

MS Excel – makra a VBA



© Autor:

RNDr. Milan Myšák

e-mail: milan.mysak@centrum.cz

Obsah:

1	Využití, výhody a nevýhody	3
1.1	Přehled VBA – objektový model	3
1.2	Metody, vlastnosti a třídy	3
2	Makra a zabezpečení	4
2.1	Nastavení zabezpečení Excelu	4
2.2	Uložení maker do sešitu a osobního sešitu maker	5
3	Vytvoření jednoduchého makra záznamníkem maker a spuštění	5
3.1	Relativní a absolutní odkazy	6
3.2	Spouštění maker	6
4	Úprava makra a prostředí editoru VBA	7
4.1	Okno projektu a okno kódu	8
4.2	Panely (Toolbars)	8
4.3	Další okna	8
4.4	Ukončení makra s chybou	9
4.5	Makra a národní prostředí	10
5	Programátorská pravidla pro makra a VBA	11
5.1	Konstanty	11
5.2	Proměnné	12
5.3	Datové typy	13
5.4	Pole	14
5.5	Operátory – výpočty a kontroly	14
5.6	Podmínky If, Then, Elseif, Else	15
5.7	Cykly, posloupnosti a opakování	16
5.8	Více možností – Select Case	20
5.9	Výstup – MsgBox	21
5.10	Vstup – InputBox	21
5.11	MsgBox - okna s tlačítky	21
5.12	Konstanty pro určení reakce uživatele	22
5.13	Monitorování reakce uživatele – stisk kláves	Chyba! Záložka není definována.
6	Práce s buňkami	Chyba! Záložka není definována.
6.1	Výběry buněk	Chyba! Záložka není definována.
6.2	Načtení hodnoty z buňky	Chyba! Záložka není definována.
6.3	Zápis hodnoty do buňky	Chyba! Záložka není definována.
7	Práce s listy	Chyba! Záložka není definována.
7.1	Termíny	Chyba! Záložka není definována.
7.2	Přidání, odstranění	Chyba! Záložka není definována.
7.3	Kopie, přesun	Chyba! Záložka není definována.
7.4	Skrývání listů	Chyba! Záložka není definována.
7.5	Zamknutí a odemknutí listů	Chyba! Záložka není definována.
7.6	Události listu	Chyba! Záložka není definována.
8	Práce se soubory	Chyba! Záložka není definována.
9	Výpočty pomocí maker	Chyba! Záložka není definována.
10	Ukázka konkrétních praktických maker	Chyba! Záložka není definována.
10.1	Nastavení formátu textu a čísel	Chyba! Záložka není definována.

10.2	Seznam listů.....	Chyba! Záložka není definována.
10.3	Sehrání hodnot z předem neznámého počtu listů.....	Chyba! Záložka není definována.
10.4	Vložení funkce pomocí makra	Chyba! Záložka není definována.
10.5	Přidání listu s kontrolou a dotazem.....	Chyba! Záložka není definována.
10.6	Výpis posledního času změny listu - událost listu	Chyba! Záložka není definována.
10.7	Výpis posledního času změny buněk.....	Chyba! Záložka není definována.
10.8	Dotaz a využití odpovědi	Chyba! Záložka není definována.
10.9	Formuláře	Chyba! Záložka není definována.
10.10	Sehrání dat z více listů.....	Chyba! Záložka není definována.

1 Využití, výhody a nevýhody

Programování maker se učíme proto, abychom uměli automatizovat často se opakující (i poměrně složité) činnosti. Abychom se to naučili, musíme rozumět kódu, který vytváří záznamník maker a musíme ho také umět upravit, abychom konkrétní zaznamenanou činnost uměli zobecnit. Pokud máme např. každý měsíc zpracovávat desítky tabulek od obchodních zástupců, je rozumné tuto činnost automatizovat pomocí makra. Potom může sehrání dat trvat pouze několik minut a následně se pouze aktualizují připravené kontingenční tabulky. V opačném případě (bez maker) to může být práce i na celý den.

Výhody:

- automatizované opakování činností
- pomocí makra lze zajistit, co v Excelu přímo nejde nebo pouze složitě
- možnost jednoduchého spouštění pomocí klávesových zkratk

Nevýhody:

- problémy se spouštěním – nelze jednoduše nastavit, protože to je věcí uživatele, nikoliv makra
- problémy se zabezpečením – makra musí být povolena a bohužel se některé viry tváří jako makra
- umístění makra – je vázané na místo uložení, může být pouze v sešitu nebo v osobním sešitu maker (pomocí maker a VBA tedy nelze vytvořit samostatný spustitelný program)
- akci, provedenou makrem, nelze vrátit zpět

Prostředí pro tvorbu maker se neliší již od verzí 2002/2003.

Pozor – makra nejsou plně kompatibilní mezi jednotlivými verzemi MS Excel a to bohužel ani do novějších, ani do starších verzí. Vše je nutné řádně vyzkoušet a otestovat. Makra nemůže obsahovat excelový soubor typu XLSX, ale musí být uložený jako XLSM.

Makra mohou být nejenom v excelových souborech, ale lze je používat ve všech programech MS Office.

Upozornění: příprava a úprava maker je programování v jazyce VBA. Ten, kdo doposud ani trochu neprogramoval, bude mít těžké začátky. Pokud ale máte čas a vůli se toto programování naučit, dokážete následně s Excelem velké věci. Pokud nikoliv, raději na makra zapomeňte a používejte pomalejší excelové postupy.

1.1 Přehled VBA – objektový model

Aplikace MS Office jsou vytvořené tak, že ukazují své objekty, které očekávají instrukce. Během programování spolupracujete s aplikací tak, že jí předáváte instrukce – přesněji řečeno předáváte instrukce objektům v aplikaci. Například pro práci s excelovým sešitem obsahuje Excel objekt Dokument. Pomocí něj je možné sešit otevřít, uložit i zavřít.

Objekty jsou seřazené v hierarchii zvané objektový model aplikace. S existujícím objektem lze pracovat nastavením jeho vlastností a používáním jeho metod. Definice objektu se také nazývá třída.

1.2 Metody, vlastnosti a třídy

Objektem je např. Application a jeho vlastností je ActiveDocument, metodou potom jeho uložení (Save):

Application.ActiveDocument.Save

nebo ještě lépe s metodou Uložit jako:

Application.ActiveDocument.SaveAs („NewFile.xlsx“)

Podobně třeba zapíšeme text Hello World do buňky A1:

```
Application.ActiveSheet.Range("A1").Select
```

```
Application.Selection.Value = "Hello World"
```

První řádek definuje objekt Range a spustí metodu pro její výběr. Výsledek se uloží do vlastnosti Selection.

Druhý řádek nastaví vlastnost Value výběru (vlastnost Selection) na „Hello World“

Ve skutečnosti lze programovat i bez těchto teoretických znalostí. Základ se vytvoří pomocí záznamníku maker a dále se kód upravuje.

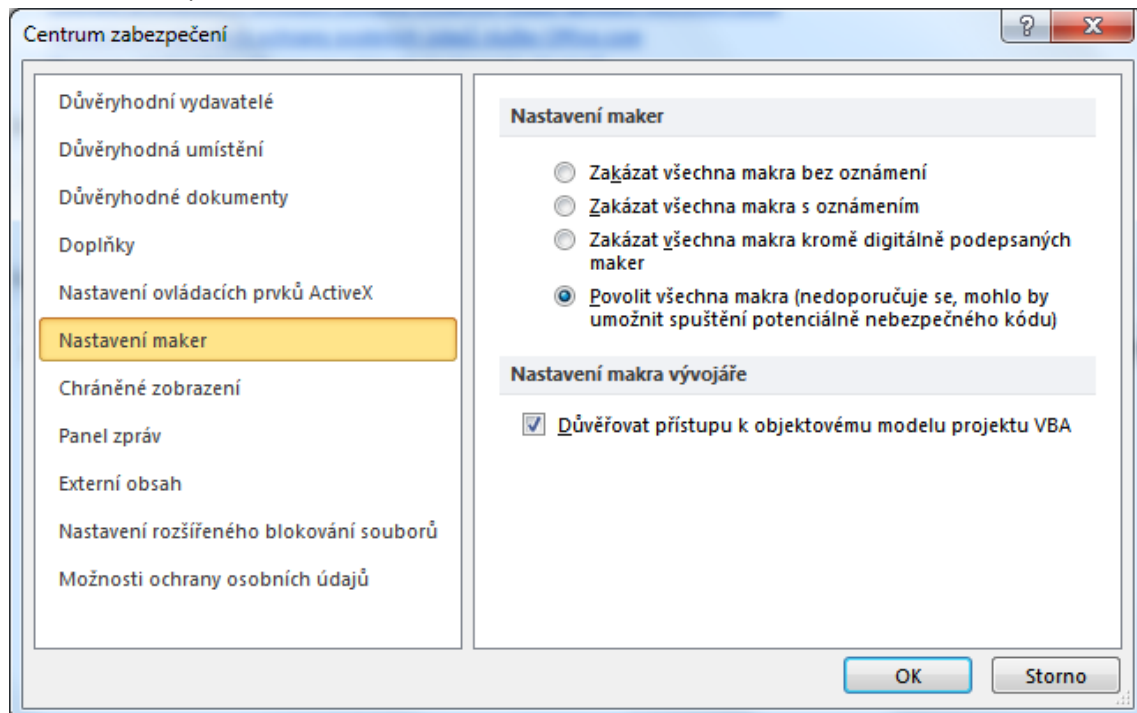
2 Makra a zabezpečení

2.1 Nastavení zabezpečení Excelu

Hlavním praktickým problémem může být to, že vaše makra na jiném počítači nemusí pracovat – mohou tam být zakázána. To je rysem počítače, autor makra to nemůže ovlivnit. Navíc může být globálně zablokováno spuštění maker pomocí skupinových pravidel.

Používání maker se řídí nastavením Excelu v Centru zabezpečení (Soubor – Možnosti – Centrum zabezpečení):

Centrum zabezpečení – Nastavení maker



Možnosti:

- Zakázat všechny makra bez oznámení: nefungují
- Zakázat všechny makra s oznámením: při otevření sešitu s makry se objeví výzva pro povolení maker (jejrozumnější, výchozí volba)
- Povolit všechna makra – vhodné pouze pro vývoj maker, jinak to je nebezpečná volba

2.2 Uložení maker do sešitu a osobního sešitu maker

Makro může být uloženo v běžném sešitu nebo v tzv. osobním sešitu maker.

Makro v běžném sešitu:

- sešit se musí uložit ve formátu „Sešit aplikace Excel s podporou maker“ – typ XLSTM
- před použitím makra se musí tento sešit otevřít. Většinou se použije makro v tomto aktuálním sešitu, lze ale také otevřít sešit s makrem a použít ho v jiném, také otevřeném sešitu

Makro v osobním sešitu maker:

- lze ho do něj uložit při vytváření makra
- makro je potom vázané k počítači a lze ho tedy použít v libovolném otevřeném sešitu, musí ale být zároveň otevřen personal.xlsb, což může být problém.
- pro úpravu takto uloženého makra se musí zobrazit skrytý sešit Personal: karta Zobrazení – Zobrazit, vybrat PERSONAL.XLSB. Pro spuštění makra nemusí být tento osobní sešit zobrazený trvale, stačí pouze poprvé – nutno vyzkoušet
- soubor PERSONAL.XLSB se nachází v uživatelském adresáři/Data aplikací/Microsoft/Excel/ExcelStart (Win 7 + MS Office 2010)

3 Vytvoření jednoduchého makra záznamníkem maker a spuštění

- Zobrazení karty Vývojář (Soubor – Možnosti aplikace – Přizpůsobit pás karet)

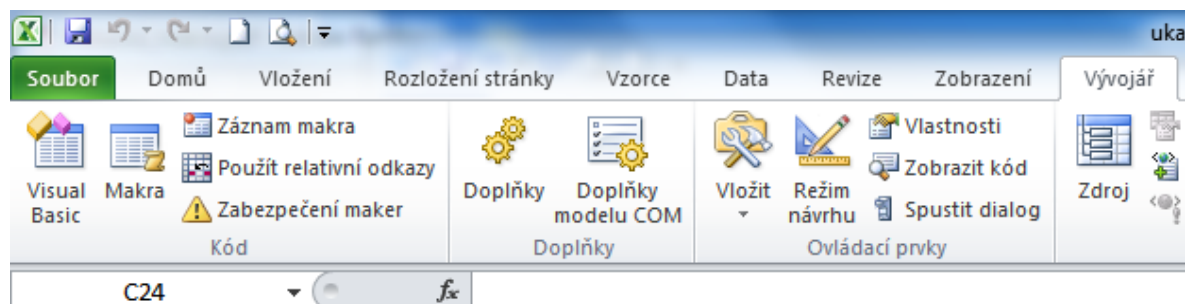
Před spuštěním záznamníku maker je vhodné:

- umístit kurzor nebo připravit výběr
- rozmyslet si posloupnost činností
- vybrat vhodnou a neobsazenou klávesovou kombinaci pro spuštění (CTRL+klávesa, popř. CTRL+SHIFT+klávesa)

Postup:

1. karta Vývojář – Záznam makra
2. zadat vhodný název (bez mezer, kromě písmen a čísel používat podtržítka, raději bez diakritiky – v případě chybného názvu Excel upozorní), např. FormatMena
3. zadat zkratku pro spuštění, umístění a popřípadě popis
4. po potvrzení se spustí záznamník, zaznamenávají se činnosti (kromě např. výběru nabídek)
5. nakonec vybrat opět výchozí buňku (A1) a karta Vývojář – Zastavit záznam

Po vytvoření makra záznamníkem je často nutné ho ještě upravit (karta Vývojář – Makra – Upravit).



Obr. – karta Vývojář

Př.: Makro pro nastavení formátu písma v oblasti (tučné, červená):

```
Sub Makro1()  
  Range("B2:B13").Select  
  Selection.Font.Bold = True  
  Selection.Font.Color = -16776961  
  Range("A1").Select  
End Sub
```

Takto vytvořené makro funguje pouze pro oblast B2:B13.

Po odstranění řádku **Range("B2:B13").Select** je makro funkční pro jakýkoliv výběr (předem vybrat oblast). Pravdou je, že v tomto případě mohou být odstraněny oba řádky **Range...** a není tedy nutné vybírat buňku A1 na začátku a na konci.

Záznamník maker to zapíše ale odlišně:

```
Sub Makro1()  
  With Selection.Font  
    .Color = -16776961  
    .TintAndShade = 0  
  End With  
End Sub
```

Poznámka: tučné písmo nastavíme podstatně jednodušeji pomocí CTRL+B. Ale např. pro číselný formát by to už bylo užitečné makro:

```
Sub Makro2()  
  Selection.NumberFormat = "#,##0.00"  
End Sub
```

Všimněte si, že při programování ve VBA se nepoužívá národní prostředí – např. i pro českou verzi Excelu se použije oddělovač tisíců čárka a jako desetinný oddělovač tečka. Výsledkem je ale např. číslo **12 350,80**

3.1 Relativní a absolutní odkazy

Makra se vytvářejí pro konkrétní oblast v sešitu. Mohou nastat situace:

- v sešitu se přidají/odstraní řádky/sloupce: makro bude fungovat špatně vzhledem k tomu, že obsahuje adresy buněk, které se automaticky nezmění – nutná oprava
- operaci máme provést pro více oblastí (např. sloupce): lze vytvořit makro pro každý sloupec nebo jediné makro, které bude používat relativní odkazy (karta Vývojář – Použit relativní odkazy)

3.2 Spouštění maker

- Nelze spouštět makra samostatně, jako programy, ale pouze „se sešitem nebo s Excelem“.
- Makra, uložená v sešitu, lze spouštět z tohoto sešitu – musí být samozřejmě otevřený.
- Makra, uložená v osobním sešitu maker, lze např. klávesovou zkratkou pouštět na příslušném počítači pouze po jeho otevření (personal.xlsb je skrytý soubor).

Možnosti spuštění:

- Karta Vývojář – Makra – Spustit
- Pomocí zadané klávesové zkratky
- Přiřazením jako akce na formulářové tlačítko
- Další možnosti: z prostředí editoru VBA, automaticky jako událost sešitu,...

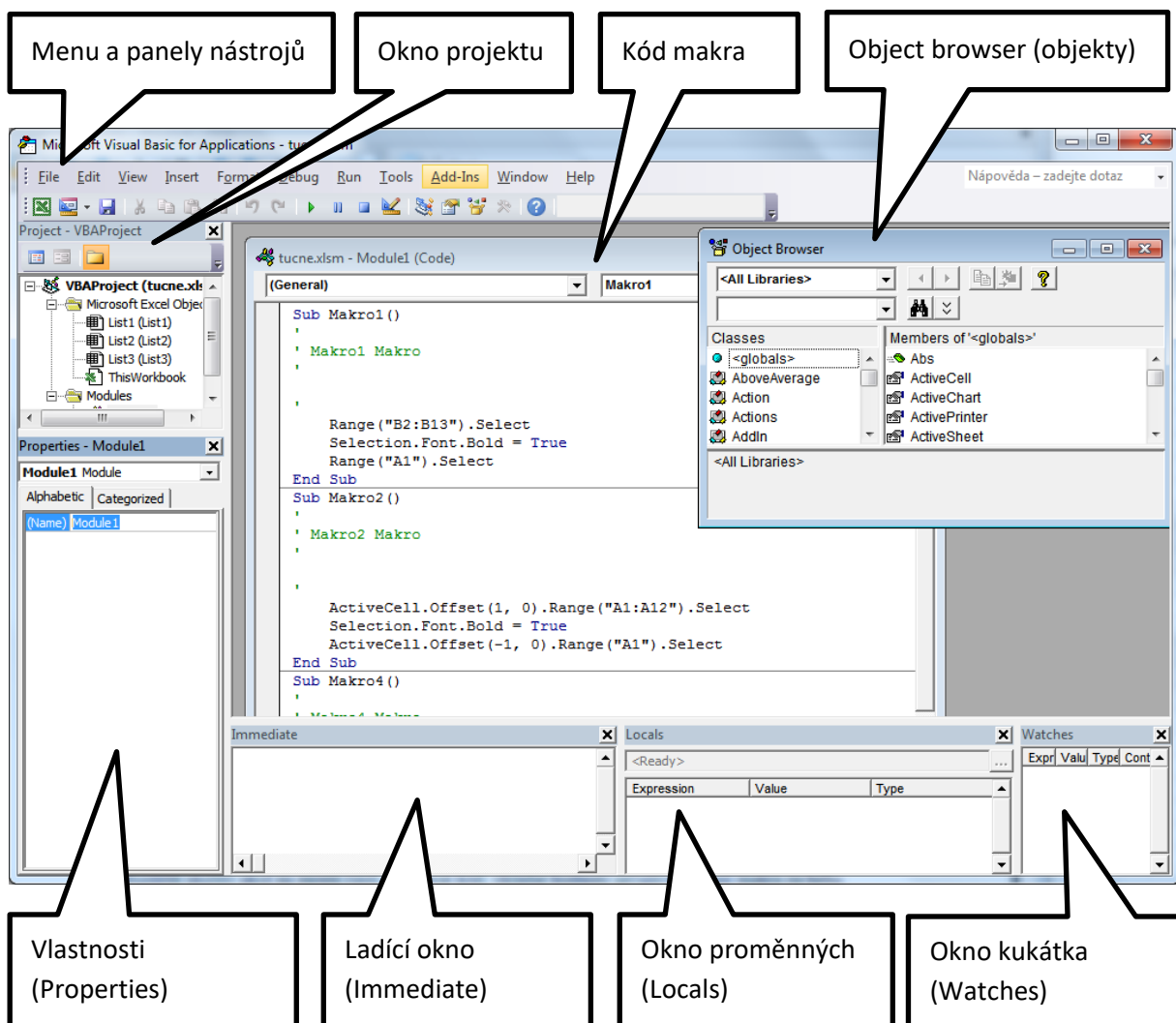
4 Úprava makra a prostředí editoru VBA

Většinu maker, které vytvoříme záznamníkem maker, je nutné dále upravit. Zároveň lze přímo vytvářet nová makra nebo např. funkce. Toto vše se provádí v klasickém prostředí editoru VBA.

Možnosti otevření editoru:

- Vývojář – Makra, vybrat a Upravit
- Vývojář – Visual Basic (pokud je makro např. událostí listu, nelze použít první postup)
- ALT + F11

Pokud je makro vytvořené záznamníkem, je součástí tzv. modulu v objektovém modelu. Tyto moduly je možné i ručně vytvářet (nabídka Insert – Module). Pokud je ale makro např. událostí listu, je umístěno jinde ve stromě – v části příslušného listu.

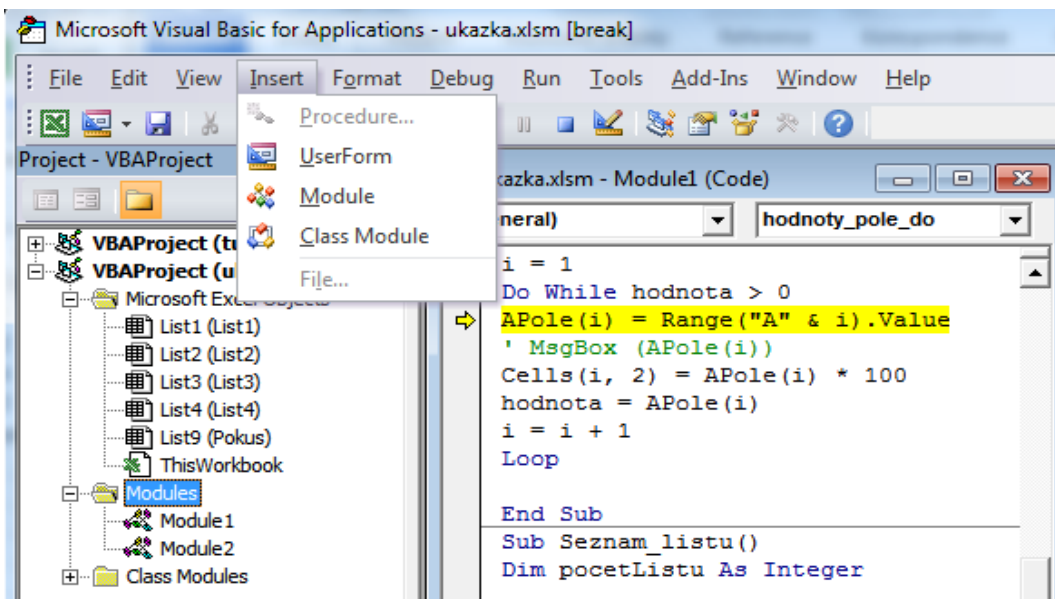


Obr. – prostředí editoru VBA

Poznámky:

- pro běžnou práci s makry postačují výchozí okna Project a Code
- okna se zobrazují / skrývají pomocí nabídek menu View
- Vzhled: panely ve volbě View – vhodné: Project Explorer, Properties Window, panely (Toolbars): Standard a Exit
- Pozor na přepínání režimu Design – Edit (tlačítko Design mode)

4.1 Okno projektu a okno kódu



Obr. – Okno projektu a okno kódu

Okno projektu obsahuje seznam všech objektů, které jsou v daném projektu k dispozici:

- objekty (objects) - Listy, Grafy - která obsahuje listy či grafy a kódy pro tyto listy
- Formuláře (Forms) - UserForm - obsahuje formulář a kódy pro tento formulář
- Moduly (Modules) - obsahuje kódy do kterých se například zaznamenávají vlastní
- funkce, či procedury společné pro celý projekt
- Class Modules

V okně kódu se zobrazuje samotný kód makra.

4.2 Panely (Toolbars)

Standardně se zobrazuje pouze lišta Standard. Pro úpravu kódu se hodí také lišta Edit. Zobrazení se nastavuje v prostředí VBA: View – Toolbars – Edit

Využití: List Properties, List Constants, Quick Info

Rychlá změna části kódu na komentář: vybrat řádky, potom Comment Block (Uncomment Block).

4.3 Další okna

Okno vlastnosti - zjistíte jaké a jak nastavené vlastnosti má konkrétní objekt

Ladicí okno (Immediate) - velice důležité při odlaďování maker a VBA aplikací. Lze do něj zapisovat např. proměnné pro otestování jejich hodnot: Debug.Print var_a

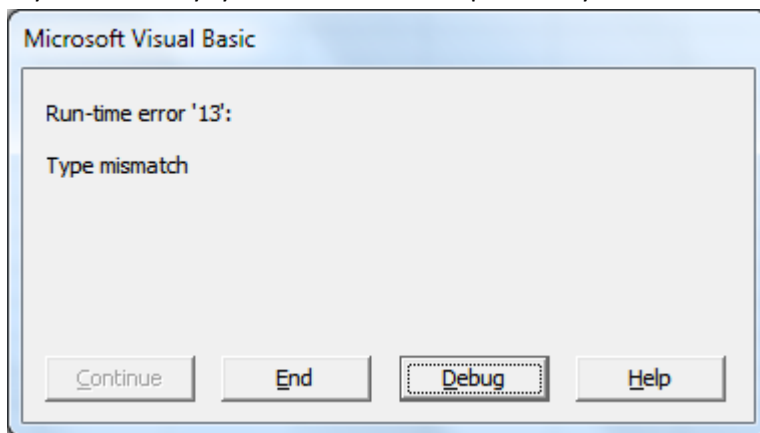
Druhou možností je zobrazení proměnné do MsgBox: MsgBox var_a, třetí možností je okno **Locals**

Object browser – přehled prvků s nápovědou

Okno Formuláře a okno s ovládacími prvky – zde se vytvářejí formuláře.

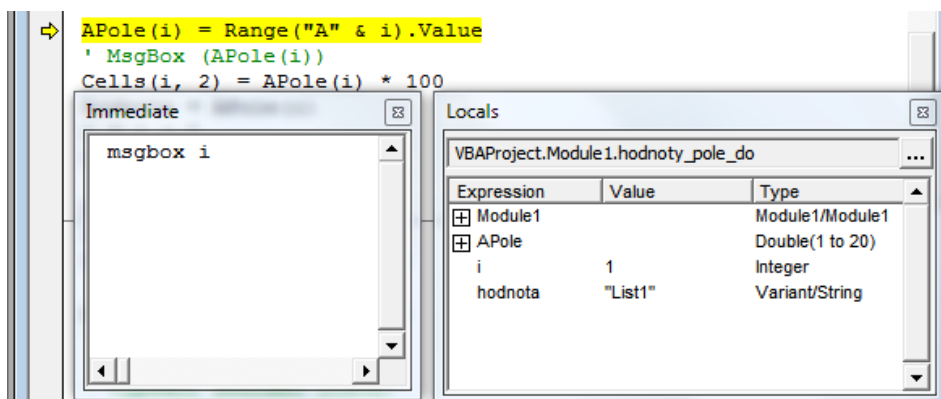
4.4 Ukončení makra s chybou

Pokud makro obsahuje chybu, tak skončí, vypíše chybu a nabídne otevření editoru (tlačítko Debug) se zvýrazněním chyby a možností kontrol proměnných. Tlačítko End makro ukončí hned.

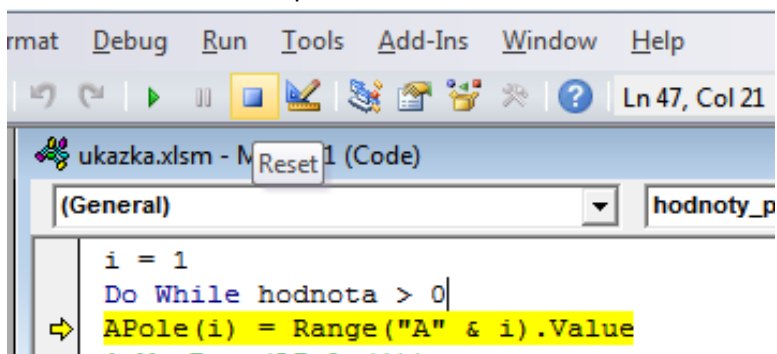


V editoru lze zobrazit další okna (Immediate, Locals) pro ladění makra – vhodné hlavně při výskytu chyby:

- Locals – zobrazuje stav všech proměnných (View – Locals)
- Immediate – k zastavenému makru lze zadávat příkazy (Debug.Print promenna nebo MsgBox promenna)



Pro zastavení makra lze použít tlačítko Reset.



4.5 Makra a národní prostředí

Číselné hodnoty proměnných se zapisují přímo, bez uvozovek a u desetinných čísel je nutné i v českém národním prostředí používat desetinnou tečku, ne čárku!

cislo = 25.50

Pokus se ale do proměnné nepřirazuje hodnota přímo, ale odkážete se např. na desetinné číslo zapsané v buňce listu, bude se jazyk MS VBA řídit místním nastavením Windows a desetinný oddělovač přečte správně. Stejně tak, pokud číslo zadáte jako text, tedy v uvozovkách:

x = "12, 34"

5 Programátorská pravidla pro makra a VBA

5.1 Konstanty

Výhodou je přiřazení konstantě (číslu) vhodný název, který je lépe přiřaditelný (ConstPInoletost) než nic neříkající číslo 18. Dále je výhodné, že po deklarování konstanty nelze tuto hodnotu změnit či dočasně přiřadit hodnotu jinou.

K nastavení (deklaraci) konstanty se používá příkaz Const.

```
Const ConstPInoletost As Integer = 18
```

5.1.1 Druhy platnosti konstant

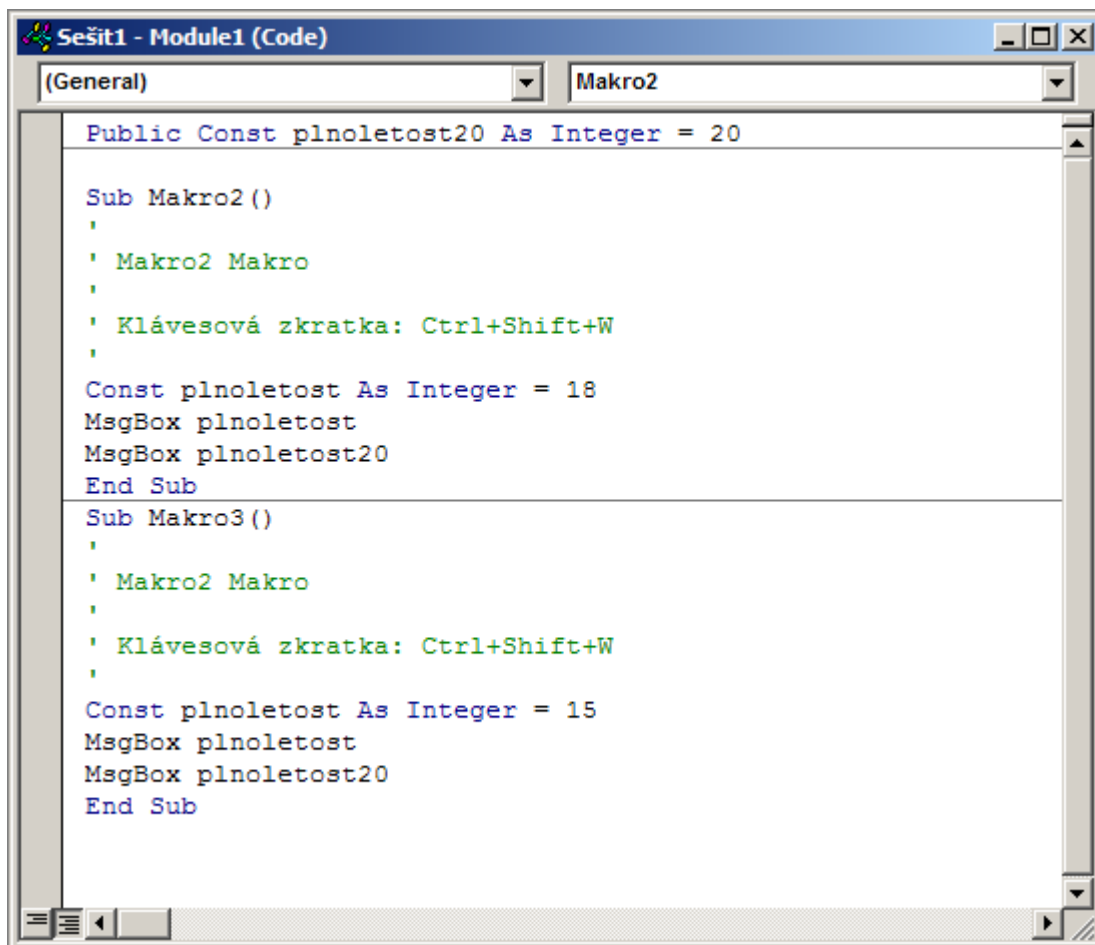
- Public Const - platnost (k dispozici) všem procedurám ve všech modulech.
- Private Const - platnost (k dispozici) pouze v rámci modulu, ve kterém byla deklarace provedena
- Const - pouze platnost v proceduře ve které je deklarována

```
Public Const ConstPInoletost As Integer = 20
```

```
Private Const ConstPInoletost As Integer = 18
```

```
Const ConstPInoletost As Integer = 18
```

Pokud se konstanta definuje jako Public nebo Private, musí být umístěna nad prvním makrem v modulu:



```
Sešit1 - Module1 (Code)
(General) Makro2

Public Const plnoletost20 As Integer = 20

Sub Makro2 ()
'
' Makro2 Makro
'
' Klávesová zkratka: Ctrl+Shift+W
'
Const plnoletost As Integer = 18
MsgBox plnoletost
MsgBox plnoletost20
End Sub

Sub Makro3 ()
'
' Makro2 Makro
'
' Klávesová zkratka: Ctrl+Shift+W
'
Const plnoletost As Integer = 15
MsgBox plnoletost
MsgBox plnoletost20
End Sub
```

5.1.2 Typy konstant:

- Boolean, Byte, Integer, Long, Currency, Single, Double
- Date, String, Variant

5.1.3 Deklarace více konstant

Jedním příkazem lze deklarovat i několik konstant.

```
Const ConstVek As Integer = 34, ConstPlat As Currency = 35000
```

Pozor na správnou deklaraci:

```
Const a, b As Integer = 34
```

a není typu Integer, ale je typu Variant, protože jeho typ nebyl deklarován,
b je typu Integer = 34.

Správně je následující:

```
Const a As Integer = 34, b As Integer = 34
```

5.1.4 Předdefinované konstanty

VBA Excel již obsahuje celou řadu předdefinovaných konstant (tj. některá jména jsou již tímto obsazena). Obsahuje například:

```
xlAnd, xlOr, xlFilterAutomaticFontColor
```

5.1.5 Pojmenovávání konstant

Jména konstant by měla mít nějaký řád ať poznáte že jde o konstantu. Například Excel využívá xl... , stejné principy platí pro proměnné, pojmenovávání objektu, atd.

5.2 Proměnné

Při psaní maker je vhodné v úvodu deklarovat proměnné. Sice VBA funguje bez tohoto deklarování, ale zbytečně alokuje pro proměnou větší prostor než je potřeba. Pro malé skripty to nevádí, ale u složitějších programů se bez deklarací neobejdeme. Jinak si zpomalíme výpočet.

Doporučení: používat názvy proměnných bez diakritiky, začínat velkým písmenem, v názvu proměnné označit její typ.

Kromě deklarace správného typu je potřeba deklarovat platnost.

- Static - Statická proměnná dokud pro danou proceduru, zachovává si platnost.
- Public - Ve všech modulech a procedurách zachovává si platnost i po skončení dané procedury.
- Private - dostupné pro všechny procedury v daném modulu
- Dim - dostupné pro jednu proceduru v daném modulu. Jen když tento modul běží. Nebo pro procedury v daném modulu. Záleží, kde je deklarace uvedena.

Příklady platnost proměnné:

```
Sub PlatnostPromenne1()  
Dim HodnotaLg1 As Long  
HodnotaLg = HodnotaLg + 1  
MsgBox "HodnotaLg = " & HodnotaLg  
End Sub
```

```
Sub PlatnostPromenne2()
Static HodnotaLg As Long
HodnotaLg = HodnotaLg + 1
MsgBox "HodnotaLg = " & HodnotaLg
End Sub
```

Pokud několikrát spustíte první kód bude HodnotaLg stále rovná jedné, ale u druhého kódu se bude hodnota HodnotaLg postupně zvyšovat.

Veřejná platnost proměnné - uvidí ji procedury ve všech modulech:

```
Public PubHodnotaLg As Long

Sub PlatnostPromenne31()
PubHodnotaLg = PubHodnotaLg + 1
MsgBox "PubHodnotaLg = " & PubHodnotaLg
End Sub
```

Pozor opět na deklaraci více proměnných:

Dim a, b As Integer

a není typu Integer, ale je typu Variant, protože jeho typ nebyl deklarován,

b je typu Integer. Správně je následující:

Dim a As Integer, b As Integer

5.3 Datové typy

Datový typ	Počet bajtů	Rozsah hodnot
Boolean	2 bajty	TRUE/FALSE
Byte	1 bajt	0 – 255
Integer	2 bajty	–32.768 – 32.767
Long	4 bajty	– 2.147.483.648 – 2.147.483.647
Single	4 bajty	
Double	8 bajtů	
Currency	8 bajtů	
Decimal	14 bajtů	
Date	8 bajtů	1.leden 0100 – 31.prosinec 9999
Object	4 bajty	Odkaz
String (proměnná délka)	10 bajtů + délka řetězce	0 – 2miliardy
String (pevná délka)	délka řetězce	1 až 65000
Variant (s čísly)	16 bajtů	
Variant (se znaky)	22 bajtů + délka řetězce	0 – 2miliardy

5.4 Pole

Pole je indexovaná skupina nějakých dat. Tato skupina se chová jako by šlo o jednu proměnou, která má několik položek. Na jednotlivé položky se poté odkazuje pomocí indexu.

Pole se začíná standardně indexovat od nuly 0.

Pole 1.

Máme pole o deseti položkách - můžeme jej deklarovat (Poznámka: nezapomeňte - začíná se počítat od nuly):

```
Dim MojePole (9) As Boolean
```

nebo

```
Dim MojePole (0 To 9) As Boolean
```

oba zápisy jsou víceméně identické.

Pole 2.

Máme seznam faktur za roky 2009 - 2011, můžeme si indexování pole posunout.

```
Dim FakturyPole (2009 To 2011) As Double
```

Pole 3.

Dynamické pole pokud neznáme přesně počet prvků. Například potřebuji načíst seznam listů.

```
Dim ListPole() As Double
```

Platnost polí

Podobně jako konstanty lze pomocí příkazů Dim, Static, Private, nebo Public nastavit platnost.

5.5 Operátory – výpočty a kontroly

5.5.1 Matematické

+ - * /

^ mocnina

ukázka:

a = 10

b = 20

vysledek = (a + b)*1,21

5.5.2 Logické

And, Not, Or, Xor

ukázka:

a = 10

b = 20

c = 30

vysledek = a > b Or b > c

5.5.3 Porovnávací

> < >= <=

=

<> není rovno

Like – vrací True nebo False, zda se v testu nachází řetězec:

Test = "F" Like "[A-Z]"

možnosti:

* - odpovídá libovolnému počtu znaků

? - odpovídá právě jednomu znaku

- odpovídá číselnému znaku
- odpovídá znaku(ům) zadaným mezi závorkami
- - odpovídá rozsahu znaku kdy mínus je pro rozsah a-c (písmena mezi a až c a je potřeba vložit do hranatých závorek [a-c]
! - negace - !b neobsahuje b

příklady:

```
MsgBox ("abcpesabc" Like "*pes*")
```

```
MsgBox ("pas" Like "p?s")
```

```
MsgBox ("pas" Like "p??s")
```

```
MsgBox ("A4C" Like "A#C")
```

5.5.4 Slučovací

& slučování textů

```
Test = "Hello" & " World"
```

5.5.5 Ostatní

_ pro rozdělení řádku s kódem

, oddělovač

vbCrLf nový řádek

:

""

' apostrof pro poznámky (lze jednoduše v prostředí VBA – Comment Block)

Příklady:

```
Sub _
```

```
PokusnyProgram()
```

```
End Sub
```

```
Dim Jméno As String, Příjmení As String, Titul As String
```

```
MsgBox("První řádek " & vbCrLf & "Druhý řádek")
```

5.6 Podmínky If, Then, Elseif, Else

5.6.1 If Then

Nejjednodušší větvení. Přeloženo do češtiny: jestliže ... pak.

```
IF Odpoved = "ano" Then  
    MsgBox „Pokračujeme“  
End If
```

5.6.2 If Then Else

Jestliže je odpověď muž pak zobrazí zprávu jinak zobrazí jinou zprávu

```
IF Odpoved = "ano" Then  
    MsgBox „Pokračujeme“  
Else  
    MsgBox „Nepokračujeme“  
End If
```


5.6.3 If Then Elseif Else

Nejsložitější, tedy relativně nejsložitější. Tímto způsobem můžeme přehledně usměrnit i velice složitý požadavek.

```
IF Vek < 15 Then
  MsgBox „Kategorie A“
Elseif Vek < 30
  MsgBox „Kategorie B“
Else Vek < 60 MsgBox „Kategorie C“
Else MsgBox „Kategorie D“
End If
```

5.7 Cykly, posloupnosti a opakování

5.7.1 With ... End With

Tato konstrukce vykonává příkazy pro jeden objekt nebo uživatelsky definovaný typ. Příkazy With zrychlují zpracovávání procedur a umožňují vyhnout se opakovaným příkazům.

With

objekt [příkazy]

End With

Doplnění: V bloku With se nemůže změnit objekt. Proto nelze použít jeden příkaz With pro změnu vlastností různých objektů.

Doporučení: Podle příruček se nedoporučuje odskakovat do nebo z bloků With. Jsou-li provedeny příkazy bez With nebo End With, mohou nastat chyby nebo nepředvídatelné chování programu.

Příklad with ... End With:

Potřebujete změnit buňce hodnotu, písmo změnit na tučné a změnit barvu písma na žlutou.

```
With Worksheets("List1").Range("A1")
  .Value = 30
  .Font.Bold = True
  .Interior.Color = RGB(255, 255, 0)
End With
```

Vnořování with ... End With

Příkazy se dají vnořovat, to nám může ulehčit práci a zpřehlednit náš program.

```
With Workbooks("MujSešit").Worksheets("List1").Cells(1, 1)
  .Formula = "=SQRT(25)"
  With .Font
    .Name = "Arial"
    .Bold = True
    .Size = 8
  End With
End With
```

5.7.2 For Next cykly

Opakuje skupinu příkazů podle zadaného počtu opakování.

Syntaxe

For čítač = začátek To konec [Step krok] [příkazy] Next [čítač]

Popis jednotlivých částí:

- čítač - Povinné - Číselná proměnná používaná jako čítač cyklů. Proměnná nemůže být typu Boolean nebo prvek pole
- začátek - Povinné - Počáteční hodnota čítače
- konec - Povinné - Koncová hodnota čítače
- krok - Volitelné. Hodnota, o kterou je čítač změněn po každém průchodu cyklem. (Není-li uvedeno, nastaví se krok na 1)
- příkazy - vlastní příkazy. Nebo úplně bez příkazu.

Poznámky:

- Čítač může i odečítat.
- Jednotlivé cykly For...Next můžeme vnořovat.
- Krok nemusí být roven jedné
- S proměnou počítadla se da uprostřed cyklu pracovat.
- Pro předčasné ukončení lze do cyklu umístit příkaz Exit For. Tento příkaz Exit For se používá pro vyhodnocení nějaké podmínky (např. If), a předá řízení příkazu následujícímu za příkazem Next.

Příklad

```
For i = 1 To 10  
...  
Next i
```

Příklad s vnořením

```
For I = 1 To 10  
    For J = 1 To 10  
        For K = 1 To 10  
            ...  
        Next K  
    Next J  
Next I
```

5.7.3 Do ... Loop opakování

Opakuje příkazy, dokud platí nebo neplatí podmínka.

Konstrukce Do .. Loop opakuje příkazy, dokud je podmínka vyhodnocena jako True, nebo dokud podmínka není True.

Syntaxe 1:

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

Syntaxe 2

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

- podmínka - Volitelné. Číselný výraz nebo řetězcový výraz, který je vyhodnocen jako True nebo False. Je-li podmínka Null, pak je vyhodnocena jako False.
- příkazy - Jeden nebo více příkazů, které jsou opakovány, dokud je nebo dokud není
- podmínka True.

Poznámky k Do ... Loop:

- Do smyčky Do ... Loop může být umístěn libovolný počet příkazů Exit Do. Tento příkaz umožňuje předčasné ukončení smyčky.
- Při použití ve vnořených příkazech Do .. Loop předá příkaz Exit Do řízení do nadřazeného cyklu.

Příklad Do...Loop

Příklad příkaz Do...Loop proběhne ve smyčce 10krát, nastaví hodnotu příznaku na False a pomocí příkazu Exit Do bude předčasně ukončen.

```
Do While Pocitadlo < 20 ' Vnitřní smyčka.
  Pocitadlo = Pocitadlo + 1 ' Zvyš počítadlo.
  If Pocitadlo = 10 Then ' Je-li podmínka True.
    Test = False ' Nastav hodnotu příznaku na False.
    Exit Do ' Opust' vnitřní smyčku.
  End If
Loop
```

5.7.4 While ... Wend posloupnost s podmínkou

Vykonává posloupnost příkazů tak dlouho, dokud je zadaná podmínka True.

Syntaxe

While podmínka [příkazy] Wend

- podmínka Povinné. Numerický výraz nebo řetězcový výraz, který je vyhodnocen jako True nebo False.
- příkazy Volitelné. Jeden nebo více příkazů provedených, dokud je podmínka True.

Je-li podmínka True, jsou provedeny všechny příkazy až k Wend. Procedura se vrací zpět k příkazu While a podmínka je znovu zkontrolována. Jestliže je stále True, proces se opakuje. Není-li True, běh pokračuje příkazem uvedeným za Wend.

Cykly While...Wend mohou být vnořeny do libovolné úrovně. Každý Wend odpovídá naposledy uvedenému While.

Poznámky:

- Je-li podmínka Null, je vyhodnocena jako False.
- Místo While...Wend lze použít příkaz Do...Loop, který umožňuje lepší a pružnější zpracování cyklů.

5.7.5 Exit

Ukončení provádění kódu.

Příkaz Exit je určen k ukončení provádění kódu v konstrukcích:

Do ... Loop

For ... Next

Function

Sub

Property

Syntaxe je dle požadované konstrukce následující:

Exit Do

Exit For

Exit Function

Exit Sub

Exit Property

Podrobnější popis

Exit Do

Poskytuje způsob ukončení smyčky příkazů Do...Loop. Může být použit jen uvnitř příkazu Do...Loop. Řízení předá příkazu uvedenému bezprostředně za příkazem Loop. Při použití ve vnořených příkazech Do...Loop předá příkaz řízení cyklu o jednu úroveň vnoření výše, než se nachází Exit Do.

Exit For

Poskytuje způsob ukončení cyklu For. Příkaz lze použít pouze uvnitř cyklů For...Next nebo For Each...Next. Příkaz Exit For předá řízení příkazu, který následuje za příkazem Next.

Při použití ve vnořených cyklech For předá Exit For řízení cyklu o jednu úroveň vnoření výše, než se nachází Exit For.

Exit Function

Opustí proceduru Function, ve které se příkaz vyskytuje. Běh programu pokračuje příkazem následujícím za příkazem volání této funkce.

Exit Sub

Ukončí provádění procedury Sub, ve které se příkaz nalézá. Běh pokračuje příkazem následujícím za příkazem volání procedury Sub.

5.8 Více možností – Select Case

Potřebujeme-li vykonat určitou posloupnost příkazu na základě hodnoty příslušného výrazu.

Například hodnota příspěvku dle věku dítěte. Velikost slevy podle hodnoty objednávky.

Poznámka: K řešení lze použít příkazy If, Then, Elseif, Else, čímž docílíte stejného efektu, ale někdy je výhodnější pro přehlednost použít Select Case. Co kdy záleží na každém programátorovi.

Syntaxe

```
Select Case testovaný_výraz
[Case seznam_výrazů-n
    [příkazy-n]]
[Case Else
    [elsepříkazy]]
End Select
```

Popis:

- testovaný_výraz - Povinné. Libovolný číselný výraz nebo řetězcový výraz.
- seznam_výrazů - Povinné, vyskytne-li se Case. Oddělený seznam položek v následujícím formátu:
 - výraz
 - výraz To výraz - Klíčové slovo To určuje rozsah hodnot. Menší hodnota musí být uvedena před To.
 - Is výraz s operátory_porovnání - Klíčové slovo Is s operátory porovnání (kromě Is a Like) lze použít pro určení rozsahu hodnot.
 - kombinace výše uvedených - odděleno čárkou
- příkazy-n - Volitelné. Jeden nebo více příkazů vykonaných, jestliže testovaný_výraz odpovídá libovolné části ze seznamu_příkazů-n.
- elsepříkazy - Volitelné. Jeden nebo více příkazů vykonaných, jestliže testovaný_výraz neodpovídá žádné klauzuli z Case.

Příklad 1 Select Case

V příkladu budeme vyhodnocovat odpověď na otázku v MsgBoxu.

```
Private Sub CommandButton1_Click()
    i = MsgBox("Chcete větší plat?", vbYesNo, "Plat")
    ' vyhodnocení odpovědi
Select Case i
    Case vbNo MsgBox ("NE")
    Case vbYes MsgBox ("ANO")
End Select
End Sub
```

Příklad 2 - Select Case

Příklad kdy je potřeba použít více podmínek pro jeden výraz:

Case 1 To 4, 7 To 9, 11, 13

Příklad 3 - kdy se kontroluje rozsah:

Case Is > MaxCislo

Příklad 4 – spuštění dalších procedur (maker – data_leden, data_unor, data_brezen)

Podle hodnoty proměnné mesic lze spouštět další makra:

```
Public Sub nahrani_mesice()  
mesic = Range("G15").Value  
Select Case mesic  
Case 1  
    data_leden  
Case 2  
    data_unor  
Case 3  
    data_brezen  
End Select  
End Sub
```

5.9 Výstup – MsgBox

Pro zobrazení okamžitých hodnot v makrech lze použít pomocná okna (Immediate, Locals). Pro výstup z makra se ale dá jednoduše využít dialogové okno MsgBox. Během chodu makra se využije na kontrolu proměnných, na konci potom pro výstup:

Příklady:

- Zobrazení proměnné: `MsgBox soucet`
- Zobrazení textu a proměnné: `MsgBox ("proměnná odp: " & odp)`
- Dialogové okno s dotazem – do proměnné vystup se vloží reakce uživatele (stisknuté tlačítko):
`vystup = MsgBox("Vaše jméno: " & odp, vbQuestion, "titulek")`

5.10 Vstup – InputBox

InputBox je způsob, jak může uživatel zadat hodnotu v průběhu makra (interaktivně):

`reakce = InputBox("Zadejte jméno: ", "Zadání jména")`

Poznámka: odchylení stisku klávesy nebo stisku tlačítka se provede jinak - MsgBox

5.11 MsgBox - okna s tlačítky

Podle 2. parametru okna MsgBox se objeví jeho konkrétní vzhled s příslušnými tlačítky. Reakce se dá odchytit (zde proměnná `vystup`):

```
vystup = MsgBox("Chcete pokračovat?", vbYesNoCancel, "Reakce")
```

parametr1: zobrazený dotaz

parametr2: typ okna s tlačítky

parametr3: titulek okna

Typy oken s tlačítky:

Konstanta	Hodnota	Popis
vbOKOnly	0	zobrazí se pouze tlačítko OK
vbOKCancel	1	OK a Storno
vbAbortRetryIgnore	2	Přerušit, Znovu, Ignorovat
vbYesNoCancel	3	Ano, Ne, Storno
vbYesNo	4	Ano, Ne
vbRetryCancel	5	Znovu, Storno
vbCritical	16	Kritické hlášení
vbQuestion	32	Varující dotaz
vbExclamation	48	Varující hlášení
vbInformation	64	Informační hlášení

Dialogy lze kombinovat:

i = MsgBox("Ano ?", vbYesNoCancel + vbInformation)

5.12 Konstanty pro určení reakce uživatele

Konstanta	Hodnota	Popis
vbOK	1	bylo stisknuto tlačítko OK
vbCancel	2	Storno
vbAbort	3	Přerušit
vbRetry	4	Znovu
vbIgnore	5	Ignorovat
vbYes	6	Ano
vbNo	7	Ne

Příklad vyhodnocení:

```
Private Sub CommandButton2_Click()  
i = MsgBox("Chcete pokračovat?", vbYesNo, "Pokracovani")  
Select Case i  
Case 1  
' vbOK  
    MsgBox ("OK, 1")  
Case 2  
' vbCancel  
    MsgBox ("Storno, 2")  
Case 3  
' vbAbort  
    MsgBox ("Přerušit, 3")  
Case 4  
' vbRetry  
    MsgBox ("Znovu, 4")  
Case 5  
' vbIgnore  
    MsgBox ("Ignorovat, 5")  
Case 6  
' vbYes  
    MsgBox ("Ano, 6")  
Case 7  
' vbNo  
    MsgBox ("Ne, 7")  
End Select  
MsgBox ("Vaše odpověď: " & i)  
End Sub
```