

Programové vybavení pro porovnávání dokumentů

Program equipment for comparison
of documents

Josef Moravčík

Bakalářská práce
2011



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

*** nescannované zadání str. 1 ***

*** nescannované zadání str. 2 ***

ABSTRAKT

Abstrakt česky

Tato práce je zaměřena na porovnávání dokumentů a k následnému zrychlení hledání plagiátů mezi dokumenty. Je zde využit programovací jazyk VBA, který je obsažen v každé sadě programů MS Office. V této práci byl využit MS Word, který rovněž obsahuje VBA. Program zde byl napsán ve formě makra pro MS Word. MS Word byl vybrán z důvodu, že je textovým editorem a obsahuje řadu funkcí pro práci s textem.

V teoretické části je popsán jazyk VBA a jeho hlavní prvky při programování. V praktické části pak postup při ovládní programu a také podrobně popsána jeho programová část. Vše je názorně zdokumentováno obrázky.

Klíčová slova: VBA, MS Office, MS Word

ABSTRACT

Abstrakt ve světovém jazyce

This work is focused on comparing documents and then to accelerate the search for the counterfeit documents. There is a programming language used by VBA, which is included in each set of Office. In this work, we used MS Word, which also included VBA. The programm here was written in the form of macros for MS Word. MS Word was chosen because it is a text editor and contains a number of functions for working with text.

The theoretical part describes VBA and its major elements in the programming. The practical part of the process control program and a detailed description of the programm. Everything is clearly documented by pictures.

Keywords: VBA, MS Office, MS Word

Poděkování, motto

Zde bych rád poděkoval svému vedoucímu bakalářské práce panu Ing. Petru Chalupovi, Ph.D. za uvedení do problému a na navedení na správnou cestu jeho vyřešení.

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 a jeden výtisk bude uložen u vedoucího práce;
- 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 s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- 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

.....
podpis diplomanta

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 VBA	11
1.1 PROGRAM.....	11
1.2 DEKLARACE	11
1.2.1 Objekty	11
1.2.1.1 Události.....	12
1.2.1.2 Akce	12
1.2.2 Proměnné.....	12
1.3 PROCEDURY A FUNKCE.....	13
1.4 PODMÍNKY A CYKLY	14
1.4.1 Cykly	14
1.4.1.1 Do – Loop	14
1.4.1.2 While – Wend	14
1.4.1.3 For – Next	15
1.4.2 Podmínky	15
1.4.2.1 If – Then – Else – End If.....	15
1.4.2.2 Select Case	15
1.4.3 With – End With	15
1.5 OPERÁTORY A MATEMATICKÉ OPERACE	16
1.6 OBJEKTY	18
2 PROSTŘEDÍ VBA	19
2.1 NAHRÁVÁNÍ MAKER	20
2.2 PROGRAMOVÁNÍ.....	21
3 ALGORITMY PRO VYHLEDÁVÁNÍ V TEXTU	22
3.1 NAIVNÍ ALGORITMUS	22
3.2 KNUTH-MORRIS-PRATTŮV ALGORITMUS	22
3.3 QUICKSEARCH.....	23
II PRAKTICKÁ ČÁST	25
4 PROGRAM POROVNÁVÁNÍ	26
5 POPIS PROGRAMU PRO UŽIVATELE	27
5.1 SLOŽKA PROGRAM	27
5.2 SLOŽKA TEST	28
5.3 SLOŽKA VÝSLEDKY.....	28
6 POPIS PROGRAMU PRO PROGRAMÁTORY	30
6.1 DOKUMENT POROVNÁVÁNÍ.....	30
6.1.1 Porov1	30

6.1.2	Porovnej	31
6.1.3	Tabulka.....	33
6.1.4	Obrazek	34
6.1.5	Aktualizovat_postup	35
6.1.6	Nacti_data.....	36
6.1.7	Formuláře	37
6.2	DOKUMENT VÝSLEDKY	42
6.2.1	Formulář šablony výsledků	44
ZÁVĚR		46
CONCLUSION		47
SEZNAM POUŽITÉ LITERATURY.....		48
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK		49
SEZNAM OBRÁZKŮ		50
SEZNAM TABULEK.....		51
SEZNAM PŘÍLOH.....		52

ÚVOD

Počítače a jejich využití proniklo skoro už do všech oborů a mnozí lidé si dnes nedokáží představit život bez počítače a jeho aplikací. Pro vytváření těchto různých aplikací vzniklo, velké množství programovacích jazyků ve kterých se různé programy programují.

Jedním z těchto programovacích jazyků je i Visual Basic pro Aplikace, který vznikl z jazyku Basic. Programovací jazyk VBA patří momentálně mezi ty méně významné programovací jazyky, protože nenabízí tolik možností. Programovací jazyk VB není nijak složitý a dají se s ním programovat jednoduché programy pro usnadnění práce na počítači.

Tento programovací jazyk bývá dnes součástí programového balíku MS Office. Proto se bude skvěle hodit pro naše účely při vytváření programu na odhalování plagiátorství, které je v dnešní době na vzestupu. Na ruční projití a vzájemné porovnání většího množství dokumentů je třeba velkého množství času. Proto byl naprogramován program, který by nám kontrolu většího množství těchto dokumentů jakýmkoliv způsobem usnadnil a zrychlil tak naši práci.

Tato práce by měla pomoci při odhalování plagiátů textových dokumentů a nastítnit dokumenty, které by plagiáty opravdu mohly být. Ty pak budou předneseny uživateli k jeho následnému a důkladnému prozkoumání.

I. TEORETICKÁ ČÁST

1 VBA

Jak již bylo zmíněno v úvodu, jazyk VBA se vyvinul z jazyka Basic. Programovací jazyk VBA vytvořila firma Microsoft a vložila jej do svého balíku Office, který je jedním z nejpoužívanějších programů na světě. VBA zde má plnit úlohu samotného dotvoření a přizpůsobení programového balíku Office na míru uživatele. VBA se tedy využívá k psaní maker, procedur a jednoduchých funkcí.

1.1 Program

Každý program, procedura, makro je jednoduchá sekvence instrukcí, která tvoří ucelenou jednotku, tzn. má svůj začátek a konec. Některé programy se vykonávají rychleji a některé pomaleji v závislosti na velikosti dat a složitosti programu. Proto se snažíme tento programový kód jakýmkoliv způsobem zjednodušit a tím pádem i zefektivnit. K řešení časově náročných operací se dají využít algoritmy, které tyto časové prodlevy zmírní. Tyto algoritmy se ovšem dají použít jen někdy. Volbu vhodného algoritmu provádíme například podle předpokládané velikosti zpracovávaných dat.

1.2 Deklarace

Deklarace přiřazuje datový typ proměnné, je důležitým prvkem každého programování. Slouží ale také k urychlení a zpřehlednění samotného kódu. Deklaraci provádíme vždy na začátku kódu nebo před samotným použitím proměnné. Ve VBA má již několik věcí předdeklarovaných. Každá proměnná reprezentuje určité jednotky paměti. V programovacím jazyce VAB není nutné vždy proměnné deklarovat. Nové proměnné deklarujeme příkazem *Dim* a nové odkazy pro nové objekty příkazem *Set*. U VBA spolu vždy můžou pracovat jen proměnné stejného typu.

1.2.1 Objekty

Objekty jsou prvky, které spojují programovací kód s interními daty.

Objekty mají své vlastnosti, na které se můžeme dále odkazovat. Na co konkrétního se právě odkazujeme, uvádí zpravidla poslední slovo výrazu např.:

Application.Name

kde: *Application* značí objekt

Name je vlastnost objektu, zde vyjadřuje jméno

Objekt *Application* je nadřazen všem ostatním objektům. Pod Objekty také patří Události a Akce. Jedná se o procedury a funkce které provádějí určitou akci.

1.2.1.1 Události

Událost je akce, která se stane v sešitu, listu, na formuláři, s prvkem formuláře (například při kliknutí myši, stisknutí klávesy, aktivace – výběr prvku). Mezi události patří např.:

Activate - při aktivaci provede akci ...

Change - při změně provede akci ...

Open - při otevření provede akci ...

1.2.1.2 Akce

Některé prvky a objekty mají kromě vlastností také další možnosti. Seznam veškerých akcí je dostupný v menu, stejně jako vlastnosti. Akce spojené s daným objektem nebo prvkem se automaticky filtrují a nabízejí při psaní kódu. Patří sem např.:

AddComment – vlož komentář

Copy - kopíruj

Find – najdi

1.2.2 Proměnné

Proměnná je název, ke kterému přiřazujeme určitou hodnotu. Proměnná se deklaruje následujícím způsobem:

Dim nazev_promenne As Integer

kde: *Dim* značí začátek deklarace

As Integer přiřazuje proměnné datový typ, zde se jedná o datový typ číslo.

Při deklarování ve VBA nemusíme předem deklarovat některé proměnné. VBA sám určí, o jaký typ se jedná, a přiřadí typ *Variant*. Přehled standardních datových typů je uveden v Tab.1.

Datový typ	Rozsah hodnot
Byte	0 až 255
Boolean	True nebo False
Integer	-32 768 až 32 767
Long	-2 147 483 684 až 2 147 483 647
Date	1. leden 0100 až 31. prosinec 9999
Object	Libovolný objekt
String	0 až zhruba 2 miliardy
Variant (čísla)	Libovolná číselná hodnota
Variant (znaky)	0 až zhruba 2 miliardy
Uživatelsky definovaný	Podle typu
Range	Definuje proměnnou jako buňku

Tab. 1. Datové typy

1.3 Procedury a funkce

Jako ve většině programovacích jazyků i zde můžeme využít při programování procedur a funkcí. Procedury a funkce by se jinými slovy daly nazvat jako podprogramy. Jejich hlavní využití pak přichází hlavně při psaní velkých a složitých programů. Také se dají využít, pokud by se část programu měla opakovat. Procedury a funkce se dají volat z kterékoliv části programu. Dnešní program už se vlastně bez procedur a funkcí ani neobejdou, protože jsou celé poskládány z menších částí, které spolu pak spolupracují. Funkcím a procedurám lze předávat parametry. Předávané parametry, jsou vlastně proměnné, které obsahují nějaké hodnoty, jež se mají předat a s nimiž pak daná funkce nebo procedura dál pracuje. Procedura nebo funkce může své parametry pozměnit a tím vrátit jiné hodnoty, než byly na jejím vstupu. Každá procedura začíná příkazem *Sub* a končí *End Sub*. Funkce zase začíná

Function a končí *End Function*. Z těmi příkazy pak následuje název procedury nebo funkce a za tímto názvem se v kulatých závorkách nacházejí případné parametry, které se budou předávat. Volání procedur a funkcí se provádí příkazem *Call*.

1.4 Podmínky a cykly

Další důležitou součástí programovacího jazyka, bez které není možné se při programování neobejít jsou podmínky a cykly. VBA rovněž jako spousta jiných programovacích jazyků využívá řadu podmínek a cyklů.

1.4.1 Cykly

Využití cyklů je potřeba zejména při procházení různých seznamů, kolekcí, polí a dokumentů. Víme, že když se vykonává nějaký program tak jdeme postupně jeden řádek kódu za druhým, ale pokud se potřebujeme k něčemu vrátit nebo zopakovat, budeme k tomu potřebovat právě nějaký cyklus. Cykly bývají opatřené nějakou podmínkou. Tato podmínka bývá uvedena buď na začátku nebo na konci cyklu. Bez této podmínky by se cyklus stal nekonečným. Teď uvedeme několik druhů cyklů.

1.4.1.1 *Do – Loop*

U tohoto cyklu může být podmínka uvedena buď na začátku nebo na konci cyklu. Cyklus začíná slovem *Do* a končí slovem *Loop*. Pokud uvedeme podmínku za slovy *Do While*, znamená to, že podmínka bude na začátku. To znamená, že cyklus se bude vykonávat pokud zadaná podmínka bude platit. Pokud platit přestane, cyklus se už znova nevykoná. Jestliže bude podmínka umístěna za slovy *Loop Until*, znamená to dokud podmínka neplatí. Cyklus se tedy bude vykonávat když podmínka nebude platit. Zde také dochází k tomu, že se tento cyklus minimálně 1x vykoná. Z cyklu *Do – Loop* lze vyskočit v kterémkoliv místě cyklu a to pomocí příkazu *Exit do*.

1.4.1.2 *While – Wend*

Tenhle cyklus je úplně stejný jako předchozí cyklus *Do – Loop*. Je však o něco starší a proto se dnes už nepoužívá a upřednostňuje se proto raději cyklus *Do – Loop*.

1.4.1.3 *For – Next*

Tento cyklus se používá například při procházení kolekcí nebo když předem známe počet opakování. Při procházení kolekcí použijeme *For Each – Next*. Příkaz *Each* nám zajistí, že projdeme všechny položky daného seznamu nebo kolekce a pak se cyklus ukončí. Pokud předem známe počet opakování, nastavíme jej pomocí *For - To – Next*. Kde před a za příkazem *To* bude hodnota, na které se má začínat a končit. Pokud bychom chtěly nečekaně ukončit cyklus při jeho běhu můžeme tak učinit pomocí příkazu *Exit For*.

1.4.2 Podmínky

Podmínky nám slouží, pokud chceme část kódu vykonat, nebo přeskočit. Nebo pokud se máme rozhodnout a podle určité podmínky vykonat určitou část kódu. Slouží nám tedy hlavně při rozhodování. Uvedeme si proto dvě hlavní podmínky.

1.4.2.1 *If – Then – Else – End If*

Tato podmínka je asi nejznámější a hlavně nejpoužívanější podmínkou vůbec. Její začátek a konec se značí *If – End If*. Za *If* je vždy uvedena naše podmínka, kterou budeme porovnávat na pravdu. Tato podmínka nemusí být úplná, stačí uvést jen větvi s *Then* a větve pro *Else* už se uvádět nemusí. Pokud chceme však větvit, tak máme vlastně k dispozici dvě varianty, neboli dvě větve našeho kódu.

1.4.2.2 *Select Case*

Podmínka *Select Case* je podobná jako podmínka *If Then*. Zde se však dá větvit do daleko více větví. Podmínka začíná příkazem *Select Case*, za kterým následuje výraz. Výsledek podmínky je buď *True* nebo *False*. Všechny větve pak začínají příkazem *Case*, za kterým je uveden výsledek. Pokud se podmínka bude shodovat s daným výsledkem, tak se provede daná větev. Celou tuto podmínku opět uzavírá příkaz *End Select*.

1.4.3 *With – End With*

Zvláštním případem je *With – End With*. Nejedná se ani o cyklus, ani o podmínku. Uvozuje blok příkazů, které v tečkové notaci začínají stejnou počáteční částí. To vede k zrychlení a zpřehlednění celého kódu. Příkaz začíná *With*, za kterým je stejná část příkazu a končí *End With*. Do těchto bloků lze dále vnořovat další bloky *With – End With*.

1.5 Operátory a matematické operace

Matematické operace a operátory nám slouží pro sčítání, odčítání, zaokrouhlování, násobení, dělení, slučování, rozdělování a porovnávání. Přehled operátorů se nachází v Tab.2.

Operátor	Kategorie	Popis
=	Přiřazení, porovnávání	Přiřazuje proměnné vypočtenou hodnotu nebo vyjadřuje rovnost výrazů na pravé a levé straně.
<	Porovnávání	Určuje, jestli je hodnota výrazu na levé straně menší než hodnota na pravé straně.
>	Porovnávání	Určuje, jestli je hodnota výrazu na levé straně větší než hodnota na pravé straně.
<=	Porovnávání	Určuje, jestli je hodnota výrazu na levé straně menší nebo rovna hodnotě na pravé straně.
>=	Porovnávání	Určuje, jestli je hodnota výrazu na levé straně větší nebo rovna hodnotě na pravé straně.
<>	Porovnávání	Dává výsledek True pokud se výraz na pravé straně nerovná výrazu na levé straně.
+	Aritmetický	Sečte hodnotu výrazu na pravé straně s hodnotou výrazu na levé straně. Může také provést zřetězení textu.
-	Aritmetický	Hodnotu výrazu na pravé straně odečte od hodnoty výrazu na levé straně. Může taky hodnotu na pravé straně změnit na opačnou.
/	Aritmetický	Vydělí hodnotu výrazu na levé straně hodnotou výrazu na pravé straně a vypočte přesný výsledek.
\	Aritmetický	Vydělí hodnotu výrazu na levé straně hodnotou výrazu na pravé straně, vrátí však jen celočíselnou část výsledku.
Mod	Aritmetický	Vydělí hodnotu výrazu na levé straně hodnotou výrazu na pravé straně, vrátí však jen hodnotu zbytku.

^	Aritmetický	Hodnotu výrazu na levé straně umocní na exponent výrazu na pravé straně.
&	Textový	Znakový řetězec na pravé straně připojí ke znakovému řetězci na levé straně.
Like	Textový	Vrací hodnotu True pokud je výraz na levé straně podobný výrazu na pravé straně.
Is	Objektově orientovaný	Vrací hodnotu True pokud se objektový výraz na levé straně odkazuje na stejný objektový výraz na pravé straně.
And	Logický	Vrací True pokud se výraz na levé i pravé straně vyhodnotí na True.
Or	Logický	Vrací True pokud se alespoň jeden výraz na levé nebo pravé straně vyhodnotí na True.
Xor	Logický	Vrací True pokud se výraz na levé nebo pravé straně vyhodnotí na True, nikoli však na obou stranách na True.
Not	Logický	Změní hodnotu logického výrazu na pravé straně na opačnou.
Eqv	Logický	Vrací True pokud se výraz na levé i pravé straně vyhodnotí na True, nebo oba na False.

Tab. 2. Operátory

Mezi matematické operace by se ještě dala zařadit funkce *Round*, která nám dokáže zaokrouhlit výraz na určený počet desetinných míst, nebo příkaz *Randomize*. Příkaz *Randomize* nám slouží jako generátor náhodných čísel.

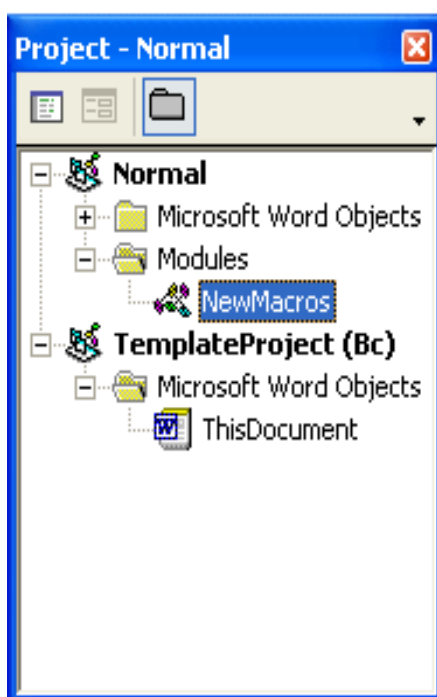
Priorita operátorů je následující. Nejdříve jsou vyhodnocovány aritmetické, pak řetězíci (&), pak srovnávací (=, >= atd.) a nakonec logické (And, Or, Eqv, Xor atd.). Srovnávací mají mezi sebou všechny stejnou prioritu, aritmetické jsou vyhodnocovány zleva doprava s tím, že násobení a dělení má přednost před sčítáním a odčítáním, logické jsou vyhodnocovány zleva doprava. Prioritu operátorů lze ovlivnit závorkami.

1.6 Objekty

Objekty a jejich vlastnosti už byly stručně nastíněny v kapitole deklarace. Objektů a jejich vlastností je ve VBA celá řada a probírat každý z nich by zabralo spoustu času. Proto si zde uvedeme dva takové významnější objekty, což jsou objekt *Range* a objekt *Selection*. Oba tyto objekty slouží k práci s dokumentem, nebo spíš s jeho obsahem. Objekt *Selection* je závislý na kurzoru a jeho pohybu. Dokáže nám označovat a načítat postupně text dokumentu. Také se jím dají zjišťovat vlastnosti označeného textu. Kdežto objekt *Range* nám zase dokáže adresovat v jakémkoliv okamžiku jakoukoliv část dokumentu nebo třeba i celý dokument. Objekt *Range* nepotřebuje ke svému pohybu kurzor. Pokud však chceme označit nějaký text, musíme o tomto textu vědět nějaké informace, které pak předáváme objektu *Range*. Ten nám podle těchto informací dokáže náš text najít a označit. Mezi tyto informace patří například o který odstavec se jedná nebo kolikátým znakem začíná a končí náš text.

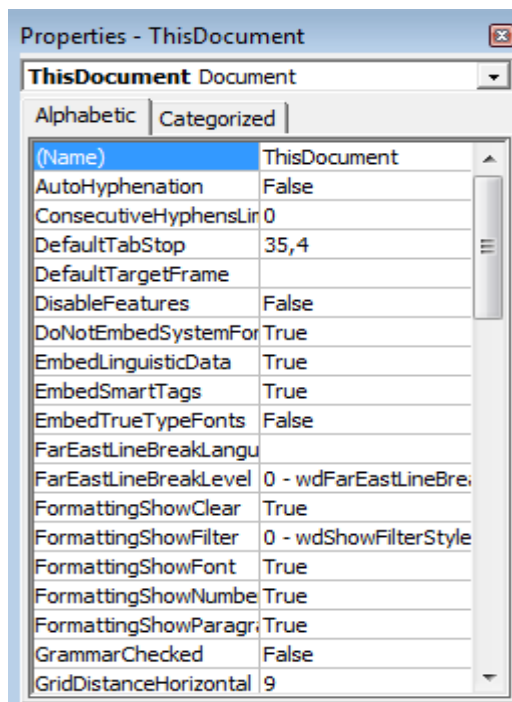
2 PROSTŘEDÍ VBA

Pro napsání kódu pro VBA se dá použít i obyčejného textového editoru, ale třeba k dalšímu testování chyb je tento editor nevhodný. Proto raději budeme využívat editoru poskytovaného v programovém balíku sady Office. Po otevření některého z programů Word, Excel, Access, PowerPoint se do editoru dostaneme buď přes horní menu nebo klávesovou zkratkou **Alt+F11**. Zobrazí se nám hlavní menu a pod ním hlavní okno s vepsaným začátkem a koncem makra. Na levé straně pak zpravidla okno *Project*, ve kterém vidíme náš projekt a také další projekty a formuláře přístupné z našeho makra.



Obr.1. Okno Project

Pod ním zpravidla bývá okno *Properties* s vlastnostmi objektů.



Obr. 2. Okno Properties

Není na škodu si do spodní části obrazovky umístit také okno *Immediate*, které nám zobrazuje průběžné výsledky zvolených proměnných.

Při psaní makra nám automaticky pomáhá knihovna objektů, která nám nabízí dostupné možnosti objektů. Do popředí si můžeme vynést jakékoliv nástroje z hlavního menu. Program je zde také možné otestovat spuštěním příkazu *Run*. Lze si také před spuštěním umístit zářezku, po kterou se nám program bude vykonávat, nebo procházet program řádek po řádku. Editor nám také kontroluje správnou syntaxi zapsaných příkazů a usnadňuje opravování chyb.

2.1 Nahrávání maker

Nahrávání maker je vytváření makra pomocí záznamu. Nahrávání spustíme z hlavního menu (Nástroje>Makro>Záznam nového makra). Po spuštění záznamu makra provádíme úpravy textu a přednastavení, které chceme aby výsledné makro taky provedlo. Pak stačí záznam zastavit a dosavadní provedené úpravy se nám uloží jako nové makro. Při záznamu makra nám dochází k automatickému psaní kódu tohoto makra. Lze také tento kód otevřít a upravit podle potřeby. Otevření provedeme klávesovou zkratkou **Alt+F11**. Spuštění makra může provést přes hlavní menu nebo klávesovou zkratkou **Alt+F8**.

2.2 Programování

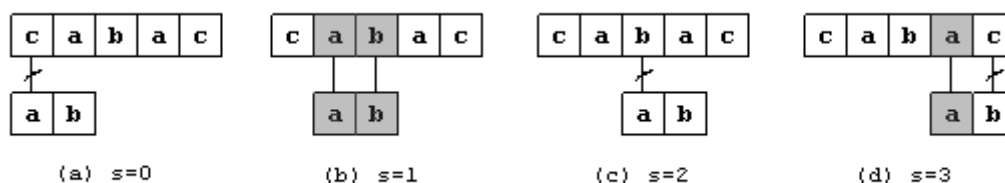
Samotné programování se pak nějak výrazně neliší od programování v ostatních programovacích jazycích. Fungují zde podobná pravidla jako třeba v C++. Procedury se dají napsat k různým částem dokumentu. Můžeme například vložit určitý objekt na pozadí dokumentu a k tomuto objektu přiřadit proceduru. Dále se dá procedura napsat jako modul, v našem případě makro a nebo pro vytvoření nějakého formuláře. To vše najdeme v okně *Project*. Tyto procedury spolu v rámci jednoho dokumentu dokážou komunikovat a volat se navzájem.

3 ALGORITMY PRO VYHLEDÁVÁNÍ V TEXTU

Algoritmy se obecně využívají hlavně pro urychlení vykonávání různých operací a tím zkrácení času potřebného k tomuto vykonávání, které vykonává například počítač nebo nějaký automat. Odborně se tomuto potřebnému času říká časová složitost. V posledních letech je to velmi důležitá věc, která se řeší při navrhování méně i více složitých systémů, které pracují s velkým množstvím dat. Mezi algoritmy, které se zabývají vyhledáváním v textu se řadí například následně popsané algoritmy.

3.1 Naivní algoritmus

Tento algoritmus je základní algoritmus při prohledávání textu. Není na něm nic složitého. Je to v podstatě prozkoumání všech možností a zkráceně řečeno teda porovnání každého vzorku (písmena nebo slova) s celým textem (každým písmenem nebo slovem jednotlivě).



Obr. 3. Naivní algoritmus

Časová složitost je:

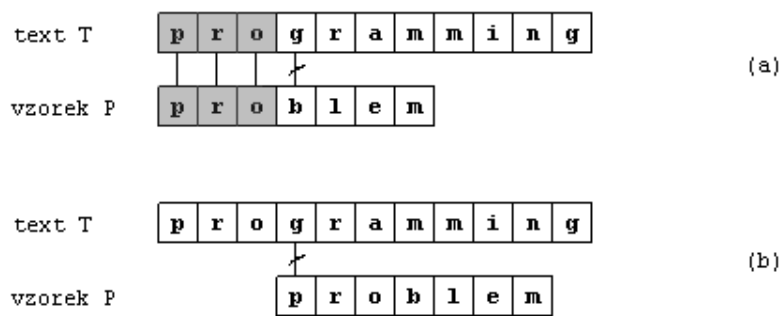
$$O((n-m+1)*m)$$

Kde: m – je délka vzoru

n – je délka textu

3.2 Knuth-Morris-Prattův algoritmus

Tento algoritmus má vylepšení od předchozího Naivního algoritmu v tom, že se nemusí posouvat doprava jen o jeden vzorek, ale o více. Velice dobře je to vidět na následujícím obrázku.



Obr. 4. Knuth-Morris-Prattův algoritmus

Pokud při procházení textu T a porovnávání vzorku P narazí program na neshodu, neposune se v textu jen o jeden vzorek, ale hned o několik a to až na místo neshody kde začíná opět porovnávat. Časová složitost je zde:

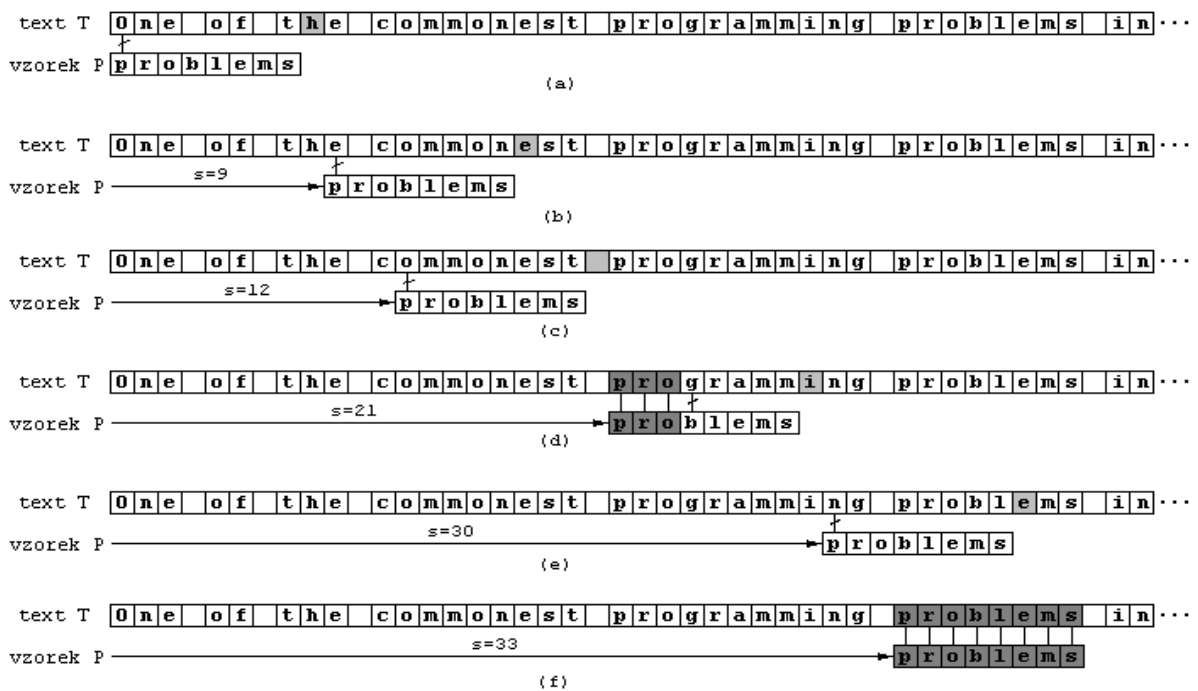
$$O(m+n)$$

Kde: m – je délka vzoru

n – je délka textu

3.3 Quicksearch

Dalším z mnoha algoritmů je Quicksearch. Je opět malinko podobný všem ostatním algoritmům. Také zde se vzorek posouvá o více než jeden znak a to maximálně až o délku o jeden znak vyšší než je délka vzorku, takže o $m+1$.



Obr. 5. Quicksearch

Je to způsobeno tím, že algoritmus porovná první znak vzorku s přiřazeným znakem textu a současně znaky obsažené ve vzorku se znakem z textu o 1 vyšším než je délka vzorku. Na ukázce (a) se porovnává znak **h** se znaky vzorku. Pokud se znak **h** nenachází ve vzorku, dojde k posunutí až za znak **h** jak je vidět na ukázce (b), kde je dalším porovnávaným znakem **e**. To se již ve vzorku nachází. Provede se posunutí vzorku, aby obě **e** byly na stejné pozici. Ale při prvním porovnání nesedí první znak. Dalším znakem pro porovnání je proto prázdné políčko, které se ve vzorku nevyskytuje a proto dochází k posunutí o $m+1$ znaků. Na ukázce (d) se sice první 3 znaky shodují, ale na čtvrtém dojde k neshodě a současně znak **i** se nenachází ve vzorku a následuje maximální posun v tomto případě o 9 pozic. Následně je testovaným znakem opět znak **e**, který se přiřadí ke znaku **e** ze vzorku. Pak probíhá porovnávání zbylých znaků a v tomto případě k nalezení celého vzorku. Časová složitost je zde přibližně:

$$O(n/(m+1))$$

Kde: m – je délka vzoru

n – je délka textu

II. PRAKTICKÁ ČÁST

4 PROGRAM POROVNÁVÁNÍ

Tato práce se zaměřuje na porovnávání dokumentů a určování míry shodnosti dokumentů. Program je schopný porovnávat dokumenty MS Office a textové soubory a dokáže také porovnávat tabulky a obrázky vložené v dokumentech. Program porovnává vždy jeden hlavní soubor s jinými porovnávanými soubory. Program je založen na vyhledávání shodných řetězců a podle toho určuje míru shodnosti. Zda se ale opravdu jedná o plagiát, by měl s definitivní platností posoudit až uživatel. Program jen naznačuje, jestli mezi dokumenty nějaká míra shodnosti opravdu existuje. Program nepoužívá žádného zvláštního algoritmu, proto při zpracovávání objemných dat může být časově náročnější. Program také využívá četných dialogů, proto by práce s ním měla být jednoduchá pro každého uživatele. Program je zde popsán jak pro uživatele tak i pro programátory.

5 POPIS PROGRAMU PRO UŽIVATELE

Tento program není nutné nijak instalovat ani nijak zvlášť složitě nastavovat a jeho ovládní by měl každý zvládnout i bez nějakého předchozího setkání. Veškeré součásti jsou umístěny v jedné složce s názvem Porovnávání. Po otevření této složky máme zde další 3 podsložky a to Program, Test a Výsledky.

5.1 Složka Program

V této složce se nachází hlavní spouštěč programu Porovnej. Je zde umístěn dokument s názvem Porovnávání.doc. Po otevření tohoto dokumentu možná budeme vyzváni k povolení maker. S tímto povolením budeme souhlasit. Pokud vyzváni nebudeme je nutné v nastavení povolit makra. Po otevření budeme mít v souboru doprovodný text a tlačítko Spustit. Tímto tlačítkem spustíme celý program. Po spuštění programu se před námi objeví formulář pro zadání vstupních dat.

Makro pro porovnání dokumentů

Vyberte množství porovnávaných souborů

Porovnání dvou souborů

Porovnání dvou a více souborů

Zadejte soubor který e bude porovnávat

C:\Users\Pepa\Desktop\Porovnáv Otevřít

Zadejte porovnávaný soubor nebo složku se soubory

C:\Users\Pepa\Desktop\Porovnáv Otevřít

Zdejte počet porovnávaných slov

3

OK Zavřít

Obr .6. Formulář vstupních dat

Je zde položka buď pro porovnávání mnoha dokumentů a nebo jen dvou dokumentů. Dále je zde kolonka pro zadání cesty k hlavnímu porovnávanému souboru. Tuto kolonku můžeme také vyplnit pomocí kliknutí na tlačítko Otevřít a zde soubor jen vybrat. Ve druhé kolonce zadáváme buď druhý porovnávaný soubor nebo složku, ve které se nachází

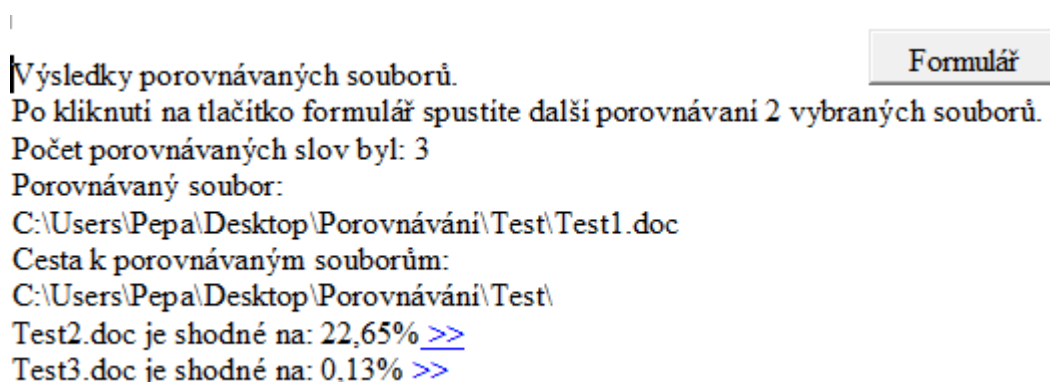
soubory, které chceme porovnávat s hlavním souborem. Dále je zde kolonka pro nastavení počtu porovnávaných slov. Po vybrání souborů a nastavení délky řetězce pak stačí kliknout na tlačítko OK a program na porovnávání se nám spustí. Při porovnávání více souborů se nám ukáže ukazatel průběhu, který zobrazuje kolik procent souborů z vybrané složky je již hotovo. Nakonec se nám zobrazí kde máme hledat výsledky našeho porovnávání a jak se tento soubor jmenuje. Soubor najdeme ve složce Výsledky. Při porovnávání jen dvou souborů se nám však ukazatel průběhu nezobrazí, ale vidíme, jak nám program postupně zvýrazňuje shodné řetězce.

5.2 Složka Test

Ve složce Test máme tři soubory na odzkoušení našeho programu, abychom věděli jak funguje. Jsou zde dokumenty Test1.doc, Test2.doc a Test3.doc. Kde Test1.doc a Test3.doc jsou rozdílné dokumenty, proto je jejich míra shoda minimální. Při srovnání dokumentů Test1.doc a Test2.doc nebo Test2.doc a Test3.doc již bude míra shody ztelně větší.

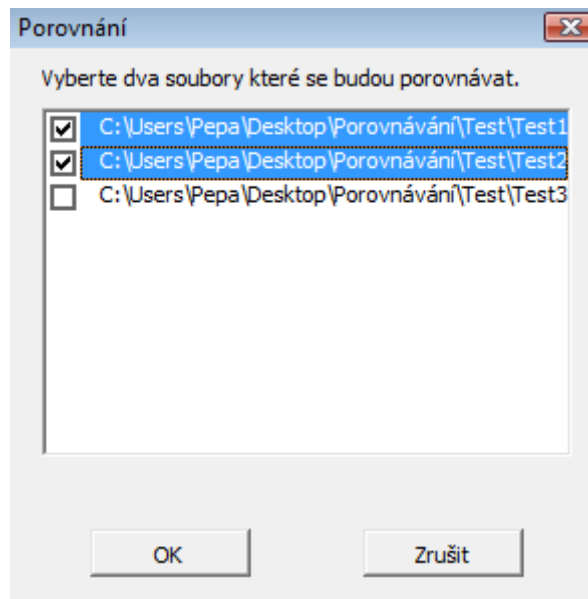
5.3 Složka Výsledky

Složka výsledky obsahuje naše výsledky porovnávání více souborů. Zde se nachází i šablona pro ukládání výsledků. Po otevření našich výsledků vidíme, který soubor se s kterými soubory porovnával a jejich procentuální shodu.



Obr. 7. Výsledky porovnávání

Za hodnotou shody je vždy uveden hypertextový odkaz na příslušný soubor. V pravém horním rohu nalezneme tlačítko formulář, které slouží pro spuštění dalšího porovnávání. Tentokrát však už jen dvou souborů. Po kliknutí na toto tlačítko se nám objeví dialog, kde vybere z nabídky dva soubory a po potvrzení našeho výběru se spustí další porovnávání.



Obr.8. Dialog výsledků

Toto porovnávání je stejné jako předchozí zmiňované porovnávání jen dvou dokumentů. Takže se nám opět ukáží oba vybrané dokumenty a zvýrazní se nám shodné řetězce v obou dokumentech.

6 POPIS PROGRAMU PRO PROGRAMÁTORY

Celkem máme dva programy, které jsou obsaženy ve dvou dokumentech přesněji v jednom dokumentu a jedné šabloně dokumentu. Hlavní program je v dokumentu *Porovnej.doc* a druhý je v šabloně *Výsledky.dot*.

6.1 Dokument Porovnávání

Tento program je rozdělen do několika procedur, které se navzájem volají a spolupracují spolu. Hlavní procedura má název *Porovl*. Procedura *Porovl* nám volá procedury *Nacti_data*, *Aktualizovat_postup* a *Porovnej*. V Proceduře *Porovnej* nám probíhá samotné porovnávání dokumentů. Tato procedura pak zase volá proceduru *Tabulka* a funkci *Obrazek*. V proceduře *Tabulka* dochází k porovnávání tabulek a ve funkci *Obrazek* k načítání parametrů obrázků. Všechny procedury a funkce voláme pomocí příkazu *Call*.

6.1.1 Porovl

Jak už bylo zmíněno procedura *Porovl* je hlavní procedura. Na začátku procedury se provede jen pár základních nastavení. Pak se hned volá procedura *Nacti_data*. Proceduře *Nacti_data* se předávají 4 parametry a to: *porovnani*, *cesta*, *soubor1*, *podbarveni*. Příkaz vypadá asi takto: *Call nacti_data(porovnani, cesta, soubor1, podbarveni)*. Po načtení dat zjistíme hodnotu *podbarveni*. Tato hodnota nám určí zda budeme podbarvovat shodné texty, nebo ne a tím pádem i zda se jedná o porovnání dvou a nebo více souborů. Takže následuje podmínka *If podbarveni*.

Když bude hodnota pravdivá (*True*), tak otevřeme *soubor1*, což je první ze dvou porovnávaných dokumentů. Otevřeme jej příkazem *Documents.Open(soubor1)*. Pak už jen voláme proceduru *Porovnej* a předáváme všechny tyto parametry. *Call Porovnej(soubor1, cesta, porovnani, cesta_s, zakladni_s, podbarveni)*

Pokud však bude hodnota nepravdivá (*False*), čeká nás více příkazů. Opět provedeme otevření prvního dokumentu, který se bude porovnávat. Pak provedeme skrytí souborů, aby nás neobtěžovalo prohlížení dokumentů pomocí příkazu *Documents(soubor1).Application.Visible = False*. Poté si nastavíme filtry, které nám budou filtrovat jen soubory s koncovkami *doc* a *txt*. Následně si vytvoříme kolekci všech souborů ze zadaného adresáře.

```
1 Set adresar = fso.getfolder(cesta)
2 Set kolekce = adresar.Files
```

Dále si na základě šablony *Vysleky.dot* vytvoříme dokument *Vysledky* s následujícím pořadovým číslem (např. *Vyseldky1.doc*) a vypíšeme do něj základní informace o našem porovnávání. Tento dokument poté uložíme do podložky Výsledky. Pak vstoupíme do cyklu *For* a postupně čteme všechny názvy souborů z naší vytvořené kolekce a porovnáváme je na typ souboru. Pokud soubor projde našimi filtry, tak se volá již zmiňovaná procedura *porovnej* a předává se jí aktuální název filtrovaného souboru s dalšími parametry. Takže se pokaždé porovnávají dva dokumenty. První zadaný a druhý dokument vybraný z kolekce dokumentů. Zároveň se v tomto cyklu aktualizuje a volá procedura *Aktualizovat_postup*. Po vyskočení z cyklu provedeme zavření druhého dokumentu bez uložení.

```
3 Documents(soubor1).Close SaveChanges:=wdDoNotSaveChanges
```

A nakonec této procedury necháme zviditelnit všechny soubory, které byly na začátku skryty.

```
4 Documents.Application.Visible = True
```

6.1.2 Porovnej

Procedura *Porovnej* je volána jen z hlavní procedury *Porov1*. V proceduře *Porovnej* se provádí hlavní porovnávání textu a také případné zvýraznění textu. Procedura *Porovnej* proto přebírá parametry *soubor1*, *soubor2*, *porovnaní*, *cesta*, *zakladni_s* a *podbarveni*. Tyto proměnné obsahují cesty k souborům, délku porovnávaného řetězce nebo jestli se má zvýrazňovat shodný text. Na začátku procedury provedeme zjištění počtu slov v aktivním dokumentu. V tomto případě je aktivní dokument první otevřený dokument. Počet slov zjistíme pomocí následujícího příkazu.

```
5 pocetslov = ActiveDocument.ComputeStatistics(wdStatisticWords)
```

Za slova jsou zde považovány i některé samostatné znaky jako jsou například čárka nebo tečka. Dále provedeme zjištění počtu obrázků v dokumentu. A to následujícím příkazem.

```
6 poc_obr1 = ActiveDocument.Shapes.Count
```

Poté ověříme zda dokument obsahuje nějaké obrázky a když ano tak voláme proceduru *Obrazek*. Volání provedeme opět pomocí funkce *Call*.

```
7 Call obrazek(pole_obr1)
```

Proceduře *Obrazek* předáváme parametr *pole_obr1*. Do tohoto pole se nám pak uloží hodnoty o nalezených obrázcích.

Pak provedeme otevření druhého dokumentu. Druhý dokument je buď námi zadaný a nebo vybraný z kolekce dokumentů. Opět provedeme zjištění počtu slov v druhém dokumentu a uložíme si jej do druhé proměnné s názvem *pocetslov2*. A také provedeme opětovné zjištění počtu obrázků. Načítání parametrů obrázků však provádíme jen v případě, že první dokument obsahoval nějaké obrázky a druhý dokument také obsahuje nějaké obrázky. Načítání nám opět obstarává procedura *Obrazek*. Pak aktivujeme náš první otevřený dokument pomocí příkazu.

```
8 Documents(soubor1).Activate
```

Dále zjistíme počet tabulek v tomto dokumentu příkazem.

```
9 poctab = ActiveDocument.Tables.Count
```

Pak vstoupíme do hlavního cyklu procedury *Porovnej*. Je to opět cyklus *For*, který nastavíme aby jel od 1 do počtu slov v našem aktuálním dokumentu. Takže cyklus projde všechny slova v dokumentu. Podle nastavení *porovnavani* zjistíme, kolik slov bude obsahovat náš porovnávaný řetězec a kolik slov máme tedy do řetězce načíst. Načtení provedeme následujícím příkazem.

```
10 ActiveDocument.Range(ActiveDocument.Words(i).Start,  
ActiveDocument.Words(i + porovnavani - 1).End).Select
```

Potom zjistíme zdali se označený text nachází v tabulce.

```
11 If Selection.Information(wdWithInTable) = True Then
```

Pokud se náš označený text bude nacházet v tabulce budeme volat proceduru *Tabulka* s jejími parametry.

```
12 Call tabulka(i, pocetret, stejnypocet, pocet_tabulek1, podbarveni,  
soubor1, soubor2)
```

Pokud se text nenachází v tabulce, tak se toto volání přeskočí a dál se pracuje s označeným textem. Nastavíme počet řetězců na 1 a pak při každém dalším cyklu počet o jedničku navýšíme. Pak aktivujeme druhý dokument, ve kterém hodláme náš označený řetězec hledat. Pro toto hledání použijeme příkazu *Find*. Nejprve však nastavíme funkci *hledej*, aby obsahovala celý dokument, a pak teprve použijeme zmiňovaný *Find*.

```
13 Set hledej = ActiveDocument.Content
```

```
14 If hledej.Find.Execute(retezec) = True Then
```

Pak nám příkaz *Find* vyhledává náš řetězec, a pokud ho najde, zvýší se stejný počet řetězců (*stejnypocret*) o 1. Zároveň zjišťujeme, zda-li je podbarvení pravdivé.

```
15 If podbarveni Then
```

Pokud je *podbarveni* rovno *True*, tak se jedná o porovnávání jen dvou dokumentů a zde provádíme podbarvení stejného textu. Zároveň pak aktivujeme první dokument a zde také provedeme zvýraznění označeného textu. Nakonec se zase přepneme do druhého dokumentu.

```
16 Selection.Start = 0
17 Selection.End = 0
18 Selection.Find.Text = řetězec
19 Selection.Find.Replacement.Text = ""
20 Selection.Find.Forward = True
21 Selection.Find.Wrap = wdFindStop
22 While Selection.Find.Execute = True
23 Selection.Range.HighlightColorIndex = wdYellow
24 Wend
25 Documents(soubor1).Activate
26 Selection.Range.HighlightColorIndex = wdYellow
27 Documents(soubor2).Activate
```

Po vyskočení z cyklu *For* následuje porovnávání obrázků. Před tímto porovnáváním však ještě zjišťujeme zda podbarvujeme, protože jestli ano, tak nemá smysl obrázky porovnávat. Pokud nepodbarvujeme, tak porovnáme hodnoty obrázků uložených do pole v několika vnořených cyklech. Následně pak provedeme konečné propočítání všech získaných hodnot a tím pádem procentuální vyjádření shodnosti dokumentů. Poté provedeme otevření našeho předem nastaveného dokumentu výsledky a zde uložíme název a procentuální shodu dokumentů. Výsledky pak uložíme a uzavřeme.

6.1.3 Tabulka

Procedura *tabulka* slouží k porovnávání tabulek v dokumentech. Pro hledání v tabulkách nemůžeme použít příkaz *Find*. Proto zde budeme porovnávat postupně každý řádek s každým řádkem. Nejprve si označíme celou jednu tabulku a pak spočítáme počet slov v označené tabulce.

```
28 posun_i = Selection.Words.Count
```

Tento počet slov nám slouží jako hodnota posunu naší pozice v dokumentu. Pokud tedy prohledáváme dokument po určitých řetězcích a najednou narazíme na tabulku, tak po vyskočení z této tabulky musíme pokračovat až za ní. K tomu nám právě poslouží počet slov v tabulce, které už budeme mít prohledané. Dále pak spočítáme počet řádků v tabulce.

```
29 pocet_radku1 = ActiveDocument.Tables(pocet_tabulek1).Rows.Count
```

Pak vstupujeme do vnějšího cyklu.

```
30 For j = 1 To pocet_radku1
```

Zde postupně načítáme řádek po řádku. Po načtení jednoho řádku se přepneme do druhého dokumentu. Zjistíme, zdali se zde nějaká tabulka nachází a když ano, tak kolik jich je. Pak vstoupíme do prvního vnořeného cyklu.

```
31 For k = 1 To pocetab2
```

Tento cyklus nám slouží k procházení všech tabulek v druhém dokumentu. Následuje zjištění počtu řádků první tabulky druhého dokumentu. Dále vstoupíme do posledního vnořeného cyklu, který nám stejně jako první cyklus slouží k procházení jednotlivých řádků. Tentokrát však v jiném dokumentu a tudíž i tabulce. Zde už se také provádí porovnání obou načtených řádků tabulek. Pokud jsou shodné připočítáme je do míry shodnosti. Zároveň opět zjišťujeme, jestli máme zvýrazňovat. Pokud ano, tak zvýrazníme načtené řádky v obou tabulkách. Nakonec celé procedury *Tabulka* provedeme již zmiňované posunutí našeho kurzoru za tabulku.

```
32 i = i + posun_i - 1
```

6.1.4 Obrazek

Funkce *Obrazek* nám zjišťuje parametry jednotlivých obrázků a ty pak podle toho porovnáváme. Funkce se volá a označuje stejně jako procedura. Funkci *Obrazek* předáváme pouze jeden parametr a to pole. Na začátku funkce provedeme zjištění počtu obrázků v dokumentu a poté provedeme nastavení velikosti našeho pole.

```
33 pocet_obrazku = ActiveDocument.Shapes.Count
```

```
34 p_obr = pocet_obrazku * 6
```

```
35 ReDim pole_obr(p_obr) As Integer
```

Počet porovnávaných parametrů u každého obrázku je 6. Mezi tyto parametry patří jas, kontrast, typ barvy, typ, výška a šířka obrázku. Nejprve provedeme označení určitého obrázku a pak podle následujících příkazů zjišťování těchto hodnot.

```
36 Contrast = Selection.ShapeRange.PictureFormat.Contrast
```

```
37 jas = Selection.ShapeRange.PictureFormat.Brightness
38 typ_barvy = Selection.ShapeRange.PictureFormat.ColorType
39 typ = Selection.ShapeRange.Type
40 sirka = Selection.ShapeRange.Width
41 vyska = Selection.ShapeRange.Height
```

Tyto hodnoty zjišťujeme v cyklu.

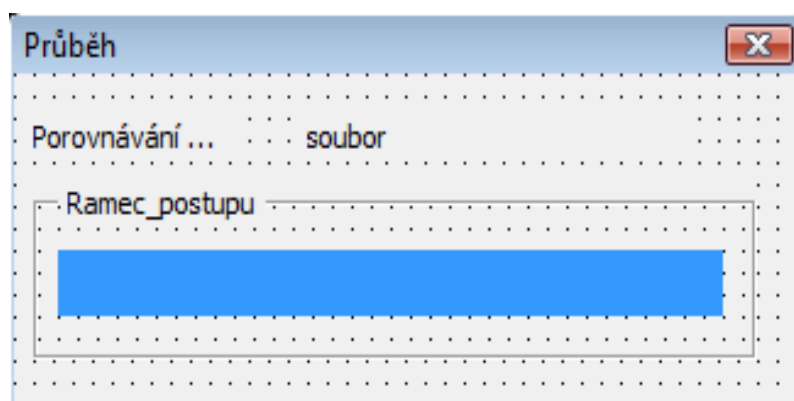
```
42 For i = 1 To pocet_obrazku
```

Zároveň zde provádíme ukládání hodnot do vytvořeného pole.

6.1.5 Aktualizovat_postup

Procedura *Aktualizovat_postup* přebírá jen jeden parametr a to proměnnou *procenta*.

V podstatě zde dochází k aktualizaci postupu průběhu. Zde také voláme druhý Formulář, který nám slouží jako ukazatel postupu a předáváme mu dál hodnoty k zobrazení.



Obr. 9. Ukazatel průběhu

Tato procedura vypadá následovně.

```
43 Sub aktualizovat_postup(procenta)
44 With UserForm1
45     .Ramec_postupu.Caption = Format(procenta, "0 %")
46     .popopisek_postupu.Width = procenta *
        (.Ramec_postupu.Width - 20)
47     .Repaint
48     .Show vbModeless
49 End With
50 End Sub
```

6.1.6 Nacti_data

Tato procedura nám slouží k načítání potřebných dat. Toto načítání se provádí pomocí formuláře, který je volán z této procedury.

Obr. 10. Formulář načítání

Proceduře jsou předávány 4 hodnoty. Ale skutečný parametr, který předáváme je jen parametr porovnávání. Slouží k přednastavení první hodnoty pro porovnávání. Procedura vypadá následovně.

```

51 Sub nacti_data(porovnaní, cesta, soubor, podbarvení)
52 UserForm2.SpinButton1.Value = porovnaní
53 UserForm2.Show
54 porovnaní = UserForm2.SpinButton1.Value
55 cesta = UserForm2.TextBox2.Value
56 soubor = UserForm2.TextBox1
57 podbarvení = UserForm2.OptionButton1
58 Unload UserForm2
59 End Sub

```

Nejdříve provedeme předání hodnoty porovnávání.

```

60 UserForm2.SpinButton1.Value = porovnaní

```

Pak necháme formulář zobrazit.

```
61 UserForm2.Show
```

Poté, co jsme formulář zobrazili, čekáme na další data. Tedy čekáme až uživatel programu zadá požadované data do formuláře a potvrdí tlačítkem OK. Následně se pokračuje v naší proceduře. Pokračujeme tím, že data z formuláře načteme zpět do parametrů procedury.

```
62 porovnani = UserForm2.SpinButton1.Value
```

```
63 cesta = UserForm2.TextBox2.Value
```

```
64 soubor = UserForm2.TextBox1
```

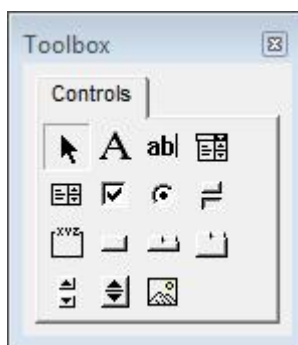
```
65 podbarveni = UserForm2.OptionButton1
```

Do proměnné *porovnani* se nám uloží hodnota, kterou zadal uživatel. Tato hodnota znázorňuje délku porovnávaného řetězce ve slovech. Pak je zde proměnná *soubor*, která obsahuje cestu k prvnímu vybranému dokumentu porovnávání. Proměnná *cesta* nám buď předává cestu k druhému souboru nebo cestu do složky, ve které jsou porovnávané soubory. Poslední proměnná *podbarveni* nám bude obsahovat hodnotu True nebo False ze které pak určíme zda budeme porovnávat buď jen 2 dokumenty nebo 1 dokument s celou složkou jiných dokumentů. Zároveň nám pak také poslouží k zjištění zda máme stejné nalezené řetězce podbarvovat. Na závěr celé procedury provedeme zavření našeho formuláře.

```
66 Unload UserForm2
```

6.1.7 Formuláře

V našem programu používáme dva formuláře. Jeden nám slouží jako ukazatel průběhu a druhý pro načítání dat. Formulář vytvoříme po kliknutí na *insert* a pak *UserForm*. Poté se nám otevře grafické prostředí a okno *Toolbox*, díky kterému můžeme formulář snadno vytvořit.



Obr. 11. Okno Toolbox

Po grafickém návržení vzhledu našeho formuláře je důležité nastavit jeho funkce. U ukazatele průběhu nebylo toto nastavení funkcí nutné. Při načítání dat nám funkce ve formuláři zajišťují snadné ovládání a také ošetření chyb.

OptionButtony umístěné v našem formuláři také nepotřebují nastavení. Jen se může v okně *Properties* přednastavit jedna položka jako již vybraná. To se provede změnou hodnoty na True. Jinak jsou zde hodnoty předávány automaticky. Při nastavení dalších dvou tlačítek Otevřít však nějaké nastavení již potřebovat budeme. První z tlačítek Otevřít má následující nastavení.

```
67 Private Sub CommandButton1_Click()  
68 With Application.FileDialog(msoFileDialogFilePicker)  
69     .Filters.Add "soubory", "*.doc;*.txt"  
70     .Title = "Vyber soubor"  
71     .Show  
72     If .SelectedItems.Count = 0 Then  
73         MsgBox "Výběr byl zrušen."  
74     Else  
75         TextBox1.Text = .SelectedItems(1)  
76     End If  
77 End With  
78 End Sub
```

Jsou zde napsané příkazy, které se vykonají po kliknutí na toto tlačítko. Jak vidíme tak použijeme přednastaveného dialogu Windows pro otevírání souborů a nastavíme filtrování souborů. Nakonec ošetříme předání cesty k zvolenému souboru do první kolonky TextBoxu.

```
79 TextBox1.Text = .SelectedItems(1)
```

Skoro to stejné nastavení platí i pro druhé tlačítko Otevřít. Toto tlačítko je však doplněno ještě o otevírání jen složek. Po kliknutí na tlačítko se nejprve zjišťuje hodnota OptionButtonu a podle ní se pak buď bude jednat o předání cesty k vybranému souboru jako tomu bylo u prvního tlačítka a nebo o předání cesty ke zvolené složce. Zde vidíme naši podmínku po které následuje stejné vybrání souboru jako bylo u prvního tlačítka.

```
80 If OptionButton1 Then
```

Změna v horní části podmínky pro získání cesty k souboru je jen v tom, že předáváme hodnotu do druhého TextBoxu.

```
81 TextBox2.Text = .SelectedItems(1)
```

Dále máme ukázanou druhou část podmínky, která slouží pro získání cesty k zadané složce. Opět využíváme přednastaveného dialogu Windows.

```

82 Else
83     With Application.FileDialog(msoFileDialogFolderPicker)
84         .Title = "Vyber složku"
85         .Show
86         If .SelectedItems.Count = 0 Then
87             MsgBox "Výběr byl zrušen."
88         Else
89             TextBox2.Text = .SelectedItems(1)
90         End If
91     End With
92 End If

```

Dalším tlačítkem je tlačítko OK. Tlačítko OK a jeho procedura nám kontrolují zda máme všechno a správně zadané. Takže začínáme touto podmínkou.

```

93 If CStr(SpinButton1.Value) = TextBox3.Text And TextBox2 <> "" And
    TextBox1.Text <> "" Then
94     TextBox3.Text = SpinButton1.Value
95     UserForm2.Hide

```

V případě, že jsou všechny potřebné kolonky vyplněné, se provede skrytí našeho formuláře. Skrytí provádíme z toho důvodu, aby při načítání dat v proceduře Nacti_data jsme mohly tyto data předat do požadovaných proměnných a pak až v procedura Nacti_data nám tento formulář zavírá. Pokud není některá z kolonek vyplněná, tak zjišťujeme, která z nich to je. Výsledek zapíšeme pomocí čísla do proměnné *zadani*.

```

96 If CStr(SpinButton1.Value) <> TextBox3.Text Then zadani = 1
97 If TextBox2 = "" Then zadani = 2
98 If TextBox1 = "" Then zadani = 3

```

Po zjištění nevyplněné kolonky se na příslušnou kolonku přesuneme.

```

99 Select Case zadani
100     Case 1
101         MsgBox "Nesprávné zadání.", vbCritical
102         TextBox3.SetFocus
103         TextBox3.SelStart = 0
104         TextBox3.SelLength = Len(TextBox3.Text)
105     Case 2
106         MsgBox "Nesprávné zadání.", vbCritical
107         TextBox2.SetFocus

```

```

108         TextBox2.SelStart = 0
109         TextBox2.SelLength = Len(TextBox2.Text)
110     Case 3
111         MsgBox "Nesprávné zadání.", vbCritical
112         TextBox1.SetFocus
113         TextBox1.SelStart = 0
114         TextBox1.SelLength = Len(TextBox1.Text)
115     End Select

```

Zde dochází k vypsání chyby a pak následnému přesunutí na tuto kolonku, kde čekáme, že ji uživatel vyplní a opět s potvrzením tlačítka *OK* provádí tato kontrola.

Dále zde máme poslední tlačítko s názvem *Zavřít*. Zde se samozřejmě nachází jen jeden příkaz k zavření celého formuláře.

```
116 Unload UserForm2
```

Pro nastavení délky porovnávaného řetězce máme ve formuláři opět dva způsoby. Uživatel tuto hodnotu zadává buď přes klávesnici do určené kolonky (v našem případě je to *TextBox3*), nebo pak pomocí posuvných šipek (*SpinButton1*). Tyto dvě položky jsou navzájem propojeny, aby po změně v jedné položce se projevila změna i ve druhé položce. Takže nastavením *SpinButtonu* se změní i *TextBox*.

```

117 Private Sub SpinButton1_Change()
118     TextBox3.Text = SpinButton1.Value
119 End Sub

```

To samé platí i naopak. Zde je ještě přidáno omezení pro minimální a maximální hodnotu.

```

120 Private Sub TextBox3_Change()
121     Dim nova_hodnota As Integer
122     nova_hodnota = Val(TextBox3.Text)
123     If nova_hodnota >= SpinButton1.Min And nova_hodnota <=
        SpinButton1.Max Then
124         SpinButton1.Value = nova_hodnota
125     End If
126 End Sub

```

Dalším Opatřením v tomto formuláři je zavírání toho formuláře jen pomocí tlačítka *Zavřít* jak již bylo uvedeno. Takže ošetříme horní křížek tak, aby se tímto způsobem nedal formulář zavřít.

```

127 Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As
    Integer)
128     If CloseMode = vbFormControlMenu Then

```

```
129         MsgBox "Tento formulář nelze zavřít tímto způsobem."  
130         Cancel = True  
131     End If  
132 End Sub
```

Ted už nám zbývá ošetřit jen poslední dvě kolonky. Těmito kolonkami jsou dva TextBoxy. Do těchto okýnek se nám zobrazují cesty vybraných souborů nebo složek. Je zde možné, aby tyto cesty zadal uživatel i ručně. V prvním políčku bude vždy vepsaná cesta k prvnímu dokumentu. Proto procedura pro TextBox1 kontroluje zda tento dokument existuje.

```
133 Private Sub TextBox1_Exit(ByVal Cancel As MSForms.ReturnBoolean)  
134     If Dir(TextBox1.Text) = "" Then  
135         MsgBox "Soubor neexistuje.", vbCritical  
136         TextBox1.SetFocus  
137         TextBox1.SelStart = 0  
138         TextBox1.SelLength = Len(TextBox1.Text)  
139     End If  
140 End Sub
```

V případě, že neexistuje, se zobrazí chybové hlášení a čeká se až cestu k souboru uživatel opraví. U druhého TextBoxu se opět kontroluje zda soubor či složka existuje. Proto máme na začátku procedury podmínku, která je ovlivněna předchozím nastavením OptionButtonu. Ten nám zjišťoval, jestli budeme porovnávat dva dokumenty a nebo jeden dokument s celou složkou dokumentů.

```
141 If OptionButton1 Then
```

Takže pokud se bude jednat o zjišťování, zda daný dokument existuje, tak procedura bude stejná jako výše uvedená, ale přidá se k ní ještě ověření zda je soubor dokumentem nebo textovým souborem. Pokud však bude OptionButton druhé možnosti tak víme, že se bude jednat o ověření, zda daná složka existuje a to ověření provedeme následovně.

```
142 If Dir(TextBox2.Text, vbDirectory) = "" Then  
143     MsgBox "Složka neexistuje.", vbCritical  
144     TextBox2.SetFocus  
145     TextBox2.SelStart = 0  
146     TextBox2.SelLength = Len(TextBox2.Text)  
147 End If
```

6.2 Dokument Výsledky

Šablona Výsledky obsahuje další makro neboli program. V pravém horním rohu je umístěno tlačítko Formulář. Toto tlačítko nám provede načtení adres dokumentů do dalšího formuláře k jejich následnému zpracování. Protože soubor výsledky byl vytvořen naším programem na základě šablony *Výsledky.dot*, známe jeho strukturu a víme umístění jednotlivých výsledků porovnávání a dalších údajů. Předpokládáme, že obsah souboru s výsledky nebyl ručně měněn. Takže si nejprve zjistíme počet řádků v dokumentu s výsledky.

```
148 poc_radku = ActiveDocument.ComputeStatistics(wdStatisticLines)
```

Poté se přesuneme na třetí řádek a načteme zde hodnotu porovnávání.

```
149 posun_naR = Selection.MoveStart(wdLine, 3)
150 posun_naS = Selection.MoveLeft(wdWord, 2, wdExtend)
151 porovnaní = Selection.Text
```

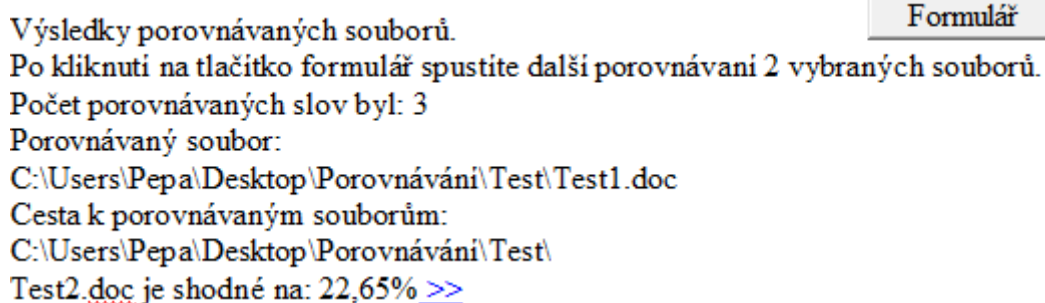
Tato hodnota bude načtena jako řetězec, proto ji musíme převést na číslo.

```
152 porovnaní = CInt(porovnaní)
```

Pak vložíme tuto hodnotu do formuláře, kde bude na tomto formuláři skryta.

```
153 UserForm1.Label1 = porovnaní
```

Následuje opět přesun o dva řádky níže, kde načteme cestu k prvnímu porovnávanému dokumentu.



Výsledky porovnávaných souborů. Formulář
 Po kliknutí na tlačítko formulář spustíte další porovnávání 2 vybraných souborů.
 Počet porovnávaných slov byl: 3
 Porovnávaný soubor:
 C:\Users\Pepa\Desktop\Porovnávání\Test\Test1.doc
 Cesta k porovnávaným souborům:
 C:\Users\Pepa\Desktop\Porovnávání\Test\
 Test2.doc je shodné na: 22,65% >>

Obr. 12. Pořadí řádků

```
154 posun_naR = Selection.MoveStart(wdLine, 2)
155 posun_naR = Selection.MoveDown(wdLine, 1, wdExtend)
156 prvni_s = Selection.Text
```

Provedeme úpravu načtené cesty, tak že oddělíme znak konce řádku.

```
157 poc = Len(prvni_s)
158 prvni_s = Left$(prvni_s, poc - 1)
```

A pak tuto upravenou cestu vložíme jako první cestu do našeho formuláře.

```
159 UserForm1.ListBox1.AddItem prvni_s
```

Opět provedeme přesun o dva řádky níže, kde načteme cestu do složky s porovnávanými dokumenty.

```
160 posun_naR = Selection.MoveStart(wdLine, 2)
161 posun_naR = Selection.MoveDown(wdLine, 1, wdExtend)
162 cesta_s = Selection.Text
```

Upravíme adresu cestu složky.

```
163 poc = Len(cesta_s)
164 cesta_s = Left$(cesta_s, poc - 1)
```

Posuneme se o řádek níže, kde vstoupíme do cyklu, který projde a načte všechny zbylé cesty k porovnávaným dokumentům.

```
165 posun_naR = Selection.MoveStart(wdLine, 1)
166 For i = 8 To poc_radku
```

Do hlavního cyklu, který prochází všechny zbylé řádky je vnořen ještě jeden pomocný cyklus, který nám slouží pro načtení jen názvu našeho dokumentu.

```
167   For j = 1 To 1
168       posun_naS = Selection.MoveRight(wdWord, 1, wdExtend)
169       radek = Selection.Text
170       znak = Right(radek, 1)
171       If znak = " " Then
172           slovo = radek
173       Else
174           j = j - 1
175       End If
176   Next
```

Po ukončení tohoto vnořeného cyklu následuje sloučení cesty s názvem dokumentu, takže získáváme úplnou cestu k dokumentu a vložení této cesty do formuláře.

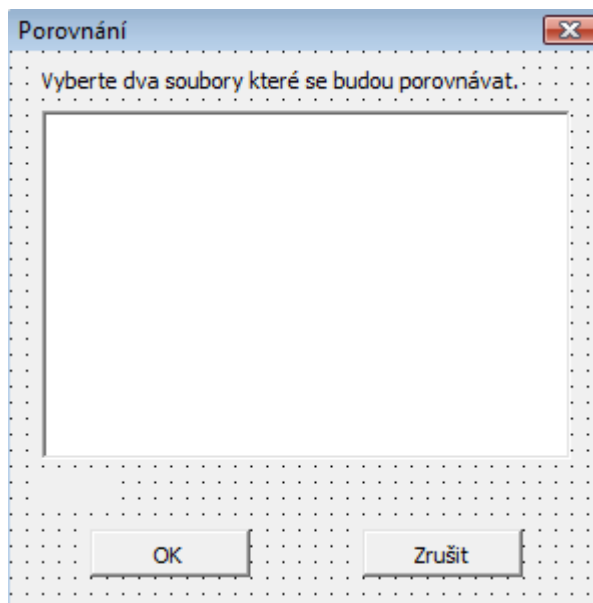
```
177 posun_naR = Selection.MoveStart(wdLine, 1)
178 radek = cesta_s & radek
179 UserForm1.ListBox1.AddItem radek
```

Nakonec po ukončení i hlavního cyklu provedeme už jen přesun kurzoru na začátek dokumentu s výsledky a zobrazení našeho formuláře i s daty.

```
180 Selection.GoTo.StartOf wdStory, wdExtend
181 UserForm1.Show
```

6.2.1 Formulář šablony výsledků

Formulář jsme vytvořili graficky ze dvou tlačítek, ListBoxu, Labelu a nyní nastavíme jen jejich procedury.



Obr. 13. Okno výsledků

U ListBoxu si v okně Properties nastavíme, aby se nám u každé položky zobrazovalo zaškrťovací políčko (CheckBox). Popisové okno (Label) nastavíme v okně Properties jako skryté a bude nám sloužit jen k předávání hodnoty velikosti porovnávání. Pro ListBox a Label už žádné procedury neprogramujeme. Následuje tedy nastavení procedur pro obě tlačítka. U tlačítka OK budeme zjišťovat, které dva dokumenty byly vybrány k porovnávání a také zde nastavíme a předáme další parametry makru *Porovnavani*, což je vlastně stejná procedura jako byla v makru *Porovl*. Po kliknutí na tlačítka OK tedy nejprve zjistíme, které dokumenty uživatel vybral.

```
182 For i = 0 To ListBox1.ListCount - 1
183     If ListBox1.Selected(i) Then
184         poc = poc + 1
185         Select Case poc
186             Case 1
187                 soubor1 = UserForm1.ListBox1.List(i)
188             Case 2
189                 soubor2 = UserForm1.ListBox1.List(i)
190             Case 3
191                 MsgBox "Bylo zadáno více jak 2 soubory."
```

```
192             End
193         End Select
194     End If
195 Next
```

Zjišťujeme to pomocí uvedeného cyklu, který prochází všechny položky seznamu a zkoumá, které jsou označeny. Následně se první dvě položky uloží do proměnných *soubor1* a *soubor2*. Pokud v tomto cyklu narazíme na 3 označenou položku, dojde k vypsání chyby a program se ukončí. Dále pak provádíme ještě kontrolu zda byly označeny dva soubory a ne jen jeden. To zjistíme testem, jestli jsou obě proměnné naplněny.

```
196 If (soubor1 = Empty) Or (soubor2 = Empty) Then
197     MsgBox "Bylo zadáno málo souborů k porovnání."
198     End
199 End If
```

Následně provedeme vložení důležitých hodnot do proměnných a zavoláme proceduru *Porovnej* pro porovnání dvou námi vybraných dokumentů. Na konci procedury tlačítka OK provedeme zavření tohoto formuláře. U tlačítka Zrušit jen nastavíme zavření formuláře stejným způsobem jako na konci procedury tlačítka OK.

```
200 Private Sub CommandButton2_Click()
201     Unload UserForm1
202 End Sub
```

ZÁVĚR

Tato bakalářská práce měla za cíl vytvořit program, který by určoval míru shodnosti mezi dokumenty. Pro tyto účely byl využit programovací jazyk VBA, který je standardní součástí každého balíku Microsoft Office. Je maximálně kompatibilní s programem MS Word a proto se pro tento úkol dobře hodil. Navíc obsahuje řadu možností a doplňků, díky kterým lze tento program realizovat. Program měl porovnávat mezi sebou dokumenty i textové soubory. Program je nastaven tak, aby při porovnávání více jak dvou souborů určil míru jejich shodnosti a při dvou souborech zvýraznil stejné části.

V teoretické části této práce je vysvětleno pár základních pojmů a základy programování v tomto jazyce. Praktické části je pak uveden popis programu z uživatelského pohledu, kde je vysvětlen postup práce s tímto programem pro uživatele. Dále pak popis programu z programátorského pohledu, kde jsou postupně rozebírány jednotlivé procedury a funkce toho programu. Tento program se celkově skládá ze dvou dílčích programů, kde ten hlavní je umístěn v dokumentu *Porovnávání*. Druhý, který slouží rychlému porovnávání dvou dokumentů, je umístěn v šabloně *Výsledky* a tudíž nahrán do každého vytvořeného dokumentu s výsledky.

Tento program lze využít jen na některé druhy obsahů dokumentů a to zejména pro dokumenty jejichž obsah je tvořen informativním textem. Pokud by například uživatel porovnával dokumenty jejichž obsahem jsou zdrojové kódy, program by vyhodnotil velkou shodu dokumentů a nebo naopak odlišnost i kdyby byly zdrojové kódy významově stejné. Proto nesmíme zapomínat na to, že program nám vyhodnocuje míru shodnosti textu a ne míru shodnosti významu textu. Tento program nám slouží spíše jako označení a pomůcka při porovnávání. I když se v nástrojích MS Word nachází funkce Porovnej nemá ani zdaleka takové možnosti jako náš program pro porovnávání. Náš program dokáže nejen porovnat a zvýraznit stejný text, ale i vyhodnotit míru shodnosti a zároveň dokáže porovnávat dokument s více dokumenty a ne jen dva mezi sebou.

CONCLUSION

This bachelor work had for the aim to create a program that would have determined the degree of commonality between the documents. For these purposes has been used programming language, VBA which is a standard part of every package Microsoft Office. It is most compatible with MS Word and, therefore it especially fit for this task. In addition, it contains set of options and accessories that make this program realized. Program had to compare with each other documents and also text files. Program is set up so that when comparing more than two files determined the extent of their commonality and to highlight two spare set of the same part.

In the theoretical part of this work is explained some basic terms and fundamentals of programming in this language. In the practical part is a description of the program from the user perspective, where is explained way of work with this program for users. Next, there is way of program from programmer's view, where they are gradually analyzed individual procedures and functions of this program. This program that generally consists of two sub-programs, where the main is located in a document comparing. The second, which is quickly compare two documents, is located in the template results and thus loaded into each document created with the results.

This program can be used only for certain types of documents, content, and especially for documents whose content consists of informative text. For example, if a user compare the documents containing the source code program by assessing the compliance of large documents, or vice versa, even if differences were semantically the same source code. Therefore, we must not forget that the program we evaluate the degree of conformity of the text and not the degree of commonality meaning of the text. This program serves us more as a suggestion, and to assist in the comparison. While in MS Word Compare function is not even far from such opportunities as our program for comparison. Our program can not compare and highlight the same text, but also assess the degree of consistency and also can compare the document with multiple documents and not just two between them.

SEZNAM POUŽITÉ LITERATURY

Monografie:

- [1] D.F. Scott - „Velká kniha programování v Microsoft Office 2000“, Computer Press, Praha (2000), 946s, ISBN 80-7226-240-8
- [2] John Walkenbach - „Microsoft Office Excel 2007 Programování ve VBA“, Computer Press, Brno (2008), 912s, ISBN 978-80-251-2011-8

Internetové odkazy:

- [3] DAVID-ZBIRAL.CZ – Počítač, tipy, informatika, makra, makra pro Word.
Dostupný z URL: <<http://www.david-zbiral.cz/vb.htm>>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

VBA Visual Basic for Applications.

A Tučným písmem jsou vyjádřeny tlačítka klávesnice.

C++ Objektivě orientovaný programovací jazyk.

CD Compact Disc.

MS Microsoft

SEZNAM OBRÁZKŮ

Obr. 1. Okno Project.....	19
Obr. 2. Okno Properties.....	20
Obr. 3. Naivní algoritmus.....	22
Obr. 4. Knuth-Morris-Prattův algoritmus.....	23
Obr. 5. Quicksearch.....	24
Obr. 6. Formulář vstupních dat.....	27
Obr. 7. Výsledky porovnávání.....	28
Obr. 8. Dialog výsledků.....	29
Obr. 9. Ukazatel průběhu.....	35
Obr. 10. Formulář načítání.....	36
Obr. 11. Okno Toolbox.....	37
Obr. 12. Pořadí řádků.....	42
Obr. 13. Okno výsledků.....	44

SEZNAM TABULEK

Tab. 1. Datové typy.....	13
Tab. 2. Operátory.....	16

SEZNAM PŘÍLOH

Příložené CD obsahuje složku, v níž jsou umístěny dokumenty se zmiňovanými programy. Je zde také složka Test, ve které jsou umístěny tři zkušební dokumenty pro otestování tohoto programu.