Univerzita Hradec Králové
Přírodovědecká fakulta
Katedra Informačních technologii

Programování v OpenOffice.org

Bakalářská práce

Autor: Michal Tydrych
Studijní program:
Studijní obor: Tělovýchovné a sportovní aktivity se zaměřením na vzdělávání
Informatika se zaměřením na vzdělávání
Vedoucí práce: doc. RNDr. Štěpán Hubálovský, Ph.D.

Univerzita Hradec Králové
Přírodovědecká fakulta

Programování v OpenOffice.org

Autor: Michal Tydrych

Studijní program: B1801

Studijní obor: Informatika se zaměřením na vzdělání
Tělovýchovné a sportovní aktivity se zaměřením na vzdělávání

Název práce: Programování v OpenOffice.org

Název práce v AJ: Programming in OpenOffice.org

Cíl a metody práce: Cílem bakalářské práce je zmapovat a ukázat možnosti programování v kancelářském software OpenOffice.org. V rámci teoretické části práce, která je východiskem praktické části práce, bude pojednáno o kancelářském SW balíku OpenOffice. org a o možnostech programování v tomto software. V rámci praktické části práce autor vypracuje jednak učebnici programování v OpenOffice.org, jednak pomocí komparativní analýzy porovná programování v Microsoft Office a OpenOffice.org.

Garantující pracoviště: katedra Informatiky Přírodovědecké fakulty UHK

Vedoucí práce: doc. RNDr. Štěpán Hubálovský, Ph.D.

Prohlášení:

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a že jsem v seznamu použité literatury uvedl všechny použité prameny a literaturu, z kterých jsem vycházel.

V Hradci Králové dne Michal Tydrych

Poděkování:

- děkuji panu doc. RNDr. Štěpánu Hubálovskému, Ph.D za poskytnuté konzultace k mé bakalářské práci

Anotace:

Cílem bakalářské práce je zmapovat a ukázat možnosti programování v kancelářském software OpenOffice.org. V rámci teoretické části práce, která je východiskem praktické části práce, bude pojednáno o kancelářském SW balíku OpenOffice. org a o možnostech programování v tomto software. V rámci praktické části práce autor vypracuje jednak učebnici programování v OpenOffice.org, jednak pomocí komparativní analýzy porovná programování v Microsoft Office a OpenOffice.org.

Annotation:

The aim of bachelor thesis is to map and show the possibilities of programming in OpenOffice.org office software. In the theoretical part, which is the basis of the practical work, will be dealt with office software package OpenOffice.org and programming options in this software. In the practical part, the author draw both textbook programming in OpenOffice.org, both through comparative analysis compares programming in Microsoft Office and OpenOffice.org.

Obsah

ÚVOD 5

1 CO JE TO OPENOFFICE.ORG 6

1.1 Historie OpenOffice.org 6

2 CO JSOU TO MAKRA A STAROFFICE BASIC 7

2.1 StarOffice Basic 8

3 ZÁZNAM A ULOŽENÍ MAKRA V OPENOFFICE.ORG 9

3.1 Nahrání a uložení makra 12

3.2 Pravidla ukládání maker do kontejnerů 13

4 PROGRAMOVÁNÍ VE STAROFFICE BASIC 14

4.1 Základní pravidla pro zápis procedury 15

4.1.1 Řádkování a mezery 15

4.1.2 Popis makra - Komentáře 15

4.1.3 Názvy proměnných a procedur 16

4.1.4 Příkazy a funkce 16

4.1.5 Operátory 17

4.1.5.1 Matematické operátory 17

4.1.5.2 Logické operátory 17

4.1.5.3 Porovnávací operátory 18

4.2 Základy jazyka StarOffice Basic 18

4.2.1 Proměnné 18

4.2.1.1 Typy proměnných 18

4.2.1.1.1 Celočíselné typy 19

4.2.1.1.2 Typ pro čísla s desetinou čárkou 20

4.2.1.1.3 Měnový typ 20

4.2.1.1.4 Datumový typ 21

4.2.1.1.5 Textový typ 21

4.2.1.1.6 Logický typ 21

4.2.1.1.7 Typ Variant 21
4.2.1.2 Deklarace proměnných 22
4.2.1.3 Globální a privátní proměnné 25
4.2.1.4 Pole (array) 28
4.2.1.5 Konstanty 30
4.2.2 Funkce a příkazy 30
4.2.2.1 Funkce pro práci s proměnnými 31
4.2.2.2 Funkce pro práci s obrazovkou 33
4.2.2.2.1 Příkaz Print 34
4.2.2.2.2 Funkce a příkaz MsgBox 34
4.2.2.2.3 Funkce InputBox 36
4.2.2.2.4 Příkazy pro řízení běhu programu 37
4.2.2.2.4.1 Příkazy pro větvení programu 37
4.2.2.2.4.2 Cykly 40
4.2.2.2.4.3 Příkaz GoTo 42
4.2.2.3 Matematické funkce 43
4.2.2.3.1 Trigonometrické funkce 43
4.2.2.3.2 Exponenciální a logaritmické funkce 44
4.2.2.3.2.1 Generování náhodných čísel 44
4.2.2.3.2.2 Ostatní číselné funkce 44
4.2.2.4 Funkce pro práci s řetězci 45

5 POROVNÁNÍ MOŽNOSTÍ PROGRAMOVÁNÍ V MS EXCEL (VBA) A OPENOFFICE.ORG (STAROFFICE BASIC) 47

ZÁVĚR 51

PŘEHLED PRAMENŮ 52

Seznam použité literatury 52

Seznam obrázků 53

Seznam tabulek 54

Seznam příkladů 54

Úvod

Tato bakalářská práce by měla ukázat možnosti maker a programování v jazyce StarOffice Basic a jejich využití při výuce programování na základních a středních školách, aniž by bylo nutno zakoupit jakýkoli komerční program.

Téma této práce jsem si vybral z toho důvodu, že prakticky neexistuje ucelený návod v českém jazyce pro práci a programování s makry v kancelářském balíku OpenOffice.org, který by mohl sloužit i jako učebnice.

Cílem této bakalářské práce je vytvořit učebnici, která čtenáři ukáže, co jsou to makra a co je to programovací jazyk StarOffice Basic a jak se s ním pracuje v kancelářském balíku OpenOffice.org. Příručka je určena pro všechny čtenáře, kteří nemají zkušenosti s programováním maker, avšak ovládají základní obsluhu OpenOffice.org. Čtenáři se v této publikaci postupně naučí, jak nahrát a posléze i naprogramovat makra, dále se naučí základy programovacího jazyka Basic.

V první části si ukážeme co je to OpenOffice.org, co jsou to makra a jak se pracuje s jazykem Basic, jakožto hlavním programovacím jazykem pro tvorbu maker.

V druhé části, která bude konkrétnější, si předvedeme způsoby jak nahrávat a ukládat makra v OpenOffice.org, jak lze makra spouštět a upravovat.

Ve třetí části, se budeme věnovat podrobně programovacímu jazyku Basic, kde se čtenář naučí základy strukturovaného programování v jazyku StarOffice Basic. Budou zde vysvětleny základní prvky jazyka, jako jsou proměnné, funkce a příkazy. Podkapitoly budou doplněny o názorné příklady, na kterých budou vysvětleny jednotlivé prvky jazyka StarOffice Basic.

Poslední čtvrtá část této bakalářské práce bude pojednávat o rozdílu mezi prostředím a samotným programovacím jazykem v programech OpenOffice.org a jeho StarOffice Basic a Microsoft Office Excel a jeho VBA (Visual Basic for Aplication), kde pomocí komparativní analýzy porovnáme možnosti obou jazyků.

1 Co je to OpenOffice.org

OpeOffice.org je plnohodnotný kancelářský balík, který se řadí po bok slavnějších předchůdců jako je například Microsoft Office. Jeho hlavní výhodou je, že je vydáván pod svobodnou licencí LGPL 3.0, která mimo jiné umožnuje tento balík programů svobodně šířit. Na stránkách openoffice.org je citát, který poukazuje na účel projektu: *„Mission statement: To create, as a community, the leading international office suite that will run on all major platforms and provide access to all functionality and data through open-component based APIs and an XML-based file format."*

Neboli v předkladu: „Cíl projektu: Jako komunita vytvořit mezinárodně používaný kancelářský balík, který bude možné provozovat na všech využívaných platformách a poskytnout přístup ke všem funkcím a datům pomocí otevřených API a souborových formátů založených na XML."[1]

1.1 Historie OpenOffice.org

Kancelářský balík OpenOffice.org jako projekt vzešel z komerčního balíku StarOffice, který byl vyvíjen firmou StarDivision, která byla založena v Německu roku 1986. V roce 1999 společnost Sun Microsystem vstupuje do vývoje balíku koupí společností StarDivision. Tím to krokem se chtěl Sun Microsystem dostat do konkurenčního boje s Microsoftem a jeho kancelářským balíkem MS Office.

Mezi jeden z důvodů, proč Sun odkoupil StarDivision, byla cena licencí MS Office. Sun v té době zaměstnával cca 42 tisíc zaměstnanců. Ve výsledku pro společnost bylo výhodnější koupit společnost StarDivision, než pořizovat licence na provoz konkurenčního produktů společnosti Microsoft a také potřeboval kancelářský balík pro svůj operační systém Solaris, s kterým MS Office není kompatibilní.

Netrval dlouho a v roce 2000 byla vydaná verze StarOffice 5.2, která byla k dispozici zdarma pro osobní použití. Poté následovalo uvolnění velké části zdrojových kódů kancelářského balíku, čímž byl položen základ komunitního projektu OpenOffice.org.

Sun dále vyvíjel i komerční verzi StarOffice, která navíc obsahovala některá rozšíření chráněna autorskými právy (např. písma, obrázky, šablony), avšak nadále podporoval komunitu zabývající se vývojem balíku openoffice.org.

Vývoj obou kancelářských balíků probíhá paralelně a navzájem se doplňuje. Od verze StarOffice 6.0 jsou zdrojové kódy, souborové formáty a jiné komponenty přebrány od komunity OpenOffice.org. Tyto dva projekty se tak navzájem doplňují. [2][3]

2 Co jsou to makra a StarOffice Basic

Slovo makro vzniklo z Řeckého slova „μακρό“ což znamená velké. Do informatiky bylo převedeno jako makroinstrukce. Počítač pracuje na základě zadaných instrukcí, které vykonává CPU. Instrukcí může být stisk klávesy nebo tlačítka myši. Instrukcí může být i příkaz napsaný v programovacím jazyce. Několik takovýchto instrukcí vytvořených v programovacím jazyku tvoří makroinstrukci – naše makro. V kancelářských balících označuje makro posloupnost akcí, funkcí nebo příkazů, které automatizují rutinní operace, umožnují jednoduší zpracování a vyhodnocování dat. Uživatel nejvíce ocení význam maker při práci s rozsáhlými datovými soubory. Laicky řečeno jsou tedy makra v kancelářském balíků OpenOffice.org programy, které mají za úkol zjednodušit uživateli práci a rozšířit možnosti kancelářského balíku nad jeho základní funkce. Obyčejnému uživateli se základní znalostí programování je dána možnost makro nahrát nebo ho naprogramovat. Tyto programy mohou být maličké, jako je například odstranění speciálních znaků nebo vložení data

a času, mohou být samozřejmě i dosti rozsáhlé. Příkladem rozsáhlého makra je Dmaths [dmaths.org], které například umožnuje editaci a vkládání matematických vzorců.

Drobným nedostatkem OpenOffice.org je, že neumí spouštět makra vytvořené v Microsoft Office. Microsoft Office používá pro tvorbu maker VBA (Visual basic for Applications) za to OpenOffice.org nabízí své vlastní, odlišné, rozhraní pro tvorbu maker kterým je StarOffice Basic, který je založený na prostředí OpenOffice.org API.

I když je programovací jazyk pro oba balíky stejný (Basic), implementace objektů, struktur je odlišná. Proto se uživatelé při importu maker z MS do OO.org setkávají s nekompatibilitou. Což nemusí být však vždy jen nevýhodou. Z druhého pohledu je to výhoda a to z hlediska možného napadení virem. MS Office jsou známi svou náchylnosti na tzv. makroviry. Ty ještě do nedávna pro OpenOffice.org vůbec neexistovali. [4]

2.1 StarOffice Basic

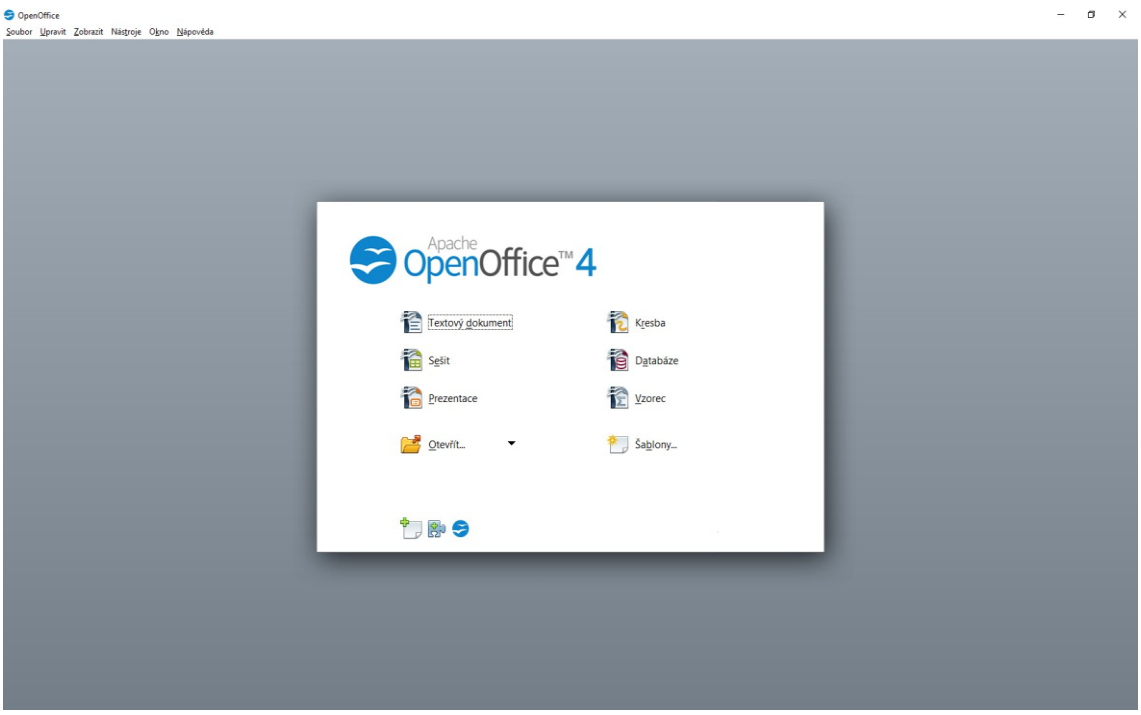
StarOffice basic je programovací jazyk vyvinutý speciálně pro OpenOffice.org, ale je integrován do celého balíku. Jak již název napovídá, jedná se o programovací jazyk rodiny Basic. Každý, kdo předtím někdy pracoval s Visual Basic nebo Visual Basic for application od Microsoftu, si rychle zvykne i na StarOffice basic. Velká část základní struktury StarOffice basic je kompatibilní s Visual Basic.

StartOffice basic lze rozdělit na 4 části. První částí je samotný jazyk StarOffice basic, který definuje základní jazykové konstruktory, například pro deklaraci proměnných a funkcí. Další částí je runtime knihovna, která poskytuje standartní funkce, které se přímo nevztahují k StarOffice, například funkce pro editaci čísel, textových řetězců a dat. Třetí část je StarOffice API (aplikační programové rozhraní), které umožnuje přístup k dokumentům, jejich vytvoření, editaci, uložení a samozřejmě vytisknutí a další možnosti. Poslední částí je Dialogové okno editoru, díky kterému je možné přidat ovládací prvky pro obsluhu událostí. [5]

3 Záznam a uložení makra v Openoffice.org

V úvodu jsme se dozvěděli, že makro je soubor instrukcí. Makro, neboli makoinstrukce, je tvořena souborem příkazů napsaných v programovacím jazyce. Instrukcí může být takřka vše od stisku klávesy po výběr z panelu nástrojů. Proto existují dva základní typy tvorby maker – a to je přímo naprogramování samotných příkazů, příkaz po příkazu, nebo jednoduší způsob a to je vytvoření makra pomocí nástroje záznam makra. Zde se dají tvořit jednoduchá makra, která nám napomáhají v nejrutinnějších operacích.

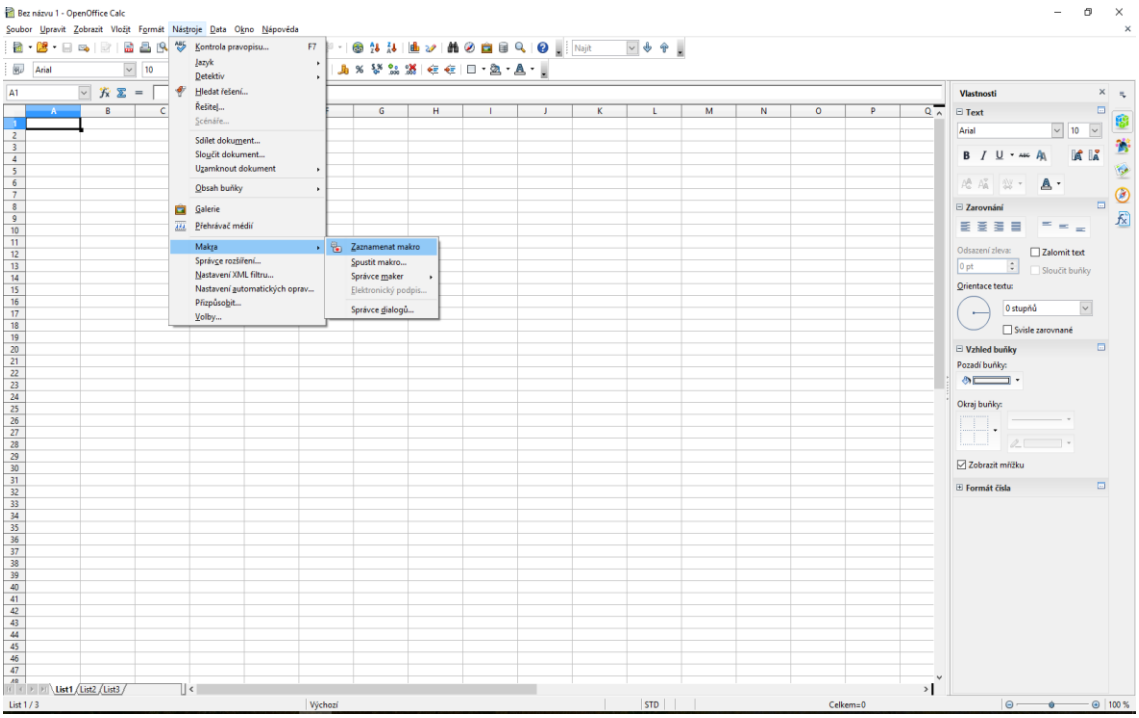
V malé ukázce si vytvoříme makro, které nám v Calcu, sloučí tři buňky, do nich napíše text HLAVIČKA TABULKY a text vycentruje do středu. První věcí co musíme udělat je spustit tabulkový editor balíku OpenOffice.org, který nese název CALC. Po instalaci balíku OpenOffice.org se na ploše vytvořila ikona, pomocí které je možné otevřít jakoukoli část kancelářského balíku, my otevřeme SEŠIT (viz obrázek 1).



Obr. 1: Úvodní strana OpenOffice.org

Automaticky se založí dokument s názvem **Bez názvu 1.** V panelu nabídek vybereme položku **Nástroje – Makra**, kde v rozbalené podnabídce vybereme **zaznamenat makro** (viz obrázek 2).

Zobrazí se v levém horním rohu okno **Zaznamenat,** s jediným tlačítkem zastavit nahrávání. Od té doby vše co v okně uděláme, se nahrává. Nezaznamenává se čas, ale pouze vykonané akce, v pořadí v kterém je uděláme. V okně označíme tři buňky v řádku, v nabídce vybereme **Formát – sloučit buňky,** dále v nabídce **Formát – Zarovnání – Na střed** nastavíme vycentrování textu. V dokumentu se nám sloučí vybrané tři buňky a kurzor se přesune do středu. Nyní na klávesnici stiskneme klávesu Caps Lock, čímž nastavíme psaní velkých písmen a napíšeme text **HLAVIČKA TABULKY.** Záznam makra ukončíme stiskem tlačítka **Zastavit nahrávání**. Zobrazí se nám dialogové okno **Makra v OpenOffice Basic.** V tomto okně vybereme v položce **Uložit makro do – Bez názvu 1,** kde se proklikáme na položku **Standard.** Poté klikneme na tlačítko **Nový modul**, kde nám vyskočí okno s předvyplněným názvem Modul 1, který změníme například na „Makra“ a potvrdíme tlačítkem OK. Nyní nám pod řádkem Standard přibyl řádek Makra. Před stiskem tlačítka **Uložit** ještě změníme v levém horním rohu okna název makra na **první_makro.**

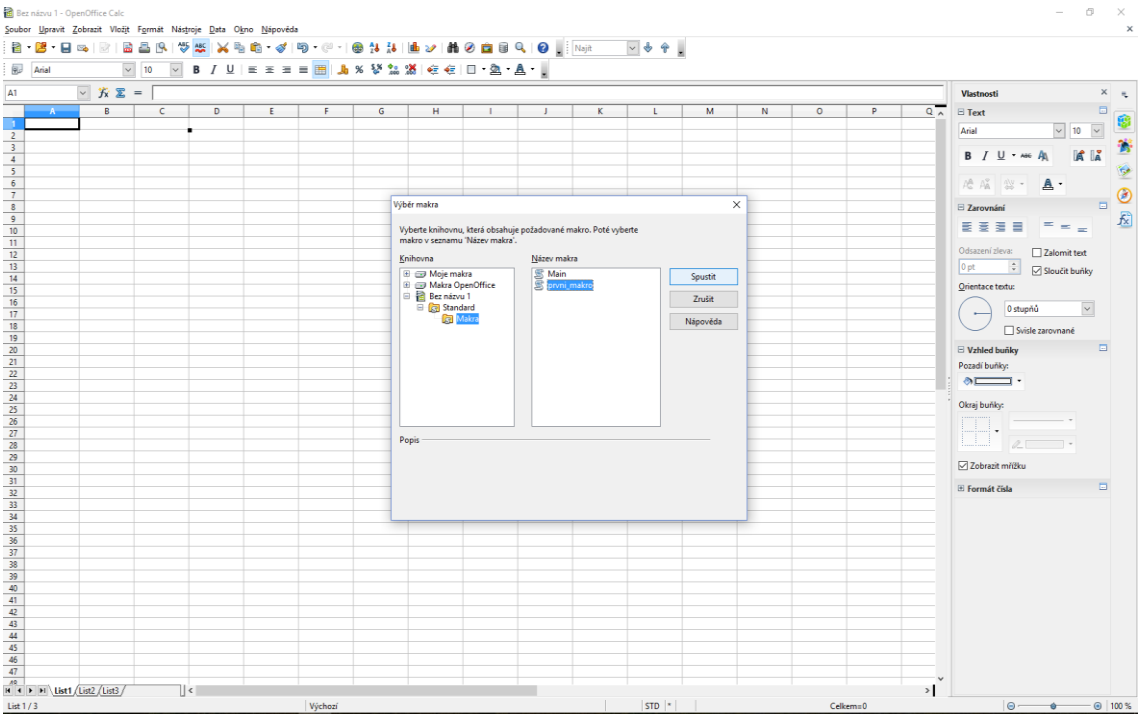


Obr. 2: Záznam makra

Nyní jsme úspěšně zaznamenali a uložili naše první makro v OpenOffice.org.

Teďka se podíváme na makro, které jsme vytvořili. V dokumentu vše smažeme. V nabídce **Nástroje – Makra – Spustit makro…** zobrazíme okno **Výběr**

makra. V knihovně rozklikneme položku **Bez názvu 1 – Standard – Makra,** poté se vpravo v položce **Název makra** objeví dvě položky Main a naše první_makro. Označíme si první_makro a stiskneme tlačítko **Spustit**. V tu chvíli se provedou krok po kroku všechny akce tak jak jsme je nahráli (viz obrázek 3).



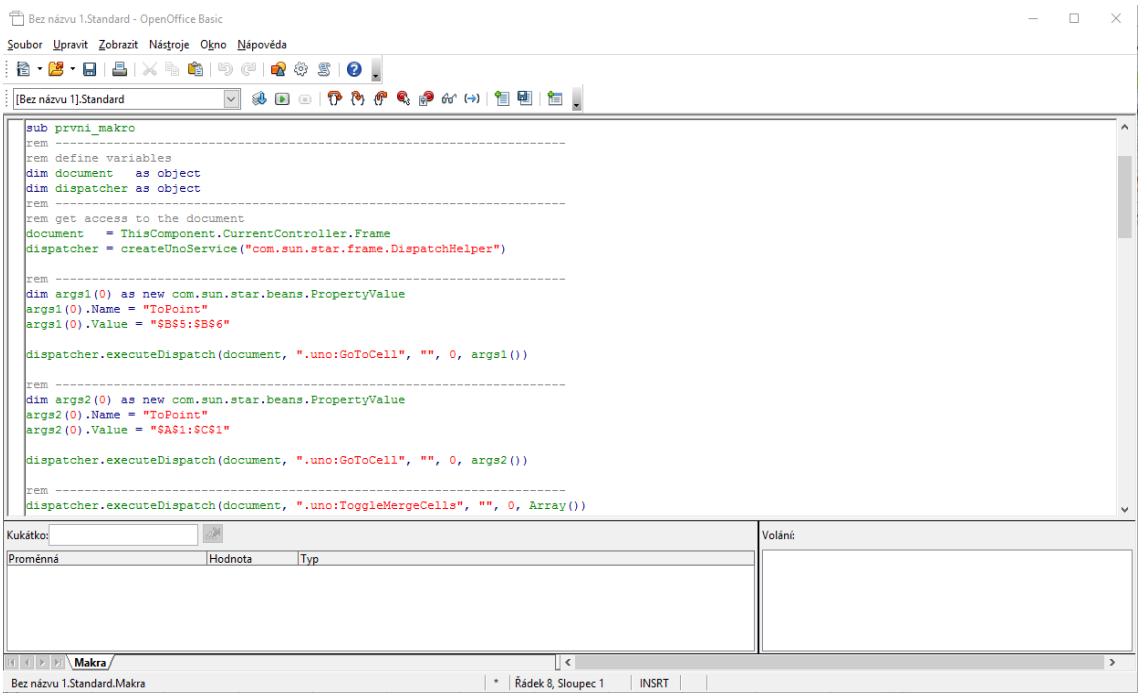
Obr 3: Spuštění makra

Tímto jsme ověřili funkčnost naše prvního makra a nyní se podíváme, jak vlastně vypadá jeho zdrojový kód. V položce **Nástroje – Makra – Správce maker – OpenOffice Basic...** Zobrazí se nám již známé okno Makra, kde v levém položce se proklikáme až k položce **Makra** a v tu chvíli se v levé položce zobrazí naše makro s názvem první_makro. Stiskneme tlačítko **Upravit.** Zobrazí se nám nové okno (viz obrázek 4) ve kterém je vidět kód makra.

Takto vypadá naše jednoduché makro v programovacím jazyce OpenOffice Basic. Možná někoho překvapí délka a složitost kódu, který je potřeba k napsání několika slov. Tímto se samozřejmě nenechte zastrašit, právě jste vytvořili své první jednoduché makro. I takováto jednoduchá makra vám mohou hodně usnadnit práci. Někdo by si mohl myslet, že makra lze vytvářet pouze v Calcu, kde

pravda mají trochu větší smysl, ale není tomu tak, makra lze vytvářet také v textovém editoru. Postup záznamu je obdobný jako v tabulkovém procesoru.

V okně programu na obr. 4 vidíme, že na začátku programu je klíčové slovo **sub,** což je zkratka vycházející z anglického slova subroutine, což v českém jazyce znamená podprocedura a za ním se nachází jméno **první_makro.** Jazyk OpenOffice Basic nerozlišuje pojmy procedura a podprocedura a proto se spíše používá pro **sub** název procedura. Každá procedura je také ukončena klíčovým slovem **End sub**. [6]



Obr 4: Zobrazení kódu makra

3.1 Nahrání a uložení makra

Makra jak jsme se již dozvěděli, jsou vlastně procedury (podprocedury). Procedury se ukládají jako data. Aby bylo vše ohledně dat přehledné, byly vytvořeny tří úrovně ukládání – makra se zapisují do modulů, moduly se ukládají do knihoven a knihovny se ukládají do kontejnerů. Kontejner pro knihovny se vytváří automaticky pro každý nový dokument. Svůj kontejner pro knihovny má i samotná aplikace OpenOffice.org. Počty procedur v modulech, modulů

v knihovnách a knihoven v kontejneru není nijak omezen. Pro předejití chyb, doporučuji názvy maker v modulech, modulů v knihovnách a knihoven v kontejnerech pojmenovávat jednoznačně. [6]

3.2 Pravidla ukládání maker do kontejnerů

Pří každém založení nového dokumentu se automaticky vytvoří s dokumentem i kontejner stejného názvu. Knihovny, moduly a samozřejmě i makra, která jsou uložena v tomto kontejneru jsou dostupná a funkční pouze v tomto dokumentu a v jiných dokumentech či aplikacích OpenOffice.org nejsou přístupná. Tento kontejner obsahuje základní knihovnu s názvem Standard. Knihovnu Standard nedoporučuji používat. Pro použití si vytvoříme vlastní knihovnu, protože knihovna Standard je obsažena v každém kontejneru a použití stejně pojmenovaných knihoven by mohlo vést k chybám.

Svůj vlastní kontejner má i samotná aplikace OpenOffice.org. Kontejner se jmenuje Makra OpenOffice. Pokud makro uložíme do toho kontejneru, bude možné ho použít z jakéhokoli dokumentu dané aplikace. Tento kontejner by se dal nazvat jako aplikační kontejner.

Umět pracovat s kontejnery je velmi důležité. Pokud vytvoříme makro, které uložíme do aplikačního kontejneru, můžeme ho používat ze všech aplikací OpenOffice.org. Problém nastává, pokud soubor přeneseme ze svého počítače do počítače jiného, makra nebudou fungovat, protože nejsou uložena v dokumentu a jeho kontejneru. To může vést k chybám, protože makra nebudou v dokumentu k dispozici. Možnost tomu předejít je uložit makra do kontejneru příslušného dokumentu. Potom budou makra i po přenesení souboru přístupná, protože spolu s dokumentem se přenáší také jeho kontejner s makry. Nevýhodou tohoto postupu je však to, že makra nejsou přístupná pro jiné dokumenty. Proto je dobré promyslet, pro jaké účely je naše makro určeno. [4]

4 Programování ve StarOffice Basic

V předchozí části jsme viděli jak vytvořit makro pomocí záznamníku maker. Mimo tohoto jednoduchého způsobu zaznamenání makra existuje ještě způsob druhý, což je přímá tvorba procedury, která se tvoří v integrovaném vývojovém prostředí IDE. Vývojové prostředí má v sobě OpenOffice.org již integrované. V podstatě jsme se s ním již potkali v předchozí kapitole, když jsme prohlíželi námi zaznamenané makro.

Nyní si znovu vytvoříme nový dokument OpenOffice.org s názvem Bez názvu 1, tentokrát však ne jako textový dokument ale jako tabulkový dokument. V nabídce Start najdeme aplikaci s názvem OpenOffice Calc a spustíme jí. V panelu nabídek vybereme Nástroje – Makra – Správce maker – StarOffice Basic…, otevře se dialogové okno Makra v StarOffice Basic. Do této nabídky se lze dostat také pomocí klávesové zkratky ALT+F11. V levé části otevřeme soubor Bez názvu 1, ve kterém prozatím nic není a klepneme na tlačítko Nový…, kterým se nám otevře okno Nový modul. Zde vyplníme název modulu, například Makra a potvrdíme tlačítkem OK. Tím se nám zobrazí vývojové prostředí IDE s modulem Makra. Název modulu můžeme vidět ve spodní části okna vývojového prostředí. V tomto vývojovém prostředí můžeme tvořit, upravovat, ukládat a spouštět vytvořené procedury. K tomu abychom mohli začít tvořit makra, je potřeba znát něco o jazyce Basic, které toto IDE využívá. Základy jazyka Basic si ukážeme v dalších kapitolách. Než k nim přejdeme, zkusíme si napsat naše první makro ve vývojovém prostředí. V okně smažeme všechen text a začneme od začátku, samotný kód je zapsán na následujícím obrázku. Nyní je potřeba naše makro zkompilovat neboli přeložit pro počítač. K tomu slouží ikona listů, kde je odshora dolů šipka. Najdeme ho v horní části, hned vedle rozbalovací nabídky. Po stisknutí proběhne přeložení, nyní můžeme makro spustit, což učiníme tlačítkem, které je vedle a je vyobrazeno jako obdélník se zeleným trojúhelníkem uprostřed. Makro můžeme také spustit klávesovou zkratku F5. Spustíme makro. Objeví se vyskakovací okno, ve kterém vidíme text, který jsme zadali „Zadej své jméno“, dvě tlačítka OK a Zrušit a box pro zapsání odpovědi uživatele vyzve k zadání jeho jména. [4]

4.1 Základní pravidla pro zápis procedury

Jak už jsme si již řekli, procedura je část programu kterou je možné volat, a to i opakovaně a z různých míst programu. Procedura obsahuje řádky textu, ve kterém jsou napsány instrukce pro počítač. Po spuštění makra se jednotlivé příkazy plní až do úplného konce. [7]

Procedura začíná klíčovým slovem **Sub** a končí slovy **End sub.** Oblast, která je vymezena těmito slovy, je vyhrazena pro samotný kód programu. [8]

4.1.1 Řádkování a mezery

V naší proceduře (podprogramu) musí být každý příkaz na svém vlastním řádku. V případě složitých (dlouhých) příkazů lze řádek rozdělit, abychom udrželi přehlednost kódu. Rozdělení uděláme tak že v místě kde chceme, příkaz rozdělit dáme mezeru a podtržítko_ . Poté můžeme pokračovat na dalším řádku. Rozdělení není možné provést uprostřed slova, pouze v místě mezery.

Basic nerozlišuje mezi mezerou nebo tabulátorem, dále mu nevadí ani prázdné řádky. Tudíž je možné využívat odřádkování, mezery a tabulátory pro přehlednost kódu dle libosti. [6]

4.1.2 Popis makra - Komentáře

Komentář je text, který interpret (překladač) OpenOffice.org basic ignoruje. Slouží pro autora nebo jeho pokračovatele k vysvětlením určitých částí kódu. Komentář v OOo Basic lze zapsat dvěma způsoby, první možnost je zápis pomocí klíčového slova REM. To může být kdekoli na řádku a za komentář se považuje text za tím to slovem. Druhou možností je označit komentář pomocí znaku apostrof '. Tedy slovo REM a apostrof mají naprosto stejný význam. [6]

Příklad 1: Komentáře

```
Dim A ' To to je komentář pro proměnnou A
Rem to to je komentář tvořen klíčovým slovem Rem

Dim B ' Komentář pro proměnnou B, který je na více řádků,
      ' je tvořen na každém řádku klíčovým znakem apostrof.
```

4.1.3 Názvy proměnných a procedur

Názvy jednotlivých proměnných, procedur nebo funkcí v OO.org Basic si většinou určuje programátor sám. Pro takový zápis neexistují žádná psaná pravidla. Přesto se vyplatí pro přehlednost několik pravidel dodržovat. Ideální je veškeré názvy pojmenovávat logicky podle toho co popisují – např. Jmeno, celkovaVaha, vysledek. Můžeme si všimnout, že je jedno zda v názvech používáme velká nebo malá písmena, protože Basic je nerozlišuje a nechává to tak na samotném programátorovi jaké použije. Zažitým pravidlem u tvorby názvů je, že úvodní písmeno je velké, pokud je název složen ze dvou slov, používá se tzv. velbloudí zápis (celkováVáha). V žádném případě se nám nesmí stát, pokud je název složen ze dvou slov abychom ho rozdělili mezerou, basic to vyhodnotí jako dva názvy a vyhodí chybu. Dále nesmí být v názvu použita klíčová slova (např. Sub, End, Rem,…). Název může mít maximální délku 255 znaků, smí obsahovat pouze latinská písmena, číslice a podtržítko. Je vhodné nepoužívat písmena s diakritikou, mohli by způsobit vážné problémy při překladu programu. [6]

Příklad 2: Názvy proměnných

```
Dim vysledek          ' správný zápis
Dim vysledek1         ' správný zápis, číslice 1 není jako první znak
Dim krestni jmeno     ' špatný zápis, název obsahuje mezeru
Dim křestníJméno      ' špatný zápis, nedoporučuje se používat diakritiku
Dim 1vysledek         ' špatný zápis, číslo nesmí být jako první znak
Dim kresni,jmeno      ' špatný zápis, v názvu nesmí být čárka
```

4.1.4 Příkazy a funkce

Základními příkazy se kterými se setkáme je např. příkaz **End sub**, který ukončuje proceduru. Dále to může být příkaz **Print,** který uživateli zobrazí dialogové okno. Pokud chceme, aby se v dialogovém okně vypsal určitý text, musíme ho za příkaz Print dát do uvozovek. Dále můžeme pomocí příkazu Print vypsat proměnnou do které, může uživatel uložit vše co je potřeba. Obecně tedy můžeme říct že, příkaz se skládá z názvu příkazu, za kterým následují jeho parametry. Parametry mohou být povinné nebo nepovinné a oddělují se čárkou.

Funkce v zásadě vypadá stejně jako příkaz, jen s tím rozdílem, že parametry se u funkce zapisují do závorek. Jednou ze základních funkcí je např. **InputBox()**, tato funkce zobrazí dialogové okno s textovým polem po stisku tlačítka OK, načte hodnotu z textového pole. Tato funkce se používá pro přiřazení hodnoty od uživatele určité proměnné. [9][6]

4.1.5 Operátory

Operátory v programovacích jazycích jsou symboly používané ve výrazech, které umožnují provádět nějaké operace s hodnotami. StarOffice basic používá tři typy operátorů, matematické, logické a porovnávací. [10]

4.1.5.1 Matematické operátory

Matematické operátory se používají na veškeré operace s čísly, pouze operátor **+** může být použit i pro operace s řetězci, přesněji pro spojení řetězců. [11]

Znak	Popis
+	Sčítání čísel a spojování řetězců
-	Odčítání čísel
*	Násobení čísel
/	Dělení čísel
\	Celočíselné dělení (tzn. Zaokrouhlení výsledku na celé číslo)
^	Mocnina
MOD	Zbytek po dělení

Tabulka 1: matematické operátory

4.1.5.2 Logické operátory

Logické operátory umožnují propojit prvky v souladu s pravidly výrokové logiky. [11]

Znak	Popis
AND	Logický součin
OR	Logický součet
XOR	Exkluzivní součet
NOT	Negace
EQV	Ekvivalence
IMP	Implikace

Tabulka 2: logické operátory

4.1.5.3 Porovnávací operátory

Porovnávací operátory lze aplikovat na všechny typy základních proměnných (čísla, řetězce a boolovské proměnné). [11]

Znak	Popis
=	Rovnost čísel
<>	Nerovnost čísel nebo řetězců
>	Větší číslo než druhé
>=	Větší nebo roven než druhé číslo
<	Menší než druhé
<=	Menší nebo roven než druhé číslo

Tabulka 3: porovnávací operátor operátory

4.2 Základy jazyka StarOffice Basic

4.2.1 Proměnné

Proměnnou si můžeme představit jako box, do kterého se ukládají různé údaje. Tento box (proměnná) má své jméno a nese určitou informaci. V běžném životě se můžeme s proměnnými setkat všude kolem nás, například cenovka v obchodě. Na cenovce je uvedena cena, což je hodnota proměnné, která se s postupem času může měnit. Stejně to také funguje v programovacích jazycích. Je vytvořena proměnná s určitým názvem „cenovka“ do které je uložená určitá hodnota, např. „50“. [4]

4.2.1.1 Typy proměnných

U proměnných se kromě názvu a hodnoty udává ještě několik parametrů, prvním je místo kde je proměnná uložena a druhým je informace a typu proměnné. Typ proměnné je u příkladu výše číselný. Nic jiného do ni uložit nelze, bylo by to i dosti nelogické a matoucí. Představte si, že bychom do proměnné „cenovka“, uložili například název zboží ,v tu chvíli by se jistě nejednalo o cenu zboží. Typy proměnných si shrneme v následující tabulce a poté si je detailněji popíšeme. [6]

Typ	Klíčové slovo	Typový znak	Popis
Celé číslo	Integer	%	Celá čísla s rozsahem -32 768 až 32 768
Celá čísla s dlouhým rozsahem	Long		Celá čísla s rozsahem -2 147 483 648 až 2 147 483 647
Reálná čísla	Singl		Čísla s desetinou čárkou s rozsahem $3{,}402823*10^{38}$ až $1{,}401298*10^{-45}$
Reálná čísla s dlouhým rozsahem	Double		Čísla s desetinou čárkou s rozsahem $1{,}79769313486232*10^{308}$ až $4{,}94065645841247*10^{-324}$
Měna	Currency		
Text	String		Řetězec písmen zapsán v uvozovkách
Datum a čas	Date		Datum a čas
Logická hodnota	Boolean		Hodnota pravda (True) a nepravda (False)
Všechny typy	Variant		Zahrnuje všechny uvedené typy

Tabulka 4: datové typy

4.2.1.1.1 Celočíselné typy

Pro zápis celých čísel máme dva typy, je to typ **Integer** a **Long.** Od sebe liší pouze rozsahem hodnot. Jak je z názvu patrné můžeme do nich uložit pouze celá čísla. Hlavní rozdíl v použití je v rychlosti provedených výpočtů, **Integer** je rychlejší, protože má menší nároky na velikost paměti počítače. [11]

Příklad 3: Celočíselné datové typy

```
Rem celá čísla v rozsahu -32 768 až 32 768
        Dim promenna as INTEGER
        Dim promenna%       ' zkrácený zápis
Rem celá čísla v rozsahu -2 147 483 648 až 2 147 483 647
        Dim promenna as LONG
        Dim promenna&       ' zkrácený zápis
```

4.2.1.1.2 Typ pro čísla s desetinou čárkou

Reálná čísla mají ze stejného důvodu jako celá čísla, také dva typy **SINGLE** a **Double.** Tyto typy proměnných mohou nabývat hodnoty reálného čísla (desetinné číslo), např. 3,14 ; 7,456. [11]

Příklad 4: Typ pro čísla s desetinou čárkou

```
Rem reálná čísla v rozsahu 3,402823*10^38 až 1,401298*10^-45
        Dim promenna as SINGLE
        Dim promenna!       ' zkrácený zápis
   Rem celá reálná čísla v rozsahu 1,79769313486232*10^308 až
4,94065645841247*10^-324
        Dim promenna as DOUBLE
        Dim promenna#       ' zkrácený zápis
```

4.2.1.1.3 Měnový typ

Typ proměnné pro uložení čísla se čtyřmi místy za desetinou čárkou je typ **Currency**, který slouží pro uložení měnových údajů. Používá se při přesným finančních výpočtech. [11]

Příklad 5: Měnový datový typ

```
Rem zápis měnového typu
        Dim promenna as CURRENCY
        Dim promenna@       ' zkrácený zápis
```

4.2.1.1.4 Datumový typ

Typ **Date** slouží pro uložení a práci s datem a časem. Při ukládání hodnot datumu, StarOffice Basic používá interní formát, který umožnuje provádět s ním matematické operace. Tento typ nemá žádný zástupná symbol, proto pro deklaraci lze použít jen klíčové slovo DATE. [11]

Příklad 6: Datový typ pro datum

```
Rem zápis datumu a času
        Dim promenna as DATE
```

4.2.1.1.5 Textový typ

Typ **String** je typ proměnné pro uložení textu. Text se vždy ukládá mezi uvozovky a může obsahovat jakékoli znaky, čísla, písmena a samozřejmě je možné také použít text s diakritikou. [11]

Příklad 7: Datový typ String

```
Rem zápis textové řetězce
        Dim promenna as STRING
        Dim promenna$        ' zkrácený zápis
```

4.2.1.1.6 Logický typ

Logický typ může nabývat pouze dvou hodnot – logickou pravdu (True) nebo logickou nepravdu (False). V tomto datovém typu je možné hodnoty zapsat také pomocí čísla 1 který značí pravdu (True) a pomocí čísla 0, které značí nepravdu (False). [11]

Příklad 8: Logický datový datový typ

```
Rem zápis booloské proměnné
        Dim promenna as BOOLEAN
```

4.2.1.1.7 Typ Variant

Jazyk StarOffice Basic, má proti jiným jazykům obrovskou výhodu v typu **Variant.** Tento typ je naprosto univerzální, lze do něho uložit jakoukoli hodnotu od čísla až po logické hodnoty. Platí tedy, co do proměnné nahrajeme, to z ní také získáme. V některých případech to však může přinést problémy. [11]

4.2.1.2 Deklarace proměnných

V předchozí kapitole jsme si řekli něco o typech proměnných, které jsou v programu jasně definovány při prvním použití, tomu se říká deklarace proměnné. Jak jsme si mohli všimnout, klíčovým slovem deklarace je slovo **Dim.** Slovo deklarace znamená něco ve smyslu „zveřejněná definice“, významově odpovídá větě „Na vědomost se dává, že proměnná bude určitého typu“. Za slovem **Dim** se objevuje název proměnné, poté klíčové slovo **as** a poté klíčové slovo datového typu**,** všechny datové typy máme vypsané v Tabulce 3.1. Jak jsme si ukázali výše, je možnost proměnnou deklarovat dvěma způsoby, které jsou navzájem rovnocenné.

Příklad 9: Deklarace proměnných

```
Rem zápis textové řetězce
        Dim Jmeno as STRING
        Dim Jmeno$                  ' zkrácený zápis
```

V jazyce StarOffice Basic, je možné deklaraci proměnné úplně vynechat. V tu chvíli programovací jazyk použije datový typ **Variant.** Stane-li se, že u proměnné není datový typ vyplněn, Basic mu automaticky přiřadí univerzální datový typ Variant. Tato možnost velmi usnadňuje práci programátora a většina programátorů si s ní vystačí. Bohužel však může v určitých situacích nastat problém, který se pak velmi špatně hledá. Nyní si ukážeme deklaraci proměnné na jednoduchém příkladu výpočtu obvodu trojúhelníku.

Příklad 10: Deklarace proměnné s dlouhým zápisem datového typu

```
Sub ObvodTrojuhelniku
        Dim A, B, C, Obvod as Double
        A = 3.5;
        B = 4;
        C = 6;
        Obvod = A + B + C;
        Print Obvod
End sub
```

Popis: V příkladu vidíme, použitou deklaraci pomocí klíčového slova DIM s přesným určením datového typu. Jako datový typ jsme u výpočtu obvodu zvolili Double. Názvy proměnných jsme zvolili dle matematických zvyklostí A, B, C a Obvod. Můžeme si všimnout, že pokud deklarujeme proměnné jednoho typu, můžeme deklaraci provést na jednom řádku, pouze si musíme dát pozor, proměnné se od sebe oddělují čárkou.

Příklad 11: Deklarace proměnné pomocí zkráceného zápisu datového typu

```
Sub ObvodTrojuhelniku
        Dim A#= 3.5
          Dim B#= 4
          Dim C#= 6
          Dim Obvod#
        Obvod = A + B + C;
        Print Obvod
End sub
```

Popis: V tomto příkladu, vidíme stejný program jako u předchozího příkladu, pouze jsme použili zápis proměnných pomocí typového znaku u prvního výskytu proměnné v programu.

Příklad 12: Deklarace proměnné pomocí zkráceného zápisu datového typu

```
Sub ObvodTrojuhelniku
        A = 3.5
        B = 4
        C = 6
        Obvod = A + B + C;
        Print Obvod
End sub
```

Popis: V tomto příkladu jsme si ukázali, že u StarOffice Basic není potřeba deklarovat proměnnou ani její datový typ. Basic v tu chvíli proměnné přiřadí univerzální datový typ Variant. Výsledek bude stejný jako u předchozích příkladů. Přesto však doporučuji se tomuto způsobu vyhnout.

V předchozích příkladech jsme si ukázali práci s jedním typem proměnných, přesněji s datovým typem Double. V dalším příkladu si ukážeme práci s různými typy proměnných. Nastíníme si možný problém s ukládáním „širších" datových typů do „užších". Například můžeme říct, že datový typ Double, obsahuje všechna celá čísla typu Integer. Je však potřeba být opatrný a mít na paměti rozsah určitých datových typů.

Příklad 13: Práce s různými datovými typy

```
Sub ObvodTrojuhelniku
        Dim A, B, C as Double
        Dim Obvod as Integer
        A = 3.5
        B = 4
        C = 6
        Obvod = A + B + C;
        Print Obvod
End sub
```

Popis: V tomto příkladu vidíme, že jsme pro proměnné, do kterých ukládáme délky stran použili datový typ Double (reálné číslo). Pro Obvod, jsme však zvolili datový typ Integer (celé číslo). Vzorec pro výpočet obvodu trojúhelníku, by měl

vyjít 13,5 ale v našem případě bude výsledek 14, protože proměnná Obvod je Integer a ten si hodnotu 13,5 zaokrouhlí a výsledkem tak je hodnota 14. [6]

4.2.1.3 Globální a privátní proměnné

Proměnná v StarOffice Basic má omezenou životnost a omezený rozsah, ze kterého lze číst a využívat ho v dalších částech programu. To jak se proměnná chová a odkud a jak je přístupná závisí na jejím umístění a na jejím typu. Dělíme je na privátní a globální. V našich příkladech jsme se zatím setkali pouze s **privátními proměnnými**. V příkladu 13 máme definované proměnné A, B, C a Obvod. Ty to proměnné mají platnost pouze v proceduře ObvodTrojuhelniku, tedy mezi klíčovými slovy Sub a End sub. Jsou to tedy proměnné lokální, pokud je použijeme v jiné proceduře, proměnná se založí znovu a nebude mít s touto nic společného.

Druhou možností je definovat proměnnou jako **globální.** Globální proměnná je definována před procedurou pomocí klíčových slov **Private**, **Public** nebo **Global**. Pomocí klíčového slova **Private** vytvoříme proměnnou, která má platnost v daném modulu, můžeme s ní tedy pracovat ve všech procedurách daného modulu. Příkaz **Public** vytváří proměnnou, která má nejširší platnost a lze jí použít ve všech otevřených knihovnách a jejich modulech. Poslední možností pro definici proměnné je klíčové slovo **Global,** taková proměnná uchovává svou hodnotu i po skončení makra. Pokud bychom chtěli uchovat hodnotu lokální proměnné i po skončení makra, je potřeba použit klíčové slovo **Static.** [4]

```
Příklad 14: Práce s globálními proměnnými
    Global Mzda as Integer
    Global Mesic as String
Sub MzdaLeden
        Mesic = "Leden"
        Mzda = 26450
        Print "Výdělek za " + Mesic + " je " + Mzda + " Kč."
End sub

Sub MzdaUnor
    Print "Výplata za minulý měsíc" + Mesic + "byla" + Mzda + "Kč"
        Mesic = "Únor"
        Mzda = 16450
        Print "Výdělek za " + Mesic + " je " + Mzda + " Kč."
End sub
```

Popis: V příkladu vidíme definované dvě proměnné mimo procedury. První proměnná je typu Global s názvem Mzda a datovým typem Integer (celé číslo). Druhou proměnnou je Mesic s datovým typem String (textový řetězec) definována jako Privat. V první proceduře MzdaLeden je do proměnné Mesic nahrána hodnota Leden a do proměnné Mzdy hodnota 26450. Poté pomocí příkazu Print vypsána věta „Výdělek za měsíc Leden je 26450 Kč“. V druhé proceduře MzdaUnor hned na prvním řádku pomocí Print vypíšeme „Výplata za minulý měsíc Leden byla 26450 Kč“. U toho si můžeme všimnout, že ani proměnná Mesic ani Mzda v této proceduře nebyla inicializována a přesto vypíše hodnotu. Je to tím, že proměnné byly definovány jako globální. Nyní se v programu zapíše do proměnných Mzda a Mesic nové hodnoty a to Únor a 16450, čímž se přepíší hodnoty z původní procedury. Posledním co se v programu udělá je znovu výpis do vyskakovacího okna pomocí příkazu Print, kdy se vypíše „Výdělek za měsíc Únor je 16450 Kč“. Je důležité si všimnout, že pro spojování řetězců používáme znak **+**, který nám umožní připojit text a proměnnou pří výpisu.

V programu se můžeme setkat se situací, kdy se budou globální a lokální proměnné překrývat. Je-li deklarována globální proměnná například Mzda a poté stejná proměnná definována jako lokální v určité proceduře se stejným názvem tak lokální proměnná bude v této proceduře tu globální překrývat viz. příklad 15.

Příklad 15: Práce s globálními a lokálními proměnnými

```
    Global Mzda as Integer
    Global Mesic as String
Sub MzdaLeden
        Mesic = "Leden"
        Mzda = 26450
        Print "Výdělek za " + Mesic + " je " + Mzda + " Kč."
End sub

Sub MzdaUnor
        Dim Mzda as Integer
    Print "Výplata za minulý měsíc" + Mesic + "byla" + Mzda + "Kč"
        Mesic = "Únor"
        Mzda = 16450
        Print "Výdělek za " + Mesic + " je " + Mzda + " Kč."
End sub
```

Popis: Tento program dělá takřka to stejné jako předchozí program 14. Jsou definovány globální proměnné. V první proceduře jsou do nich uloženy určitá data a poté vypsány. V druhé proceduře je změna hned na začátku. Je založena lokální proměnná se stejným názvem jako globální proměnná – Mzda. Poté pomocí Print vypisujeme větu, která se vypíše ve formátu „Výplata za minulý měsíc Leden byla 0 Kč“. A to proto, že globální proměnná Mzda byla v té to proceduře nahrazena lokální proměnnou Mzda, které nemá deklarovanou hodnotu, tudíž je hodnota automaticky nastavena na hodnotu 0. Zbytek je zase stejný jako v příkladu 5.

4.2.1.4 Pole (array)

V předchozí kapitole jsme se bavili o proměnných. Proměnná obsahuje pouze jednu hodnotu a ta se může měnit v průběhu programu, v určitou chvíli však může obsahovat pouze jednu hodnotu. Někdy můžeme pracovat s velkým množstvím hodnot stejného typu, abychom nemuseli deklarovat velké množství proměnných je možno v jazyce StarOffice Basic použít datové pole (pole proměnných). Pole je datová struktura, která sdružuje vždy konečný počet prvků (čísel, textovým řetězců, …) stejného datového typu. K jednotlivým prvkům pole se přistupuje pomocí jejich indexu. Na rozdíl od proměnné musí být pole vždy před použitím deklarováno. Index prvku pole u StarOffice Basic se uvádí do kulatých závorek. Více si to vysvětlíme v příkladu 16. [8]

Příklad 16: Pole - array

```
Sub PoleDny
        Dim Dny(7) as String
        Den(0) = “Ponděli“
        Den(1) = “Úterý“
        Den(2) = “Středa“
        Den(3) = “Čtvrtek“
        Den(4) = “Pátek“
        Den(5) = “Sobota“
        Den(6) = “Neděle“
        Print Den(5)
End sub
```

Popis: Jak jsme si říkali, každé pole musí být definováno, k jeho definici dochází pomocí klíčového slova Dim. Za slovem Dim se nachází jméno pole a poté v kulatých závorkách počet prvků pole (v našem případě je to 7). Přiřazení jednotlivých prvků do pole, probíhá stejně jako u jednoduchých proměnných pouze s tím rozdílem, že za názvem pole je v závorkách uveden index prvku. Je důležité si dát pozor na to, že indexace pole začíná od čísla 0, takže indexy naších dní v týdnu jsou od 0 – 6. Příkaz Print Den(5) vypíše 6 den v týdnu Sobotu. Pokud by jsme chtěli, aby pole bylo indexované od 1. Bylo by potřeba upravit zápis deklarace pole na **Dim Dny(1 to 7) as String**.

V příkladu 16 jsme si ukázali deklaraci základního pole. Pole má ale několik možností zapsání. Od jednorozměrného po vícerozměrné.

Příklad 17: Pole - array

```
Sub Pole
Dim a(5) as Integer                ' pole se 6 prvky (0 až 5)
Dim b (-3 to 3) as Integer         ' pole se 7 prvky (-3 až 3)
Dim c(1 to 6, 1 to 6) as Long      ' více rozměrné pole 6*6
Dim d ()                           ' pole bez specifikace
d = array (5,6,7,8,9)              ' přiřadil do pole d 5 prvků
End sub
```

Pro práci s poli je několik základních funkci. První jsme si již zmínili, je to funkce array(). Funkce array() slouží pro naplnění pole. Další funkcí je LBOUND() A UBOUND(), ty to funkce vrací první respektive poslední index pole. [8]

Příklad 18: Funkce pro práci s polem

```
Sub Auta
Dim znackyAut() as String
znackyAut() = array("škoda", "audi", "mercedes", "Volvo")
Dim Prvni_index = LBOUND(znackyAut())
Dim Posledi_index = UBOUND(znackyAut())
Dim Pocet_znacek = Posledni_index + 1
Print "Počet značek aut v seznamu je " + Pocet_znacek
End sub
```

Popis: Prázdné pole znackyAut, je pomocí funkce array() naplněno. Do proměnné Prvni_index je uložena hodnota, kterou vrací funkce LBOUND(), která vrací první hodnotu indexu v poli. Do proměnné Posledni_index je uložena hodnota, kterou vrací funkce UBOUND(), která vrací poslední hodnotu indexu v poli. Do proměnné Pocet_znacek nahráváme hodnotu posledního indexu a zvětšujeme ji o 1, abychom dosáhli celkového počtu značek v poli. Je to tím, že pole je indexováno od 0, proto je potřeba poslední index o 1 povýšit. Výsledek se pomocí Print vypíše ve vyskakovací okně.

4.2.1.5 Konstanty

Konstanta je v programovacích jazycích označení pro identifikátor, který je spojený s určitou hodnotou, kterou není možné během provádění programu měnit. Většina programovacích jazyků umožňuje syntakticky rozlišit proměnné a konstanty. Konstanty v StarOffice Basic se stejně jako pole musí deklarovat. K deklaraci se používá klíčové slovo **Const** u deklarace je potřeba uvést zároveň i hodnotu dané konstanty. Nepsané pravidlo pro definování konstant v programu je, že jejich název je psán velkými písmeny. Přestože je konstanta definována pouze jednou, může být v programu použita mnohokrát. Použití konstanty místo přímého použití její hodnoty nejenom usnadňuje údržbu programu, ale také umožňuje předejití chybám, které by způsobila nechtěná změna této hodnoty. [12]

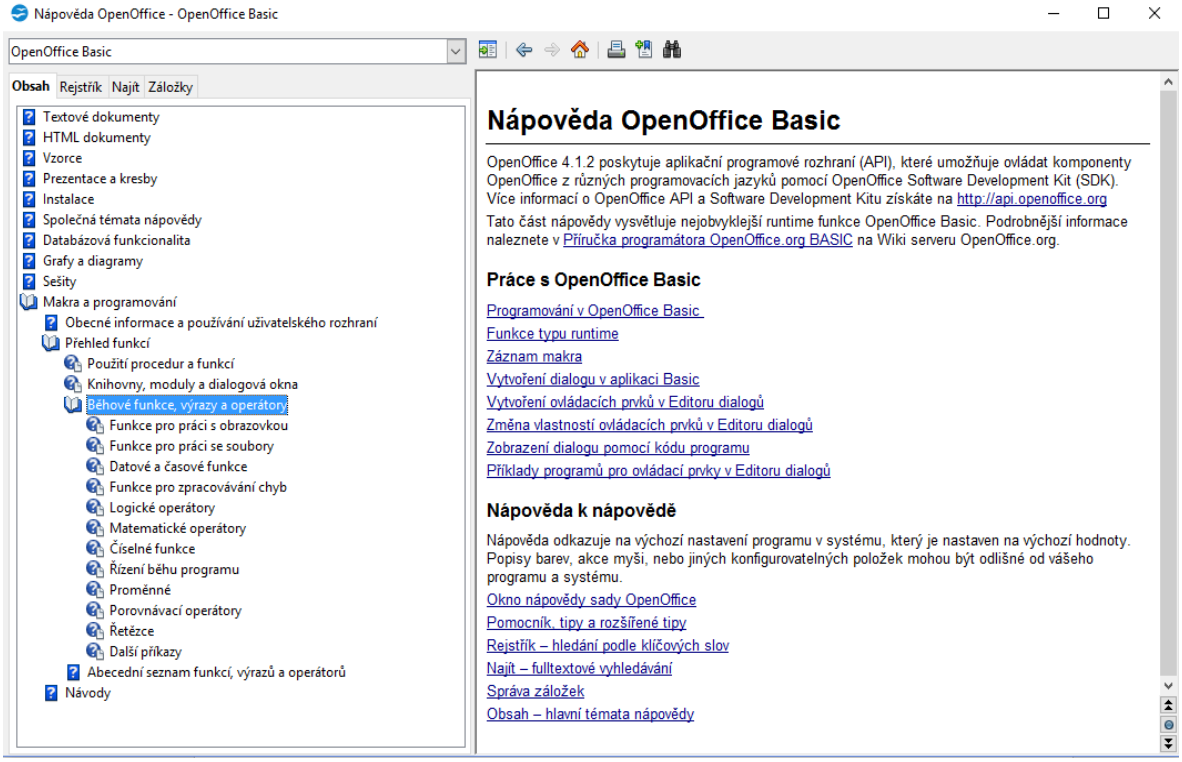
Příklad 19: Definice konstanty

```
Rem zápis konstanty PI
        Const PI = 3,1415926535
```

4.2.2 Funkce a příkazy

O tom co jsou a jak fungují funkce a příkazy, jsme si říkali v dřívější kapitole číslo 4.1.4. V té to kapitole se podíváme už na určité příkazy a funkce, které pracují s obrazovkou, se soubory atd.

Veškeré informace o funkcích a příkazech, jsou velmi přehledně sepsány v nápovědě OpenOffice.org. Do nápovědy se dostaneme klávesou **F1**, při otevřeném okně OpenOffice.org. Můžeme si tam všimnout nápovědy, pro všechny části programu, my se však nyní budeme zabývat nápovědou pro funkce a příkazy. Tu to část najdeme v záložce **Obsah**, poté odrážka Makra a programování – Přehled funkcí – Běhové funkce, výrazy a operátory. [6]



Obr 5: Nápověda OpenOffice.org

4.2.2.1 Funkce pro práci s proměnnými

Funkce a příkazy, které pracují s proměnnými je v StarOffice Basic celá řada. Některé základní příkazy jsme si již uvedli v kapitole o proměnných. Byli to například Dim, Private, Const a další.

Mezi základní funkce pro práci s proměnnými je funkce pro zjištění datového typu proměnné. Tato funkce má zápis TypeName() a VarType().

Příklad 20: Funkce pro určení datového typu

```
Sub DatovyTyp
        Dim A
        Dim B
        A = 3.14
        B = VarType(A)
        C = TypeName(B)
        print "Datový typ proměnné B je "+B
        print "Datový typ proměnné C je "+C
End sub
```

Popis: Na příkladu vidíme, deklaraci proměnných, bez určení datového typu. Do proměnné A je poté nahrána hodnota 3.14. Pomocí funkce VarType() zjišťujeme datový typ proměnné a vrací nám číselnou hodnotu, v našem příkladu je to hodnota 5, což nám bez znalosti tabulky prozatím moc neříká. Funkce TypeName() nám vrací textový řetězec s názvem datového typu. V našem příkladu je to Integer.

TypeName	VarType	Typ proměnné
Boolean	11	Logická proměnná
Date	7	Proměnná data
Double	5	Proměnná s dvojitou přesností a plovoucí desetinnou čárkou
Integer	2	Celočíselná proměnná
Long	3	Dlouhá celočíselná proměnná
Objekt	9	Objektová proměnná
Single	4	Proměnná s jednoduchou přesností a plovoucí desetinnou čárkou
String	8	Řetězec
Variant	12	Proměnná typu Variant (může obsahovat všechny typy a zadává se definicí)
Empty	0	Proměnná není inicializována
Null	1	Žádná platná data

Tabulka 5: Typy proměnných

Další funkce, jsou spjaty s datovým typem **Variant.** Velmi často je potřeba zjistit jaký typ hodnot je v proměnné uložen. K tomu slouží funkce IsNumeric, IsDate, IsEmpty, IsNull, IsArray. Funkce nám zjišťují, zda proměnná obsahují hodnoty číselné, datové a další. Všechny zmíněné funkce vrací jako hodnotu buď True (pravda) nebo False (nepravda).

Další sérii funkcí jsou funkce pro přetypování neboli změnu datového typu proměnné. Jazyk StarOffice Basic to umí dělat automaticky, pokud bychom to ale chtěli udělat přesně tak jak my chceme je možnost použít funkce **CCur, CBool,**

CDate, CDec, CDbl, CInt, CLng, CSng, Cstr, Cvar. Tyto funkce převádí proměnnou na určitý datový typ. Už z názvu funkce, můžeme vyčíst na co bude proměnná přetypována. První písmeno **C** nám říká Change (změna) a poté následují 3-4 písmena, které značí zkratku datového typu **Cur**(Curren), **Bool**(Boolien), atd. V dalším příkladu si ukážeme, jak se s těmito funkcemi pracuje.

Příklad 21: Funkce pro přetypování proměnných

```
Sub Pretypovani
        Cislo = 7*3.14
        Print "Proměnná Cislo = " + Cislo
        Print "Proměnná Cislo je typu " + TypeName(Cislo)
        Print "Funkce IsNumeric zjistí, zda se jedná o číslo" +
IsNumeric(Cislo)
        TypInt = CInt(Cislo)
        Print "Proměnná Cislo po přetypování je " + TypInt
        Print "Proměnná TypDouble je typu " + TypeName(TypInt)
```

Popis: V tomto příkladu, jsme použili takřka všechny funkce, které jsme zmínili. Definovali jsme si 2 proměnné Cislo a TypInt. Poté pomocí příkazu Print vypíšeme hodnotu proměnné Cislo, což je v našem případě hodnota 21,98. Pomocí funkce TypeName(), zjistíme datový typ proměnné Cislo, kterým je Double. Díky funkci IsNumeric() víme, že v proměnné je uložené číslo. Poté jsme vytvořili proměnnou TypInt a pomocí funkce CInt jsme do ní přetypovali proměnnou Cislo na datový typ Integer. Poté jsme vypsali pomocí Print její hodnotu, která se z hodnoty 21,98 díky přetypování změnila na hodnotu 22 a datový typ, kterým je Integer. [14]

4.2.2.2 Funkce pro práci s obrazovkou

Pro komunikaci s uživatelem StarOffice Basic, používá základní příkaz, který jsme již několikrát v naších příkladech použili **Print** a dvě základní funkce. Jedná se o **MsgBox** a **InputBox**.

4.2.2.2.1 Příkaz Print

Prakticky nejpoužívanějším příkazem je příkaz Print. Print je jazykový příkaz – nikoli funkce, nemusíme tedy u něj používat závorky. Tento příkaz má jeden úkol a to vypsat řetězec nebo číselný výraz v dialogovém okně případně v uložit ho do souboru. [4]

4.2.2.2.2 Funkce a příkaz MsgBox

MsgBox se v StarOffice Basic objevuje ve formě příkazu a ve formě funkce. Mezi nimi je pouze jediný rozdíl a to v tom, že funkce v sobě obsahuje ještě návratovou hodnotu. I tento příkaz i funkce mají svůj význam a své přesné použití.

Příkaz MsgBox se používá pro zobrazení informace v dialogovém okně. Má tří parametry a zadává se ve forma viz. příklad.

Příklad 22: Funkce MsgBox

```
MsgBox Zprava, [Vzhled], [Nadpis]
```

Zpráva je datového typu String, Vzhled je číslo ve formě Integer a Nadpis je opět String. Poslední dva parametry nejsou povinné. Nejdůležitější částí je parametr Zprava, které obsahuje informace, které chceme zobrazit v dialogovém okně. Vzhled dialogového okna se nastavuje pomocí čísel viz. Tabulka 6.

0	Zobrazí se pouze tlačítko OK.
1	Zobrazí se tlačítka OK a Zrušit.
2	Zobrazí se tlačítka Zrušit, Opakovat a Ignorovat.
3	Zobrazí se tlačítka Ano, Ne a Zrušit.
4	Zobrazí se tlačítka Ano a Ne.
5	Zobrazí se tlačítka Opakovat a Zrušit.
16	Do dialogového okna je přidána ikona Zastavit.
32	Do dialogového okna je přidána ikona Otazník.
48	Do dialogového okna je přidána ikona Vykřičník.
64	Do dialogového okna je přidána ikona Informace.
128	První tlačítko v dialogu je použito jako výchozí tlačítko.

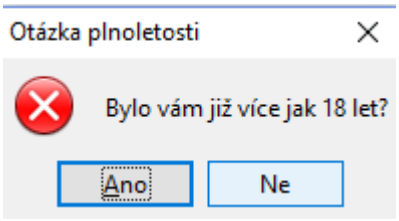
256	Druhé tlačítko v dialogovém okně je použito jako výchozí tlačítko.
512	Třetí tlačítko v dialogovém okně je použito jako výchozí tlačítko.

Tabulka 6: Styl dialogového okna MsgBox

Příklad 23: Příkaz MsgBox

```
Sub Main
    MsgBox "Bylo vám již více jak 18 let?", 20, "Otázka plnoletosti"
End Sub
```

Popis: Tento příklad zobrazí dialogové okno s tlačítky Ano a Ne s textem „Bylo vám již více jak 18 let?“ a nadpisem „Otázka plnoletosti“. U dialogového okna jsme zvolili vzhled, který odpovídá číslu 20. Pokud se podíváme do tabulky 6, zjistíme, že číslo 20 tam není. Je to tím, že jednotlivé vzhledy lze sčítat. U nás tedy vzhled 20 znamená, vzhled 4 + vzhled 16.



Obr 6: Dialogové okno MsgBox se vzhledem 20

Po stisku jakéhokoliv tlačítka v tomto dialogovém okně se okno zavře bez vrácení výsledku. Právě proto existuje funkce MsgBox, která nám dokáže vrátit určitý číselný výsledek. Výsledkem jsou tedy číselné hodnoty viz. Tabulka .

Návratová hodnota	Tlačítko
1	OK
2	Zrušit
3	Zrušit
4	Opakovat
5	Ignorovat
6	Ano
7	Ne

Tabulka 7: Návratové hodnoty fce MsgBox

Zápis funkce MsgBox je stejná jako u příkazu MsgBox jen s tím rozdílem, že funkce je zapsána do kulatých závorek. Použití příkazu a funkce MsgBox si ukážeme v dalším příkladu. [6]

Příklad 24: Použití příkazu a funkce MsgBox

```
Sub zpravaMsgBox
    Dim Zprava1, Nadpis, Zprava2 as String
    Zprava1 = "Bylo vám již 18 let?"
    Zprava2 = "Vybrali jste tlačítko: "
    Nadpis = "MsgBox"
    Tlacitko = MsgBox (Zprava1, 20, Nadpis)
    MsgBox Zprava2 + Tlacitko, 64, Nadpis
End Sub
```

Popis: V tomto příkladu, vidíme použitý příkaz a funkci MsgBox. Definovali jsme si proměnné, do kterých jsme vložili text. Poté proměnnou Tlacitko do které se uloží výstupní hodnotu funkce MsgBox, v našem příkladu se do proměnné Tlacitko může dostat hodnota 6 nebo 7 čili Ano nebo Ne. Poté pomocí příkazu MsgBox je uživateli předána samotná hodnota.

4.2.2.2.3 Funkce InputBox

Pří programování ve StarOffice Basic máme k dispozici jen jednu vstupní funkci, kterou je „InputBox“. Funkce slouží pro zachycení textu, zadaného uživatelem. Inputbox zobrazuje dialogové okno s polem pro text a tlačítko OK a Zrušit, po stisku tlačítka OK je obsah textové pole uložen do proměnné, po stisku tlačítka Zrušit, je do proměnné uložen prázdný řetězec ("") Tato funkce má několik základních parametrů, které si můžete nastudovat v Nápovědě OpenOffice.org, kde najdeme i jednoduchý příklad. [4]

Příklad 25: Použití InputBox

```
Sub Inputbox
    Jmeno = InputBox("Zadej své jméno","Formulář pro vložení jména")
    MsgBox "Vaše jméno je " + Jmeno
End sub
```

Popis: V příkladu, vidíme funkci inputbox, která má jako nadpis větu „Formulář pro vložení jména“ a větu „Zadej své jméno“, pod kterou je textové pole pro zapsání jména. Obsah funkce se zapíše do proměnné, která je následně vypsána pomocí funkce msgbox.

4.2.2.2.4 Příkazy pro řízení běhu programu

Prozatím jsme se setkali s částmi kódu, které se prováděli jedna po druhé. Velmi často ale potřebujeme, aby se část programu provedla několikrát nebo aby se část programu úplně přeskočila nebo aby se program větvil dle vstupních údajů. K tomu se používají příkazy pro řízení běhu programu. Nápovědu k této oblasti můžeme najít v nápovědě OpenOffice.org v části Makra a programování – Přehled funkcí – Běhové funkce, výrazy a operátory – Řízení běhu programu. [4]

Program StarOffice Basic má tři typy řídících struktur.

- Větvení, příkazy které ke svému chodu využívají podmínky.
- Cykly, vytváří smyčky v programu.
- Příkazy skoku, které provádějí skoky v programu.

4.2.2.2.4.1 Příkazy pro větvení programu

Jedná se o příkazy, který se provedou pouze tehdy, pokud je splněna podmínka.

Větvení má v jazyce StarOffice Basic následující formát, jedná se klíčová slova IF – Then - Else.

Příklad 26: Větvení programu

```
Sub Vetveni
    Dim A
    A = 10
    IF A > 0 THEN
    Prikaz_1 REM Příkazy, které se provedou pokud je podmínka splněna
    Else
    Prikaz_2REM Příkazy, které se provedou pokud podmínka není splněna
    End IF
```

Popis: Pokud je A větší než nula, proveď Prikaz_1. Pokud A je menší než 0 provede se Prikaz_2. První je klíčové slovo IF, potom podmínka. Na základě této podmínky se rozhodne, zda se následující příkazy provedou či nikoliv.

Příkazů, které se mají provést po vyhodnocení podmínky, může být několik. Důležité je také, jak sestavíme podmínku, podmínka musí vždy vracet buď hodnotu True nebo False. Pokud je podmínka True provede se příkaz nebo série příkazů za klíčovým slovem THEN. Pokud je podmínka False provede se část kódu za klíčovým slovem Else.

Příklad 27: Použití větvení programu

```
Sub Vetveni
        Dim Vek
        Vek = InputBox("Napis svůj věk","Kontrola věku")
        Vek = CInt(Vek)
        IF Vek >= 18 THEN
        Print "Vstup povolen"
        Else
        Print "Vstup zakázan"
    End IF
    End sub
```

Popis: Do proměnné Vek, jsme přiradili získanou hodnotu z funkce InputBox(vstup uživatele). Chceme získat věk uživatele, předpokládáme, že vstupem bude číslo, ale InputBox nám vrací hodnotu jako datový ty string. Proto proměnnou Vek přetypujeme na Integer. Abychom mohli použít do podmínky matematické operátory. Podmínka je postavena následovně, pokud je obsah proměnné menší než 18, vypíše se větev s příkazem „Vstup zakázán“, pokud je věk větší nebo roven 18 vypíše se „Vstup povolen“.

Často se nám stává, že jeden výraz v podmínce může nabývat mnoha hodnot. Pokud bychom chtěli všechny hodnoty ošetřit, museli bychom použít velmi dlouhou a složitou strukturu příkazů if a elseif. Existuje však i jiné řešení. Příkaz **Select,** ten funguje poněkud odlišným způsobem než podmínky. Skládá se ze selektoru a jednotlivých větví označených hodnotou. Příkaz se pak snaží najít větev označenou stejnou hodnotou jakou má selektor. Ukážeme si jednoduchý příklad:

Příklad 28: Select - case

```
Sub selectJmena
Dim Uzivatel
Uzivatel = InputBox("Napiš své jmeno","Jméno")
Select Case Uzivatel
    Case "Michal"
    Zprava = "Vítel Michale"
    Case "Lukáš"
    Zprava = "Vitej Lukáši"
    Case "Tomáš"
    Zprava = "Vítěj Tomáši"
    Case Else
    Zprava = "Vítej neznámí uživateli"
    End Select
    Print Zprava
End sub
```

Popis: V tomto příkladu, vidíme už známé konstrukce. Do proměnné Uzivatel je uložen výstup z funkce InputBox. Poté následuje klíčové slovo Select, Case a poté proměnná Uzivatel. Poté následuje vždy klíčové slovo

Case a za ním možná hodnota a příkazy, které se mají případně provést, v našem případě to znamená uložit do proměnné Zprava větu „Vítej a jméno uživatele“. Příkaz Select má také prázdnou větev, podobně jako příkaz if má větev else. Ta se použije, pokud žádný z předchozích Case neodpovídá, použije se větev Case Else. [11]

4.2.2.2.4.2 Cykly

Cyklus je řídící struktura, která umožnuje opakovaně provádět posloupnost příkazů. Opakování a ukončování cyklu je řízení nějakou podmínkou. V programovacím jazyku StarOffice Basic máme několik druhů cyklů. Prvním typem je cyklus Do Loop, tento cyklus má tří různé modifikace. Všechny tři modifikace, patří do kategorie cyklu s neznámým počtem opakování.

Do While ... Loop

Blok příkazů se provádí tak dlouho, dokud platí podmínka. Pokud podmínka na začátku není splněna, cyklus se neprovede ani jednou.

Příklad 29: Cyklus – Do While

```
Sub cyklus
    Do While podmika
    Prikaz_1
    Prikaz_2
    Loop
End sub
```

Do Until ... Loop

Blok příkazů se provádí tak dlouho, dokud podmínka neplatí. Pokud podmínka na začátku je splněna, cyklus se neprovede ani jednou

Příklad 30: Cyklus – Do Untile

```
Sub cyklus
    Do Until podmika
    Prikaz_1
    Prikaz_2
    Loop
End sub
```

Do ... Loop While

Blok příkazů se provádí tak dlouho, dokud je podmínka platná. Blok příkazů se provede, alespoň jednou. (Podmínka se vyhodnocuje po prvním průchodu cyklem.)

Příklad 31: Cyklus – Do Loop While

```
Sub cyklus
    Do
    Prikaz_1
    Prikaz_2
    Loop While podminka
End sub
```

Do kategorie cyklů s přeným počtem opakování spadá cyklus For ... Next. Nejlépe si jeho fungování vysvětlíme na příkladu.

Příklad 32: Cyklus – For

```
Sub cyklusFor
    Soucet = 0
    For i=3 to 11 step 2
        Soucet = Soucet + i
    Next i
Print Soucet
End sub
```

Popis: Tento program sečte lichá čísla od 3 do 21. Pokud není uvedeno slovo Step, přičítá se v každém kroku 1. Po posledním průchodu cyklem, vypíše součet čísel.

Na příkladu 32 si vysvětlíme co cyklus For-Next, krok po kroku dělá. Za klíčovým slovem For máme do proměnné i uloženo číslo 3. Od tohoto čísla začíná cyklus počítat. Za slovem **To** následuje další číslo, které udává konečnou hodnotu

počítadla a délku kroku udává číslo za slovem **Step**. Pří spuštění makra, Soucet je 0, i = 3. Při první průchodu cyklem, se do proměnné Soucet dostane 0 + i což v prvním průchodu je 3. Next i přičte k i hodnotu step v našem případě 2. Takže Soucet = 3, i = 5. V druhém průchodu se stane prakticky to stejné. i = 5, Soucet = 3 +5 = 8. Ve třetím průchodu je i = 7, Soucet = 8 +7 = 15. Ve čtvrtém průchodu i = 9, Soucet = 15 + 9 = 24. V pátém průchodu i = 11, Soucet = 24+11 = 35. V šestém průchodu je i = 13, což nesplňuje podmínku i = 3 to 11. Cyklus se tedy ukončí a makro pokračuje dál. Kde pomocí Print vytiskne proměnnou součet s výsledkem 35. [8]

4.2.2.2.4.3 Příkaz GoTo

Příkaz GoTo slouží ke skoku z daného místa (místa, kde tento příkaz použijeme) programu do jiného (označeného tzv. návěštím) od kterého je program dále zpracováván. Příkaz GoTo se používá pouze výjimečně, jelikož se bez něj vždy lze obejít a navíc použití tohoto příkazu většinou činí program poněkud nepřehledný a tudíž obtížněji odladitelný. Používání tohoto příkazu je ve sporu s myšlenkou strukturovaného programování. Funci tohoto příkazu si ukážeme na příkladu. [6]

Příklad 33: Příkaz GoTo

```
Sub Skoky
    Zacatek:
    Jmeno = InputBox("Zadejte své jméno")
    DatTyp= IsNumeric(Jmeno)
    If DatTyp = True Then GoTo Chyba
    GoTo Konec
Chyba:
Print "Chyba, zadejte dvé jméno!"
GoTo Zacatek
Konec:
Print "Vaše jméno je " + Jmeno
End sub
```

Popis: V tomto programu by nám měli do oka padnout tří místa, Zacatek, Chyba a Konec. Jedná se o tzv. návěstí, která sama o sobě do programu nezasahují. Dále je potřeba si všimnou klíčového slova GoTo. První místo

kde se GoTo vyskytuje, je za podmínkou IF. Pokud je podmínka IF splněna (DatTyp je číslo) pomocí GoTo Chyba skočí na navěstí Chyba odkud program pokračuje dále. Pomocí Print se vypíše věta a dále se pokračuje na příkaz GoTo Zacatek. Tím program skočí na úplná začátek programu. Pokud je podmínka nesplněna, pomocí GoTo Konec program skočí na navěstí Konec, kde už se jen vypíše pomocí Print věta „Vaše jméno je ….

4.2.2.3 Matematické funkce

Pro práci s čísly jsou v programovacím jazyce StarOffice Basic matematické funkce. Jak jsme již zmínily dříve, velmi často se u matematických funkcí používají i matematické a logické operátory viz. Tabulka 1 a 2. Matematické funkce mají stejný zápis jako jakékoli jiné funkce, čili jméno funkce a v kulatých závorkách parametr a celá funkce vrátí výsledek.

4.2.2.3.1 Trigonometrické funkce

Můžeme je znát také pod názvem goniometrické funkce. Slovo goniometrie z řečtiny znamená měření uhlů, trigon se pak překládá jako trojúhelník. Tedy goniometrické funkce pracují s úhly v trojúhelníku. Rozlišujeme čtyři základní goniometrické funkce sinus, cosinus, tangens, kotangens. Všechny tyto čtyři funkce nám umožnuje použít i StarOffice Basic. Jejich syntaxe je takřka stejná viz. Příklad 34.[6]

Příklad 34: Trigonometrické funkce

```
Sub goniometrie
cislo = InputBox("zadej číslo")
sinus= sin(cislo)
cosinus = cos (cislo)
tangens = cos (cislo)
atangens = atn (cislo)

Print "Hodnota Sinu " +sinus+ ", Hodnota Cosinu " + cosinus +", Hodnota tangens " + tangens +", Hodnota arcostangens " + atangens
End sub
```

Popis: Vidíme zde trigonometrické funkce kde jejich zpětnou hodnotu, ukládáme do proměnné, funkce se skládá z klíčového slova a parametru, kterým je číslo, ze kterého má funkce získat výslednou hodnotu. Funkce sinus, cosinu a tangens jsou funkce, které vrací hodnotu v rozsahu -1 až 1. Jejich parametr je zadáván v radiánech. Funkce Ant vrací arkustanges svého parametru.

4.2.2.3.2 Exponenciální a logaritmické funkce

Do této skupiny patří dvě funkce.

Exp(hodnta) - Vrátí základ přirozeného logaritmu (e = 2,718282) umocněný na zadané číslo.

Log(hodnota) - Vrátí přirozený logaritmus čísla.

4.2.2.3.2.1 Generování náhodných čísel

Randomize [Číslo] - Inicializuje generátor náhodných čísel.

Rnd [(Výraz)] - Vrátí náhodné číslo mezi 0 a 1.

4.2.2.3.2.2 Ostatní číselné funkce

Fix(Výraz) - Vrátí celočíselnou hodnotu číselného výrazu, z něhož odstraní zlomkovou část.

Int(Číslo) - Vrátí celočíselnou část čísla.

Hex(číslo) – Vrátí řetězec, který představuje šestnáctkovou hodnotu čísla.

Oct(číslo) - Vrátí osmičkovou hodnotu čísla. [15]

4.2.2.4 Funkce pro práci s řetězci

Pro práci s řetězci existuje velká spousta funkcí, ty základní si v této kapitole popíšeme. První skupinou jsou funkce, které slouží pro převod řetězců mezi ASCII a ANSI.

První funkcí je Asc(), tato funkce vrací hodnotu ASCII prvního znaku řetězce. Další funkcí je Chr(), ta vrací znak odpovídající zadanému kódu. Funkcí pro práci je ještě několik, ale už nejsou tak významné, je možné je nastudovat s návodu OpenOffice.Org.

Další skupinou jsou funkce pro úpravu řetězce, zde si řekneme něco o těch nejpoužívanějších. Zřejmě nejpoužívanější je funkce LCase(text), která nám převede všechna velká písmena na malá, opačně nám to umožnuje funkce UCase(text). Funkce Trim(text) nám smaže mezery na začátku a konci řetězce, tato funkce má i své obměny jedná se o LTrim (text) a RTrim(text), které mažou mezery buď na začátku, nebo na konci řetězce.

Mezi funkce pro porovnání řetězců patří funkce Len(text) která vrací délku řetězce nebo funkce StrComp(text1,text2), která nám porovnává dva řetězce.

Užití některých funkcí pro práci s řetězcem si ukážeme na příkladu 35.[4]

Příklad 35: Funkce pro práci s řetězci

```
Sub Retezce
Vstup = InputBox("Zadejte jakýkoliv text")
Uprava_vstup = Trim(Vstup)
DelkaTextu = Len(Uprava_vstup)
PrvniZnak = Left(Uprava_vstup,1)
PosledniZnak = Right(Uprava_vstup,1)
KodPrvni = Asc(PrvniZnak)
KodPosledni = Asc(PosledniZnak)
MsgBox("Zadali jste text: " + Uprava_vstup + " o délce " + DelkaTextu +
_
Chr(13) + "ASCII kódy prvního a posledního znaku jsou " + _
KodPrvni + " a " + KodPosledni)
End Sub
```

Popis: Pomocí funkce InputBox jsme do proměnné Vstup vložili libovolný text od uživatele. Funkce InputBox nám vrací vždy String, Proto můžeme používat jednotlivé funkce pro práci s řetězci. První věc co uděláme je, že odstraníme případné mezery před a za textem. Poté pomocí funkce Len() určíme délku řetězce. Poté do proměnných přiřadíme první a poslední znak řetězce. Pomocí fukce Asc() určíme číselnou hodnotu daného znaku. A poté to pomocí MsgBox vypíšeme uživateli.

5 Porovnání možností programování v MS Excel (VBA) a OpenOffice.org (StarOffice Basic)

Programovací jazyky VBA a StarOffice Basic si jsou v mnohém velmi podobné. Oba dva jazyky patří do rozsáhlé rodiny jazyků Basic. Pokud uživatel ovládá alespoň jeden z jazyků basic zejména Visual Basic nebo Visual Basic for Applications je schopen po krátké aklimatizaci plně používat a orientovat se i v jazyce StarOffice basic, platí to samozřejmě i obráceně.

Jak jsme si řekli v jedné z prvních kapitol, OpenOffice.org vznikl jako možná náhrada za Microsoft Office. Stejně tak to bylo i s programovacím jazykem StarOffice Basic. Bohužel však z důvodů licencování nemohl použít všechny konstrukce, příkazy a ostatní časti jazyka z VBA. Přesto velká část základních konstrukcí je s VBA kompatibilní. Například deklarace proměnných je v obou programovacích jazycích stejná. Deklaruje se pomocí klíčového slovo DIM a jména proměnné a datového typu. V podstatě by se dalo říct, že veškeré základní konstruktory jsou stejné. Drobnou odlišnost si ukážeme v následujícím příkladu. [16]

Jako zajímavost je také porovnání hardwarových požadavků obou programů.

Požadavky OpenOffice.Org Windows

- Windows 2000 (Service Pack 2 a vyšší), Windows XP, Windows 2003, Windows Vista
- naprosté minimum pro spuštění je 256 megabajtů RAM, doporučeno 512 MB
- 650 megabajtů na disku pro běžnou instalaci (včetně Javy); po instalaci a smazání dočasných souborů cca 440 megabajtů
- rozlišení obrazovky 1024×768 a minimálně 256 barev

Požadavky MS Office

- RAM: 1 GB paměti pro x86 nebo 2 GB paměti pro x64
- HDD: 3 GB místa
- Operační systém: Windows 7, Windows 8, Windows Server 2008 R2 nebo Windows Server 2012

Příklad 36: StarOffice Basic - Kvadratická rovnice

```
Sub kvadratickaRovnice

    Dim A, B, C as Double
    Dim D as Double
    Dim x, x1, x2 As Double

A = InputBox("Zadej hodnotu kvadratického členu A", "Kvadratická rovnice")
B = InputBox("Zadej hodnotu lineárníhoo členu B", "Kvadratická rovnice")
C = InputBox("Zadej hodnotu absolutního členu C", "Kvadratická rovnice")

    A = Cdbl(A)
    B = Cdbl(B)
    C = Cdbl(C)

    D = ((B*B) - (4*A*C))

    If D < 0 then
    Print "Kvadratická rovnie nemá řešení v oboru reálných čísel"
    elseIf D = 0 then
    x = (-B + Sqr(D))/(2*A)
    Print "hodnota x je " + x
    else
    x1 = (-B + Sqr(D))/(2*A)
    x2 = (-B - Sqr(D))/(2*A)
    Print "Hodnota x1 je " + x1+ " Hodnota x2 je " + x2

    EndIf
```

Příklad 37: VBA – Kvadratická rovnice

```
Sub kvadratickaRovnice()

Dim A, B, C As Double
Dim D As Double
Dim x, x1, x2 As Double

A = InputBox("Zadej hodnotu kvadratického členu A", "Kvadratická rovnice")
B = InputBox("Zadej hodnotu lineárníhoo členu B", "Kvadratická rovnice")
C = InputBox("Zadej hodnotu absolutního členu C", "Kvadratická rovnice")

A = CDbl(A)
B = CDbl(B)
C = CDbl(C)

D = ((B * B) - (4 * A * C))

If D < 0 Then
MsgBox ("Kvadratická rovnie nemá řešení v oboru reálných čísel")
ElseIf D = 0 Then
x = -B + Sqr(D) / (2 * A)
MsgBox ("Hodnota x je " & x)
Else
x1 = (((-B) + Sqr(D)) / (2 * A))
x2 = (((-B) - Sqr(D)) / (2 * A))
MsgBox ("Hodnota x1 je" & x1 & "hodnota x2 je" & x2)

End If
```

Popis: Můžeme si všimnout, že programy jsou takřka stejné, jediné dvě drobné odlišnost jsou, že VBA používá pro vypsání do dialogového okna funkci MsgBox a pro spojování řetězců znak **&** naproti tomu StarOffice Basic používá **+** a příkaz Print**.**

Větší odlišnost můžeme spatřit u složitějších příkazů. Například pokud budeme chtít načítat data z určité buňky nebo do buňky budeme chtít něco vložit.

Příklad 38: VBA – Načtení dat z buňky

```
Sub soucetBunek()
A = Cells(1, 1)
B = Cells(1, 2)

vysledek = A + B
MsgBox (vysledek)
Cells(2,2) = vysledek
End Sub
```

Příklad 39: StarOffice Basic – Načtení dat z buňky

```
Sub soucetBunek
A = thisComponent.Sheets(0).getCellByPosition(0,0).Value
B =  thisComponent.Sheets(0).getCellByPosition(1,0).Value
vysledek = A + B
thisComponent.Sheets(0).getCellByPosition(1,1).Value= vysledek
Print vysledek
End Sub
```

Popis: V tomto příkladu, už vidíme větší rozdíl mezi jazyky. VBA používá pro přístup k buňkám klíčové slovo Cells(řadek, sloupec) na rozdíl od StarOffie Basic, který má přístup k buňkám složitější. Používá se pro ni konstrukce thisComponent.Sheets(0), která nám určuje, že budeme pracovat s listem 1. getCellByPosition(sloupec, řádek) je odkaz na buňku na pozici A1. Můžeme si

všimnout, že čísla sloupců a řádků se zadávají v opačném pořadí než u MS Excel. Navíc jsou řádky i sloupce v OpenOffice.org značeny od 0.[8]

Závěr

Cíle této bakalářské práce bylo vytvořit učebnici strukturovaného programování v programovacím jazyku StarOffice Basic. Při tvorbě této učebnice, nebyl předpoklad, že čtenář ovládá jakýkoli jiný programovací jazyk, přesto však má základní povědomost o algoritmizaci a programu OpenOffice.org. Tato učebnice vznikla jako možnost pro rozvoj programování na základních a středních školách, které nemají možnost používat výkonný hardware, na kterém by šlo plnohodnotně provozovat složité vývojové prostředí jako je například Visual Studio od společnosti Microsoft.

Jako druhý cíl mé bakalářské práce bylo porovnat možnosti programování v MS Excel a OpenOffice.org. Možnosti programovat v těchto programovacích jazycích jsou takřka rovnocenné.

Myslím, že oba programy jsou vhodné pro výuku základů programování, avšak pro OpenOffice.org mluví cena pořízení a hardwarová náročnost.

Jako rozšíření své práce bych uvažoval, dopsat k učebnici základů strukturovaného programování také učebnici základů objektově orientovaného programování v StarOffice Basic.

Přehled pramenů

Seznam použité literatury

[1] Apache OpenOffice: The Free and Open Productivity Suite. *Apache OpenOffice: The Free and Open Productivity Suite* [online]. [cit. 2016-07-19]. Dostupné z: http://www.openoffice.org/about/

[2] Bakalářská práce OpenOffice.org Writer: Tvorba výukových materiálů. *Bakalářská práce OpenOffice.org Writer: Tvorba výukových materiálů* [online]. [cit. 2016-07-18]. Dostupné z: http://mcodl.wz.cz/bcprace/cz2.htm

[3] BUDINSKÁ, Jana. Stručná historie OpenOffice.org: Jak se hvězda otevřela. *ABC Linuxu* [online]. 2009, , 3 [cit. 2016-07-19]. Dostupné z: http://www.abclinuxu.cz/clanky/recenze/strucna-historie-openoffice.org-jak-se-hvezda-otevrela

[4] ŠPIROCH, Petr. *OpenOffice.org: programování maker* [online]. České Budějovice, 2007 [cit. 2016-07-16]. Dostupné z: http://www.theses.cz/id/vu7d3i/downloadPraceContent_adipIdno_3592. Bakalářská práce. Jihočeská univerzita v Českých Budějovicích.

[5] *StarOfficeTM 7 Office Suite: A SunTM ONE Software Offering* [online]. 1. Santa Clara, 2003 [cit. 2016-07-16]. ISBN 817-1826-10. Dostupné z: http://www.staroffice.com/SO7/so-7-ga-en-BASIC_GUIDE.PDF

[6]SOBEK, Milan. *OpenOffice.org: tipy a triky pro záznam a úpravy maker*. Praha: Grada, 2006. Průvodce (Grada). ISBN 80-247-1374-8.

[7] Podprogram. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-07-16]. Dostupné z: https://cs.wikipedia.org/wiki/Podprogram

[8] PONÍŽIL, Petr. *Makra v OpenOffice.org Calc* [online]. In: . Kroměříž, 2012 [cit. 2016-07-16]. Dostupné z: http://www.gymkrom.cz/web/ict/materialy/Makra_OO-Calc.pdf

[9] Příkaz (programování). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-07-16]. Dostupné z: https://cs.wikipedia.org/wiki/Příkaz_(programování)

[10] Operátor (programování). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-07-16]. Dostupné z: https://cs.wikipedia.org/wiki/Operátor_(programování)

[11] PASTIERIK, Július. *Makrá v OpenOffice.org* [online]. Dačov Lom, 2010 [cit. 2016-07-16]. ISBN 978-80-970479-0-0. Dostupné z: http://www.liberix.cz/doplnky/knihy/ooomakra.pdf

[12] Konstanta (programování). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-07-16]. Dostupné z: https://cs.wikipedia.org/wiki/Konstanta_(programování)

[13]Řídicí struktura. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-07-16]. Dostupné z: https://cs.wikipedia.org/wiki/Řídicí_struktura

[14] Programovanie makier v LibreOffice: Funkcie určené na prácu s premennými. *Programovanie makier v LibreOffice: Funkcie určené na prácu s premennými* [online]. 2016 [cit. 2016-07-19]. Dostupné z: http://www.openoffice.cz/navody/programovanie-makier-v-libreoffice-funkcie-urcene-na-pracu-s

[15] Nápověda OpenOffice.org

[16] LAURENČÍK, Marek. *Programování v Excelu 2007 & 2010: záznam, úprava a programování maker*. Praha: Grada, 2011. Průvodce (Grada). ISBN 978-80-247-3448-4.

Seznam obrázků

Obr. 1: Úvodní strana OpenOffice.org 9
Obr. 2: Záznam makra 10
Obr 3: Spuštění makra 11
Obr 4: Zobrazení kódu makra 12
Obr 5: Nápovědo OpenOffice.org 31

Obr 6: Dialogové okno MsgBox se zvhledem 20 35

Seznam tabulek

Tabulka 1: matematické operátory 17
Tabulka 2: logické operátory 17
Tabulka 3: porovnávací operátor operátory 18
Tabulka 4: datové typy 19
Tabulka 5: Typy proměnných 32
Tabulka 6: Styl dialogového okna MsgBox 34
Tabulka 7: Návratové hodnoty fce MsgBox 35

Seznam příkladů

Příklad 1: Komentáře 15
Příklad 2: Názvy proměnných 16
Příklad 3: Celočíselné datové typy 20
Příklad 4: Typ pro čísla s desetinou čárkou 20
Příklad 5: Měnový datový typ 20
Příklad 6: Datový typ pro datum 21
Příklad 7: Datový typ String 21
Příklad 8: Logický datový typ 21
Příklad 9: Deklarace proměnných 22
Příklad 10: Deklarace proměnné s dlouhým zápisem datového typu 23
Příklad 11: Deklarace proměnné pomocí zkráceného zápisu datového typu 23
Příklad 12:Deklarace proměnné pomocí zkráceného zápisu datového typu 24
Příklad 13: Práce s různými datovými typy 24
Příklad 14: Práce s globálními proměnnými 26
Příklad 15: Práce s globálními a lokálními proměnnými 27
Příklad 16: Pole – array 28
Příklad 17: Pole – array 29
Příklad 18: Funkce pro práci s polem 29
Příklad 19: Definice konstanty 30
Příklad 20: Funkce pro určení datového typu 31

Příklad 21: Funkce pro přetypování proměnných 33
Příklad 22: Funkce MsgBox 34
Příklad 23: Příkaz MsgBox 35
Příklad 24: Použití příkazu a funkce MsgBox 36
Příklad 25: Použití InputBox 36
Příklad 26: Větvení programu 38
Příklad 27: Použití větvení programu 38
Příklad 28: Select – case 39
Příklad 29: Cyklus – Do While 40
Příklad 30: Cyklus – Do Untile 40
Příklad 31: Cyklus – Do Loop While 41
Příklad 32: Cyklus – For 41
Příklad 33: Příkaz GoTo 42
Příklad 34: Trigonometrické funkce 44
Příklad 35: Funkce pro práci s řetězci 45
Příklad 36: StarOffice Basic - Kvadratická rovnice 48
Příklad 37: VBA – Kvadratická rovnice 49
Příklad 38: VBA – Načtení dat z buňky 50
Příklad 39: StarOffice Basic – Načtení dat z buňky 50