

# Multimediální průvodce základů práce ve VBA pro MS Excel

Radim Halfar

---

Bakalářská práce  
2023



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Radim Halfar**  
Osobní číslo: **A20363**  
Studijní program: **B0613A140020 Softwarové inženýrství**  
Forma studia: **Prezenční**  
Téma práce: **Multimediální průvodce základů práce ve VBA pro MS Excel**  
Téma práce anglicky: **Multimedia Guide to VBA Basics for MS Excel**

## Zásady pro vypracování

1. Provedte literární průzkum z oblasti VBA pro MS Excel a software použitého při tvorbě multimediálního průvodce.
2. Vypracujte sadu 13 výukových materiálů pro výuku základů práce ve VBA pro MS Excel.
3. Vytvořte multimediálního průvodce na základě výukových materiálů vytvořených dle zásady č. 2.
4. V praktické části práce popište vytvořené podklady ve VBA pro MS Excel a multimediálního průvodce.
5. Vytvořte vzorová zadání a vypracování ukázkových cvičení.

Forma zpracování bakalářské práce: **tištěná/elektronická**

**Seznam doporučené literatury:**

1. LAURENČÍK, Marek a Michal BUREŠ. Programování v Excelu 2019: záznam, úprava a programování maker. Praha: Grada Publishing, 2021. Průvodce (Grada). ISBN 978-80-271-3145-7.
2. KRÁL, Martin. Excel VBA: výukový kurz. Brno: Computer Press, 2010. ISBN 978-80-251-2358-4.
3. MUNEEER, Tariq a IVANOVA, Stoyanka. Excel-VBA: From Solving Mathematical Puzzles to Analysing Complex Engineering Problems. Cham: Springer Nature Switzerland, 2022. ISBN 978-3030940843.
4. JELEN, Bill a SYRSTAD, Tracy. Microsoft Excel 2019 VBA and Macros. Microsoft Press, 2018. ISBN 978-1509306114.
5. WILLIAMS, David A. Excel Programming: The Ultimate Collection to Learn Excel VBA & Excel Macros Step by Step. Independently Published, 2019. ISBN 978-1676827481.

Vedoucí bakalářské práce: **Ing. Karel Perůtka, Ph.D.**  
Ústav řízení procesů

Datum zadání bakalářské práce: **28. července 2023**  
Termín odevzdání bakalářské práce: **25. srpna 2023**



**doc. Ing. Jiří Vojtěšek, Ph.D. v.r.**  
děkan

**prof. Mgr. Roman Jašek, Ph.D., DBA v.r.**  
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

### **Prohlašuji, že**

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

### **Prohlašuji,**

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 24. 8. 2023

.....  
podpis studenta

## **ABSTRAKT**

Tato bakalářská práce se zaměřuje na multimedialního průvodce jazykem VBA a jeho implementaci v Microsoft Excel. Byla vytvořena multimedialní aplikace průvodce pro Windows. Průvodce probírá témata jako jsou procedury, proměnné, výpočty, kalendáře, větvení, cykly, stylování, výběry, práci se sešitem a další. V rámci práce bylo vytvořeno 13 pomocných studijních materiálů, cvičení a ukázek pro základy práce s VBA, které by měli sloužit jako pomůcka při výuce předmětu Kancelářský software 2.

Klíčová slova: VBA, výukový materiál, návod, Excel, Visual Basic, makra, Office Scripts, základy, aplikace

## **ABSTRACT**

This bachelor thesis focuses on a multimedia VBA wizard and its implementation in Microsoft Excel. A multimedia wizard application was created for Windows. The wizard discusses topics such as procedures, variables, calculations, calendars, branching, loops, styling, selections, working with the workbook and more. As part of the work, 13 post-work study materials, exercises and demonstrations for the basics of working with VBA have been created to serve as an aid in teaching the Office Software 2 course.

Keywords: VBA, tutorial, manual, Excel, Visual Basic, macros, Office Scripts, basics, application

Rád bych poděkoval mému vedoucímu bakalářské práce panu Ing. Karlovi Perůtkovi, Ph.D. za trpělivost, odborné vedení, užitečné rady a poskytnuté informace.

Poděkování patří také mé rodině, přítelkyni a přátelům za trpělivost a podporu po dobu studia, a především při psaní této práce.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

# OBSAH

<b>ÚVOD</b> .....	<b>8</b>
<b>I TEORETICKÁ ČÁST</b> .....	<b>9</b>
<b>1 VISUAL BASIC PRO APLIKACE</b> .....	<b>10</b>
1.1    PROSTŘEDÍ .....	10
1.1.1    Microsoft 365 .....	11
1.1.2    Microsoft Excel .....	11
1.1.3    Visual Basic Editor .....	12
1.1.4    Ostatní aplikace s implementací VBA .....	12
1.2    OSTATNÍ VERZE VISUAL BASIC .....	13
1.2.1    Classic Visual Basic .....	13
1.2.2    .NET Visual Basic.....	14
1.3    VÝVOJ A BUDOUCNOST VBA .....	14
1.3.1    Verze .....	15
1.3.2    Office Scripts .....	15
1.4    ZÁKLADNÍ KONCEPTY JAZYKA .....	16
1.4.1    Struktura projektu.....	16
1.4.2    Procedury .....	17
1.4.3    Proměnné.....	19
1.4.4    Datové typy .....	19
1.4.5    Operátory.....	20
1.4.6    Podmíněné příkazy.....	23
1.4.7    Cykly a rekurze .....	24
1.4.8    Třídy a objekty .....	26
<b>2 NÁSTROJE K TVORBĚ APLIKACE</b> .....	<b>27</b>
2.1    FRAMEWORK TAURI.....	27
2.1.1    Framework SvelteKit .....	27
2.1.2    Rust .....	28
2.1.3    TypeScript .....	28
2.1.4    Tailwind CSS .....	28
2.1.5    Knihovna Skeleton .....	29
2.1.6    Node.js .....	29
2.1.7    Markdown .....	29
2.2    VIDEO .....	30
2.2.1    OBS Studio.....	30
2.2.2    Audacity .....	30
2.2.3    Adobe Premiere Pro .....	30
<b>II PRAKTICKÁ ČÁST</b> .....	<b>31</b>
<b>3 APLIKACE MULTIMEDIÁLNÍHO PRŮVODCE</b> .....	<b>32</b>

3.1	TECHNOLOGIE APLIKACE.....	32
3.2	ČTENÍ LEKCÍ V SOUBORECH MARKDOWN.....	33
3.3	RESPONZIVNÍ ZOBRAZENÍ.....	35
3.4	SESTAVENÍ APLIKACE.....	35
3.5	NAHRÁVÁNÍ A STRÍH VIDEÍ.....	35
<b>4</b>	<b>STUDIJNÍ MATERIÁLY .....</b>	<b>37</b>
4.1	LEKCE 0 – ÚVOD DO VBA .....	38
4.2	LEKCE 1 – PROCEDURY .....	39
4.3	LEKCE 2 – PROMĚNNÉ, DATOVÉ TYPY A VÝPOČTY .....	40
4.4	LEKCE 3 – VĚTVENÍ A PODMÍNĚNÉ PŘÍKAZY .....	41
4.5	LEKCE 4 – KALENDÁŘNÍ A ČASOVÉ HODNOTY .....	42
4.6	LEKCE 5 – OPAKOVÁNÍ, CYKLY A POLE.....	43
4.7	LEKCE 6 – PRÁCE S OBLASTMI VÝBĚRU.....	44
4.8	LEKCE 7 – STYLOVÁNÍ TEXTU A BUNĚK .....	45
4.9	LEKCE 8 – PRÁCE SE SEŠITEM.....	46
4.10	LEKCE 9 – PRÁCE S GRAFY .....	47
4.11	LEKCE 10 – TŘÍDY A OBJEKTY.....	48
4.12	LEKCE 12 – PRÁCE S UŽIVATELSKÝMI DIALOGY A FORMULÁŘI.....	49
4.13	LEKCE 13 – PRÁCE S TEXTOVÝMI SOUBORY .....	50
	<b>ZÁVĚR .....</b>	<b>51</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>56</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>57</b>
	<b>SEZNAM TABULEK.....</b>	<b>58</b>
	<b>SEZNAM PŘÍLOH.....</b>	<b>59</b>

## ÚVOD

Prakticky každý z nás, ať už ve školním, pracovním nebo i domácím prostředí, se setkal s kancelářskými programy balíku Office. V dnešní moderní době jsou tyto aplikace nezbytné pro všechny možné profesní obory. Při používání těchto aplikací dochází velmi často k repetitivním úkolům, které by se s pomocí jazyka Visual Basic pro Aplikace daly zautomatizovat.

Cílem této bakalářské práce je poskytnout podporu pro získání základních znalostí a dovedností pro práci s tímto jazykem. Multimediální průvodce a materiály jsou primárně určeny pro studenty předmětu Kancelářský Software II na Fakultě aplikované informatiky ve Zlíně. Nicméně vytvořené lekce by měly být pochopitelné i pro ostatní, kteří se s programováním (i ve VBA) teprve seznamují.

Snahou této bakalářské práce je rozšíření procesu učení se s jazykem Visual Basic pro Aplikace skrz interaktivního multimediálního průvodce.

Teoretická část je složena ze dvou hlavních kapitol. První kapitola, s názvem Visual Basic pro Aplikace, je věnována historii jazyka, prostředí, ve kterém běží, jeho ostatním verzím z rodiny Visual Basic, na které mohou uživatelé narazit, a možném budoucím rozvoji. Kapitola zároveň rozebírá základní koncepty, které jazyk využívá.

Druhá kapitola, s názvem Nástroje k tvorbě aplikace se věnuje použitým softwarovým nástrojům pro tvorbu aplikace multimediálního průvodce. Rozebírá použité aplikační frameworky a nástroje pro vytvoření videí.

Praktická část nejprve popisuje některé složitější procesy tvorby aplikace v kapitole s názvem Aplikace multimediálního průvodce. V této části je popsán i proces stříhání a nahrávání videí pro průvodce. Druhá kapitola s názvem Studijní materiály popisuje jednotlivě vytvořené lekce a úkoly.

## **I. TEORETICKÁ ČÁST**

## 1 VISUAL BASIC PRO APLIKACE

Visual Basic pro Aplikace (VBA) je programovací jazyk, který umožňuje rozšířit funkce aplikací v balíku kancelářského softwaru Microsoft 365 (dříve označován jako balík Microsoft Office nebo Office 365). Mezi tyto aplikace patří například Excel, Access, Power-Point, Word a další. Jedná se o objektově orientovaný programovací jazyk, který je zároveň v kódu řízen různými událostmi. [1]

Microsoft s pomocí VBA umožňuje uživatelům zautomatizovat jakoukoliv operaci, kterou lze provést pomocí myši nebo klávesnice. Cílem je usnadnit práci, kterou uživatelé v těchto programech opakují neustále dokola.

VBA dovoluje i propojení mezi jednotlivými kancelářskými aplikacemi, například lze převést kontakty z aplikace Outlook přímo do tabulky v aplikaci Excel. Mezi další cíle jazyka patří jednoduše pochopitelná syntaxe pro vývojáře (vycházející z anglického jazyka) nebo udržení dlouhodobé podpory pro aplikace vytvořené v tomto jazyce.

Mezi další rozšířené funkce programovacího jazyka VBA určitě patří podpora Windows API a možnost připojení dynamicky linkovaných knihoven DLL. Pomocí volání funkcí z těchto součástí může vývojář přistupovat k operacím, které by jinak v jazyce nebyly dostupné.

### 1.1 Prostředí

Tradiční programovací jazyky, jako například jazyk C, překládají kód vytvořený vývojářem do kompilovaných, samostatně spustitelných souborů, které jsou pro běh programů mnohem rychlejší. Visual Basic (a Visual Basic pro Aplikace) spojuje prvky kompilovaných a interpretovaných jazyků. Interpretace jazyka běží na pozadí již při psaní kódu, z tohoto důvodu je možné otestovat kód vůči chybám ještě před jeho spuštěním. [2]

Během vývoje kódu VBA dochází zároveň k jeho kompilaci do kombinace kompilovaného a interpretovaného pseudokódu (někdy označovaného jako p-code). Tento pseudokód umožňuje kódu VBA běžet rychleji než pouze interpretovaný kód. Ke kompilaci kódu dochází při prvním běhu programu nebo při zvolení možnosti kompilovat kód v editoru. [2]

Programovací jazyk VBA nelze použít k vytvoření samostatných aplikací, které by běžely na operačním systému. Vždy je nutné mít hostitelskou aplikaci, která s pomocí *Visual Basic Runtime Library*, umožňuje spustit napsaný kód. Ke spuštění kódu dochází ve virtuálním stroji hostující aplikaci.

### 1.1.1 Microsoft 365

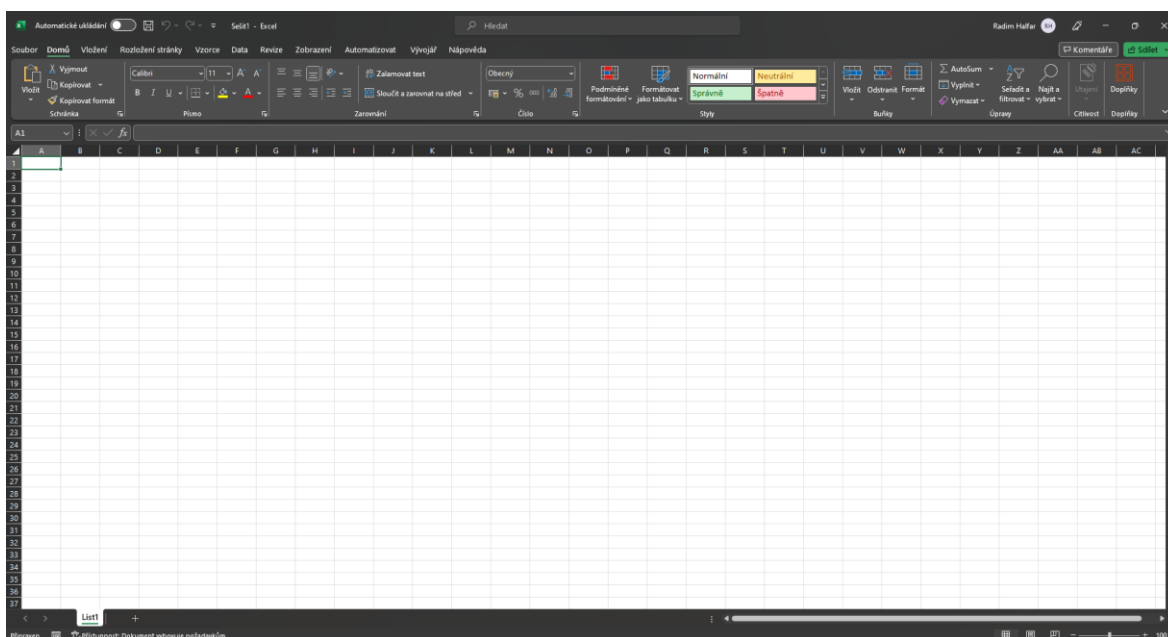
Primárním prostředím pro běh jazyka VBA jsou aplikace kancelářského balíku Microsoft 365: Word, PowerPoint, Access, Outlook a zejména Excel. Každá z těchto aplikací integruje jazyk VBA vlastním způsobem pomocí svého OLE Automation rozhraní a objektového modelu COM.

Balík primárně poskytuje řadu aplikací a služeb, které jsou nezbytné pro produktivitu a spolupráci v moderním pracovním prostředí. Aplikace nabízí uživatelům možnost vytvářet a upravovat dokumenty, tabulky či prezentace, nebo například posílat e-maily.

### 1.1.2 Microsoft Excel

Microsoft Excel je tabulkový program, jehož hlavním účelem je usnadnit uživatelům práci s tvorbou tabulek a grafů, manipulací číselných dat nebo s datovou analýzou. Program obsahuje širokou škálu funkcí, které umožňují provádět různé výpočty a organizovat data.

Data programu jsou ukládána do sešitů. Jedná se o soubory aplikace, končící koncovkou .xlsx, nebo .xlsm, pokud jsou v sešitu použita makra. Sešity se skládají z minimálně jednoho listu (Obrázek 1), i když jich jeden sešit může obsahovat více. Jednotlivé sešity jsou tvořeny buňkami, které mají souřadnice – každá buňka má přiřazené číslo řádku a písmeno sloupce, ve kterém se nachází.

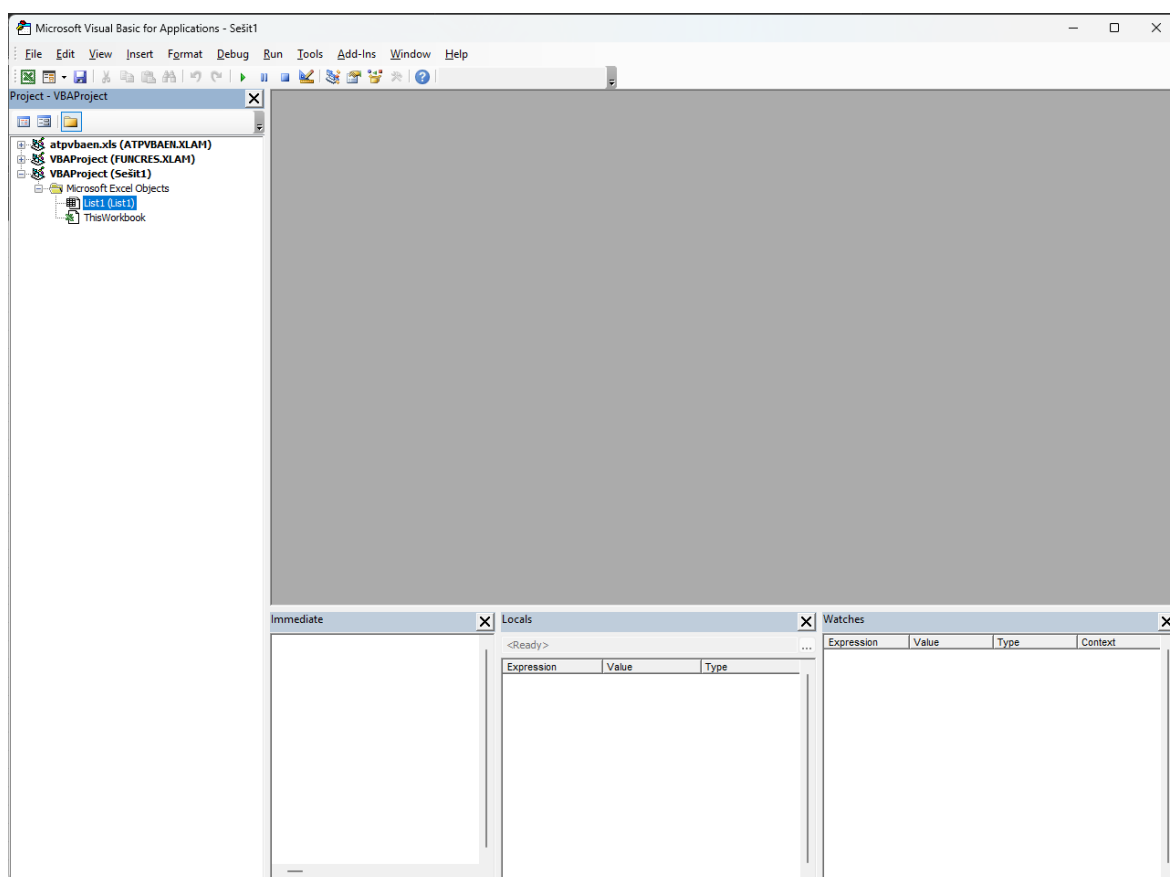


Obrázek 1 – Prázdný list Microsoft Excel

### 1.1.3 Visual Basic Editor

Visual Basic Editor (VBE) je integrované vývojové prostředí pro tvorbu maker. To znamená, že uživatelům v tomto prostředí umožňuje vytvářet, upravovat a ladit kód VBA. Editor zahrnuje barevné zvýrazňování jazyka, lehké našeptávání kódu, editor pro uživatelské rozhraní a možnost práce se strukturou projektu.

Základní rozložení programu (Obrázek 2) obsahuje ve vrchní panelu ovládací lištu a menu, v levé části projektový strom se soubory, v hlavní šedé části okna s otevřeným kódem souborů a ve spodní části okna *Immediate*, *Locals* a *Watches* pro ladění kódu.



Obrázek 2 – Základní rozložení Visual Basic Editoru

### 1.1.4 Ostatní aplikace s implementací VBA

Přestože je VBA software s uzavřeným kódem, Microsoft po určitou dobu poskytoval ostatním vývojářům softwaru licence k používání tohoto jazyka i v jiných programech než v balíčku Microsoft Office.

Alespoň částečně implementovaný lze jazyk VBA zaznamenat v programech jako jsou:

- UNICOM System Architect [3] – nástroj pro vizualizaci podnikové architektury.
- Autodesk AutoCAD 2023 [4] - 2D a 3D modelovací software pro projektování
- SolidWorks [5] – 3D CAD strojírenský software
- OpenText™ Reflection [6] – software pro emulování terminálů
- CorelDRAW [7] – profesionální grafický software pro designéry
- ArcGIS – software geografického informačního systému, podpora VBA ukončena
- LibreOffice – nekompletní podpora maker VBA

## 1.2 Ostatní verze Visual Basic

Pod pojmem Visual Basic se v rámci historie objevila řada několika programovacích jazyků, které spolu do jisté míry souvisejí. Při programování s VBA je nutné rozlišovat tyto jazyky, protože některé z nich mohou mít velmi podobnou syntaxi, každopádně nemusí zaručovat obdobnou funkci. Jedná se například o jazyky Classic Visual Basic (VB 6.0), .NET Visual Basic (VB.NET) či Visual Basic Script (VBScript).

### 1.2.1 Classic Visual Basic

Klasickou verzí VB se označují verze od první vytvořené v roce 1991 až po poslední verzi 6.0 vytvořenou v roce 1998. Základní struktura jazyka byla inspirovaná programovacím jazykem BASIC, který na rozdíl od VB neobsahuje integraci s vývojem grafického rozhraní. Byl vyvíjen společností Microsoft primárně pro potřebu tvorby aplikací pro operační systém Windows.

Spuštění aplikací vytvořených v poslední verzi 6.0 je stále podporováno ve všech aktuálně podporovaných verzích operačního systému Windows (včetně nového operačního systému Windows 11). Vývojářský tým klasické verze je zavázán podporovat prostředí pro spuštění aplikací VB po celou dobu podpory operačních systémů Windows. Podpora integrovaného vývojářského prostředí byla ukončena 8. dubna 2008. [8]

Důsledkem ukončení podpory IDE vyplývá, že v této verzi není vhodné do budoucna vyvíjet jakýkoliv software a je vhodné přejít na jednu z novějších adaptací jazyka.

### 1.2.2 .NET Visual Basic

V rámci balíčku .NET je Visual Basic (VB.NET) objektově orientovaný programovací jazyk. Používá se na rychlé a snadné vytvoření aplikací s grafickým uživatelským rozhraním pro operační systém Windows s datově typovou bezpečností. [9]

Mezi hlavní strategie VB.NET je zajištění snadno přístupného a srozumitelného jazyka se stabilním designem. Pokud v knihovně .NET přibude nová funkce, VB.NET ji převezme, ovšem takovým způsobem, aby se vyhnul nové syntaxi a zpětně nepoškodil již existující kód. [10]

První uvedená verze VB.NET byla vydána v roce 2002 společně se sadou vývojářských nástrojů (SDK) Visual Studio 2002. Od této doby byl jazyk rozšířen o typ My (zjednodušující práci s aplikací, sítí, souborovým systémem), knihovnu LINQ (zjednodušuje práci s daty), lambda výrazy, asynchronním programováním a dalšími. Tyto funkce nejen ovlivnily jazyk VB.NET, ale i jiné programovací jazyky jako například C#.

Jazyk .NET Visual Basic lze využít pro vývoj doplňků pro aplikaci Excel spolupracující pomocí objektového modelu COM aplikace Excel (COM object model). Takovéto propojení dovoluje obdobnou ovládací funkci programu Excel, kterou má programovací jazyk VBA.

## 1.3 Vývoj a budoucnost VBA

Firma Microsoft umožňovala od června 1996 firmám licencování VBA pro ty, kteří chtěli začlenit VBA do svých vlastních aplikací. Nicméně později oznámila, že neplánuje zásadní vylepšení VBA. Od 1. července 2007 pak Microsoft zrušil možnost distribuce licence VBA pro nové zákazníky. [11]

Společnost Microsoft si je vědoma, že uživatelská komunita věnovala mnoho hodin vývoji VBA nástrojů pro své podnikání v desktopových aplikacích Excel, Word a dalších aplikacích Office. Microsoft nemá v plánu zmařit tuto investici a učinit ji zbytečnou. Z těchto důvodů se předpokládá, že existence programovacího jazyka VBA se pravděpodobně v brzké době neukončí. [12]

### 1.3.1 Verze

První uvedení VBA na trh bylo v roce 1993 společně s aplikací Microsoft Excel 5.0. Tato verze představovala posun ve vývoji pomocných skriptů ke kancelářským programům a velmi rychle se stala populární.

V roce 1996 byla na trh uvedena verze VBA 4.0, která proběhla kompletní inovací implementováním pomocí jazyka C++. Tato verze přinesla do jazyka VBA objektově orientovaný přístup k programování.

Jednotlivé verze od roku 1997 byly pravidelně aktualizovány, po verzi VBA 6.5, která vyšla s Office 2007. Od této doby byl vývoj jazyka podstatně utlumen. Následující verzí byla verze 7.0, která přinesla pouze podporu 64bitů v balíčku Office 2010.

Aktuální vydanou verzí VBA v roce 2023 je verze 7.1, která je obsažena v balících Office 2013, Office 2016, Office 2019 a Office 2021. Tato verze neobsahuje příliš mnoho významných změn od předchozí verze 7.0.

### 1.3.2 Office Scripts

Office skripty a VBA makra mají mnoho společných prvků. Oba tyto nástroje umožňují uživatelům automatizovat svá řešení prostřednictvím snadno použitelného záznamníku akcí a také umožňují úpravy těchto záznamů. Oba rámce jsou navrženy s ohledem na uživatele, kteří nemají zkušenosti s programováním, aby mohli vytvářet jednoduché programy v aplikaci Excel. [13]

Hlavním rozdílem oproti makrům VBA je oblast použití. Skripty Office se dají použít pouze v souvislosti s aplikací Excel a jsou spíše zamýšleny pro použití v multiplatformních a cloudových prostředích, zatímco makra VBA jsou vždy mířena na použití na desktopových počítačích s operačním systémem Windows.

Skripty Office jsou velmi omezené ve funkčnosti, jako například v použití Windows formulářů nebo v přístupu k souborům počítače, na kterém se aplikace provozuje. Obě tyto funkce a jiné jsou do jazyka VBA implementovány.

Výhody skriptů zahrnují jejich použití ve webových prohlížečích, snadné sdílení mezi uživateli a nižší bezpečnostní riziko ve srovnání s makry VBA. Office skripty jsou psány v programovacím jazyce TypeScript (který je založen na JavaScriptu). [14]

Office skripty je vhodné použít v případech, kdy uživatelé nepotřebují funkce, které jsou jinak dostupné pouze v jazyce VBA. Zároveň se Office skripty jeví jako budoucnost automatizace kancelářského softwaru balíku Office na základě nedávného rozvoje v této oblasti firmou Microsoft.

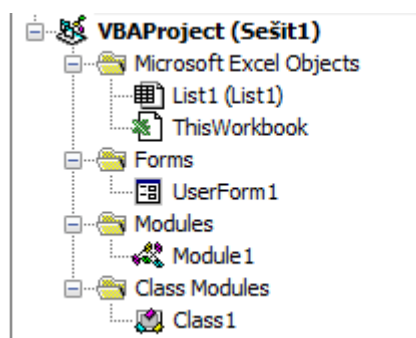
## 1.4 Základní koncepty jazyka

Podle společnosti Microsoft existuje pouze řada základních konceptů, které je pro používání jazyka VBA nutné pochopit. Mezi tyto koncepty patří správné volání procedur, správná deklarace proměnných a příslušných datových typů, strukturování podmiňovacích příkazů, efektivní využívání cyklů a práce s objekty, jejich vlastnostmi, metodami a událostmi. [15]

Ostatní možnosti aplikování jazyka VBA jsou primárně odvozovány od hostující aplikace a jejím ovládní. Výše uvedené koncepty jsou klíčové pro tvorbu funkčních VBA skriptů pro rozšíření jakýchkoliv aplikací, které mají jazyk VBA implementovaný.

### 1.4.1 Struktura projektu

Projekt VBA je skoro vždy vázán na konkrétní sešit Excelu a makra se ukládají přímo do tohoto sešitu v binární podobě. Toto tvrzení by platilo i o ostatních programech s rozdílem, že binární podoba kódu by byla implementována do příslušného dokumentu dané aplikace. Základní strukturu projektu lze vidět v levé části VBE (Obrázek 3), kde se nachází všechny soubory celého projektu, popř. dalších projektů.



Obrázek 3 – Struktura projektu VBA

V postranním menu, které zobrazuje strukturu projektu, jsou zobrazeny složky *Microsoft Excel Objects*, *Forms*, *Modules* a *Class Modules*. Tyto složky představují rozložení jednotlivých vývojářských kódů dle použití a pomáhají s přehlednou organizací.

Složka *Microsoft Excel Objects* převážně obsahuje objekt právě otevřeného sešitu a pak jednotlivé objekty listů sešitu. Dvoj kliknutí na jeden z objektů zobrazí nové podokno ve VBE, kde lze psát kód k příslušnému objektu. Tato složka je výchozí pro Excel VBA projekt a jako jediná v novém projektu zobrazená.

Soubor modulů (*Module*) by měl obsahovat procedury, kterými vývojář rozšiřuje funkčnost svého sešitu. Tyto procedury by měly zahrnovat makra a funkce, které reagují na změny a vytvořené události v používaném sešitě aplikace Excel. Moduly lze zároveň exportovat a importovat mezi různými sešity.

Pro implementaci objektově orientovaného programování v jazyce VBA se využívá soubor modulů tříd (*Class Modules*). V těchto souborech by se třídy měly definovat společně s jejich vlastnostmi, metodami a událostmi.

Soubor uživatelských formulářů (*Forms*) by měl obsahovat část s grafickým rozhraním formuláře a druhou část s kódem, který toto rozhraní obsluhuje.

#### 1.4.2 Procedury

V rámci jazyka VBA existují dva základní typy procedur: makra a funkce. Všechn spustitelný kód musí být umístěn uvnitř procedur. Deklarace procedur nelze zanořovat do jiných procedur. [16]

Makrem je označována každá procedura, která je definována v kódu pomocí klíčového slova *Sub* (vychází z anglického názvu subroutine – v češtině subrutina, podprogram nebo makro). Uvnitř makra se nachází sada instrukcí, které určují činnost, kterou se počítač pokusí provést. Všechny instrukce uvnitř kódu musí být správně syntakticky zapsány, jinak dojde k chybě při běhu programu.

Procedury typu funkce na rozdíl od maker vracejí tzv. návratovou hodnotu, jinými slovy nějaký finální výsledek, ke kterému během svého chodu dospěli. Tento výsledek může být výsledek výpočtu, hláška, datum a jiné.

Definice funkce probíhá klíčovým slovem *Function*, které je následované jedinečným názvem (Obrázek 4), jinak by mohlo dojít ke kolizím funkcí. Funkci lze zavolat v kódu a její výsledek uložit do proměnné nebo ji lze použít přímo v sešitě Excel. Návratová hodnota se v takovém případě objeví v buňce, ze které funkce byla volána.

```
Function MojeFunkce()  
    'Kod funkce  
    MsgBox ("Zdravím, z funkce!")  
    'Přiřazení návratové hodnoty funkce  
    MojeFunkce = "Pozdrav"  
End Function
```

Obrázek 4 – Ukázka definice funkce

Před ukončením funkce se musí uložit návratová hodnota. Tento proces proběhne přiřazením dané hodnoty názvu funkce pomocí operátoru rovná se. V případě nepřirazení funkce nebude vracet žádnou hodnotu.

### Modifikátory a atributy procedur

Každá procedura má nadefinovanou úroveň přístupu. Pokud není přímo určena pomocí modifikátoru přístupu, automaticky se přiřadí modifikátor *Public*.

Modifikátor *Public* umožňuje veřejný přístup k proceduře ze všech modulů, na rozdíl od modifikátoru *Private*, který omezuje přístup k proceduře pouze na modul, ve kterém je definována. Modifikátor *Friend* může být použit pouze u procedur v modulu třídy. [17]

Pokud se deklaruje procedura s atributem *Static*, lokální proměnné jsou zachovány ve stejném stavu mezi jednotlivými voláními procedury. Tato změna se netýká globálních proměnných, i když jsou v proceduře použity.

### Parametry procedur

Argumenty procedur jsou hodnoty (nebo proměnné), které se předávají do procedury jako parametry při jejím volání. Ty jsou použity ke změně chování procedury. Procedura může mít jeden nebo více parametrů, které jsou ve volání procedury odděleny čárkami.

Pokud je při deklaraci parametru použito klíčové slovo *ByVal*, argument je z volající procedury předán jako zkopírovaná hodnota. Deklarovaná procedura pak nemodifikuje původní proměnnou mimo proceduru.

Pokud nebyla zvolena indikace způsobu předávání proměnné nebo bylo při deklaraci použito klíčové slovo *ByRef*, argument je z volající procedury předán jako reference na jeho proměnnou. Deklarovaná procedura pak modifikuje i původní proměnnou mimo proceduru.

Při deklaraci posledního parametru je možné použít klíčové slovo *ParamArray*, které označuje, že konečným argumentem je pole prvků datového typu *Variant*. Toto označení umožňuje zadat libovolný počet argumentů. [17]

### 1.4.3 Proměnné

Při práci s procedurami je zapotřebí ukládat určité hodnoty, výsledky výpočtů a jiná data, které algoritmus zapsaný v kódu potřebuje. K tomuto účelu slouží proměnné, které alokují místo v paměti počítače a uloží do něj potřebné hodnoty.

Implicitní deklarace proměnných ve VBA proběhne, pokud se dříve nepoužitému názvu (názevu proměnné) přiřadí hodnota. Přiřazení hodnoty probíhá napsáním názvu proměnné následovaného znakem rovná se a hodnotou, kterou chce vývojář do proměnné uložit.

Při explicitní deklaraci pomocí klíčového slova *Dim* je nutné určit datový typ proměnné pomocí klíčového slova *As* a názvu datového typu či třídy.

Proměnné deklarované pomocí klíčového slova *Static* namísto klíčového slova *Dim* zachovávají svou hodnotu, i když program opustí danou proceduru. [18]

### 1.4.4 Datové typy

Kontext hodnot v proměnných je udržován pomocí datových typů. V paměti počítače jsou tyto hodnoty zapsány pomocí jedniček a nul a bez kontextu se nedá určit, jestli je tato kombinace číslo, text, datum nebo něco kompletně odlišného (jak je zobrazeno v Tabulka 1). Pro vývojáře i počítač zároveň značí, jak s těmito hodnotami zacházet (alokovat místo v paměti, používání určitých funkcí, ...).

<b>Binární číslo v paměti</b>	0100 0001 0100 0000
<b>Číslo (desítková soustava)</b>	16 704
<b>Text (UTF-16; znak)</b>	A ( <i>U+0041</i> )
<b>Datum</b>	24.09.1945

Tabulka 1 – Ukázka reprezentace dat se stejnou hodnotou

Pokud je proměnná definována implicitně bez datového typu, přiřadí se jí automaticky datový typ *Variant*, který se přizpůsobuje na základě vložené hodnoty. To vyžaduje vyšší náklady na paměť a řízení práce s proměnou

V dnešní době jsou tyto vyšší náklady při vyšších výkonech počítačů a větších kapacit paměti zanedbatelné, ale v minulosti mohly značit problémy. Datový typ *Variant* zároveň umožňuje vývojářům představit v chodu programu chyby v podobě načtení dat jiného datového typu do proměnné, než je potřeba. Z těchto důvodů se většinou vývojáři v jazyce VBA snaží implicitní deklaraci vyvarovat.

Datový typ	Význam	Hodnoty
<b>Boolean</b>	Logická hodnota	True / False (Pravda / Nepravda)
<b>Integer (%)</b>	Celé číslo	-32 768 až 32 767
<b>String (\$)</b>	Text (řetězec znaků)	0 až 2 <sup>32</sup>
<b>Double (#)</b>	Desetinné číslo	-1,79769313486231E308 až -4,94065645841247E-324 pro záporná čísla 4,94065645841247E-324 až 1.79769313486232E308 pro kladná čísla
<b>Date</b>	Datum	1. ledna 100 až 31. prosince 9999
<b>Currency</b>	Měna	-922 337 203 685 477,5808 až 922 337 203 685 477,5808
Long (&)	Celé číslo	-2 147 483 648 až 2 147 483 647
LongLong	Celé číslo	-9 223 372 036 854 775 808 až 9 223 372 036 854 775 807
Byte	Bajt	0 až 255
Single (!)	Desetinné číslo	-3.402823E38 až -1.401298E-45
<b>Variant</b>	Automaticky přizpůsobitelný datový typ	Podle přiřazené hodnoty
<b>Object</b>	Objekt	Podle vlastností objektu
Type	Uživatelsky definovaný datový typ	Libovolné (dle použitých typů)

Tabulka 2 – Základní datové typy ve VBA

Výše uvedené (Tabulka č.2) datové typy jsou pouze základní. Programovací jazyk VBA obsahuje i další datové typy kromě výše uvedených. Deklarace datových typů proměnných lze zkrátit přiřazením typově deklaračního znaku, uvedeného za názvem datového typu ve zmíněné tabulce.

### 1.4.5 Operátory

Do proměnných lze ukládat i výsledky výpočtu, které lze provést pomocí aritmetických operátorů (viz. Tabulka 3 – Aritmetické operátory Tabulka 3), jiných proměnných a konstant. Číselné konstanty mohou být celá nebo desetinná čísla, která se ovšem musí ve VBA vždy psát s desetinnou tečkou a nikoliv čárkou, jak bývá v aplikaci Excel zvykem.

Operátor	Popis
+	Sečte dvě čísla (na levé a pravé straně)
-	Odečte od levého čísla pravé číslo
*	Vynásobí dvě čísla (na levé a pravé straně)
/	Podělí číslo na levé straně pravým číslem
Mod	Modulo, zjistí celočíselný zbytek po dělení levého čísla pravým číslem
^	Umocní levé číslo pravým číslem

Tabulka 3 – Aritmetické operátory

Výpočet může obsahovat závorky pro nejvyšší přednost ve výpočtu. Počítač nadále vypočítá veškerá umocnění ve výpočtu. Pokud jich je v příkladu více, postupuje z levé strany na stranu pravou. Po umocnění následuje násobení, dělení a modulo vše v pořadí zleva doprava. Na závěr počítač provede veškerá sčítání a odečítání opět ve stejném pořadí z levé strany. Ukázka priority operátorů ve výpočtu v kódu je zobrazena v Obrázek 5.

```
Dim x As Integer
x = 10 + 5 * 2 ^ 3 / 4
'Výsledek x bude 20
```

Obrázek 5 – Priorita výpočtu

Pro práci s řetězcí znaků v proměnných existují pouze dva operátory (zobrazené v Tabulka 4), které řetězce spojují. Ostatní funkce pro práci s textem nelze provádět pomocí operátorů a musí být použity specializované funkce.

Operátor	Popis
+	Spojí řetězec znaků na levé straně s řetězcem na straně pravé
&	Spojí řetězec znaků na levé straně s řetězcem na straně pravé

Tabulka 4 – Spojovací operátory

K dosažení logické hodnoty se v příkazech používají porovnávací operátory, které porovnávají dvě (většinou číselné) hodnoty. Podle výsledku porovnání navracejí hodnotu, která odpovídá požadovanému tvrzení. Tyto operátory jsou zobrazeny společně s popisem jejich funkcí v Tabulka 5.

Operátor	Popis
=	Pokud je levá strana rovna pravé straně, vrátí <i>PRAVDA (True)</i> , jinak <i>NEPRAVDA (False)</i> .
<	Pokud je levá strana menší než pravá strana, vrátí <i>PRAVDA (True)</i> , jinak <i>NEPRAVDA (False)</i> .
>	Pokud je levá strana větší než pravá strana, vrátí <i>PRAVDA (True)</i> , jinak <i>NEPRAVDA (False)</i> .
<=	Pokud je levá strana menší než pravá strana nebo rovna pravé straně, vrátí <i>PRAVDA (True)</i> , jinak <i>NEPRAVDA (False)</i> .
=>	Pokud je levá strana větší než pravá strana nebo rovna pravé straně, vrátí <i>PRAVDA (True)</i> , jinak <i>NEPRAVDA (False)</i> .
<>	Pokud není levá strana rovna pravé straně, vrátí <i>PRAVDA (True)</i> , jinak (pokud se rovnají) <i>NEPRAVDA (False)</i> .

Tabulka 5 – Porovnávací operátory

Využití logických operátorů (zobrazených v Tabulka 6) nastává v případě, kdy je potřeba použít více podmínek dohromady a přidat k nim další možnou rozšiřující logiku pro vznik podmínky nové.

Operátor	Popis
AND	Pokud je levá strana <i>PRAVDA (True)</i> a pravá strana je <i>PRAVDA (True)</i> , vrátí <i>PRAVDA (True)</i> , jinak <i>NEPRAVDA (False)</i> .
OR	Pokud je levá strana <i>PRAVDA (True)</i> nebo pravá strana <i>PRAVDA (True)</i> , vrátí <i>PRAVDA (True)</i> , jinak <i>NEPRAVDA (False)</i> .
XOR	Pokud je levá strana <i>PRAVDA (True)</i> a pravá strana <i>NEPRAVDA (False)</i> nebo pokud je levá strana <i>NEPRAVDA (False)</i> a pravá strana <i>PRAVDA (True)</i> vrátí <i>PRAVDA (True)</i> , jinak pokud jsou obě strany <i>PRAVDA (True)</i> nebo <i>NEPRAVDA (False)</i> , vrátí <i>NEPRAVDA (False)</i>
NOT	Pokud je hodnota na pravé straně <i>PRAVDA (True)</i> , vrátí <i>NEPRAVDA (False)</i> . Pokud je hodnota na pravé straně <i>NEPRAVDA (False)</i> , vrátí <i>PRAVDA (True)</i> .

Tabulka 6 – Logické operátory

### 1.4.6 Podmíněné příkazy

Ve VBA existují tři hlavní způsoby pro vypořádání se s větvením algoritmu. Jedná se o příkazy *If*, *Else* a *Else If*, příkaz *Select Case* nebo podmiňovací funkce *IIf*, *Switch* a *Choose*.

Příkaz *If* umožňuje podmíněně spustit blok kódu, který se vyskytuje v jeho těle. Při větvení na základě více podmínek lze příkaz rozšířit o další variantu pomocí *Else If*. Nakonec lze využít příkazu *Else*, který spustí svůj blok kódu, pokud žádná z předchozích podmínek nebyla naplněna. Syntaxe příkazů je znázorněna v Obrázek 6.

```
' Proměnná s číslem
cislo = 11

If (cislo > 0) Then
    ' Číslo je větší jak nula => číslo je kladné
    MsgBox ("Číslo je kladné!")
ElseIf (cislo < 0) Then
    ' Číslo je menší jak nula => číslo je záporné
    MsgBox ("Číslo je záporné!")
Else
    ' Číslo je nula, protože jiná možnost neexistuje
    MsgBox ("Číslo je nula!")
End If
```

Obrázek 6 – Syntaxe příkazů *If*, *Else* a *Else If*

Alternativou k příkazům *If*, *ElseIf* a *Else* je příkaz *Select Case*, který jako vstup přijme hodnotu (většinou z proměnné), kterou se následně snaží přirovnat ke všem možnostem, které má nadefinované.

Možnosti jsou nadefinované pomocí klíčového slova *Case* následované možnou hodnotou. Pokud žádná hodnota nevyhovuje, lze nadefinovat možnost *Case Else*, která bude výchozí možností, pokud nebudou odpovídat možnosti jiné.

```
Function VysledekZBodu(Optional pocetBodu = 100)
    ' Určí výsledek podle dosaženého počtu bodů
    Select Case pocetBodu
        Case Is > 100
            VysledekZBodu = "Exceloval"
        Case 50 To 100
            VysledekZBodu = "Uspěl"
        Case Else
            VysledekZBodu = "Neuspěl"
    End Select
End Function
```

Obrázek 7 – Syntaxe příkazu *Select Case*

K dosažení jednoduchých podmiňovacích příkazů lze i za pomoci podmiňovacích funkcí *Switch*, *If* a *Choose*. Podmiňená funkce *If* vrací jednu ze dvou definovaných hodnot na základě podmínky, která je v prvním parametru funkce.

Funkce *Switch* zjednodušuje formát příkazu *Select Case* a definici jeho možností, pokud je potřeba vrátit pouze jednu hodnotu. Princip funkce *Choose* spočívá ve vybírání z možností na základě prvního číselného parametru a ostatních parametrů, které určují návratové hodnoty.

### 1.4.7 Cykly a rekurze

V programování (nejen) VBA kódu je občas nutné vykonat určitou činnost několikrát. Cykly jsou příkazy, které opakují svůj vnitřní blok kódu, dokud není dosaženo nějaké podmínky. Ve VBA jsou celkově 3 základní cykly: *While*, *For* a *Do* (ve variantách *Until* a *While*).

Definice cyklu *While* začíná klíčovým slovem *While* a končí klíčovým slovem *Wend*, jak lze vidět v Obrázek 8. V cyklu se vyskytuje ukončovací podmínka, která řídí cyklus a jeho ukončení. Pokud má daná podmínka hodnotu *PRAVDA*, cyklus se bude opakovat. Ke kontrole podmínky dochází vždy při začátku cyklu a pokud její hodnota bude *NEPRAVDA*, cyklus bude v daném místě ukončen.

```
Dim i As Integer
i = 0

' Počítání do 20
While (i <= 20)
    MsgBox (i)
    i = i + 1
Wend
```

Obrázek 8 – Syntaxe cyklu *While*

Cyklus *For* (Obrázek 9) má definovanou proměnou, která se posunuje k určitému cíli. Při běhu cyklu se kontroluje, zda hodnota definované proměnné nepřesáhla definované číselné omezení. Cyklus po dokončení svého vnitřního kódu vždy navýší proměnnou o určitou hodnotu (krok *Step*) a spustí vnitřní kód znovu, dokud nedosáhne omezení.

```
zprava = ""
' Počítání do 20 po dvou
For i = 0 To 20 Step 2
    zprava = zprava & CStr(i) & " "
Next i

' Zobrazí "0 2 4 6 8 10 12 14 16 18 20 "
```

Obrázek 9 – Syntaxe cyklu *For*

U příkazu cyklu *Do* existují dvě varianty jeho použití. První z nich je možnost použít dodatečné klíčové slovo *While*, které zajistí stejnou funkcionalitu cyklu s podmínkou, jako předem jmenovaný cyklus *While* (Obrázek 10). Obě varianty cyklu jsou vždy ukončeny klíčovým slovem *Loop*, které označuje konec opakujícího se bloku.

```
Dim i As Integer
i = 0

' Počítání do 20
Do While (i <= 20)
    MsgBox (i)
    i = i + 1
Loop
```

Obrázek 10 – Syntaxe cyklu *Do While*

Druhou variantou je cyklus *Do Until*, který na první pohled vypadá obdobně jako cyklus *Do While* (Obrázek 11). Rozdíl je v tom, že cyklus pokračuje, když je podmínka hodnoty *NEPRAVDA*

a ukončí se v momentě, kdy je kontrolní podmínka hodnoty *PRAVDA*.

```
Dim i As Integer
i = 0

' Počítání do 20
Do Until (i > 20)
    MsgBox (i)
    i = i + 1
Loop
```

Obrázek 11 – Syntaxe cyklu *Do Until*

Dalším způsobem opakování jsou rekurzivní volání funkce (příklad v Obrázek 12). V principu jde o volání dané funkce uvnitř svého bloku kódu. Volání musí mít určitou ukončovací podmínku, která zamezí nekonečnému volání funkce. Kdyby volání nebylo ukončeno, program by vyčerpal veškerou paměť, kterou by měl k dispozici, a nakonec by spadl.

```
' Výpočet faktoriálu, např. 6! = 720
Function FaktorialRekurzivne(cislo)

    If cislo = 1 Then
        FaktorialRekurzivne = 1
    ElseIf cislo <= 0 Then
        FaktorialRekurzivne = 0
    Else
        FaktorialRekurzivne = cislo * FaktorialRekurzivne(cislo - 1)
    End If

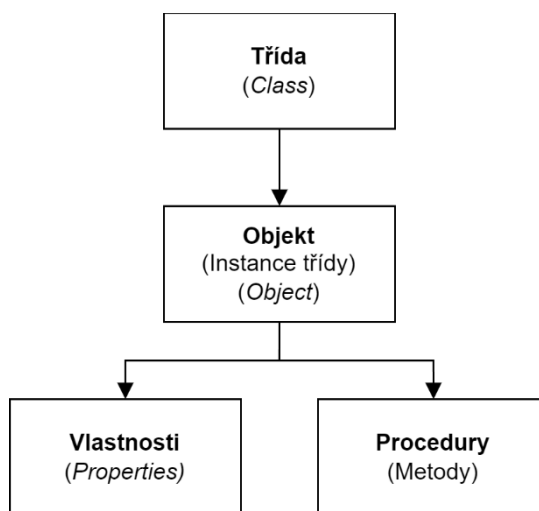
End Function
```

Obrázek 12 – Příklad rekurzivní funkce

### 1.4.8 Třídy a objekty

Pojem objekt můžeme chápat jako ucelenou kombinaci kódu a dat, která může být určitou částí aplikace, jako například ovládací prvek nebo okno formuláře. V principu může být i celá aplikace praktickým objektem. Při programování ve VBA se pracuje s objekty neustále, i když to na první pohled nemusí být zřejmé. [19]

Každý objekt je ve VBA definován třídou, která popisuje proměnné, vlastnosti, procedury a události v objektu (viz. Obrázek 13). Před použitím těchto prvků je potřeba vytvořit danou instanci třídy (daný objekt). Přes vytvořený objekt pak lze přistupovat k jednotlivým metodám a vlastnostem. Vytvořených instancí třídy může být v kódu libovolné množství.



Obrázek 13 – Diagram tříd, objektů a jejich vlastností a metod

## 2 NÁSTROJE K TVORBĚ APLIKACE

Aplikace multimedialního průvodce umožňuje uživateli interagovat s obsahem pomocí interaktivních prohlídek, textu, obrázků a videí. Uživatelé pomocí aplikace objevují a získávají informace. Ve výsledku je uživateli poskytnuto větší zapojení s obsahem a zároveň vylepšen celkový zážitek k porozumění obsahu.

### 2.1 Framework Tauri

Pro tvorbu aplikací existuje v dnešním světě vývojářů spousta možností. Ideální framework by měl jít bez problému spustit na všech platformách a zároveň poskytovat velké množství nástrojů, které by aplikace mohla využít. Ve sféře multiplatformních aplikací je například velmi populární framework Electron, který tyto možnosti nabízí.

Tauri, obdobně jako Electron, je framework určený pro vývoj multiplatformních aplikací, který dovoluje spustit aplikaci na hlavních desktopových platformách. Vytvořené aplikace dosahují malých velikostí a extrémní responzivity.

Vývojáři dostávají možnost integrovat do Tauri libovolný front-end webový framework pro vytvoření uživatelského rozhraní. Backend aplikace tvoří zkompileovaný kód v programovacím jazyce Rust. Framework Tauri poskytuje API pro zajištění komunikace mezi oběma konci aplikace. [20]

Framework podporuje distribuci vytvořené aplikace do operačních systémů Windows 7 a výše, macOS 10.15 a výše, či některých distribucí operačního systému Linux. Pro zobrazení grafického rozhraní framework používá systémové rozhraní WebView.

V porovnání s populárním frameworkem Electron, postaveném na jádře Chromium, lze konstatovat, že Tauri pomocí rozhraní WebView přináší několik výhod. Tauri nejenže efektivně využívá systémové zdroje při provozu aplikace, ale nabízí výrazně vyšší rychlost. Balíček Tauri aplikace je zároveň mimořádně kompaktní, protože potřebné zdroje se nachází již v operačním systému.

#### 2.1.1 Framework SvelteKit

Pro potřeby frameworku Tauri pro front-end je možné použít moderní webový framework SvelteKit. Framework umožňuje rychlý vývoj robustních a výkonných aplikací pomocí technologie Svelte. [21]

Pomocí Svelte jsou tvořeny komponenty uživatelského rozhraní, které jsou zkompileovány do spustitelného JS souboru a stylového CSS souboru. Vyexportovaný skript následně vykresluje HTML stránky s komponenty. [21]

Při vývoji aplikace dochází k okamžitému zobrazení změn kódu ve spuštěném okně. Svelte a SvelteKit poskytují vývojáři spoustu užitečných možností pro vývoj webu i aplikací a celková struktura projektu a syntaxe je velmi příjemná.

### 2.1.2 Rust

Rust je programovací jazyk, který slouží k vytváření kompilovaných a paměťově bezpečných aplikací. Kombinuje jednoduchost s vysokým výkonem na nízké úrovni. Je hojně využíván při vývoji systémů, kde je výkon naprosto klíčový, jako jsou herní enginey, databáze nebo operační systémy. [22]

Na rozdíl od jiných nízko úrovněových programovacích jazyků Rust nepoužívá automatickou správu paměti pomocí *garbage collection*. Automatická správa paměti je zde nahrazena systémem modelu vlastnictví, čímž je zajištěna paměťová bezpečnost.

Rust disponuje integrovaným nástrojem pro správu balíčků přídatných rozšíření jménem Cargo. Balíčky jsou v tomto nástroji označovány jako *crate*. Ve frameworku Tauri zajišťuje tento programovací jazyk backend aplikace a komunikaci s operačním systémem, na kterém aplikace běží.

### 2.1.3 TypeScript

Programovací jazyk JavaScript postrádá funkce jako jsou statické typování, třídy, rozhraní a další možné prvky z objektového programování. Nastavba TypeScript tyto prvky do JS přidává a usnadňuje vývojářům práci s pomocí datových typů, napovídání v kódu a automatickou kontrolou. TS kód používá tzv. transpiler pro přeložení kódu do JS. [23]

Přidané funkce TS se využívají při programování aplikace k rychlejší identifikaci chyb, přehlednosti v kódu a možného pozdějšího rozšíření aplikace.

### 2.1.4 Tailwind CSS

Na rozdíl od tradičních CSS frameworků (například Bootstrap) open-source framework Tailwind CSS nenabízí předdefinovanou sadu CSS tříd pro prvky tlačítka, tabulky aj. Místo toho přináší koncept „užitkových“ atomických tříd, které umožňují vytvářet styly pomocí jejich kombinace. [24]

Kromě těchto tříd Tailwind nabízí jejich varianty, například pokud je na element zaměřen *focus*, *hover* a jiné možné stavy. Zároveň podporuje světlý a tmavý režim aplikace, umožňuje použít a nastavit vlastní téma (barevné palety, velikosti a další), nebo podporuje Just In Time způsob generování CSS stylů. Způsob vygeneruje pouze třídy, které se používají a tím snižuje velikost finálního CSS souboru.

Důsledkem použití frameworku je optimalizace CSS stylů, jednotná grafická úprava s relativně příjemným grafickým rozhraním a zjednodušení psaní stylů SvelteKit komponentů pomocí znovu aplikovatelných atomických tříd.

### 2.1.5 Knihovna Skeleton

Knihovna komponentů Skeleton nabízí často používané elementy aplikací spojením světů frameworků Tailwind CSS a Svelte. Společně s touto stavební architekturou nabízí snadno přizpůsobitelné, responzivní a reaktivní komponenty pro všechny typy projektů. [25]

Použití knihovny a jejich komponentů šetří vývojářům aplikací převážně čas, protože již nemusí tvořit ustálené prvky znovu, ale zároveň je mohou graficky upravit dle potřeb.

### 2.1.6 Node.js

Node.js je hostitelské prostředí, které umožňuje asynchronně spustit kód programovacího jazyka JavaScript. Primární použití aplikace je cíleno na vytváření síťových aplikací, které jsou schopny efektivního škálování. [26]

V případě použití frameworků SvelteKit a Tauri, je Node.js použito jen při vývoji aplikace. Prostředí poskytuje změny na front-endu v reálném čase. SvelteKit jej používá při kompilaci výsledného souboru JavaScriptu, který je následně hostován v aplikaci Tauri.

Neodmyslitelnou součástí Node.js je Node Package Manager (NPM), které vývojářům umožňuje uživatelům vyhledávat přídatné balíčky. NPM se spouští v terminálu pomocí CLI, které představuje primární způsob ovládání rozhraní. Na serveru je dostupná veřejná databáze obsahující softwarové balíčky pro JavaScript a jejich související metainformace. [27]

### 2.1.7 Markdown

Markdown je univerzální a snadno použitelný značkovací jazyk, který umožňuje formátovat jednoduché textové dokumenty. Jeho tvůrcem je John Gruber, který ho vytvořil v roce 2004. Od této doby je Markdown jedním z nejpopulárnějších značkovacích jazyků používaných po celém světě. [28]

Soubory dokumentů napsané v tomto jazyce jsou označeny koncovkou *.md* nebo *.markdown*. Tyto textové soubory jsou později pomocí Markdown procesoru naformátovány do značkovacího jazyka HTML nebo dokumentu PDF.

Na základě velmi jednoduché syntaxe a možnosti snadné a rychlé úpravy textových souborů činí Markdown ideální volbu pro dokumenty aplikace, které lze později jednoduše upravovat. Zároveň mohou uživatelé aplikace použít tyto soubory pro své vlastní zápisy a upravit si je dle svých potřeb.

## 2.2 Video

### 2.2.1 OBS Studio

Jeden z možného softwaru pro nahrávání videa je volně dostupný program OBS Studio (Open Broadcaster Software). Spuštěná aplikace umožňuje snadno zachytit video určitého programu, vstupního zařízení nebo celé obrazovky. Při nahrávání videa může současně docházet i k zachycení zvuku z připojeného mikrofону. [29]

### 2.2.2 Audacity

Volně dostupný program Audacity umožňuje nahrávat zvuk z připojených audio zařízení. Zároveň lze nahrané zvukové stopy upravovat, vyčistit od šumu a stříhat. Program podporuje editování více stop zároveň a je dostupný pro operační systémy Windows, macOS, GNU/Linux a další operační systémy. [30]

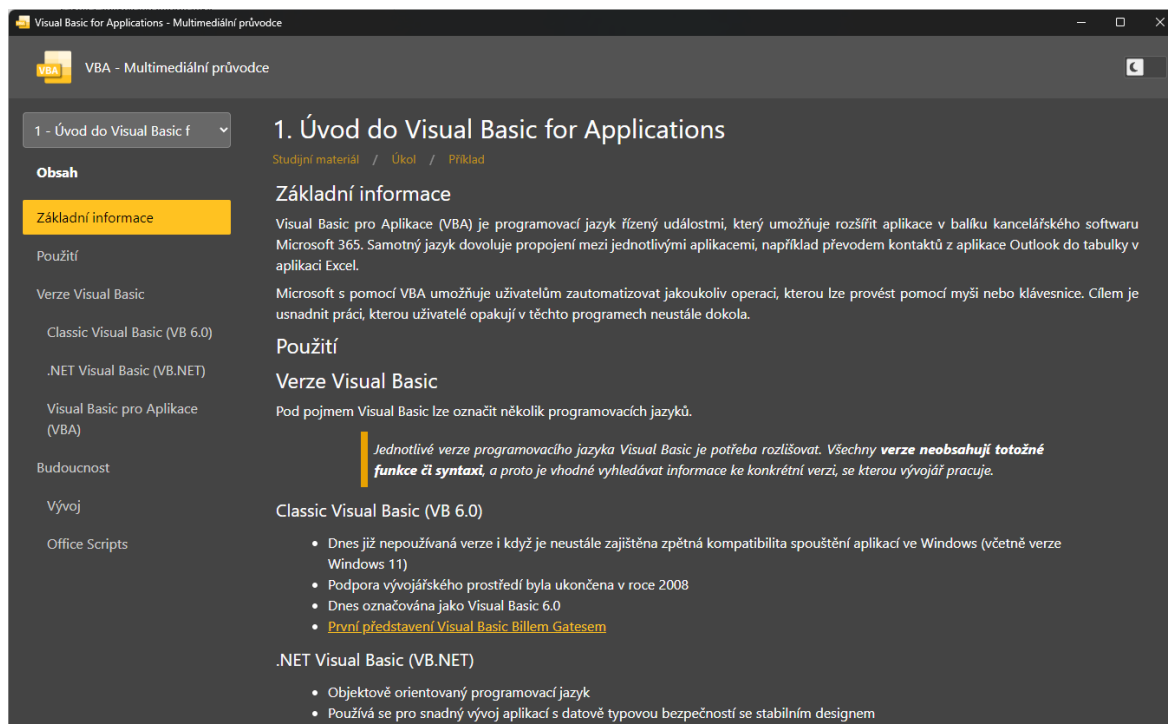
### 2.2.3 Adobe Premiere Pro

Pro stříh videa lze použít placený profesionální video editor Adobe Premiere Pro. Program umožňuje přidávat přechody, efekty a nadpisy do editovaných videí. Dále nabízí možnost pro mixování audia nebo funkce pro úpravu a korekci barev. Program nabízí exportovat konečné video v různých formátech, např. pro sociální síť YouTube a další. [31]

## **II. PRAKTICKÁ ČÁST**

### 3 APLIKACE MULTIMEDIÁLNÍHO PRŮVODCE

Primárním cílem aplikace je možnost procházet jednotlivé lekce a jejich videa pomocí snadno použitelného rozhraní (vyskytující se na Obrázek 14). Uživatel aplikace si v rozhraní může vybrat požadovanou lekci a projít si její textovou dokumentaci s přímým propojením na příklady do aplikace Excel. Každá lekce navíc obsahuje jedno video s probranou látkou a příklady vysvětlení. Video lze přímo v aplikaci přehrát.



Obrázek 14 – Rozhraní aplikace multimediálního průvodce

#### 3.1 Technologie aplikace

Přestože framework Tauri není přímo určený pro stavbu multimediálních CD průvodců, je poměrně vybavený funkcemi, které takový průvodce může využívat. Toto vybavení pochází z komponenty systému *WebView*, kterou framework využívá. Vytvořená aplikace se chová jako internetový prohlížeč pro svoji interní webovou strukturu. Prohlížečové zobrazení umožňuje číst HTML dokumenty, zobrazovat obrázky a pouštět videa.

Komunikace s backendem probíhá pomocí knihovny Tauri API, která dovoluje komunikovat s programovacím jazykem Rust. Backend v této aplikaci primárně zajišťuje čtení z požadovaných souborů a otevírání programu Excel s požadovaným materiálem.

O činnost frontendu se stará framework SvelteKit, který zajišťuje správné zobrazování dat a upravuje je dle potřeby. SvelteKit výsledný projekt kompiluje do nativního kódu JavaScriptu, který nezatěžuje spuštěné prostředí externí knihovnou.

Pro syntaktickou analýzu lekcí napsaných v Markdown souborech do HTML kódu byla použita knihovna *marked*. Sekce s programátorskými kódy byly vylepšeny pomocí knihovny *highlight.js*, která graficky zpracovává syntaxi kódů.

Vzhled aplikace byl upraven pomocí CSS frameworku Tailwind CSS a grafické knihovny Skeleton pro framework SvelteKit. Využitím těchto dvou technologií lze jednoduše dosáhnout relativně důstojného grafického rozhraní s možností responzivního zobrazení aplikace.

### 3.2 Čtení lekcí v souborech Markdown

Při načítání aplikace si zprvu aplikace vyžádá ve složce *materials* soubor *meta.json*. Tento soubor lze označit jako knihovnu celé aplikace, která popisuje, kde se jednotlivé lekce nacházejí a v jakém pořadí k nim aplikace má přistupovat.

Vyžádání probíhá pomocí metody *Invoke* funkce *read\_meta\_menu\_data* z frontendu při načítání stránky. Odpovídající funkce Rustu (kód funkce na Obrázek 15) vrátí přečtenou hodnotu zpět na frontend. Z přečtené hodnoty je vytvořeno menu se všemi lekcemi.

```
#[tauri::command]
fn read_meta_menu_data() -> String {
    match fs::read_to_string( get_current_working_dir() + "\\materials\\meta.json") {
        Ok(contents: String) => contents,
        Err(_) => String::from("{\"error\": \"Error, file couldn't be opened or found.\\\"}")
    }
}
```

Obrázek 15 – Naprogramovaná funkce *read\_meta\_menu\_data* v Rust

Načítání konkrétně vybrané lekce probíhá v souboru *+page.ts* pomocí asynchronní funkce *load* (ukázka kódu na Obrázek 16). Funkce vytvoří data lekce pro zobrazovanou stránku, které následně používají jednotlivé komponenty na stránce.

```
export const load = (async ({ url, parent }) => {
  let parentData = await parent();

  let currentIndex = getIndex(url);
  let nextPage = getNextURL(url, currentIndex, parentData.library.length);
  let previousPage = getPreviousURL(url, currentIndex);
  let meta : Material = parentData.library[currentIndex];

  return {
    markdownHTML : await loadSrcs(await readMaterialTextFile(meta.
      markdown_file)),
    meta : meta,
    index : currentIndex,
    next : nextPage,
    previous : previousPage,
  };
}) satisfies PageLoad;
```

Obrázek 16 – Asynchronní funkce load v +page.ts

Surový Markdown soubor je při načítání stránky nejdříve přečten jako text, pomocí naprogramované funkce *readMaterialTextFile*. Tento text je následně upraven ve funkci *loadSrcs*, která vytvoří HTML kód textu, upraví v něm všechny odkazy pro otevírání zvlášť v prohlížeči, dále označí všechny bloky kódu pro knihovnu highlight.js a přidá pomocí formátování v base64 všechna videa a obrázky. Čtení souborů v base64 probíhá v jazyce Rust pomocí funkce *read\_file\_as\_base64* (Obrázek 17).

```
fn read_file_as_base64(filepath: &str) -> io::Result<String> {
  let mut file: File = File::open(path: filepath)?;

  let mut buffer: Vec<u8> = Vec::new();
  file.read_to_end(buf: &mut buffer)?;

  let base64_data: String = general_purpose::STANDARD.encode(input: &buffer);

  Ok(base64_data)
}
```

Obrázek 17 – Funkce *read\_file\_as\_base64* v Rust

Na stránce je výsledný HTML kód zobrazen pomocí Svelte komponentu *MarkdownComponent*, který ještě upravuje původní HTML kód (např. přidáním tlačítek pro zkopírování bloků kódu). Svelte provede automatické převedení HTML v proměnné pomocí označení proměnné pomocí značky *@html*.

### 3.3 Responzivní zobrazení

Jak již bylo v teoretické části práce uvedeno, hlavní princip frameworku Tailwind CSS spočívá v používání „užitkových“ tříd. Pro zobrazení určité třídy v určité velikosti okna (s pomocí CSS media query) se používají varianty užitkových tříd. Varianty pro aplikování třídy v určitém zobrazení se zapisují před danou užitkovou třídou. Aplikování variant třídy lze vidět na Obrázek 18.

```
<AppShell
  class="h-screen"
  slotPageFooter="p-4 flex flex-row justify-between justify-items-center"
  slotSidebarLeft="lg:p-4"
  regionPage="p-4 flex-auto sm:pl-8 sm:pr-6 xl:px-22 2xl:px-44 overflow-x-hidden h-full"
>
```

Obrázek 18 – Aplikování variant media query užitkových tříd

Menu v malém zobrazení je tvořeno komponentou *Drawer* z knihovny *Skeleton*. Při malém zobrazení okna se zobrazí tlačítko, které po kliknutí zavolá metodu *open* z *drawerStore*, která otevře tuto komponentu vykreslenou nad všemi ostatními. Otevřený šuplík je nastylován pomocí užitkových tříd frameworku Tailwind.

### 3.4 Sestavení aplikace

Sestavení aplikace probíhá pomocí příkazů v příkazové řádce (CLI rozhraní) s pomocí aplikace Node.js a balíčkovacího softwaru npm. Pro vývoj aplikace se používá příkaz *npm run tauri dev*, který sestaví základ aplikace a umožňuje vývojáři okamžitě vidět veškeré změny ve zdrojovém kódu aplikace.

Pro finální sestavení se použije příkaz *npm run tauri build*, který sestaví samostatnou aplikaci podle operačního systému, na kterém sestavení běží. Sestavená aplikace je v adresáři projektu *src-tauri/release*.

### 3.5 Nahrávání a stříh videí

Při nahrávání videa v programu OBS Studio byl nejprve nastaven požadovaný zdroj – obrazovka zaměřující program Excel. Poté bylo spuštěno nahrávání a začal se zaznamenávat požadovaný obsah. Během nahrávání můžete lze přepínat mezi různými scénami a upravovat překryvy, to ovšem nebylo pro danou aplikaci potřeba. Jakmile se nahrávání dokončilo, video bylo uloženo v dříve nastaveném požadovaném formátu.

Při snímání obrazovky běželo v pozadí nahrávání zvuku z mikrofonu v aplikaci Audacity a po skončení záznamu byl zvukový soubor uložen. Pro dosažení optimální kvality zvuku byla použita funkce pro odstranění šumu, která eliminuje nežádoucí pozadí z odposlechu části šumu v nahrávce. Po dokončení úprav byl zvuk exportován do formátu, který podporuje stříhací software Adobe Premiere Pro.

Ve stříhovém programu Adobe Premiere Pro byly naimportovány všechny nahraná videa a zvuky. Na časové ose byla sestavena sekvence videa, kde byl prováděn potřebný střih, posun klipů a přidání přechodů. Dále byla provedena práce se zvukem, přesněji synchronizace s obrazem a úprava hlasitosti. Po dokončení editace bylo video vyexportováno do finální formátu odpovídající standardu pro nahrávání videí na sociální síť YouTube se souborovou koncovkou *.mp4*.

## 4 STUDIJNÍ MATERIÁLY

Pomocné studijní materiály pro potřeby předmětu Kancelářský software II jsou tvořeny z několika částí. Celá látka je rozdělena na 14 lekcí, které se týkají seznámení se základy programovacího jazyka VBA a jeho propojení s aplikací Microsoft Excel.

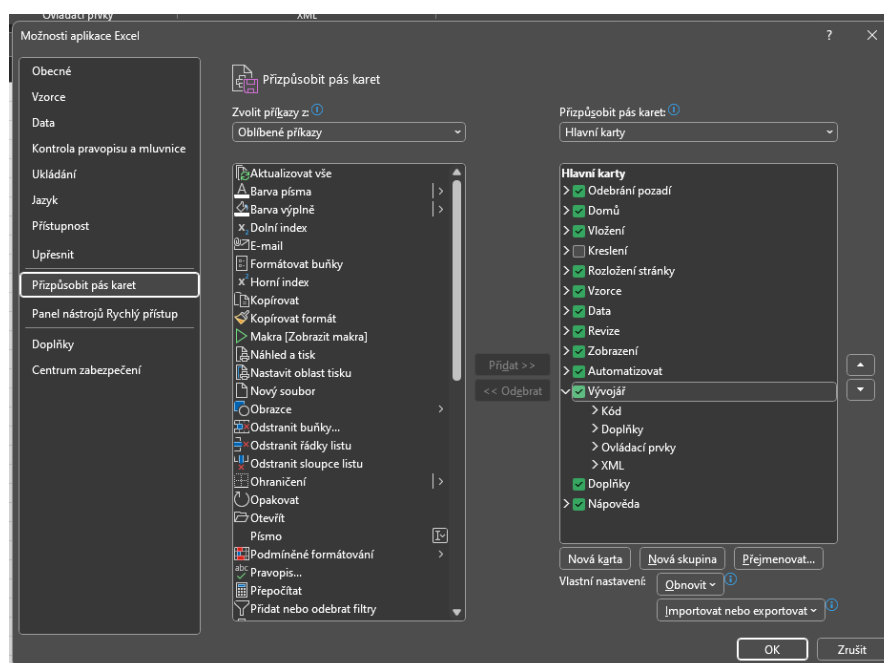
Každou lekci tvoří čtyři hlavní soubory. První soubor je textový materiál ve formátu PDF, který popisuje příslušné téma lekce. Dále se v lekci vyskytují dva soubory sešitů s makry, první s ukázkami použití tématu a druhý s vypracovanými úkoly k lekci. Zpracování úkolů je zaheslované. Poslední soubor obsahuje video, které se řídí strukturou textových materiálů a vysvětluje koncepty.

Z uvedených textových materiálů jsou vyhotoveny soubory pomocí značkovacího jazyka Markdown, které jsou použity v přiložené aplikaci multimediálního průvodce. V aplikaci jsou dostupná i videa, které jsou k dispozici ve složce lekce. Obsah Markdown souborů je kromě formátování totožný se soubory PDF.

## 4.1 Lekce 0 – Úvod do VBA

Úvodní lekce představuje jazyk VBA společně s jeho polem působnosti. Lekce představuje základní cíle jazyka, funkce jazyka a proč je vhodné jej využívat. Dále poukazuje na rozdíl mezi jednotlivými programovacími jazyky z rodiny Visual Basic a popisuje kompatibilitu jazyka mezi operačními systémy.

Lekce následně popisuje vývojové prostředí v aplikaci Excel a jak uvést do provozu využití jazyka VBA (Obrázek 19) pomocí Visual Basic Editoru. Představená je i základní struktura VBA projektu.



Obrázek 19 – Nastavení pásu karet v MS Excel pro přístup k VBA

Hlavní cíle lekce:

- Seznámení se s kartou vývojáře v MS Excel
- Seznámení se s vývojářským prostředím (VBE)
- Seznámení se s rozložením VBA projektu
- Procvičit nové znalosti

Úkoly lekce:

- Vytvoření prvního projektu VBA
- Přejmenování projektu s přidáním vlastního popisu
- Uložení projektu k sešitu Excelu

## 4.2 Lekce 1 – Procedury

Úvodní lekce se zabývá rozšiřováním funkcí sešitu nebo listu pomocí dvou hlavních procedur: maker a funkcí. Text nadále popisuje základní práce s procedurami, jak vytvořit první makro pomocí nabídky v programu Excel.

Látka nadále probírá možnost záznamu makra pomocí akcí uživatele v programu a jak jej později upravit. Nadále je v textu ukázáno, jak nahraná a jiná makra uložit do sešitu. Na závěr lekce je rozebrán princip funkce (Obrázek 20) a poukázán na rozdíl oproti klasickému makru.

```
' Nadefinovana funkce s názvem "MojeFunkce"  
Function MojeFunkce(Jmeno, Optional Prijmeni = "ve VBA")  
    ' Návrátová hodnota funkce - Vrátí vložené jméno a příjmení, pokud bylo vloženo  
    ' Pokud Prijmeni nebylo vloženo, vrátí místo něj "ve VBA"  
    MojeFunkce = Jmeno & " " & Prijmeni  
End Function
```

Obrázek 20 – Ukázka z příkladu funkce s použitím parametru *Optional*

Hlavní cíle lekce:

- Seznámení se s procedurami
- Vytvoření a použití makra
- Záznam makra
- Vytvoření a použití funkce
- Využití parametrů funkce

Úkoly lekce:

- Vytvoření prvního subrutiny
- Zaznamenání nového makra
- Vytvoření první funkce
- Vytvoření složitější s aplikací funkce v sešitě na datech

### 4.3 Lekce 2 – Proměnné, datové typy a výpočty

V první části lekce je vysvětlen význam proměnných v programování, po kterém je následně popsán proces deklarace a přiřazování hodnot proměnným. Druhá část textu se věnuje datovým typům a jejich významu při udržování kontextu hodnot v proměnných.

Třetí část se zabývá operátory používanými při práci s proměnnými. Jsou zde uvedeny aritmetické operátory pro provádění výpočtů a spojovací operátory pro práci s řetězci znaků. Dále je zde ilustrována priorita výpočtů a práce s řetězci na příkladu (Obrázek 21).

```
Dim x As Integer
x = 10 + 5 * 2 ^ 3 / 4
'Výsledek x bude 20
```

Obrázek 21 – Příklad priority operací operátorů

Konečně, čtvrtá část textu zahrnuje převody mezi datovými typy. Jsou zde uvedeny funkce převodu, které umožňují konverzi mezi různými datovými typy, například z řetězce na číslo.

Hlavní cíle lekce:

- Seznámení se s proměnnými a jejich využitím
- Seznámení se se základními datovými typy
- Seznámení se s aritmetickými operátory
- Seznámení se se spojovacími operátory
- Seznámená se s převody mezi datovými typy a funkcemi pro převod

Úkoly lekce:

- Vytvoření převodová funkce s využitím operátorů
- Vytvoření funkce s výpočetním vzorcem a konstantou
- Vytvoření funkce používající datový typ *Boolean*
- Vytvoření funkce s převodem datových typů

## 4.4 Lekce 3 – Větvení a podmíněné příkazy

Lekce se zabývá konceptem větvení a podmíněných příkazů při tvorbě algoritmů. V textu je popsána potřeba vývojářů rozdělit kód na různé cesty v závislosti na splnění určitých podmínek. V případě splnění podmínky je vybrána příslušná cesta, zatímco ostatní jsou ignorovány a později jsou spojeny.

```
Function VysledekZBodu(Optional pocetBodu = 100)
' Určí výsledek podle dosaženého počtu bodů
Select Case pocetBodu
    Case Is > 100
        VysledekZBodu = "Exceloval"
    Case 50 To 100
        VysledekZBodu = "Uspěl"
    Case Else
        VysledekZBodu = "Neuspěl"
End Select
```

Obrázek 22 – Ukázka *Select Case* příkazu

Popsány jsou podmíněné příkazy *If*, *Else* a *Else If*, porovnávací operátory pro logické vyhodnocení podmínek, logické operátory (*AND*, *OR*, *XOR* a *NOT*), a také příkaz *Select Case* pro alternativní způsob větvení algoritmu (Obrázek 22). Dále vysvětluje zjednodušení vytváření podmíněných příkazů použitím logických funkcí *Iif*, *Switch* a *Choose*.

Hlavní cíle lekce:

- Představení větvení
- Použití příkazů *If*, *Else* a *Else If*
- Použití podmínek
- Seznámení se s porovnávacími operátory a logickými operátory
- Použití příkazu *Select Case* a jeho speciálních operátorů *Is* a *To*
- Seznámení se s podmíněnými funkcemi *Switch*, *Iif* a *Choose*

Úkoly lekce:

- Vytvoření první funkce využívající podmínky
- Vytvoření funkce s příkazem *Select Case*
- Vytvoření funkce s kombinací několika podmínek
- Vytvoření výpočetní funkce se složitější logikou

## 4.5 Lekce 4 – Kalendářní a časové hodnoty

Lekce se zaměřuje na způsoby práce s datem a časem ve VBA, včetně zaměřením na specializované funkce pro práci s časem a jeho formátování. První část popisuje, jakým způsobem lze přiřadit čas a datum do proměnné s datovým typem *Date*. Další části se zaměřují na funkce pro získání aktuálního času v sešitě a na funkce počítání s časem, například přičítáním počtu týdnů a hledání rozdílu, jak je na uvedeném Obrázek 23.

```
Dim datum As Date
datum = #2/17/2001#
' o 8 týdnů později
noveDatum = DateAdd("ww", 8, datum)

' Spočítá počet týdnů mezi daty
pocetTydnu = DateDiff("ww", datum, noveDatum, vbMonday, vbFirstJan1)

' Zobrazí 8
MsgBox (pocetTydnu)
```

Obrázek 23 – Ukázka z příkladu užití funkcí *DateAdd* a *DateDiff*

Hlavní cíle lekce:

- Představení kalendářních hodnot a jejich princip zápisu v kódu
- Představení časových hodnot a jejich princip zápisu v kódu
- Použití aritmetických operátorů pro počítání s časem
- Seznámení se se získávkami funkcemi z formátů hodnot času
- Použití serializace kalendářních hodnot
- Seznámení se s problémem pojetí kalendáře ve VBA pro jiné země a jeho řešením
- Seznámení se s rozšiřujícími funkcemi pro práci s datem

Úkoly lekce:

- Vytvoření makra pro výpočet časového úseku
- Vytvoření funkce s využitím aktuálního času
- Vytvoření funkce se serializací datumu
- Vytvoření funkce s využitím funkcí dnů v týdnu

## 4.6 Lekce 5 – Opakování, cykly a pole

Lekce pojednává o různých způsobech opakování bloků kódu v programovacím jazyce VBA. Představeny jsou všechny vyskytující se typy cyklů ve VBA, zahrnující příkazy *For*, *For Each*, *While*, *Do While*, *Do Until*. V rámci další sekce jsou vysvětleny vnořené cykly a jak cyklus opustit, před jeho ukončením (Obrázek 24). Konec lekce tvoří představení polí a vícedimenzionálních polí, společně s vysvětlením rekurzivních volání funkcí.

```
' Počítání do 20
Do Until (i > 20)
    MsgBox (i)

    ' Pokud má i hodnotu 15 - ukončí se cyklus
    If (i = 15) Then
        Exit Do
    End If

    i = i + 1
Loop
```

Obrázek 24 – Syntaxe Do While s předčasným ukončením

Hlavní cíle lekce:

- Seznámení se s logikou cyklů
- Použití příkazu *While* a *Wend*
- Použití cyklu *For* a jeho výhody
- Použití cyklu *Do* a jeho variant
- Seznámení se s vnořenými cykly a příkazem opuštění cyklu
- Použití rekurzivních funkcí
- Použití pole a vícedimenzionálních polí
- Použití cyklu *For Each* a seznámení se s jeho přístupem k datům

Úkoly lekce:

- Vytvoření funkce s využitím vnořených cyklů
- Vytvoření funkce s polem
- Manipulace s polem s využitím cyklů

## 4.7 Lekce 6 – Práce s oblastmi výběru

Tato lekce se zaměřuje již na propojení jazyka VBA s tabulkovým softwarem MS Excel. Přesněji, vysvětluje možnosti aktivního označení buněk a oblastí pomocí objektu *ActiveCell* a *Selection*, včetně čtení a zapisování do těchto buněk. Dále lekce navazuje vysvětlením přístupu pasivního čtení listů pomocí příkazu *Range* a možností nakopírováním hodnot přímo do nově vytvořeného pole (Obrázek 25).

```
' Čtení dat jako pole
Dim Data As Variant
Data = Range("List1!A1:A10").Value

' Průchod polem
For Each Item In Data
    MsgBox (Item)
Next Item

' Ukázka zápisu s pomocí Array
Range("A1:D1") = Array("Titul Před Jménem", "Jméno", "Příjmení", "Titul za jménem")

' Ukázka čtení různých buněk s pomocí Array
hlavicka = Array(Range("A1"), Range("B1"), Range("C1"), Range("D1"))
```

Obrázek 25 – Příklad získání hodnot oblasti jako pole

Hlavní cíle lekce:

- Seznámení se s objekty aktivního označení dat v MS Excel
- Čtení a zápis z objektů aktivního označení, posun objektů aktivního označení
- Seznámení se s objekty pasivního přístupu k datům v MS Excel
- Použití polí pro rychlý přepis dat z tabulek v MS Excel
- Seznámení se s vnitřním výběrem vybraných dat a příkazy
- Použití odsazení výběru a rozšíření
- Vybrání celých řádků a sloupců v sešitě
- Manipulace (smazání, odstranění, přidání, skrytí) s řádky a sloupci v sešitě

Úkoly lekce:

- Vytvoření makra s přidáním nových sloupců
- Vytvoření několika maker s ovládáním funkce *Offset* a zapisováním hodnot
- Vytvoření makra, které zkopíruje hodnoty z jiného sloupce a upraví je

## 4.8 Lekce 7 – Stylování textu a buněk

V MS Excel lze graficky stylovat jednotlivé buňky sešitu. Lekce nejdříve seznamuje čtenáře s klauzulí *With*, která zjednodušuje změnu (nejen) grafických vlastností buněk (Obrázek 26). Lekce následně navazuje na jednotlivé vlastnosti a jak je měnit, například výšku a šířku buňky, zalamování textu v buňce, ohraničení buněk, slučování a rozdělení buněk, barvu a nastavení fontu buňky a další. Na konci se lekce věnuje formátování v MS Excel.

```
' Horní ohraničení
With Range("C12").Borders(xlEdgeTop)
    .Weight = xlThick
    .LineStyle = xlDash
    .ColorIndex = 3
End With

' Dolní ohraničení
With Range("C12").Borders(xlEdgeBottom)
    .Weight = xlThick
    .LineStyle = xlDash
    .ColorIndex = 4
End With
```

Obrázek 26 – Příklad využití klauzule *With* pro částečné ohraničení buňky

Hlavní cíle lekce:

- Seznámení se s příkazem *With*
- Nastavení šířky a výšky buněk v sešitě
- Nastavení zalomení textu v buňce
- Nastavení ohraničení buněk v sešitě
- Nastavení sloučení a rozdělení buněk v sešitě
- Nastavení barvy pozadí buňky
- Nastavení fontu buňky a jeho vlastností
- Seznámení se s principem formátování buněk a hodnot v MS Excel pomocí VBA
- Nastavení uzamknutí buňky a skrytí vzorců uvnitř buňky

Úkoly lekce:

- Vytvoření makra pro nastavení určité barvy v označené buňce
- Vytvoření makra, které vybrané buňce změní formát
- Vytvoření makra nastavující styl fontu a otočení textu v buňce
- Vytvoření makra upravující ohraničení buňky

## 4.9 Lekce 8 – Práce se sešitem

Lekce je věnována objektům *Worksheet*, *Sheet*, přes které lze měnit vlastnosti listů v sešitě. Vysvětleny jsou koncepty označování aktivního listu, procházení kolekce listů, zamykání a odemykání listů pomocí hesla (Obrázek 27), zobrazení a skrytí listů, přidávání nových listů a kopírování listů. Na závěr materiál popisuje události, které se mohou v listech vyskytovat a jak je napojit na běžný kód.

```
' Zápis zamknutí pouze s heslem
Worksheets(1).Protect ("heslo")
' Alternativně
Worksheets(1).Protect Password:="heslo"

' Ostatní parametry funkce Protect
' DrawingObjects, Contents, Scenarios, UserInterfaceOnly,
' AllowFormattingCells, AllowFormattingColumns, AllowFormattingRows,
' AllowInsertingColumns, AllowInsertingRows, AllowInsertingHyperlinks,
' AllowDeletingColumns, AllowDeletingRows, AllowSorting, AllowFiltering,
' AllowUsingPivotTables
```

Obrázek 27 – Příklad zamykání listů s pomocí hesla

Hlavní cíle lekce:

- Použití přístupu k listu sešitu
- Seznámení se s procházením listů sešitu
- Nastavení zamykání a odemykání listů s pomocí hesla
- Seznámení se s vlastnostmi skrytí listů
- Seznámení se a použití událostí v sešitě

Úkoly lekce:

- Vytvoření vlastního makra s uzamčením aktivního listu pod svým heslem
- Vytvoření nového makra s novým listem s hlavičkou nové tabulky
- Vytvoření makra, které skryje před uživateli nově vytvořený list
- Přidání reaktivního kódu na základě vzniklé události v sešitě

## 4.10 Lekce 9 – Práce s grafy

Lekce se zaměřuje na jednu z hlavních funkcí programu Excel – tvorbu grafů. Nejdříve popisuje, jak vytvořit na základě určitých dat nový jednoduchý graf s pomocí označení *ActiveChart* (Obrázek 28). Následně trochu rozvinutěji popisuje možné prvky grafu, jako například nadpis, popisky os, legendy a tabulky dat. Lekce je zakončena vytvořením samostatného grafu na novém prázdném listu.

```
Sub VytvorGraf()  
  
    ' Graf Prodeje a Nákladů  
  
    ' Vybrání listu pro graf  
    Sheets("ListGrafu").Select  
  
    ' Vytvoří prázdný graf na určené pozici  
    ' Přidání jména pro pozdější identifikaci  
    Sheets("ListGrafu").Shapes.AddChart  
  
    ' Přejmenování grafu pro další reference  
    Sheets("ListGrafu").Shapes( _  
        Sheets("ListGrafu").Shapes.Count _  
    ).Name = "GrafProdejeNakladu"  
  
    ' Vybrání grafu pro ActiveChart  
    Sheets("ListGrafu").Shapes("GrafProdejeNakladu").Select  
  
    ' Určení typu grafu  
    ActiveChart.ChartType = xl3DColumn  
  
    ' Přidání dat ke grafu  
    ActiveChart.SetSourceData Source:=Range("DataGrafu!B4:D16")  
  
End Sub
```

Obrázek 28 – Příklad vytvoření jednoduchého grafu

Hlavní cíle lekce:

- Vytvoření grafu pomocí kódu
- Vložení grafu na list sešitu v MS Excel
- Manipulace s prvky grafu a nastavení vlastností grafu
- Přidání grafu jako list sešitu

Úkoly lekce:

- Vytvoření makra s vytvořením sloupcovým grafem
- Rozšíření makra o přidání nadpisu ke grafu
- Vytvoření makra s vytvořením dvou řadového bodového grafu
- Rozšíření makra o přidání legendy a tabulky dat ke grafu
- Vytvoření nového makra s novým listem sešitu zaměřeným na zkopírovaný graf

## 4.11 Lekce 10 – Třídy a objekty

Lekce nejdříve vysvětluje koncept objektového programování a co to vlastně objekty a třídy ve VBA jsou a jak fungují. Následně se lekce zaměřuje na používání proměnných s objekty a příkazem *Set*. Nechybí ani popis vytvoření vlastních tříd a objektů (Obrázek 29). Vysvětleny jsou i události objektu a kolekce, které umožňují objekty uchovávat podobně jako pole.

---

```
' Třída pro reprezentaci zaměstnance
Public Jmeno As String
Public Prijmeni As String
Public HodinovaMzda As Double

' Konstruktor třídy
Public Sub Initialize(Jmeno_ As String, Prijmeni_ As String, HodinovaMzda_ As Double)
    Jmeno = Jmeno_
    Prijmeni = Prijmeni_
    HodinovaMzda = HodinovaMzda_
End Sub

' Metoda pro výpočet hrubé mzdy
Public Function VypocetHrubeMzdy(odpracovaneHodiny As Double) As Double
    VypocetHrubeMzdy = odpracovaneHodiny * HodinovaMzda
End Function
```

---

Obrázek 29 – Příklad vytvářené třídy *Zamestnanec*

Hlavní cíle lekce:

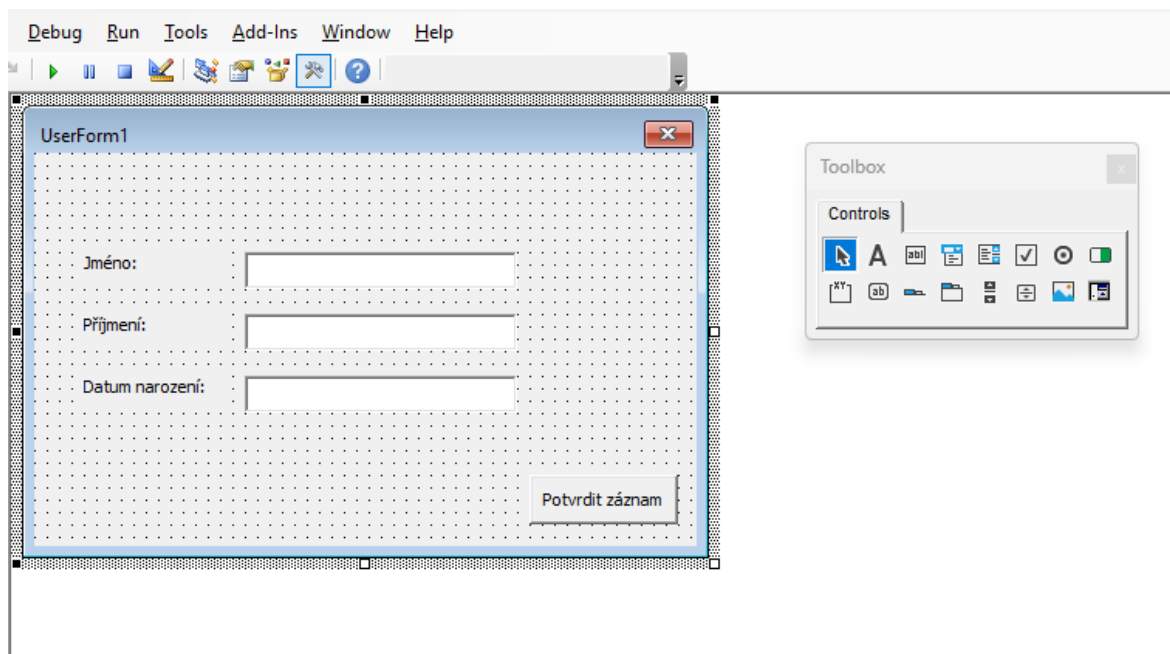
- Seznámení se s objekty a přístupem k nim pomocí příkazu *Set*
- Vytvoření vlastní třídy a objektů
- Vytvoření vlastních metod a vlastností objektu
- Seznámení se s kolekcemi objektů

Úkoly lekce:

- Vytvoření makra, které vytvoří nový objekt *Worksheet*
- Vytvoření vlastní třídy
- Přidání vlastností do vlastní třídy
- Přidání metod do vlastní třídy
- Vytvořit metodu s kolekcí objektů vlastní třídy

## 4.12 Lekce 12 – Práce s uživatelskými dialogy a formuláři

V úvodu se lekce zaměřuje na funkce *MsgBox* a *InputBox*, které umožňují velmi rychle interagovat s uživatelem Excelu a získat tak důležitá rozhodnutí pro funkce a makra. V další části lekce popisuje ovládací prvky pro vývoj vlastních Windows formulářů ve VBE, všechny prvky, které lze ve formulářích využít. Nakonec se lekce zabývá designováním formulářů a propojením prvků s funkcemi v kódu.



Obrázek 30 – Příklad tvorby formuláře

Hlavní cíle lekce:

- Seznámení se s prvky uživatelských dialogů *MsgBox* a *InputBox*
- Seznámení se s ovládací prvky pro tvorbu formulářů
- Seznámení se s prvky formuláře a jejich využitím
- Použití propojení kódu VBA a designu oken

Úkoly lekce:

- Vytvoření makra s užitím *MsgBox* a odpovědi uživatele
- Vytvoření nového makra s užitím *InputBox*
- Vytvoření nového formuláře, který přidá nový řádek do tabulky zákazníci
- Vytvoření reakčního kódu formuláře při vložení nových dat

### 4.13 Lekce 13 – Práce s textovými soubory

V závěrečné lekci je popsán přístup k souborovému systému skrz programovací jazyk VBA, jak je možné přečíst například .csv soubory a jak je otevřít přímo v programu MS Excel. Zároveň lekce ukazuje způsob exportu sešitu jako souboru PDF a představuje způsob, jak smazat soubory pomocí VBA.

```
' Export PDF
Sub ExportPDF()
    Dim FilePath As String
    Dim PDFFileName As String

    ' Cesta k souboru a název PDF souboru
    FilePath = "C:\Users\Cesta\Pro\Uložení\"
    PDFFileName = "exportovana_tabulka.pdf"

    ' Exportování sešitu do PDF
    ActiveSheet.ExportAsFixedFormat Type:=xlTypePDF, _
        Filename:=FilePath & PDFFileName, Quality:=xlQualityStandard, _
        IncludeDocProperties:=True, IgnorePrintAreas:=False, OpenAfterPublish:=False

    MsgBox ("Sešit byl úspěšně exportován do PDF.")
End Sub
```

---

Obrázek 31 – Ukázka makra pro exportování do PDF

Hlavní cíle lekce:

- Seznámení se s přístupem k souborovému systému
- Otevření souboru do VBA
- Export dat a uložení do souboru
- Smazání souboru

Úkoly lekce:

- Vytvoření makra s přečtením csv souboru
- Vytvoření makra s exportováním sešitu jako PDF
- Vytvoření makra, které smaže vyexportovaný soubor

## ZÁVĚR

Cílem této bakalářské práce bylo poskytnout podporu pro získání základních znalostí a dovedností pro práci s tímto jazykem. Vytvořený průvodce a vytvořené materiály byly primárně určeny pro studenty předmětu Kancelářský Software II na Fakultě aplikované informatiky ve Zlíně.

Snahou této bakalářské práce bylo rozšíření procesu učení se s jazykem Visual Basic pro Aplikace skrz interaktivního multimediálního průvodce. V rámci této práce byl zároveň vytvořen multimediální průvodce, který se zaměřuje na základy práce ve VBA pro MS Excel.

Teoretická část tvořili dvě hlavní kapitoly. První kapitola, s názvem Visual Basic pro Aplikace, je věnována historii jazyka, prostředí, ve kterém běží, jeho ostatním verzím z rodiny Visual Basic, na které mohou uživatelé narazit, a možném budoucím rozvoji. Kapitola zároveň rozebírá základní koncepty, které jazyk využívá.

Druhá kapitola, s názvem Nástroje k tvorbě aplikace se věnovala použitým softwarovým nástrojům pro tvorbu aplikace multimediálního průvodce. V kapitole byly popsány použité aplikační frameworky a nástroje pro vytvoření videí.

V praktická části jsou popsány některé složitější procesy tvorbu aplikace včetně popisu procesu střihání a nahrávání videí pro průvodce. Druhá vytvořená kapitola s názvem Studijní materiály popsala jednotlivě vytvořené lekce a úkoly.

Do budoucna by tato práce mohla otevřít možnost rozvíjet jiné možnosti pro práci s automatizací v aplikaci Excel jako například pomocí OfficeScript a jiných alternativ. Autorovou snahou bylo co nejlépe přiblížit problematiku programovacího jazyka VBA. Videá byla tvořena dle intuice a stejně tak hlavní textový výstupní materiál.

**SEZNAM POUŽITÉ LITERATURY**

- [1] JERABEK, Alex, Linda CAPUTO a Kim BRANDL. Office VBA Reference. In: *Microsoft Learn* [online]. Seattle: Microsoft, 2023 [cit. 2023-04-11]. Dostupné z: <https://learn.microsoft.com/en-us/office/vba/api/overview/>
- [2] MICROSOFT. ACC: Visual/Access Basic Is Both a Compiler and an Interpreter. In: *Wayback Machine: Microsoft Support* [online]. Redmond, Washington: Internet Archive, 2012 [cit. 2023-08-18]. Dostupné z: <https://web.archive.org/web/20121021000129/http://support.microsoft.com/kb/109382>
- [3] Extending product function with VBA: Automating System Architect. In: *SA help* [online]. Mission Hills (California): UNICOM Systems, 2023 [cit. 2023-08-20]. Dostupné z: [https://support.unicomsi.com/manuals/systemarchitect/11482/starthelp.html#page/Extending\\_product\\_function\\_with\\_VBA%2FExtendMSVBAforApps.1.002.html%23](https://support.unicomsi.com/manuals/systemarchitect/11482/starthelp.html#page/Extending_product_function_with_VBA%2FExtendMSVBAforApps.1.002.html%23)
- [4] About Developing Applications With VBA. In: *AutoCAD 2023 Developer and ObjectARX Help* [online]. San Francisco: Autodesk, 2023 [cit. 2023-08-20]. Dostupné z: <https://help.autodesk.com/view/OARX/2023/ENU/?guid=GUID-2CD40631-D67B-4DF0-A2C4-606E9B613252>
- [5] VBA and SOLIDWORKS x64. In: *Solidworks API HELP* [online]. Aachen: Dassault Systèmes Solidworks, 2023 [cit. 2023-08-20]. Dostupné z: [https://help.solidworks.com/2021/english/api/sldworksapiproguide/Overview/VBA\\_and\\_SolidWorks\\_x64.htm](https://help.solidworks.com/2021/english/api/sldworksapiproguide/Overview/VBA_and_SolidWorks_x64.htm)
- [6] Reflection Desktop. In: *Reflection Desktop* [online]. Waterloo (Ontario): opentext, 2023 [cit. 2023-08-20]. Dostupné z: <https://www.microfocus.com/en-us/products/reflection-desktop/overview>
- [7] Using VBA Macros in CorelDRAW. In: *Corel* [online]. Ottawa (Ontario): Corel, 2023 [cit. 2023-08-22]. Dostupné z: <https://kb.corel.com/en/128155>

- [8] Support Statement for Visual Basic 6.0 on Windows. In: *Microsoft Learn* [online]. United States: Microsoft, 2023 [cit. 2023-04-11]. Dostupné z: <https://learn.microsoft.com/en-us/previous-versions/visualstudio/visual-basic-6/visual-basic-6-support-policy>
- [9] Visual Basic documentation. In: *Microsoft Learn* [online]. United States: Microsoft, 2023 [cit. 2023-05-26]. Dostupné z: <https://learn.microsoft.com/en-us/dotnet/visual-basic/>
- [10] DOLLARD, Kathleen a Bill WAGNER. Annotated Visual Basic language strategy. In: *Microsoft Learn* [online]. United States: Microsoft, 2023 [cit. 2023-04-11]. Dostupné z: <https://learn.microsoft.com/en-us/dotnet/visual-basic/getting-started/strategy>
- [11] Visual Basic for Applications: Discontinuation of the VBA Licensing Program. In: *Microsoft Learn* [online]. United States: Microsoft, 2023 [cit. 2023-05-03]. Dostupné z: [https://learn.microsoft.com/en-us/previous-versions/msdn10/bb190538\(v=msdn.10\)](https://learn.microsoft.com/en-us/previous-versions/msdn10/bb190538(v=msdn.10))
- [12] YUNDT, Brad. Why has Microsoft stopped supporting VBA? And what language will take its place?. In: *Quora* [online]. California: Quora, 2023 [cit. 2023-05-10]. Dostupné z: <https://www.quora.com/Why-has-Microsoft-stopped-supporting-VBA-And-what-language-will-take-its-place>
- [13] JERABEK, Alex, Elizabeth SAMUEL a Keyur PATEL. Differences between Office Scripts and VBA macros. In: *Microsoft Learn* [online]. United States: Microsoft, 2023 [cit. 2023-05-10]. Dostupné z: <https://learn.microsoft.com/en-us/office/dev/scripts/resources/vba-differences>
- [14] JERABEK, Alex, Elizabeth SAMUEL, Sudhi RAMAMURTHY a Nancy WANG. Office Scripts Code Editor environment. In: *Microsoft Learn* [online]. United States: Microsoft, 2023 [cit. 2023-05-10]. Dostupné z: <https://learn.microsoft.com/en-us/office/dev/scripts/overview/code-editor-environment>
- [15] JERABEK, Alex, Linda CAPUTO, Kim BRANDL a Saisang CAI. Visual Basic for Applications: Visual Basic conceptual topics. In: *Microsoft Learn* [online]. Redmond, Washington: Microsoft, 2023 [cit. 2023-08-22]. Dostupné z:

- <https://learn.microsoft.com/en-us/office/vba/language/reference/user-interface-help/visual-basic-conceptual-topics>
- [16] CANNON, Linda Lu, Alex JERABEK, Nick SCHONNING, Kim BRANDL, Linda CAPUTO a Saisang CAI. VBE Glossary: procedure. In: *Microsoft Learn* [online]. Seattle: Microsoft, 2023 [cit. 2023-04-24]. Dostupné z: <https://learn.microsoft.com/en-us/office/vba/language/glossary/vbe-glossary#procedure>
- [17] CANNON, Linda Lu, Alex JERABEK, Kim BRANDL, Saisang CAI a Linda CAPUTO. Function statement. In: *Microsoft Learn* [online]. Seattle: Microsoft, 2023 [cit. 2023-04-25]. Dostupné z: <https://learn.microsoft.com/en-us/office/vba/language/reference/user-interface-help/function-statement>
- [18] CANNON, Linda, Alex JERABEK, Kim BRANDL a Saisang CAI. Static statement. In: *Microsoft Learn* [online]. Seattle: Microsoft, 2023 [cit. 2023-04-26]. Dostupné z: <https://learn.microsoft.com/en-us/office/vba/language/reference/user-interface-help/static-statement>
- [19] DOLLARD, Kathleen, Kent SHARKEY, Youssef VICTOR, Genevieve WARREN, Maira WENZEL, Nick SCHONNING, Luke LATHAM a Steve HOAG. Objects and classes in Visual Basic. In: *Microsoft Learn: Objects and classes - Visual Basic* [online]. Redmond, Washington: Microsoft, 2023 [cit. 2023-08-20]. Dostupné z: <https://learn.microsoft.com/en-us/dotnet/visual-basic/programming-guide/language-features/objects-and-classes/>
- [20] Tauri-apps/tauri: Build smaller, faster, and more secure desktop applications with a web frontend. In: *GitHub* [online]. GitHub, 2023 [cit. 2023-05-26]. Dostupné z: <https://github.com/tauri-apps/tauri>
- [21] Introduction. In: *SvelteKit: Docs* [online]. 2023 [cit. 2023-05-20]. Dostupné z: <https://kit.svelte.dev/docs/introduction>
- [22] DELANEY, Jeff. Rust in 100 Seconds. In: *YouTube* [online]. Mountain View (California): Google, 2021 [cit. 2023-06-22]. Dostupné z: [https://www.youtube.com/watch?v=5C\\_HPTJg5ek](https://www.youtube.com/watch?v=5C_HPTJg5ek)

- [23] Lekce 1 - Úvod do TypeScriptu. In: *Itnetwork.cz* [online]. itnetwork.cz, 2023 [cit. 2023-05-25]. Dostupné z: <https://www.itnetwork.cz/javascript/typescript/uvod-do-typescriptu>
- [24] TAILWIND LABS. Utility-First Fundamentals: Core concepts. In: TAILWIND LABS. *Tailwind CSS* [online]. Galveston (Texas): Tailwind Labs, 2023 [cit. 2023-08-20]. Dostupné z: <https://tailwindcss.com/docs/utility-first>
- [25] Introduction. In: *Skeleton - UI Toolkit for Svelte + Tailwind* [online]. Skeleton Labs, 2023 [cit. 2023-06-19]. Dostupné z: <https://www.skeleton.dev/docs/introduction>
- [26] About. In: *Node.js* [online]. San Francisco: OpenJS Foundation, 2023 [cit. 2023-05-20]. Dostupné z: <https://nodejs.org/en/about>
- [27] About npm. In: *Npm Docs* [online]. 2023 [cit. 2023-05-20]. Dostupné z: <https://docs.npmjs.com/about-npm>
- [28] CONE, Matt. *Markdown Guide* [online]. New Mexico, 2023 [cit. 2023-05-26]. Dostupné z: <https://www.markdownguide.org/getting-started/>
- [29] *Open Broadcaster Software | OBS* [online]. Lain, 2023 [cit. 2023-08-20]. Dostupné z: <https://obsproject.com/cs>
- [30] *Audacity*® | *Free, open source, cross-platform audio software for multi-track recording and editing.* [online]. Audacity Team, 2023 [cit. 2023-08-20]. Dostupné z: <https://www.audacityteam.org/>
- [31] Professional video editing software | Adobe Premiere Pro. In: *Adobe: Creative, marketing and document management solutions* [online]. San Jose (California): Adobe, 2023 [cit. 2023-08-20]. Dostupné z: <https://www.adobe.com/products/premiere.html>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

VBA	Visual Basic pro Aplikace
VBE	Visual Basic Editor
VB	Visual Basic
IDE	Integrované vývojářské prostředí
SDK	Sada vývojářských nástrojů
COM	Component Object Model
HTML	Hypertext markup language
PDF	Portable Document Format
JS	JavaScript
CSS	Cascading Style Sheet
API	Application programming interface
npm	Node Package Manager
CLI	Rozhraní příkazového řádku

**SEZNAM OBRÁZKŮ**

Obrázek 1 – Prázdný list Microsoft Excel .....	11
Obrázek 2 – Základní rozložení Visual Basic Editoru .....	12
Obrázek 3 – Struktura projektu VBA .....	16
Obrázek 4 – Ukázka definice funkce .....	18
Obrázek 5 – Priorita výpočtu .....	21
Obrázek 6 – Syntaxe příkazů <i>If</i> , <i>Else</i> a <i>Else If</i> .....	23
Obrázek 7 – Syntaxe příkazu <i>Select Case</i> .....	23
Obrázek 8 – Syntaxe cyklu <i>While</i> .....	24
Obrázek 9 – Syntaxe cyklu <i>For</i> .....	24
Obrázek 10 – Syntaxe cyklu <i>Do While</i> .....	25
Obrázek 11 – Syntaxe cyklu <i>Do Until</i> .....	25
Obrázek 12 – Příklad rekurzivní funkce .....	25
Obrázek 13 – Diagram tříd, objektů a jejich vlastností a metod.....	26
Obrázek 14 – Rozhraní aplikace multimediálního průvodce.....	32
Obrázek 15 – Naprogramovaná funkce <i>read_meta_menu_data</i> v Rust.....	33
Obrázek 16 – Asynchronní funkce <i>load</i> v <i>+page.ts</i> .....	34
Obrázek 17 – Funkce <i>read_file_as_base64</i> v Rust .....	34
Obrázek 18 – Aplikování variant <i>media query</i> uživatelských tříd.....	35
Obrázek 19 – Nastavení pásu karet v MS Excel pro přístup k VBA.....	38
Obrázek 20 – Ukázka z příkladu funkce s použitím parametru <i>Optional</i> .....	39
Obrázek 21 – Příklad priority operací operátorů .....	40
Obrázek 22 – Ukázka <i>Select Case</i> příkazu .....	41
Obrázek 23 – Ukázka z příkladu užití funkcí <i>DateAdd</i> a <i>DateDiff</i> .....	42
Obrázek 24 – Syntaxe <i>Do While</i> s předčasným ukončením.....	43
Obrázek 25 – Příklad získání hodnot oblasti jako pole .....	44
Obrázek 26 – Příklad využití klauzule <i>With</i> pro částečné ohraničení buňky.....	45
Obrázek 27 – Příklad zamykání listů s pomocí hesla .....	46
Obrázek 28 – Příklad vytvoření jednoduchého grafu .....	47
Obrázek 29 – Příklad vytvářené třídy <i>Zamestnanec</i> .....	48
Obrázek 30 – Příklad tvorby formuláře .....	49
Obrázek 31 – Ukázka makra pro exportování do PDF .....	50

**SEZNAM TABULEK**

Tabulka 1 – Ukázka reprezentace dat se stejnou hodnotou .....	19
Tabulka 2 – Základní datové typy ve VBA .....	20
Tabulka 3 – Aritmetické operátory .....	21
Tabulka 4 – Spojovací operátory .....	21
Tabulka 5 – Porovnávací operátory .....	22
Tabulka 6 – Logické operátory .....	22

## SEZNAM PŘÍLOH

Příloha P I: CD s bakalářskou prací a soubory, obsahující studijní materiály v PDF, ukázky kódu, ukázky řešených úloh se zamknutým řešením, videa, finální program a zdrojové kódy