JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

PŘÍRODOVĚDECKÁ FAKULTA
ÚSTAV APLIKOVANÉ INFORMATIKY

Sbírka řešených a neřešených úloh ze skriptování ve WSH pro výuku předmětu Operační systémy 2

BAKALÁŘSKÁ PRÁCE

Vedoucí práce: Mgr. Jiří Pech Ph.D.
Autor práce: Tomáš Černý

České Budějovice 2013

Bibliografické údaje

Černý, T., 2013: Sbírka řešených a neřešených úloh ze skriptování ve WSH pro výuku předmětu Operační systémy 2 [Collection of solved and unsolved tasks of scripting in WSH for teaching the subject Operating Systems 2] – 23 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Práce se zabývá problematikou v oblasti skriptování ve VBScriptu. Je vytvořena sbírka řešených a neřešených úloh, která obsahuje jednotlivé příkazy, které VBScript obsahuje a k těmto příkazům jsou vytvořeny ukázkové úlohy. V další části jsou zkonstruovány čtyři neřešené úlohy, na kterých si čtenář otestuje látku, kterou se naučil ze sbírky. V poslední řadě jsou k těmto úlohám přiložena řešení. Sbírka byla testována na studentech předmětu Operační systémy 2. Tato práce může být použita pro výuku, samostudium a testování.

The thesis deals with the problem of scripting in VBScript. The collection of solved and unsolved tasks is created, which contains various commands that VBScript contains and to these commands are created sample examples. In the next section are designed four unsolved tasks, where the reader tests his skills, which he had learned from the collection. Finally, there are solutions, which are attached to this tasks. The collection was tested on students of Operating systems 2. This thesis can be used for teaching, self-study and testing.

Prohlášení

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

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

V Českých Budějovicích dne 2. prosince 2013

Tomáš Černý

1 Úvod

Tato práce je zaměřena na Windows Script Host (skriptování ve Windows). Dále se dělí na dva skriptovací jazyky, JScript (podmnožina Java Scriptu) a VBScript (podmnožina Visual Basicu). Zaměřil jsem se na jeden z těchto jazyků VBScript. Hlavním bodem této práce je vytvoření sbírky úloh, která bude nápomocná při výuce Operačních systémů 2, kde je skriptování ve Windows jednou z kapitol učiva. Sbírka by měla usnadnit výuku této problematiky, porozumění látce a v neposlední řadě posloužit. jako užitečný materiál pro vyučujícího. Materiály jsou postaveny dle rozsahu učiva zmíněného předmětu a jsou určeny pro úplné začátečníky. Pro ověření užitečnosti sbírky jsem vytvořil dotazník, který jsem následně po probrání dané látky předložil studentům a výsledky zpracoval. (Viz kapitola zhodnocení práce.)

Sbírka úloh je rozdělena do jednotlivých kapitol. V první kapitole jsou uvedeny úplné základy skriptování ve VBScriptu, pravidla pro psaní kódu a užitečné funkce, které VBScript obsahuje v základním balíčku. V následujících kapitolách jsem se zaměřil na základní příkazy. U jednotlivých příkazů vysvětlím funkčnost, použití a následný příklad, který je prezentován jako funkční kód. U každého kódu je uveden popis, jak daný skript funguje. Dalším krokem sbírky jsou neřešené úlohy, u kterých je napsáno stručné zadání. Jako poslední bod jsou přidaná řešení k jednotlivým úlohám.

Cílem této bakalářské práce je vytvoření sbírky řešených a neřešených úloh, která má za cíl usnadnit a lépe pochopit tvorbu skriptů ve Windows a studentům tak ulehčit práci při studiu této problematiky. V poslední řadě jsou pro studenty zhotoveny čtyři neřešené úlohy, na kterých si student procvičí nastudované znalosti. Sbírka nemusí sloužit jen pro studenty a vyučujícího, ale i pro jedince, kteří chtějí začít ovládat skriptovací jazyk VBScript a chtějí se naučit základy tohoto jazyku.

Bakalářská práce je rozdělena do dvou hlavních částí. Obě části se dále dělí do jednotlivých podkapitol.

První část práce je zaměřena spíše teoreticky. Obsahuje stručné informace o skriptování obecně. Krátká historie VBScriptu, jak se postupně vyvíjel a jeho použití. Dále obsahuje

informace o tom, jaký byl použit software a na jaké verzi systému Windows byly skripty vytvářeny. Následně je zde podrobný popis celé sbírky. Udává, jak by měl čtenář nejlépe postupovat při studování a jakou má sbírka strukturu. Dalším krokem bylo vytvoření dotazníku pro aktuální studenty předmětu Operační systémy 2, který obsahuje několik otázek, zda byla sbírka úspěšná a jak jí zhodnotili sami studenti. Poté následuje vyhodnocení těchto dat a celkové klady a zápory sbírky.

Druhá část se zaobírá samotnou sbírkou úloh, která je prezentována jako příloha k bakalářské práci. Tato část obsahuje vysvětlení jednotlivých příkazů VBScriptu, které jsou rozděleny do podkapitol. U každého příkazu je také přiložen ukázkový příklad, jak daný příkaz funguje v praxi. Příklady jsou uvedeny přímo ve funkční syntaxi VBScriptu a lze je hned vyzkoušet, aby čtenář lépe pochopil funkčnost příkazu. Kód je také vždy detailně popsán, jak přesně funguje a co se děje při jeho spuštění. V poslední části sbírky jsou pro studenty zkonstruovány čtyři úlohy, na kterých by si měli procvičit znalosti, které se naučili při studování těchto materiálů a tím si lépe zapamatovat učivo. Jsou zde přiloženy i mé řešení, které mohou studenti použít v případě, že si nebudou vědět rady jak začít.

2 Cíle práce

- Sestavit sbírku řešených a neřešených úloh ze skriptování ve VBScriptu
    - Popsat jednotlivé příkazy a vytvořit k nim ukázkové úlohy
    - Vytvořit čtyři neřešené úlohy
    - Přiložit řešení k jednotlivým úlohám
    - Pomocí sbírky ulehčit studium této problematiky a poskytnout tak zdroj informací pro studenty Operačních systémů 2
- Uvést použité nástroje při tvorbě skriptů, otestovat funkčnost skriptů na jednotlivých verzích operačního systému Windows a následně sbírku podrobně popsat a uvést doporučený postup studia
- Vytvořit dotazník a předložit ho studentům Operačních systémů 2
- Dotazník zpracovat, data vyhodnotit a ověřit užitečnost sbírky

3 Teorie

3.1 Skriptování

Skript je posloupnost příkazů, které se provádějí v takovém pořadí, v jakém jsou napsány. Může to být jednoduchý program nebo součást většího programu, ve kterém je obsaženo i několik skriptů. Skript je reprezentován v podobě spustitelného souboru, který má určitou příponu. Koncovka souboru je zvolena podle toho, jaký skriptovací jazyk byl použit. Například ve Windows můžeme použít buď jazyk JScript (přípona *.js) nebo VBScript (přípona *.vbs).

Od plnohodnotného programu se však skript liší ve dvou bodech. Skript je textový soubor, který obsahuje příkazy a je pro nás okem čitelný. Pokud tedy známe příkazy, víme přesně, co skript provede. Na rozdíl od programu, který vyžaduje kompilaci a obsažené instrukce čte přímo procesor. V dalším bodě se liší tím, že skript ke svému spuštění potřebuje interpreta neboli program, který skript zpracuje a chová se podle něj.

3.2 Windows Script Host (WSH)

Windows Script Host je modul, který je obsažen v systému Microsoft Windows. Tento modul slouží k vytvoření prostředí, ve kterém skriptovací stroj postupně spouští příkazy, které obsahuje soubor s kódem skriptu. WSH podporuje několik skriptovacích strojů, včetně skriptů, které jsou napsány skriptovacími jazyky JScript a VBScript.

Umožňuje spouštění těchto skriptů přímo z prostředí Windows nebo pomocí příkazového řádku. Chceme-li spustit skript z příkazového řádku, najdeme cestu k danému skriptu a poté stačí napsat jen jeho přesný název a stisknout klávesu enter. Pro spuštění s prostředí Windows klikneme dvakrát na soubor se skriptem.

Tento modul lze nalézt ve Windows ve dvou podobách. První podobu nalezneme pod názvem programu Wscript.exe, kde vlastnosti skriptu nastavujeme přímo na kartě vlastností. Druhá podoba je založena pouze na příkazovém řádku (nalezneme ji pod názvem programu Csript.exe) a vlastnosti skriptu poté nastavujeme pomocí jednotlivých příkazů s parametry přímo do příkazového řádku. Obě verze je možné spustit z příkazového řádku, kam napíšeme název programu příslušné verze modulu.

3.3 VBScript

Visual Basic Script je skriptovací jazyk, který je obsažen v modulu Windows Script Host. Pokud již máte zkušenosti s programovacím jazykem Visual Basic, syntaxe VBScriptu je stejná, jen s rozdílem že Visual Basic je rozsáhlejší. Existuje ještě další podmnožina jazyka Visual Basic a tím je Visual Basic for Application (VBA). Tento jazyk je integrován s aplikacemi rodiny Microsoft Office, který obsahuje Microsoft Word, Microsoft Excel, atd. VBA si lze představit jako vrstvu mezi Jazykem Visual Basic a skriptovacím jazykem VBScript.

V modulu WSH je obsažen i další skriptovací jazyk a tím je JScript. Tato bakalářská práce je zaměřena především na VBScript, ale zmínění o existenci dalšího skriptovacího jazyka je nutné. Co se týče použitelnosti je JSCript stejný jako VBScript. Jediný rozdíl je v syntaxi zapisování kódu, která je naprosto odlišná a má jiná pravidla. Jak jsem již zmínil VBScript je syntaxí stejný jako programovací jazyk Visual Basic. Stejně tak JScript je podmnožinou JavaScriptu.

Možnosti VBScriptu jsou takřka totožné jako u JavaScriptu. Rozdíl mezi těmito skriptovacími jazyky je především v syntaxi a některé menší rozdíly (funkce, které má JavaScript, nemá VBScript a naopak). Pohlédneme-li do historie, JavaScript vytvořila společnost Netscape a tím rozšířila možnosti svých prohlížečů. Microsoft sice reagoval podporou JavaScriptu ve vlastním prohlížeči (Internet Explorer), ale vytvořil svůj vlastní skriptovací jazyk a tím je právě VBScript. Jen s tím rozdílem, že VBScript funguje pouze v prohlížečích firmy Microsoft tedy Internet Explorer.

3.4 Bezpečnost

VBScript si při spuštění skriptu vyhradí část zdrojů a paměti počítače. V této části si v podstatě může dělat, co chce, ale nesmí tuto část opustit a tím poškodit data v ostatních částech paměti. To znamená, že čistý VBScript nemůže přistupovat k paměti počítače a také nemůže pracovat se soubory na pevném disku. To znamená, že všechny příkazy, které byly nějak nebezpečné (mohly být použity pro vytvoření viru) byly z VBScriptu odstraněny. Tím sice získáme bezpečné prostředí, ale přicházíme o mnoho důležitých a potřebných prvků. Samozřejmě, že existuje řešení. Můžeme použít objekty, ve kterých je

spoustu dalších metod a tím provádět prakticky vše. Takto nás VBScript posouvá až za jeho hranice možností.

Windows Script Host je velmi flexibilní nástroj, který může být použit k automatizaci Windows, ale také může být velmi jednoduše zneužit hackery. S novou verzí WSH 5.6 přichází nový bezpečnostní prvek, který má za úkol zvýšit bezpečnost skriptů, ale na druhou stranu nesnížit funkčnost a moc tohoto nástroje. Uživatelé skriptu mohou nyní ověřit pravost skriptu předtím, než dojde k jeho spuštění. Vývojáři mohou své skripty podepsat, aby nedošlo k neoprávněné úpravě, která by mohla mít za následek rozšíření potencionálně škodlivého kódu. Správci sítí mohou dále nastavit přísné zásady, které určují, kteří uživatelé mohou skripty spouštět.

3.5 Vývoj

Modul Windows Script Host je od verze Windows 98 obsažen v každé instalaci. Od verze 1.0 zaznamenal vývoj WSH mnoho změn. Aktuální informace o změnách v nové verzi najdeme na oficiálních stránkách Microsoftu. [9] Aktuální verze je 5.8. V následující tabulce si ukážeme jakou verzi WSH obsahují jednotlivé systémy Windows.

Verze systému Windows	**Verze WSH**				
	1.0	2.0	5.6	5.7	5.8
Microsoft Windows 98	X				
Microsoft Windows NT 4 Option Pack	X				
Microsoft Windows 2000		X			
Microsoft Windows XP			X		
Microsoft Windows Vista				X	
Microsoft Windows Server 2008				X	
Microsoft Windows 7					X
Microsoft Windows Server 2008 R2					X
Microsoft Windows 8					X

Pro zjištění aktuální verze WSH lze použít jednoduchý skript, který si ukážeme. Použil jsem VBScript.

```
dim ver
ver=Wscript.version
MsgBox "Vaše verze WSH je: "&ver
```

Pro zjištění verze použijeme funkci již obsaženou ve VBScriptu. Do proměnné „ver“ uložíme hodnotu, kterou jsme zavolali touto funkcí a následně ji vypíšeme pomocí *MsgBoxu*.

3.6 Použití

VBScript má velice široké možnosti použití. Může sloužit jako automatizovaný nástroj pro Windows. Stačí napsat po sobě jdoucí příkazy, co přesně se má stát a poté skript jednoduše spustit. Výhodou je, že skript lze jednoduše přenést do jiného počítače. Příklad použití může být hromadná instalace určitého programu. Na cílovém počítači se jen spustí skript a systém Windows sám nainstaluje program podle parametrů zadaných ve skriptu. Dalším použitím může být zrychlení práce se systémem. Příkladem takového skriptu může být automatické vypnutí Windows. Stačí jen spustit skript a systém Windows se sám vypne po nějakém čase.

Dále lze VBScript použít k obohacení chování webové stránky. Skript se vloží do hlavičky HTML kódu a odehrává se na straně prohlížeče, proto VBScript funguje i v offline režimu. Jeho funkčnost se dá přirovnat k JavaScriptu (pohybující se text, hodiny, kalendář, atd.). Jediná hlavní a velice zásadní nevýhoda je, že VBScript lze použít pouze v prohlížeči společnosti Microsoft, kterým je Internet Explorer. Odpověď na otázku který skriptovací jazyk použít na webové stránce je takto jednoznačná. Pokud použijeme VBScript, uživatelé ostatních prohlížečů (Mozzila, Opera, Chrome) budou o funkčnost stránky ochuzeni, což je velice nežádoucí. Efektivní použití je v uzavřené firemní sítí, kde víme, že zde není nainstalovaný jiný prohlížeč než Internet Explorer. Pokud tvoříte stránky pro širokou veřejnost VBScript rozhodně nepoužijte.

4 Použitý Software

Při psaní VBScriptu nepotřebujete žádný speciální nástroj. Stačí vám jednoduchý textový editor, který je již integrován ve Windows. Pokud chcete větší přehled v kódu, doporučoval bych použít volně šiřitelný textový nástroj PSPad, který je univerzální a používá se také při psaní HTML stránek. Pro přehled PSPad barevně odlišuje proměnné, komentáře, funkce, atd. V nastavení tohoto programu si dále můžete zvolit vlastní odlišení barev přímo pro skriptovací jazyk VBScript. Dále v tomto programu najdeme i přednastavené šablony pro mnoho programovacích jazyků včetně VBScriptu.

Skripty obsažené ve sbírce jsou vypracovány v nástroji PSPad ve verzi 4.5.7. Výhodou je, že je dostupný zdarma a přehled, který díky tomuto programu máme, nám velice usnadní práci a hlavně rychlost při psaní kódu. Díky široké škále nastavení si prostředí upraví každý podle svých představ tak, aby to programátorovi maximálně vyhovovalo. Velikou výhodu je zde i orientace v kódu. Pokud někde uděláme chybu, systém Windows nám v chybové hlášce napoví, na jakém řádku se chyba nachází. Díky číslování řádků, se dá velice snadno chyba lokalizovat a následně opravit. PSPad bych určitě při psaní skriptů doporučil, oproti jednoduchému textovému dokumentu má tento program mnoho velice užitečných kladů. Jiné textové editory k psaní skriptů v jazyce VBScript vyzkoušeny nebyly, protože nástroj PSPad byl plně dostačující.

Všechny skripty obsažené ve sbírce jsou vytvořeny na operačním systému Microsoft Windows 8 Professional ve 32 bitové verzi. Aktuálně nejnovější verze operačního systému je Windows 8.1, ve které je také obsažen modul WSH ve verzi 5.8. Protože tato nejnovější verze obsahuje stejnou verzi WSH jako Windows 8, nebyly zde nalezeny žádné rozdíly ve funkčnosti. Skripty byly také otestovány na systému Microsoft Windows 7 a XP, kde jsem také nezaznamenal jediný rozdíl ve funkčnosti a kompatibilitě. Jsem přesvědčen, že v dnešní době běží většina strojů na operačním systému MS Windows 8 nebo 7, popřípadě Vista a v ojedinělých případech Windows XP. Testování skriptů na systémech starších než Windows XP ztrácí smysl.

Od verze operačního systému Microsoft Windows 7 je v každé instalaci obsažen VBScript 5.8. Byla použita nejnovější verze, vzhledem k použité verzi Windows. Tato verze je i nově podporována v prohlížečích Internet Explorer veze 8.0 a více. Všechny změny a

přidané funkce do novějších verzí VBScriptu jsou zveřejněny na oficiálních stránkách Microsoftu. [10]

5 Popis práce

5.1 Struktura sbírky

Struktura sbírky je vymyšlena tak, aby čtenář nejprve podrobně nastudoval teoretickou část o skriptování ve VBScriptu a následně si nastudované znalosti otestoval na přiložených úlohách. Tím lépe vstřebá danou problematiku a vyzkouší si skriptování v praxi.

První část sbírky se věnuje kapitole, ve které je krátce uvedeno co vlastně VBScript je a ke kterému jazyku by se dal porovnat. Pokud čtenář již zná jiný příbuzný skriptovací jazyk, studování problematiky VBScriptu bude pro něj snadnější a získá tak lepší přehled. Poté následují informace o základních pravidlech psaní kódu, jakou musí mít skript příponu a jak se spouští. Tato část obsahuje i způsob jak přímo v kódu psát komentáře a tím zvýšit přehlednost a orientaci nejen pro tvůrce skriptu, ale i pro ostatní, aby lépe porozuměli, jak daný skript má fungovat. Následuje seznámení čtenáře s kapitolou, kde se dozví, jaká jsou pravidla pro názvosloví v syntaxi VBScriptu. Zde se naučí, jak má vybírat správný název proměnných, funkcí a procedur. V poslední části seznámení s VBScriptem je obsaženo několik funkcí, které VBScript obsahuje již v základu. Tato část je zde především pro to, aby se k ní čtenář při vytváření skriptu mohl vrátit a nalézt zde popis a použití právě pro funkci, kterou zrovna potřebuje.

5.1.1 Rozdělení příkazů do kapitol

V dalším kroku sbírky již následuje vysvětlení jednotlivých příkazů s vytvořeným příkladem, aby si mohl čtenář vyzkoušet funkčnost v praxi. Příklad je vždy podrobně popsán, pro lepší pochopení co se přesně v daném příkladu děje. Rozdělení těchto kapitol jsem volil tak, aby šly od těch jednodušších po složitější. Co se čtenář naučí v předchozích kapitolách, použije v těch následujících. To znamená, že v každé následující kapitole je předpoklad, že čtenář již umí látku z předchozích kapitol.

Z tohoto důvodu jsem jako první kapitolu zvolil operátory. V této oblasti je uvedeno jaké všechny operátory může programátor využít a jaké vlastnosti jednotlivé operátory mají. Používání a vlastnosti těchto operátorů je ve všech programovacích a skriptovacích jazycích podobné. Tedy znalost jiného jazyku přináší značnou výhodu při studiu nejen této kapitoly, ale i celé problematiky VBScriptu.

Podobně jako z předchozí kapitoly, i v této se nacházejí informace, které čtenář bude potřebovat ve všech následujících kapitolách. Jedná se o část zaměřenou na to, jak se v kódu vytvářejí proměnné a jaký datový typ v takových proměnných můžeme uchovávat. Stejně tak jako u operátorů i datové typy jsou v ostatních jazycích podobné s rozdílem jiného značení. Datový typ proměnné můžeme mezi sebou převádět právě pomocí funkcí, které jsou již ve VBScriptu obsaženy. Tyto funkce najdeme ve sbírce pod kapitolou *2.1.2 Další užitečné příkazy a funkce*. V této kapitole je dále obsažena podkapitola, kde je popsán složitější typ proměnných. Tato podkapitola nese název Pole a je o něco složitější než jednoduché proměnné. Najdeme zde popis jak pole deklarovat, jeho funkčnost a způsob jak do něj ukládat jednotlivá data. Pole má několik druhů, které jsou zde zmíněny a následující příklad, na kterém je vidět jak pole deklarovat a jak si lze jednotlivá data v poli představit.

Další kapitola již popisuje jednu z hlavních funkcí VBScriptu a tou je vyskakovací okno, které má zkratku *Msgbox*. Tato zkratka je odvozena od anglického slova Message (zpráva) a Box (krabice neboli okno s určitými vlastnostmi). Jejím hlavním účelem je zobrazit uživateli například hodnotu nějaké proměnné nebo informovat o stavu skriptu. V této kapitole se čtenář dozví základní vlastnosti tohoto vyskakovacího okna a jeho syntaxi. Tato funkce obsahuje mnoho vlastností, které může programátor nastavit. Například odlišné druhy vzhledu tohoto okna a přizpůsobit tlačítka tak jak je pro daný skript vhodné.

V následující kapitole je popsána další hlavní funkce, která je syntaxí a funkčností velice podobná funkci *Msgbox* z předchozí kapitoly. Nazývá se *Inputbox* a již podle názvu lze rozpoznat rozdíl od *Msgboxu*. Podle anglického slova Input (vložit) lze usoudit, že půjde o vkládání hodnot ve skriptu. Tato funkce je také reprezentována vyskakovacím oknem, ale má jiné vlastnosti. Hlavním rozdílem je, že obsahuje pole, do kterého uživatel zadá hodnotu, která se uloží do proměnné a tím ovládá skript. Na rozdíl od *Msgboxu*, kde je možné měnit zobrazovaná tlačítka, zde je měnit nelze. *Inputbox* zobrazuje pouze tlačítka *OK*, kterým uživatel potvrdí zadanou hodnotu a *Storno*.

Další kapitoly se již věnují příkazům uvnitř skriptu. Tyto příkazy již nemají žádná vyskakovací okna jako předešlé funkce, ale bez nich by se neobešly, protože by zde nebyla možná interakce s uživatelem.

První takové příkazy, které sbírka obsahuje, jsou podmínky. Zde se čtenář dozví co to podmínka je, k čemu se používá a jak správně podmíněný příkaz naprogramovat. Tato kapitola byla vybrána jako první, protože syntaxe podmínek není tolik složitá a je snadněji pochopitelná. Podmínky jsou dále rozděleny do dvou podkapitol, kterými jsou podmínka *If* a podmínka *Case*. Tyto dvě podmínky jsou od sebe úplně odlišné. Jedinou společnou část tvoří jejich funkčnost, oba příkazy fungují právě tehdy, kdy je splněna určitá podmínka a poté jsou splněny i příkazy za podmínkou. Obě podmínky jsou popsány v těchto podkapitolách tak, aby čtenář viděl rozdíl mezi nimi a dokázal se rozhodnout, jakou podmínku je vhodnější použít v různých situacích. Na přiložených příkladech je možno obě podmínky vyzkoušet a poznat rozlišnost v praxi.

Dalšími, velice důležitými příkazy ve VBScriptu jsou cykly. Nejprve je ve sbírce uveden stručný popis cyklu, za jakých podmínek cyklus funguje a co ovlivňuje počet opakování. Cykly jsou dále rozděleny do třech podkapitol. V těchto podkapitolách se čtenář naučí používat tři typy cyklů.

Jako první byl zvolen cyklus *Do---Loop*, v překladu Do (dělej) a Loop (opakuj). Tento cyklus opakuje daný kus kódu do té doby, dokud je nebo není splněna určitá podmínka. Čtenář je dále obeznámen s dvěma možnými způsoby deklarace. Je možno použít buď klíčové slovo While (zatímco), které opakuje cyklus, zatímco je splněna určitá podmínka nebo Until (než), které opakuje cyklus tak dlouho, než začne nějaká podmínka platit. Následně jsou zde uvedeny dva typy syntaxe, které lze použít u obou těchto klíčových slov. Buď se bude podmínka vyhodnocovat na začátku cyklu, nebo na konci. Obě tyto varianty jsou ve sbírce demonstrovány na příkladech, kde je vysvětleno, jaký je mezi nimi rozdíl. Tento cyklus obsahuje ještě další příkaz, kterým je Exit (konec). Tímto příkazem je možno cyklus ukončit dříve než dojde k cíli. Zde je zhotoven další příklad ve formě funkčního skriptu, kde lze vidět, jak tento příkaz funguje.

V pokračující podkapitole byl zvolen cyklus *While---Wend*, který v překladu znamená (*zatímco---zamiř*). To znamená, že zatímco platí daná podmínka, proveď následující část

kódu. V této podkapitole se čtenář v podstatě neučí nic nového, pokud zvládl předchozí kapitolu bez problému, s tímto cyklem nebude mít žádný problém. Jedná se o cyklus, který je zjednodušenou verzí předchozího. Čtenář zde nalezne informace o odlišnosti od předchozího cyklu, použití a také názorný příklad jak tento zjednodušený cyklus použít.

Pro poslední podkapitolu, která se věnuje problematice cyklů, byl zvolen cyklus *For---Next*. Pokud přeložíme z anglického znění, dostaneme znění cyklu, který znamená, pro jeden krok udělej následující příkazy a poté následuje další krok cyklu. Tento cyklus je na pochopení nejsložitější a proto byl vybrán do poslední kapitoly, kdy už čtenář má základná znalosti o cyklech obecně. Obsahuje větší množství klíčových slov, které jsou čtenářovi podrobně vysvětleny, tak aby je pochopil s nejmenšími obtížemi a následně je znázorněna přesná syntaxe tohoto cyklu. Na rozdíl od předchozích dvou cyklů, tento nemá žádnou podmínku, pod kterou se cyklus spustí, ale počet cyklů ovládá sám programátor. Následující příklad byl zvolen tak, aby na něm byly demonstrovány všechna klíčová slova, která cyklus obsahuje a čtenář tak mohl vidět co se přesně v cyklu děje krok za krokem. Ukázkový příklad je také podrobně popsán.

Po rozsáhlejší kapitole, která se zabývala problematikou cyklů, následuje další větší kapitola s názvem Procedury. Zde se čtenář dozví základní informace o procedurách a jejich použití. Procedury slouží k ulehčení práce, pokud potřebujeme spustit určitou část kódu na více místech skriptu. Abychom nemuseli dokola psát stejný kód, použijeme proceduru. Tato kapitola se zabývá tím jak proceduru správně napsat a poté jí v kódu zase zavolat. Jsou zde popsány dva způsoby, jak může být procedura zavolána a informace o tom, kam správně napsat argument procedury. Na následném příkladu je deklarována jednoduchá procedura, na které si čtenář vyzkouší jak s procedurou zacházet a jakým způsobem jí zavolat.

Poslední kapitolou, která se zabývá příkazy ve VBScriptu je kapitola s názvem Funkce. Funkce jsou velice příbuzné procedurám, větší část kódu spustíme jedním příkazem několikrát ve skriptu, ale jejich použití je trochu odlišené. Tyto odlišnosti od procedur, jsou obsaženy právě v této kapitole. Stejně tak jako u procedur, i zde je nástin správné syntaxe těla funkce a její následné zavolání. Tato kapitola byla vybrána jako poslední, protože její pochopení je složitější než u procedur a v následném ukázkovém příkladu se čtenáři vyplatí znát funkčnost procedur. Aby ukázkový příklad fungovat tak jak má, byl zde

převeden typ proměnných na typ Double. Popis této funkce na převod typu proměnných najde čtenář ve sbírce. Viz kapitola *2.1.2 Další užitečné příkazy a funkce*.

5.2 Doporučený postup při studiu

Sbírka úloh byla sestavena tak, aby čtenář v části, kde se nachází teorie s jednotlivými příkazy, postupoval od začátku ke konci, protože jednotlivé kapitoly jdou po sobě, tak aby na sebe postupně navazovali. Čtenář tak bude množství informací na sebe postupně nabalovat a tím si naučenou látku lépe zapamatuje. Výjimkou ve sbírce úloh je kapitola *2.1.2 Další užitečné příkazy a funkce*. Tato kapitola se nachází na začátku sbírky a obsahuje jednotlivé funkce, které již VBScript obsahuje. Čtenář se tak hned setká s pojmem funkce a nemusí zcela jednotlivým funkcím porozumět. Tuto kapitolu si čtenář pamatovat nemusí, ale je nutné, aby si ji přečetl a měl na paměti, že je ve VBScriptu obsaženo několik vlastních funkcí. Důvod proč je tato kapitola obsažena na začátku sbírky je prostý, při postupném studování sbírky čtenář na tyto funkce, u nějakých ukázkových příkladů, narazí a již bude mít na paměti, že takové funkce existují a poté se může k této kapitole vrátit a zjistit jak přesně daná funkce funguje.

Poté co je čtenář obeznámen o základních informacích o VBScriptu, kam lze tento programovací jazyk zařadit a kterému programovacímu jazyku je podobný, navazuje sbírka na základy VBScriptu a jaké jsou pravidla pro vybírání názvů proměnných, procedur nebo funkcí. Těmto kapitolám by se měl čtenář pečlivě věnovat a nastudovat je, protože nadále je předpoklad, že je již zvládá.

Na tuto část jsou již navázány kapitoly, které obsahují jednotlivé příkazy, jejich vysvětlení a následný ukázkový příklad. U každého příkazu je uveden popis funkčnosti, který by si měl čtenář pečlivě nastudovat. Dále ukázkový skript, který si sám spustí a pokud potřebuje, přečte si u každého příkladového skriptu přiložený podrobný popis, co se v daném skriptu přesně děje. U složitějších příkazů na pochopení, jako jsou například cykly nebo funkce, je doporučeno ukázkový skript nejen spustit, ale i vyzkoušet měnit různé hodnoty a sledovat jak se skript při změně zachová. Pokud čtenář narazí na něco, čemu dostatečně nerozumí, vrátí se ve sbírce zpět a v kapitole, kde se tato problematika nachází, najde vysvětlení. Pokud ve sbírce nenajde informace, které potřebuje, obrátí se na vyučujícího nebo se pokusí vyhledat jiný zdroj.

Po nastudování všech příkazů, které sbírka obsahuje, jsou pro čtenáře sestaveny čtyři neřešené úlohy, které jsou sestaveny tak, aby si čtenář vyzkoušel většinu látky, kterou si nastudoval ze sbírky. Úlohy jsou především k tomu, aby čtenář nad danou problematikou přemýšlel a tím si učivo lépe zapamatoval. Pří řešení úloh je možno použít pouze vlastní hlavu a informace, které se čtenář naučil ve sbírce nebo se vrátit k danému příkazu a tyto informace si obnovit. U každé úlohy je připravena i nápověda, která má za úkol čtenáře nasměrovat jaký příkaz může při vypracovávání použít. Dále je možno použít i ostatní zdroje jako internet, rady vyučujícího, atd. V poslední řadě jsou k úlohám vypracovány funkční řešení, v podobě kódu skriptu. Tato řešení by měl čtenář použít jako poslední zdroj informací k vypracování zadaných úloh.

5.3 Úlohy

Součástí sbírky jsou, již zmíněné, čtyři neřešené úlohy. Tyto úlohy jsou, dle mé úvahy, sestaveny podle obtížnosti a obsahově by měly pokrýt většinu látky, kterou se čtenář naučí ze sbírky.

První úlohou je vytvořit skript, který bude řešit, zda je jedno číslo dělitelné druhým. Obě zmíněná čísla by si měl uživatel skriptu zadat sám. Tato úloha byla zadána jako první, protože je považována za tu nejlehčí ze všech zadaných. Je zde uvedena malá nápověda, která čtenáře nasměruje na kapitolu s operátory, kde by, v případě, že by si nevěděl rady, měl najít případnou pomoc.

Úloha dva má zadání vytvořit skript, který bude počítat obvod a obsah u zmíněných geometrických tvarů, které jsou vypsány v zadání. Tato úloha není složitá na přemýšlení, ale na to že čtenář bude muset napsat větší množství kódu a dát pozor aby zde neudělal chybu. Čtenář je naveden tak, aby pro výběr jednotlivých geometrických tvarů použil podmínku *Case* a Pokud by byla potřeba převádět typ proměnné, nápovědu nalezne v příslušné kapitole, která je zde uvedena.

V následující třetí úloze má čtenář za úkol vytvořit skript, který bude počítat faktoriál ze zadaného čísla. Tento skript je považovaný za mírně obtížnější a je zde předpoklad, že čtenář ví z matematiky, jak se faktoriál počítá. Měly by na něm být otestovány schopnosti z cyklických příkazů. V nápovědě je čtenář směřován tak, aby použil cyklus *For---Next* a zamyslel se nad tím, jak přesně tento příkaz funguje.

Poslední úlohou je vytvořit skript, který bude převádět číslo z dvojkové soustavy do desítkové. Tato úloha je dle mého názoru nejsložitější, a proto je zadána jako poslední. Skript musí být ošetřen proti zadání čísla, které není ve dvojkové soustavě. Na této úloze si čtenář procvičí nejen podmínkové příkazy, ale i cykly a bude muset použít funkce, které VBScript obsahuje. Je zde přiložena nápověda, ve které jsou uvedeny funkce, které byly při tvoření skriptu použity, ale není podmínkou, aby je čtenář využil.

5.4 Dotazník pro studenty

Poté co byla sbírka úloh předložena studentům Operačních systémů 2, byl následně vytvořen dotazník, který obsahuje devět stručných otázek. Tento dotazník má za úkol dostat zpětnou vazbu od studentů a zjistit tak, jestli byly dosaženy cíle práce, jak se studentům se sbírkou pracovalo a zda se vyskytly nějaké problémy při vypracovávání jednotlivých úloh. Pro vyplnění dotazníku byl použit výukový server Moodle.

Dotazník byl anonymní a studentům byly položeny tyto otázky.

1) Pracoval/a jste se sbírkou úloh?
   a) Ano
   b) Ne

2) Jste ochoten/na odpovědět na pár otázek?
   a) Ano
   b) Ne

3) Setkali jste se již s podobnou sbírkou? Pokud ano, napište čím je ve srovnání s jinou lepší či horší.
   a) Ano
   b) Ne

4) Vyskytl se nějaký problém při řešení zadaných úloh? Pokud ano, napište jaký.
   a) Ano
   b) Ne

5) Jak moc obtížené pro Vás bylo vypracování úloh?

a) Velmi těžké
b) Těžké
c) Středně těžké
d) Jednoduché
e) Naprosto jednoduché

6) Jak jste postupoval/a při řešení jednotlivých úloh? Můžete zaškrtnout více odpovědí.
   a) Vypracoval jsem je z hlavy
   b) Stačilo mi jen to, co jsem si zapamatoval při čtení sbírky
   c) Vracel jsem se k teorii ve sbírce
   d) Použil jsem přiložená řešení
   e) Použil jsem i jiné zdroje. Uveďte jaké

7) Byla sbírka dostatečně přehledná?
   a) Přehledná
   b) Částečně přehledná
   c) Nepřehledná

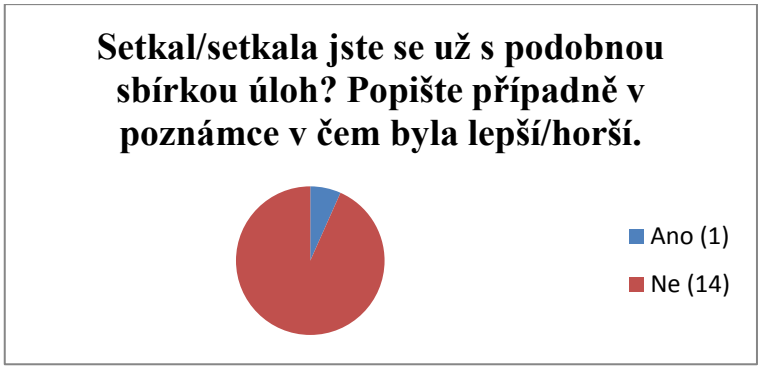
8) Co si myslíte, že by se dalo zlepšit? Napište krátkou odpověď.

9) Pomohla Vám tato sbírka při studiu problematiky VBScriptu?
   a) Ano
   b) Ne

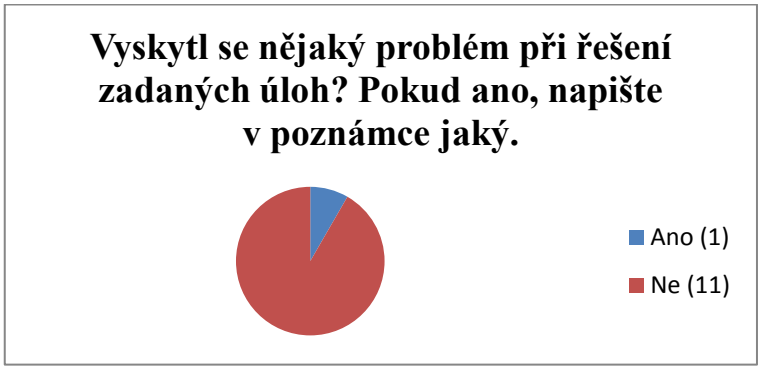
5.5 Vyhodnocení dat

Z celkového počtu 32 studentů, kteří měli aktuálně zapsaný předmět Operační systémy II, vyplnilo dotazník 16 studentů. Pro vyhodnocení dat z jednotlivých otázek byl použit program Microsoft Excel. První dvě otázky v dotazníku nebyly použity do vyhodnocování

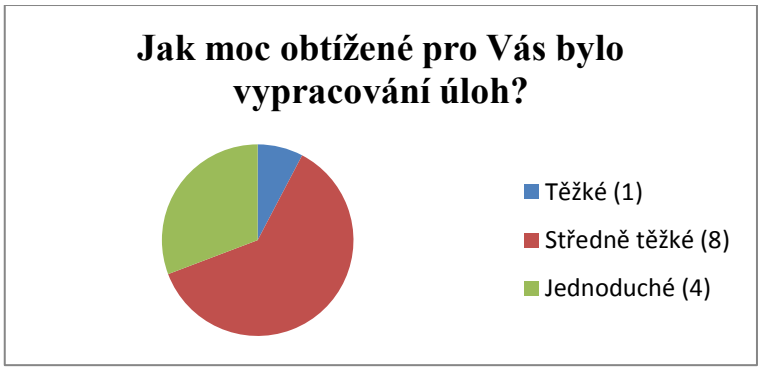
jednotlivých dat. Otázky nebyly povinné, pokud student nechtěl na danou otázku odpovědět, měl možnost ji přeskočit.



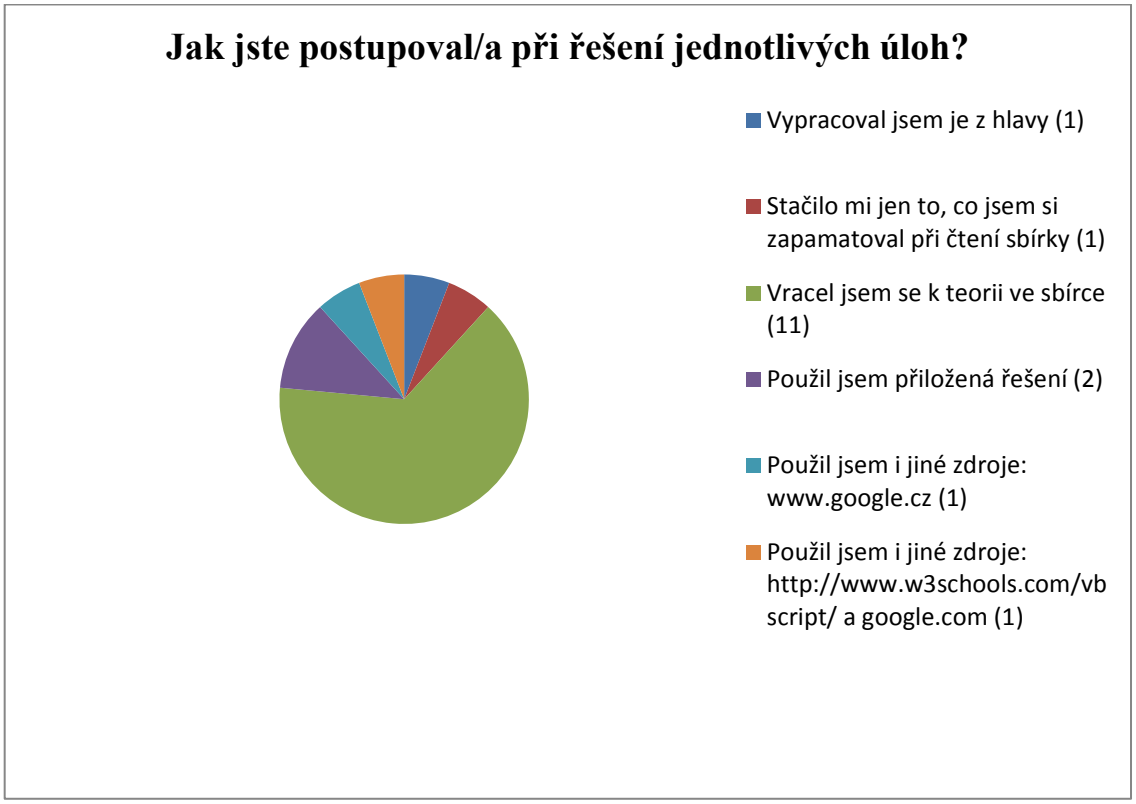
Jediný student uvedl, že se již s podobnou sbírkou setkal, ale v následném popisu uvedl, že viděl mnoho sbírek, ale žádnou na toto téma.



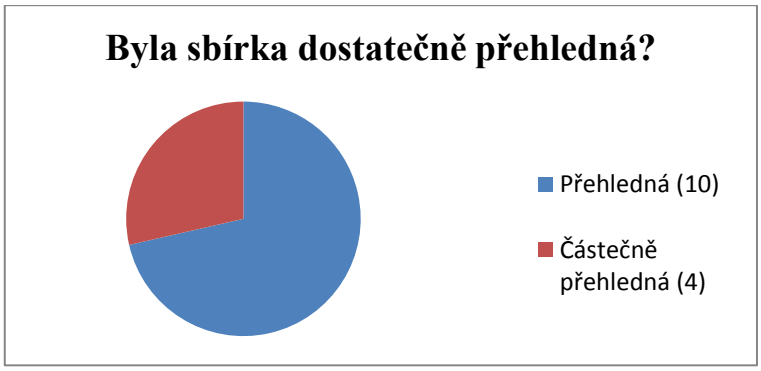
Opět jediný student uvedl, že při řešení se vyskytl nějaký problém, ale bohužel do poznámky neuvedl, o jaký problém se jednalo, a tím zamezil možnost tento problém vyřešit.



Jednotlivé úlohy byly postaveny tak, aby nebyly zcela jednoduché a student tak musel nad jednotlivými úkoly přemýšlet a zároveň nebyly tolik obtížné, aby měl čtenář motivaci úlohy vypracovat. Tento cíl byl splněn dle očekávání.



Obsažené úlohy ve sbírce odpovídali rozsahu učiva v teoretické části sbírky. Cílem bylo, aby čtenář, pří vypracovávání úloh, použil informace, které se naučil ze sbírky, popřípadě se vracel zpět k teorii. U této otázky byla možnost zaškrtnutí více odpovědí.



Sbírka byla sestavena tak, aby se v ní čtenář dokázal bez problému orientovat. Všechny informace byly psány stručně a výstižně a oddělení kódu skriptu od textu bylo provedeno změnou fontu písma kódu a kód je vložen do šedivého rámečku, který je též vyplněn šedou

barvou. Lze tak jednoduše odlišit vlastní kód skriptu od textu. Tento cíl byl z větší části splněn.

Hlavním cílem sbírky úloh bylo usnadnění studia problematiky VBScriptu. Z dat z dotazníku vyplynulo, že tento cíl byl splněn.

5.6 Ohlasy na sbírku

Po vyhodnocení dat z dotazníku bylo usouzeno, že většina názorů na sbírku byla kladná. Na otázku co by studenti na sbírce úloh změnili, odpovídali, že pro naučení základů o skriptování ve VBScriptu je sbírka plně dostačující a přehledná. Díky této sbírce studenti lépe pochopili, jak mají s tímto skriptovacím jazykem pracovat. Studenti uvedli, že doplňující informace dále hledali na internetu, převážně v anglicky psaných zdrojích a dále také z instruktážních videí, které byly nalezeny na serveru Youtube. Jedinou věc, kterou sbírce vytkli, bylo zvýraznění důležitých informací v textu.

S vypracováním vytvořených neřešených úloh neměli studenti větší obtíže. Úlohy pro ně nebyly jednoduché, ale ani nijak složité a za pomoci informací ve sbírce, ke kterým se většinou při vypracovávání úloh vraceli, zvládli úlohy splnit, bez toho aniž by se vyskytl větší problém.

6 Závěr

Byla vytvořena sbírka řešených a neřešených úloh ze skriptování ve VBScriptu, ve které je obsažen popis jednotlivých příkazů v rámci učiva předmětu Operační systémy 2 a k nim sestaveny ukázkové příklady. Následně zde byly vytvořeny čtyři neřešené úlohy, určené k testování skriptů v praxi a k nim přiložena jednotlivá řešení. Tato sbírka byla studentům zveřejněna pomocí výukového serveru Moodle a je přílohou této bakalářské práce.

V této práci je dále uvedeno, na jakém systému Windows byly skripty vytvářeny a na jakých dalších platformách byly skripty testovány a zkoumány odlišnosti ve funkčnosti. Je zde uvedeno, jaké nástroje byly použity při tvorbě skriptů a které z nich se ukázaly jako nejvhodnější. Dále je sbírka podrobně popsána a uveden doporučený postup při studiu.

Hlavním cílem měla být užitečnost sbírky při studiu předmětu Operačních systémů 2. Pro zjištění, zda byl tento cíl splněn, byl vytvořen dotazník, který byl studentům zveřejněn, stejně jako sbírka úloh, na výukovém serveru Moodle. Po vyhodnocení dat bylo zjištěno, že sbírka studentům pomohla při studiu problematiky VBScriptu a tím byl tento cíl splněn dle očekávání. Sbírka nemusí být použita pouze pro studenty Operačních systémů 2, ale i pro jedince, kteří se chtějí naučit základy VBScriptu.

7 Použitá literatura

[1] POKORNÝ, Jan. *Programování ve Visual Basicu 6.0*. 1. vyd. České Budějovice: KOPP, 2001, 373 s. ISBN 80-723-2044-0.

[2] BRŮHA, Luboš. *Začínáme programovat v jazyce Visual Basic .NET*. Vyd. 1. Praha: Computer Press, 2002, xiv, 302 s. ISBN 80-722-6785-X.

[3] HALVORSON, Michael. *Microsoft Visual Basic 6.0 Professional: krok za krokem*. Vyd. 1. [sic]. Praha: Computer Press, 2001, xxxii, 545 s. Programování. ISBN 80-722-6445-1.

[4] *VBScript* [online]. 2013 [cit. 2013-11-28]. Dostupné z: http://msdn.microsoft.com/en-us/library/t0aew7h6%28v=vs.84%29.aspx

[5] NĚMEC, Marek. *Microsoft Windows Script* [online]. 19. 12. 2001 [cit. 2013-11-28]. Dostupné z: http://www.zive.cz/clanky/microsoft-windows-script/sc-3-a-104388/default.aspx

[6] *VBScript* [online]. [cit. 2013-11-28]. Dostupné z: http://www.promotic.eu/cz/pmdoc/ScriptLangs/VBScript/VBScript.htm

[7] *VBScript - Visul basic Scripting* [online]. [cit. 2013-11-28]. Dostupné z: http://www.tvorba-webu.cz/javascript/vbscript-visual-basic-script.php

[8] *VBSCRIPT IN A NUTSHELL* [online]. [cit. 2013-11-28]. Dostupné z: http://www.oreilly.de/catalog/vbscriptian2/chapter/ch07.pdf

[9] *What's New In Windows Script Host* [online]. [cit. 2013-11-28]. Dostupné z: http://msdn.microsoft.com/en-us/library/50ss7kw2%28v=vs.84%29.aspx

[10] *VBScript Version Information* [online]. [cit. 2013-11-28]. Dostupné z: http://msdn.microsoft.com/en-us/library/4y5y7bh5%28v=vs.84%29.aspx

8 Příloha

Přílohou k této práci je samotná sbírka řešených a neřešených úloh, která je zde dále uvedena v nezkrácené verzi.

PŘÍLOHA K BAKALÁŘSKÉ PRÁCI

Sbírka řešených a neřešených úloh ze skriptování ve VBScriptu pro výuku předmětu Operační systémy 2

Tomáš Černý

České Budějovice 2013

1 Úvod do VBScriptu

VBScript (Visual Basic Script) je v podstatě zjednodušená verze VisualBasicu, která se používá pro MS Office. Je to skriptovací jazyk, který nepotřebuje kompilaci a spadá pod Microsoft. Lze říci, že je to určitá Microsoftová verze JavaScriptu. Vznikl spolu s jazykem JScript (obdoba VBScriptu, ale má jinou syntaxi). Od verze Windows 98 je VBScript obsažen v každé instalaci. Při použití v HTML stránkách běží VBScript přímo v prohlížeči uživatele, ale lze použít pouze v prohlížeči Internet Explorer verze 3 a vyšší. VBScript je použitím velice podobný JavaScriptu, s tím rozdílem, že JavaScript je možné použít i v ostatních prohlížečích a má odlišnou syntaxi. Spojením těchto dvou jazyků vzniká velice silný a efektivní nástroj.

2 Teorie

2.1 Základy VBScriptu

VBScript lze jednoduše napsat v obyčejném poznámkovém bloku. Pro lepší orientaci je možné použít například PSPad, který je volně šiřitelný. Vytvořený skript lze spustit přímo z prostředí Windows tak, že jako příponu souboru zvolíme vbs (příklad: MujScript.vbs) a na tento soubor dvakrát klikneme. Velká a malá písmena v kódu nehrají žádnou roli (na rozdíl od JScriptu). To znamená, že například proměnná ABC je stejná jako abc. Ani mezery se nepočítají, pokud nejsou obsaženy v řetězci mezi uvozovkami (a=5 je stejné jako a = 5). Pro přehled je občas lepší mezery použít. Jak jsem již uvedl, řetězce se píší mezi uvozovky (a="Ahoj" – proměnná a bude obsahovat řetězec Ahoj). Pokud chceme do řetězce napsat uvozovky, napíšeme jednoduše dvě uvozovky hned za sebou (a="Řekni: ""Ahoj""." – proměnná a bude obsahovat tento řetězec: Řekni: "Ahoj".). Pokud chceme v kódu napsat nějaký popisek tak, aby si ho VBScript nevšímal, použijeme symbol ' (apostrof) nebo příkaz REM.

```
'Takto vypadá příklad komentáře
REM Takto taky vypadá příklad komentáře
```

Každý řádek kódu obsahuje jeden příkaz. Samozřejmě, že lze napsat na jeden řádek více příkazů, ale musíme jednotlivé příkazy oddělit dvojtečkou (a=5 : b=6). Pro přehlednost bych to však nedoporučoval. Lze i jeden příkaz rozdělit na více řádků a to použitím podtržítka (_), které se vloží na konec řádky a na dalším bude příkaz pokračovat.

```
If a=0 and _
b=0 Then _
MsgBox "a i b se rovnají nule"
```

Na řádku, který obsahuje podtržítko, nesmí být komentář. Toto bych opět vzhledem k nepřehlednosti nedoporučoval. Pokud je to možné, psát co řádek to jeden příkaz.

2.1.1 Pravidla pro názvy

Pojmenovat proměnnou, proceduru nebo funkci můžeme jakkoliv podle našich představ, ale musíme dodržovat určitá základní pravidla tak, aby VBScript poznal, že jde o jakýsi námi vymyšlený název.

Základní pravidla pro názvy jsou tato:

- Nesmí obsahovat mezery
- Začáteční znak musí být vždy písmeno
- Smí obsahovat podtržítko (_), ale nesmí jím začínat
- Začáteční znak nesmí být Vb, vB, VB nebo vb, protože VBScript má vestavěné konstanty viz kapitola 2.4 MsgBox
- Nesmí být žádný z již rezervovaných výrazů (Error, Array, Exit, Next, Mid, atd.)

Pro přehlednost názvů je dobré použít právě podtržítko místo mezery (moje_procedura) nebo, jak preferuji já, použít velká a malá písmena (mojeProcedura).

2.1.2 Další užitečné příkazy a funkce

Funkce si můžeme vytvářet sami, ale mnoho velmi užitečných funkcí již VBScript obsahuje. Další modifikované funkce můžeme do VBScriptu dodat pomocí knihoven. My si zde ukážeme jen pár příkladů, které VBScript obsahuje v základu.

Funkce pro práci s řetězcem:

- len(řetězec) - spočítá délku řetězce
    - př.: zadáme číslo x=10101 potom len(x) bude 5
- mid(řetězec, x, y) - odkáže na určitou pozici v řetězci

    - "x" označuje pozici v řetězci a "y" označuje počet zobrazovaných znaků od pozice
    - př.: zadáme číslo x=10101 potom mid(x,3,1) bude 1, mid(x,2,2) bude 01 atd.
- StrReverse(řetězec) - převrátí zadaný řetězec
    - př.: zadáme x="Ahoj" potom StrReverse(x) bude "johA"
- LCase(řetězec) – převede všechny velká písmena na malá
    - př.: zadáme x="ABC" potom LCase(x) bude "abc"

Funkce pro práci s datovými typy:

- CDbl(proměnná) – funkce která převádí zadané číslo na typ Double
    - př.: zadáme x=5 (typ Variant), potom příkazem x = CDbl(x) převedeme číslo 5 na typ Double
    - obdobě převádíme na ostatní typy (CBool, CByte, atd.)
- IsNumeric(proměnná) – vrací logickou hodnotu, která určuje zda je proměnnou, která je uvedena v závorce lze vyhodnotit jako číslo
    - obdobné funkce vypadají takto (IsArray, IsDate, atd.)

Matematické funkce:

- Cos(číslo) – vypočítá kosinus zadaného úhlu, úhel se zadává v radiánech
    - zadávané číslo v radiánech musí být typ double
    - podbně se bude počítat i sinus a tangens (Sin, Tan)
- Sgr(číslo) – vrací druhou mocninu zadaného čísla
    - zadávané číslo musí být typu Double
    - př.: a = Sgr(4) vrátí hodnotu 2 do proměnné a

Funkce s datem a časem:

- Time – vrací aktuální systémový čas
    - Můžeme použít funkce jako Hour, Minute, Second, které nám vrátí pouze jednu hodnotu (hodinu, minutu, sekundu)
- Date – vrací aktuální systémový čas
    - Můžeme použít funkce jako Month, Day, Year, které nám vrátí pouze jednu hodnotu (měsíc, den, rok)

2.2 Operátory

Operátory dělíme do 4 základních kategorií:

Aritmetické

+	Sčítání
-	Odčítání
*	Násobení
/	Dělení
Mod	Modulo (zbytek při dělení)

Logické

- Používají se v logických výrazech

And	A současně
Or	Nebo
Xor	Exklusive or
Not	Negace

Porovnávací

<	Menší než
>	Větší než
<=	Menší nebo rovno
>=	Větší nebo rovno
=	Rovná se
<>	Nerovná se

Spojovací

&	Spojení řetězců
+	Spojení řetězců (pokud nejsou oba prvky číslo)

2.3 Proměnné a datové typy

Proměnná je část paměti, ve které budeme uchovávat nějakou hodnotu. Proměnnou vytvoříme jednoduchým příkazem:

```
a=20
```

Tímto jsme do proměnné přiřadili hodnotu 20. Přiřazení se dělá pomocí =. Můžeme vytvářet proměnné, které zatím nemají přiřazenou žádnou hodnotu pomocí příkazu dim.

```
dim a
dim b
dim soucet
```

Vytvořili jsme 3 proměnné s názvy a, b, součet, které zatím nemají přiřazenou žádnou hodnotu. Takovouto deklaraci třech proměnných můžeme udělat i na jednom řádku.

```
dim a,b,soucet
```

Dále do našich třech proměnných můžeme přiřadit nějakou hodnotu

```
a=4
b=5
soucet=9
```

Postupně jsme do našich vytvořených proměnných přiřadili uvedené hodnoty.

Typy proměnných:

Ve VBScriptu je pouze jeden typ proměnné a tím je *variant.* Může obsahovat buď číselnou, nebo řetězcovou hodnotu. Proto při deklaraci není nutné definovat, zda bude proměnná obsahovat číselnou nebo řetězcovou hodnotu a je možné v ní skladovat různé typy hodnot.

Typy hodnot	Popis:

Date	Datum ve formátu den.měsíc.rok.
String	Řetězec.
Object	Objekt.
Error	Číslo chybového hlášení.
Empty	Proměnná bez hodnoty (inicializace). Může obsahovat číslo 0 nebo řetězec nulové délky “”.
Null	Úplně prázdná hodnota. Neobsahuje ani číslo 0.
Boolean	Hodnota typu true/false (ano/ne).
Byte	Celé číslo: 0 - 255.
Integer	Celé číslo: -32768 - 32767.
Currency	Peněžitá hodnota: -922337203685477,5808 - 922337203685477,5807.
Long	Celé číslo: -2147483648 - 2147483647.
Single	Rozsah čísla: -3.402823E38 - -1.401298E-45 nebo 1.401298E-45 - 3.402823E38.
Double	Rozsah čísla: -1.79769313486232E308 - -4.94065645841247E-324 nebo 4.94065645841247E-324 - 1.79769313486232E308.

2.3.1 Pole

Většinou potřebujeme přiřadit do proměnné (skalár) jednu hodnotu, ale někdy je vhodnější přiřadit do jedné proměnné více hodnot a tím nám vzniká pole. Pole je několik hodnot, které mají stejné jméno. Na jednotlivé hodnoty v poli lze odkazovat pomocí čísla, které se nazývá index. Index odpovídá pořadí hodnoty v poli a začíná od nuly, takže první hodnota bude mít index 0, druhá 1 atd.

Máme několik druhů pole:

Fixní pole - Deklaruje se podobně jako proměnná avšak s tím rozdílem, že po názvu proměnné následují závorky, které udávají velikost pole (např. novePole(15) - tímto deklarujeme pole o rozměru 16 prvků)

Dimenzní pole - V poli nemusíme mít jen proměnné, ale jako proměnná může být další pole. Pole, které obsahuje jen proměnné, má jednu dimenzi. Pole o 2 dimenzích si lze představit jako tabulku (toto si ukážeme na příkladu). Pole o 3 dimenzích si lze představit jako souřadnice krychle.

Zde máme příklad deklarace pole o 2 dimenzích - je to tabulka, která má 4 sloupce a 3 řádky:

```
Dim pole(3,2)
pole(0,0) = 1
pole(1,0) = 2
pole(2,0) = 3
pole(3,0) = 4
pole(0,1) = 1
pole(1,1) = 2
pole(2,1) = 3
pole(3,1) = 4
pole(0,2) = 1
pole(1,2) = 2
pole(2,2) = 3
pole(3,2) = 4
```

Takto deklarované pole s takovými přiřazenými hodnotami si můžeme představit takto:

```
1234
1234
1234
1234
```

2.4 MsgBox

MsgBox je funkce, která zobrazí okno, ve kterém bude zpráva nebo nějaká otázka. MsgBox má 4 vlastnosti, které můžeme měnit.

Má tuto syntaxi:

MsgBox (prompt[, buttons][, title][, helpfile, context])

- prompt - zobrazovaný text (jediný povinný argument)
- buttons – určuje, jaká tlačítka se zobrazí
- title - titulek zobrazující se nahoře v okně
- helpfile - řetězec, který identifikuje soubor s nápovědou
- context - číslo, které autor nápovědy dal rubrice nápovědy, helpfile a context se musí vždy zadat současně

Tato funkce má několik možností ohledně toho, jak se okno zobrazí a s jakými tlačítky.

Název konstanty	**Hodnota konstanty**	**Okno, které MsgBox zobrazí**
vbCritical	16	Kritická zpráva
vbQuestion	32	Otazník
vbExclamation	48	Vykřičník
vbInformation	64	Informace
vbOKOnly	0	Pouze tlačítko OK
vbOKCancel	1	Tlačítka OK a Storno
vbAbortRetryIgnore	2	Tlačítka Přerušit, Opakovat, a Ignorovat
vbYesNoCancel	3	Tlačítka Ano, Ne, a Storno
vbYesNo	4	Tlačítka Ano a Ne
vbRetryCancel	5	Tlačítka Opakovat a Storno

Typ okna a tlačítka můžeme kombinovat tehdy, když mezi hodnoty konstanty vložíme +.

```
MsgBox "Toto okno zobrazí Otazník a tlačítka Ano a NE", 32+4
```

MsgBox má také pro každé tlačítko hodnotu, s kterou můžeme dále pracovat.

Tlačítko	Hodnota	Název tlačítka
Ok	1	vbOK
Storno	2	vbCancel
Přerušit	3	vbAbort
Opakovat	4	vbRetry
Ignorovat	5	vbIgnore
Ano	6	vbYes
Ne	7	vbNo

Ukážeme si jednoduchý příklad přímo v kódu.

```
anone=MsgBox("Ano nebo Ne?",64+4)
if anone=6 then
msgbox "Zvolili jste ANO"
elseif anone=7 then
msgbox "Zvolili jste NE"
end if
```

Při spuštění tohoto skriptu se nám zobrazí MsgBox s informační ikonou a tlačítky Ano a Ne. Podle akce uživatele se do proměnné anone uloží buď hodnota 6 (pokud uživatel klikne na tlačítko Ano) nebo hodnota 7 (pokud uživatel klikne na tlačítko Ne). Pomocí jednoduché podmínky If (porovná hodnotu proměnné anone) se nám zobrazí další MsgBox, ve kterém se ukáže, na jaké tlačítko uživatel kliknul.

2.5 InputBox

InputBox je funkce, která zobrazí okno, do kterého uživatel zadá hodnotu a poté klikne na tlačítko OK.

Má tuto syntaxi:

InputBox (prompt[, title][, default][, xpos][, ypos][, helpfile, context])

- prompt – text, který chceme zobrazit (povinný údaj)
- default - text, který je uveden v zadávacím okně jako příklad
- title - titulek zobrazený nahoře v okně
- xpos - pozice InputBoxu od levého okraje monitoru (udává se v twipech)
- ypos - pozice InputBoxu od horního okraje monitoru (udává se v twipech)
- helpfile - řetězec, který identifikuje soubor s nápovědou
- context - číslo, které autor nápovědy dal rubrice nápovědy. Helpfile a context se musí vždy zadat současně

Příklad InputBoxu:

```
vstup=InputBox(“Zadejte řetězec, který chcete vypsat:", "Příklad", "sem vložte řetězec")
MsgBox ("Napsali jste: " & vstup)
```

Skript, který při spuštění zobrazí InputBox, kam uživatel zadá text (ten se uloží do proměnné vstup), který bude zobrazen na dalším MsgBoxu.

2.6 Podmínky

Příkazy v skriptu jsou většinou prováděny za sebou v pořadí, v jakém jsou napsány. Vždy to tak však být nemusí. Podmíněné příkazy jsou prováděny pouze tehdy, když je splněná nějaká podmínka. Podmínka je většinou logický výraz, který porovnává nějakou hodnotu. Máme 2 typy podmínek – If a Case.

2.6.1 Podmínka If

Příkaz má 2 syntaxe:

- první, jestliže máme jeden příkaz, který se splní pod podmínkou
    - If podmínka Then kód pod podmínkou
- druhý, jestliže máme více příkazů, které se splní pod podmínkou
    - If podmínka Then
        - kód pro podmínku
    - End If

Pokud chceme provést příkazy právě tehdy, když podmínka platit nebude, přidáme příkaz Else. Pokud máme několik podmínek, další přidáváme za příkaz ElseIf.

Ukážeme si příklad přímo v kódu.

```
rychlost = InputBox("Zadejte rychlost auta v km/h:")
If rychlost <= 0 Then
MsgBox "Zadali jste zápornou hodnotu nebo 0"
ElseIf rychlost <= 50 Then
MsgBox "Pomalá rychlost"
ElseIf rychlost <= 100 Then
MsgBox "Střední rychlost"
```

```
ElseIf rychlost <= 180 Then
MsgBox "Velmi vysoká rychlost"
ElseIf rychlost >= 180 Then
MsgBox "Smrtelná rychlost"
End If
```

Skript, který po spuštění vyzve uživatele zadat hodnotu rychlosti auta. Ta se uloží do proměnné rychlost. Dále se pomocí příkazu If porovnává zadaná hodnota a podle toho se do MsgBoxu vypíše, jakou rychlost uživatel zadal.

2.6.2 Podmínka Case

Select Case porovnává zadaný výraz s výrazy v příkazu Case. Příkazy budou splněny pouze tehdy, když se zadaný výraz bude plně shodovat s nějakým výrazem v celém příkazu Case. Příkaz Select Case vyhodnotí všechny výrazy pouze jednou, proto je rychlejší.

Ukážeme si příklad přímo v kódu:

```
c=inputbox("Zadejte, co chcete udělat s čísly 8 a 2:"
&VbCrLf& "Vložte operátor:" &VbCrLf& "+ pro součet" &VbCrLf&
"- pro rozdíl" &VbCrLf& "/ pro podíl" &VbCrLf& "* pro součin"
)
Select Case c
Case "+"    MsgBox("Výsledek: "& 8 + 2)
Case "-"    MsgBox("Výsledek: "& 8 - 2)
Case "/"    MsgBox("Výsledek: "& 8 / 2)
Case "*"    MsgBox("Výsledek: "& 8 * 2)
Case Else    MsgBox "Zadaný vstup není operátor, skript se
ukončí!" & jmeno,48,"Allert"
End Select
```

Tento skript nám po spuštění zobrazí InputBox, do kterého uživatel vloží jeden ze 4 vypsaných operátorů (+, -, /, *). Operátor se uloží do proměnné c a dále se porovnává s jednotlivými řetězci v příkazu Case a podle toho se provede určitá akce (v tomto případě

se pracuje s čísly 8 a 2, které se podle zadaného operátoru buď sčítá, odečítá, dělí nebo násobí). Příkaz Case Else se provede právě tehdy, když zadaný řetězec nebude odpovídat ani jednomu řetězci v Case.

2.7 Cykly

Často je zapotřebí provádět nějaké akce až do té doby, než se dosáhne nějakého cíle. K tomu nám slouží cykly. Cykly obsahují nějaký podmíněný výraz, podle kterého několikrát proběhne daný kus kódu a vyhodnocuje se do té doby, než bude podmínka false a poté se cyklus ukončí. Některé cykly fungují opačně a opakují se dokud nezačne určitá podmínka platit.

2.7.1 Cyklus Do---Loop

'"Do" v překladu dělej a "Loop" v překladu opakuj. Do značí začátek cyklu a Loop konec (místo, ze kterého se vracíme zpět na začátek cyklu).

Příkaz má 2 syntaxe:

- While - opakuje cyklus, dokud je podmínka pravdivá
- Until - opakuje cyklus do té doby, než začne podmínka platit

Obě verze mají 2 další možnosti vyhodnocení podmínky. Buď na začátku, nebo na konci cyklu.

Syntaxe s podmínkou na začátku cyklu:

- Do [{While | Until} podmínka]
    - [příkazy]
    - [Exit Do]
    - [příkazy]
- Loop

Syntaxe s podmínkou na konci cyklu:

- Do
    - [příkazy]
    - [Exit Do]

    - [příkazy]
- Loop [{While | Until} podmínka]

Ukážeme si jednoduchý příklad s podmínkou Until na konci.

```
Do
odpoved = MsgBox("Opakovat skript?", 32+4)
Loop Until odpoved=7
```

Tento jednoduchý skript nám opakuje MsgBox s tlačítky Ano a Ne. Podle akce uživatele se do proměnné odpověď uloží buď hodnota 6 (Ano) nebo 7 (Ne). A cyklus se opakuje do té doby, než uživatel klikne na tlačítko ne a do proměnné uloží hodnotu 7.

Příklad, kde použijeme příkaz While:

```
Dim pocitadlo
Do
odpoved = MsgBox("Opakovat smyčku?", 32+4)
pocitadlo = pocitadlo + 1
Loop While odpoved=6
MsgBox "Smyčka proběhla " &pocitadlo& "x krát"
```

Kdybychom použili podmínku na začátku, smyčka by neproběhla ani jednou, protože na začátku nemáme žádnou proměnnou "odpoveď", která má hodnotu 6. Takto nám MsgBox vyhodnotí proměnnou "odpoveď" podle kliknutí uživatele a poté teprve porovnává podmínku. To znamená, že smyčka proběhne alespoň jednou.

Příklad, kde použijeme příkaz Until:

```
Dim pocitadlo
Do Until odpoved=7
odpoved = MsgBox("Opakovat smyčku?", 32+4)
pocitadlo = pocitadlo + 1
Loop
MsgBox "Smyčka proběhla " &pocitadlo& "x krát"
```

V tomto případě můžeme použít podmínku na začátku i na konci. Cyklus proběhne tolikrát, dokud uživatel neklikne na tlačítko NE. Cyklus proběhne vždy alespoň jednou.

Příklad použití příkazu Exit:
Ve všech variantách můžeme použít příkaz Exit, který způsobí skok na příkaz, který následuje po Loop.

```
Dim pocitadlo
Do Until odpoved=7
odpoved = MsgBox("Opakovat smyčku?" &VbCrLf& "Pokud smyčka proběhne 10x, sama se ukončí", 32+4)
pocitadlo = pocitadlo + 1
If pocitadlo = 10 Then Exit Do
Loop
If pocitadlo = 10 Then
MsgBox "Smyčka proběhla " &pocitadlo& "x krát a sama se ukončila"
Else
MsgBox "Smyčka proběhla " &pocitadlo& "x krát"
End If
```

Tento skript stejně jako předešlé příklady opakuje MsgBox, ale s tím rozdílem, že pokud smyčka proběhne 10x, přeskočí se rovnou na příkazy, které následují po Loop. Proměnná pocitadlo nám počítá, kolikrát smyčka proběhla.

2.7.2 Cyklus While---Wend

Pokud se vám nelíbil příkaz Do---Loop, tak pro začátek můžete použít jen příkaz While---Wend, který by měl stačit ve většině případů. Je jednodušší, srozumitelnější a odpovídá příkazu Do While---Loop, pouze s tím rozdílem, že nemá příkaz Exit.

Jednoduchá ukázka použití.

```
Dim pocitadlo
While pocitadlo < 10
pocitadlo = pocitadlo + 1
```

```
MsgBox "Smyčka se bude opakovat do té doby, než se napočíta
do 10" &VbCrLf& "Počítání: " & pocitadlo
Wend
```

Skript opakuje MsgBox do té doby, dokud je hodnota v proměnné pocitadlo menší než 10.

2.7.3 Cyklus For---Next

Další typ smyčky, který provede určité příkazy. Počet opakování ovlivňují klíčová slova To a Step

Syntaxe:

- For pocitadlo = start To konec [Step krok]
  - [příkazy]
  - [Exit For]
  - [příkazy]
- Next

- pocitadlo - číselná proměnná, která ovládá cyklus
- start - počáteční hodnota pocitadla
- konec - konečná hodnota pocitadla
- krok - o kolik se změní pocitadlo, když proběhne cyklus (pokud není zadáno, je nastavena hodnota 1)

Stejně jako cyklus Do---Loop má i tento příkaz Exit, který cyklus ukončí před dosažením konečné hodnoty.

Ukážeme si konkrétní příklad.

```
Dim pocitadlo,soucet
cislo = InputBox("Cyklus proběhne od 0 do 40. Zadejte hodnotu
kroku, po kterém se bude skákat.")
If cislo=0 Then
```

```
MsgBox "Zadali jste krok 0 - udělali byste nekonečnou smyčku
:)"
wscript.quit
End If
For pocitadlo = 0 To 40 Step cislo
soucet = soucet + pocitadlo
MsgBox pocitadlo & ". krok" &VbCrLf& "Součet hodnot v
pocitadle: " & soucet
Next
```

Tento skript nám demonstruje cyklus For, který proběhne od 0 do 40, ale s tím rozdílem, že uživatel zadá krok, po kterém bude cyklus skákat. Skript je opatřen podmínkou, která zamezuje zadání kroku 0 (cyklus by se opakoval do nekonečna). Proměnná krok nám ukazuje, kde se cyklus zrovna nachází v zadaném rozmezí 0 – 40 a proměnná soucet nám provádí součet aktuálních hodnot, které se zrovna nacházejí v daném cyklu.

2.8 Procedury

Procedury slouží k opakovanému spuštění sekvence příkazů, bez nutnosti je znovu kódovat. Procedura může mít vstupní parametr, ale neumí vrátit hodnotu.

Procedury mají tuto syntaxi:

- Sub nazevProcedury()
    - ---kód---
- End Sub

Proceduru dále voláme tímto způsobem:

- Call nazevProcedury(argument) - 1. způsob
- nazevProcedury argument - 2. způsob

Zde si ukážeme příklad použití procedury.

Tělo procedury může vypadat takto:

```
Sub pokus(a)
MsgBox "Příklad:" &VbCrLf& "Tvé oblíbené číslo je: "&a
End Sub
```

Zde proceduru zavoláme s parametrem 444

```
Call pokus(444)
```

2.9 Funkce

Stejně tak jako procedury spouští funkce sekvenci příkazů jedním zavoláním. Avšak narozdíl od procedur mají funkce většinou za úkol dělat nějaké výpočty. Mají tedy vstup a výstup. Vstupní hodnoty se jmenují argumenty nebo také parametry. Výstupem je hodnota, kterou funkce vrací programu. Funkce nemusí mít vstup a výstup, ale poté fungují stejně jako procedury.

Mají tuto syntaxi:

- Function nazevFunkce()
    - ---kód funkce---
- End Function

Ukázka funkce, která nám vypočítá průměr dvou čísel:

```
Function prumer(a, b)
dim c
c = (a + b) / 2
prumer=c
End Function

c = InputBox("Zadej první číslo:")
d = InputBox("Zadej druhé číslo:")
c = CDbl(c) 'převádí zadaný řetězec na typ Double
```

```
d = CDbl(d)
MsgBox "Vysledek: " & prumer(c,d)
```

V tomto skriptu vidíme kód funkce, která má za úkol spočítat průměr ze dvou zadaných čísel. Uživatel tedy pomocí dvou InputBoxů zadá dvě čísla, které se uloží do proměnných c, d. Vidíme, že proměnné nemusí být stejné jako v deklaraci funkce. Následně si převedeme zadané hodnoty na typ double, abychom s nimi mohli počítat. V posledním řádku kódu vypíšeme MsgBox a zavoláme funkci prumer s parametry c,d. Funkce nám vrátí výsledek, který je průměrem dvou zadaných čísel.

3 Neřešené úlohy

3.1 Úloha 1 – dělitelnost čísel

Vytvořte skript pro zjištění, zda je jedno číslo dělitelné druhým. Nápovědu naleznete v kapitole 2.2 – aritmetické operátory.

3.2 Úloha 2 – obvod a obsah

Napište skript pro počítání obvodu a obsahu geometrických tvarů (obdélník, kruh, trojúhelník). Pro vybrání geometrického tvaru, s kterým se bude pracovat, použijte podmínku Case a pro případné převedení na jiný typ proměnné naleznete dané funkce v kapitole 2.1.2 Další užitečné příkazy a funkce.

3.3 Úloha 3 – faktoriál

Vytvořte skript, který ze zadaného čísla spočítá faktoriál.

Malá nápověda - použijte cyklus For---Next

3.4 Úloha 4 – převod z dvojkové do desítkové soustavy

Vytvořte skript, který bude převádět číslo z dvojkové do desítkové soustavy. Ošetřete proti špatnému zadání čísla.

Nápověda:

- Já osobně jsem ve skriptu použil následující funkce – popis jednotlivých funkcí viz kapitola 2.1.3 Další užitečné příkazy a funkce

- Použité funkce: len(proměnná), mid(promenná, x, y), StrReverse(proměnná)
- příkaz wscript.quit - předčasně ukončí skript
- Každá implementace může být jiná, takže je možné, že tyto pomůcky potřebovat nebudete

4 Možné řešení k úlohám

Zde jsou má řešení zadaných úloh, které se samozřejmě od těch vašich mohou lišit avšak fungovat stejně. Tato řešení berte jen jako nápovědu v případě, pokud se někde zaseknete a nebudete si jistí, jak pokračovat dál.

4.1 Úloha 1 – dělitelnost čísel

```
dim a,b
a = inputbox("Zadejte číslo, u kterého chcete zjistit
dělitelnost")
b = inputbox("Zadejte čím chcete číslo dělit")
If a mod b = 0 Then
MsgBox "Číslo " & a & " je dělitelné číslem " & b & "."
Else
MsgBox "Číslo " & a & " není dělitelné číslem " & b & "."
End If
```

4.2 Úloha 2 – obvod a obsah

```
dim obvod,obsah
tvar = inputbox("Zadejte jaký tvar chcete počítat (napište:
obdelnik, kruh nebo trojuhelnik)")
Select case tvar
Case "obdelnik"
     a = inputbox("Zadejte první stranu obdélníku")
     b = inputbox("Zadejte druhou stranu obdélníku")
```

```
     a = CDbl(a)
     b = CDbl(b)
     obvod = 2*(a+b)
     obsah = a*b
     MsgBox "Obvod = " & obvod &VbCrLf& "Obsah = " & obsah
Case "kruh"
     dim pi
     pi = 3.14
     r = inputbox("Zadejte poloměr kruhu")
     obvod = pi*2*r
     obsah = pi*r*r
     MsgBox "Obvod = " & obvod &VbCrLf& "Obsah = " & obsah
Case "trojuhelnik"
     oo = inputbox("Zadejte co chcete počítat (napište: obvod
nebo obsah")
     Select case oo
     Case "obsah"
     a = inputbox("Zadejte stranu trojúhelníku")
     v = inputbox("Zadejte výšku k zadané straně")
     obsah = (a*v)/2
     MsgBox "Obsah = " & obsah
     Case "obvod"
     a = inputbox("Zadejte první stranu trojúhelníku")
     b = inputbox("Zadejte druhou stranu trojúhelníku")
     c = inputbox("Zadejte třetí stranu trojúhelníku")
     a = CDbl(a)
     b = CDbl(b)
     c = CDbl(c)
     obvod = a+b+c
     MsgBox "Obvod = " & obvod
     Case Else MsgBox "Zadaný vstup neodpovídá žádnému tvaru"
     End Select
Case Else MsgBox "Zadaný vstup neodpovídá žádnému tvaru"
End Select
```

4.3 Úloha 3 – faktoriál

```
Dim pocitadlo,faktorial
faktorial = 1
cislo = InputBox("Zadejte číslo, ze kterého chcete vypočítat
faktoriál")
if not isnumeric(cislo) then
MsgBox "Zadaný vstup není číslo, skript se ukončí."
wscript.quit
End If
For pocitadlo = 2 To cislo
faktorial = faktorial * pocitadlo
Next
MsgBox "Faktoriál čísla " & cislo & " je: " & faktoriál
```

4.4 Úloha 4 – převod z dvojkové do desítkové soustavy

```
cislodvoj = inputbox("Zadejte číslo ve dvojkové soustavě")
If not isnumeric(cislodvoj) Then MsgBox "Zadané číslo není ve
formátu dvojkové soustavy",16 & wscript.quit

For i=1 to len(cislodvoj)
If mid(cislodvoj, i, 1)>1 or mid(cislodvoj, i, 1)<0 Then
MsgBox "Zadané číslo není ve formátu dvojkové soustavy",16
wscript.quit
End If
Next



dim cislodes,x,prevraceny
prevraceny = StrReverse(cislodvoj)
cislodes = 0
For j=1 to len(cislodvoj)
If mid(prevraceny, j, 1) = 1 Then
```

```
x = j - 1
cislodes = cislodes + 2^x
End If
Next

MsgBox "Zadané číslo: (2)" & cislodvoj &VbCrLf& "Převedené
číslo: (10)" & cislodes,,"Výsledek"
```