program prepocita mnozstvi ovoce, nebo ovo. stavy, cukru a vody
 Cells.ClearContents
 Kvasinky = 0.1
 Sul = 0.1
 Tab1 = Array("šťáva ", "cukr", "voda", "kvasinky", "živná sůl")
 Tab2 = Array("ovoce", "cukr", "voda", "kvasinky", "živná sůl")

 'počíteční hodnoty ve formuláři
 OK = False
 frmMenuVolba.Caption = "Volba druhu vína a jeho množství"
 Druh = frmMenuVolba.Controls(1).Caption
 frmMenuVolba.Controls(1).Value = True
 A = frmMenuVolba.Controls("txtMnozstvi").Value

 'zobrazení formuláře
 frmMenuVolba.Show

 'blok proběhne po stisku tl. OK
 If OK Then

     'test zadání množství
     If A = 0 Then
         MsgBox "Bylo zadáno chybné, " & _
          "nebo nulové množství, aplikace bude ukončena"
         Exit Sub
     End If
     Ingr = Ingred_prepoc(A) 'přepočet

     'zápis do tabulky
     Select Case Druh
         Case "borůvkové", "ostružinové", "malinové"
             Cells(1, 1) = Druh & " na " & A & " l"
             Zapis1 (1)
             Cells(7, 1) = "nebo"
             Zapis2 (7)
         Case "rybízové", "angreštové", "třešňové", _
          "višňové", "jablečné", "hruškové"
             Cells(1, 1) = Druh & " na " & A & " l"
             Zapis1 (1)
         Case Else '"šípkové", "trnkové", "jeřabinové", _
          '"ze sušených šípků", "ze sušeného ovoce", _
          '"z povidel", "z chlebových kůrek"
             Cells(1, 1) = Druh & " na " & A & " l"
             Zapis2 (1)
     End Select
 End If
End Sub
'_____________________________________________________________________________

Private Function Ingred_prepoc(Mnozstvi As Single) As Variant
    'přepočet surovin
    Dim Pom, I As Byte
    Pom = Array(Stava, Cukr, Cukr1, Voda, Ovoce, Kvasinky, Sul)
    For I = 0 To 6
        Pom(I) = Pom(I) * Mnozstvi
    Next I
    Ingred_prepoc = Pom
End Function
'_____________________________________________________________________________
```

```
Private Sub Zapis1(I As Byte)
    For J = 1 To 5
        Cells(I + 2, J) = Tab1(J - 1)
    Next J
    Cells(I + 3, 1) = Ingr(0) & " l"
    Cells(I + 3, 2) = Ingr(1) & " Kg"
    Cells(I + 3, 3) = Ingr(3) & " l"
    Cells(I + 3, 4) = Ingr(5) & " balení"
    Cells(I + 3, 5) = Ingr(6) & " balení"
End Sub
'_____________________________________________________________________________

Private Sub Zapis2(I As Byte)
    For J = 1 To 5
        Cells(I + 2, J) = Tab2(J - 1)
    Next J
    Cells(I + 3, 1) = Ingr(4) & " Kg"
    Cells(I + 3, 2) = Ingr(2) & " Kg"
    Cells(I + 3, 3) = Ingr(3) & " l"
    Cells(I + 3, 4) = Ingr(5) & " balení"
    Cells(I + 3, 5) = Ingr(6) & " balení"
End Sub
```

Makro č. 9 Příprava vín 2 – kód formuláře

```
Option Explicit
'_____________________________________________________________________________

Private Sub cmdOK_Click()
    OK = True
    Unload Me
End Sub
'_____________________________________________________________________________

Private Sub optRybiz_Click()
    If Druh <> "rybízové" Then
        Druh = ActiveControl.Caption
    End If

    Stava = 0.35
    Cukr = 0.225    'množství surovin na 1 litr
    Voda = 5 / 10
End Sub
'_____________________________________________________________________________

Private Sub optAngrest_Click()
    Druh = ActiveControl.Caption
    Stava = 5 / 10
    Cukr = 0.225
    Voda = 0.35
End Sub
'_____________________________________________________________________________

Private Sub optTresne_Click()
    Druh = ActiveControl.Caption
    Stava = 7 / 10
    Cukr = 0.175
    Voda = 0.2

End Sub
'_____________________________________________________________________________

Private Sub optVisne_Click()
    Druh = ActiveControl.Caption
```

```
    Stava = 4 / 10
    Cukr = 0.225
    Voda = 0.45

End Sub
'_____________________________________________________________________________

Private Sub optBoruvky_Click()
    Druh = ActiveControl.Caption
    Stava = 6 / 10
    Cukr = 2 / 10
    Voda = 3 / 10
    Ovoce = 0.75
    Cukr1 = 0.225
End Sub
'_____________________________________________________________________________

Private Sub optJablka_Click()
    Druh = ActiveControl.Caption
    Stava = 8 / 10
    Cukr = 0.175
    Voda = 0.1
End Sub
'_____________________________________________________________________________

Private Sub optHrusky_Click()
    Druh = ActiveControl.Caption
    Stava = 8 / 10
    Cukr = 0.175
    Voda = 0.1
End Sub
'_____________________________________________________________________________

Private Sub optOstruziny_Click()
    Druh = ActiveControl.Caption
    Stava = 0.55
    Cukr = 0.225
    Voda = 0.3
    Ovoce = 0.7
    Cukr1 = 0.25
End Sub
'_____________________________________________________________________________

Private Sub optMaliny_Click()
    Druh = ActiveControl.Caption
    Stava = 0.55
    Cukr = 0.225
    Voda = 0.3
    Ovoce = 0.35
    Cukr1 = 0.25
End Sub
'_____________________________________________________________________________

Private Sub optSipky_Click()
    Druh = ActiveControl.Caption
    Ovoce = 0.3
    Cukr1 = 0.25
End Sub
'_____________________________________________________________________________

Private Sub optTrnky_Click()
    Druh = ActiveControl.Caption
    Ovoce = 0.35
    Cukr1 = 0.25
End Sub
'_____________________________________________________________________________

Private Sub optJerabiny_Click()
```

```
    Druh = ActiveControl.Caption
    Ovoce = 0.3
    Cukr1 = 0.25
End Sub
'_____________________________________________________________________________

Private Sub optSussipky_Click()
    Druh = ActiveControl.Caption
    Ovoce = 0.1
    Cukr = 0.25
End Sub
'_____________________________________________________________________________

Private Sub optSusovoce_Click()
    Druh = ActiveControl.Caption
    Ovoce = 0.15
    Cukr1 = 0.225
End Sub
'_____________________________________________________________________________

Private Sub optPovidla_Click()
    Druh = ActiveControl.Caption
    Ovoce = 0.1
    Cukr1 = 0.2
End Sub
'_____________________________________________________________________________

Private Sub optChleba_Click()
    Druh = ActiveControl.Caption
    Ovoce = 0.06
    Cukr1 = 0.25
End Sub
'_____________________________________________________________________________

Private Sub spbMnozstvi_Change()
    txtMnozstvi.Value = spbMnozstvi.Value
    A = spbMnozstvi.Value
End Sub
'_____________________________________________________________________________

Private Sub txtMnozstvi_Change()
    On Error Resume Next
    spbMnozstvi.Value = txtMnozstvi.Value
    If Err <> 0 Then
        A = 0
    End If
End Sub

Private Sub UserForm_Click()

End Sub
```

6.6 Makro č. 10 Výkaz 2010

Úkolem posledního makra je překopírovat souhrnné údaje z pracovních deníků všech zaměstnanců za jednotlivé měsíce do sešitu, kde výkazy představující měsíce jsou obsaženy v jednotlivých listech od 1. měsíce (ledna) do posledního (prosince) za rok 2010. V každém listu (měsíci) je seznam všech zaměstnanců a ke každému zaměstnanci

jsou zkopírovány součty položek (úkolů) z pracovního deníku, který zaměstnanec vyplní na konci každého měsíce.

První část makra otevře dialog pro výběr adresáře s pracovními deníky (viz CD adresář MAKRA\08-Výkaz_2010-vst_data), dále je otevřen formulář pro výběr měsíce. Následuje cyklus, který má 2 průběhy. Poprvé postupně otevírá vstupní soubory a kontroluje to, zda souhlasí jména zaměstnanců vyplněná v denících se jmény ve výkazu. Pokud se některé jméno neshoduje nebo chybí ve výkazu (přibyl nový zaměstnanec), otevře se formulář, který vypíše názvy vstupních souborů s neshodnými jmény a po stisknutí tlačítka OK nebo křížku je aplikace ukončena. Pokud se všechna jména shodují, tak v druhém průběhu cyklu již dochází ke kopírování dat do listu vybraného měsíce.

Makro č. 10 Výkaz 2010 – kód modulu

```
Option Explicit

Public strPom As String, Nazev_Mesice As String
Public OK As Boolean, Chybna_jmena(100)

'-----------------dialog pro výběr adresáře [2] str. 284,285------------------
'deklarace API
Declare Function SHGetPathFromIDList Lib "shell32.dll" _
 Alias "SHGetPathFromIDListA" (ByVal Pidl As Long, _
 ByVal pszPath As String) As Long

Declare Function SHBrowseForFolder Lib "shell32.dll" _
 Alias "SHBrowseForFolderA" (lpBrowseInfo As BROWSEINFO) As Long
'.............................................................
Public Type BROWSEINFO
    hOwner As Long
    pidlRoot As Long
    pszDysolayName As String
    lpszTitle As String
    ulFlags As Long
    lpfn As Long
    lparam As Long
    lImage As Long
End Type
'_____________________________________________________________________________

Private Function Getdirectory(Optional Msg) As String
'funkce pro výběr adresáře
    Dim bInfo As BROWSEINFO, Path As String
    Dim R As Long, X As Long, Pos As Integer
    bInfo.pidlRoot = 0&

    'titulek dialogu
    If IsMissing(Msg) Then
        bInfo.lpszTitle = "Vyberte adresář"
    Else
        bInfo.lpszTitle = Msg
    End If

    'typ adresáře, který se vrací
    bInfo.ulFlags = &H1
```

```
    'zobrazení dialogu
    X = SHBrowseForFolder(bInfo)

    Path = Space$(512)
    R = SHGetPathFromIDList(ByVal X, ByVal Path)
    If R Then
        Pos = InStr(Path, Chr$(0))
        Getdirectory = Left(Path, Pos - 1)
    Else
        Getdirectory = ""
    End If
End Function
'-----------------------dialog pro výběr adresáře konec-----------------------
'
'_____________________________________________________________________________

Private Function CisloM() As String
'funkce vyvolá formulář pro výběr měsíce a vrátí
 'číslo vybr. měsíce
    strPom = ""
    frmMenuVyber.Controls("lstMesice").List = Array("leden", _
     "únor", "březen", "duben", "květen", "červen", _
     "červenec", "srpen", "září", "říjen", "listopad", "prosinec")
    frmMenuVyber.Show
    CisloM = strPom
End Function
'_____________________________________________________________________________

Sub Kopirovani_udaju()

Dim Cesta As String, Soubor As String, Polozka As String
Dim Jmeno_zam As String, Nazev_s1 As String, Nazev_s2 As String
Dim Cislo_mesice As String, Komentar As String
Dim J As Byte, Pozice_jmena As Byte, K As Byte
Dim Pocet_cyklu As Byte, Pol_celkem As String
Dim Nenasel As Boolean, Cyklus_probehl As Boolean

OK = False
Cyklus_probehl = False
Nazev_s1 = ActiveWorkbook.Name

'výběr adresáře
'Cesta = InputBox("", , "F:\0-diplomka\výkaz_09\2010\")
Komentar = "Vyber adresář s výkazy zaměstnanců"
Cesta = Getdirectory(Komentar)
If Cesta = "" Then
    Exit Sub
End If
Cesta = Cesta & "\"
Cislo_mesice = CisloM 'fce vyvolá formulář pro výběr měsíce
If Not OK Then
    Exit Sub
End If
Worksheets(Nazev_Mesice).Activate
Application.ScreenUpdating = False
Pocet_cyklu = 0
For K = 0 To 1 'poprvé probíhá kontrola jmen, podruhé kopie dat
    Nenasel = False
    Soubor = Dir(Cesta)
    Do While Soubor <> ""

        'otevírání souborů podle vybraného měsíce
        If Mid(Soubor, 3, 2) = Cislo_mesice Then
            Workbooks.Add (Cesta + Soubor)
            Nazev_s2 = ActiveWorkbook.Name

            'test na správnost souboru
            If Cells(1, 1) = "Pracovní deník" Then
```

```
                Jmeno_zam = Cells(2, 3)
                Workbooks(Nazev_s1).Activate
                Pozice_jmena = 3

                'nalezení řádku se souhl jménem zaměstnance
                Do While Cells(Pozice_jmena, 2) <> Jmeno_zam
                    Cyklus_probehl = True
                    If Cells(Pozice_jmena, 2).Value = Empty _
                     Then
                        Chybna_jmena(Pocet_cyklu) = Nazev_s2
                        Pocet_cyklu = Pocet_cyklu + 1
                        Nenasel = True
                        Exit Do
                    End If
                    Pozice_jmena = Pozice_jmena + 1
                Loop
                If K = 1 Then
                    Workbooks(Nazev_s2).Activate
                    J = 3

                    'kopírování údajů
                    Do While Cells(4, J + 1) <> "poznámky"
                        Cells(37, J).Copy
                        Workbooks(Nazev_s1).Activate
                        Cells(Pozice_jmena, J).PasteSpecial _
                         Paste:=xlPasteValues 'vložení hodnot
                        Application.CutCopyMode = False
                        Workbooks(Nazev_s2).Activate
                        J = J + 1
                    Loop
                End If
            End If
            Workbooks(Nazev_s2).Close
        End If

        Soubor = Dir 'skok na další soubor
    Loop
    If Nenasel Then
        frmHlaseni.Controls("lstSoubory").List = Chybna_jmena
        frmHlaseni.Show
        Exit Sub
    End If
Next K
Application.ScreenUpdating = True
If Not Cyklus_probehl Then
    MsgBox "V zadaném adresáři nebyly nalezeny soubory " & _
           "výkazů za " & Nazev_Mesice, vbCritical
End If

End Sub
```

Makro č. 10 Výkaz 2010 – kód formuláře výběru měsíce

```
Private Sub cmdOK_Click()
    If strPom <> "" Then
        OK = True
        Unload Me
    End If
End Sub
'_____________________________________________________________________

Private Sub cmdStorno_Click()
    Unload Me
End Sub
'_____________________________________________________________________
```

```
Private Sub lstMesice_Click()
    If (lstMesice.ListIndex) + 1 < 10 Then
        strPom = "0" & (lstMesice.ListIndex) + 1
    Else
        strPom = (lstMesice.ListIndex) + 1
    End If
    Nazev_Mesice = (lstMesice.List(lstMesice.ListIndex))
End Sub


Private Sub UserForm_Click()

End Sub
```

Makro č. 10 Výkaz 2010 – kód formuláře hlášení neshod jmen

```
Private Sub cmdOK1_Click()
    Unload Me
End Sub

Private Sub UserForm_Click()

End Sub
```

7 Závěr

Účelem práce bylo popsání a porovnání možností vytváření maker v tabulkových procesorech, které vyústilo ve vytvoření několika ukázkových aplikací. Při procesu jejich vzniku jsem si ověřil teoretické zásady programování v praxi, seznámil jsem se s novým programovacím jazykem.

V současné době je VBA součástí již řady aplikací nejenom od firmy Microsoft®. Jako příklad mohu uvést Corel® Graphics Suit, Statistica, Bentley® Microstation, Arc Gis, Kokeš, aj. Díky vlastní zkušenosti jsem dospěl k závěru, že pro uživatele, který se chce naučit vytvářet makra v různých aplikacích, je většinou výhodnější začít s MS Excel. Tento editor nabízí řadu pomocníků jako záznamník maker, ladící nástroje pro odhalování chyb, prohlížeč objektů aj. Kromě toho je o vytváření maker v MS Excel na našem trhu více literatury.

Hlubší poznání zásad programování ve VBA a získané zkušenosti při tvorbě vlastních maker mně významně přispějí i k lepší efektivitě zpracování dat v mém zaměstnání.

8 Použitá literatura

[1] ČERNÝ, Jaroslav. *Excel 2000-2007 : záznam, úprava a programování maker*. 2.,aktualizované vydání. Praha : Grada Publishing, a.s., 2008. 192 s. ISBN 978-80-247-2305-1.

[2] WALKENBACH, John. *Microsoft Excel 2000 & 2002 : programování ve VBA*. Vydání druhé. Brno : Computer press, 2004. 705 s. ISBN 80-7226-547-4, EAN: 9788072265473.

[3] DODGE, Mark; STINSON, Craig. *Mistrovství v Microsoft Office Excel 2007*. Vydání první. Brno : Computer Press, a.s., 2008. Část IX Automatizace Excelu, s. 781-818. ISBN 978-80-251-1980-8.

[4] VÍT, Svatopluk. OpenOffice.org útočí potřetí. *OpenMagazin*. 10. 1. 2009, 2009, 1, s. 1-3. Dostupný také z WWW: <http://www.openmagazin.cz/2009/01/openmagazin-12009/>.

[5] PECINOVSKÝ, Josef. *OpenOffice.org 2.0 : kompletní průvodce*. Praha : Grada Publishing, a.s., 2006. 292 s. ISBN 80-247-1016-1.

[6] THOMPSON, James M. *Porting Excel/VBA to Calc/StarBasic* [online]. [s.l.] : OpenOffice.org , 6.6. 2004 [cit. 2010-02-10]. Dostupné z WWW: <http://documentation.openoffice.org/HOW_TO/various_topics/VbaStarBasicXref.pdf>.

[7] SOBEK, Milan. *OpenOffice.org : tipy a triky pro záznam aúpravu maker*. První vydání. Praha : Grada Publisching, a.s., 2006. 152 s. ISBN 80-247-1374-8.

[8] ČERNÝ, Jaroslav. *2000, 2002, 2003*. První vydání. Praha : Grada Publishing, a.s., 2005, 1. dotisk 2006. 228 s. ISBN 80-247-0922-8.

[9] MAŠÁT, Karel; NĚMEČEK, Jan; TOMIŠKA, Zdeněk. Metodika vymezování a mapování bonitovaných půdně ekologických jednotek. třetí přepracované a doplněné. Praha : VÚMOP Praha, 2002. Aktualizační karta BPEJ, s. 38-41. ISBN 80-238-9095-6.

9 Obsah přiloženého CD

- text bakalářské práce v tištěné podobě ve formátu PDF
- soubory s makry ve formátu xls, ods
- soubory se vstupními daty k některým makrům