

1.6 Lekce 5 – Opakování, cykly a pole

V programování (nejen) VBA kódu je občas počítači nutné říct, aby vykonal určitou činnost několikrát. Jedná se například o případy, kdy je potřeba projít několik podobných dat nebo je potřeba zopakovat určitou činnost, dokud není dosaženo požadovaného výsledku.

Jedním ze způsobů implementace opakování jsou cykly. Cykly jsou příkazy, které opakují svůj vnitřní blok kódu, dokud není dosaženo nějaké podmínky. Ve VBA jsou celkově 3 základní cykly: *While*, *For* a *Do* (ve variantách *Until* a *While*).

1.6.1 Cyklus While

Definice cyklu *While* začíná klíčovým slovem *While* a končí klíčovým slovem *Wend*. Za příkaz *While* následně patří podmínka, která bude celý cyklus řídit. Pokud má daná podmínka hodnotu *PRAVDA*, cyklus se bude opakovat. Ke kontrole podmínky dochází vždy při začátku cyklu a pokud její hodnota bude *NEPRAVDA*, cyklus bude v daném místě ukončen.

```
Dim i As Integer
i = 0

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

Obrázek 34 – Ukázka kódu *While* cyklu

1.6.2 Cyklus For

Na rozdíl od cyklu *While* se cyklus *For* snaží mít proměnou, která se posunuje k určitému cíli. Může se jednat o číselný index nebo jinou hodnotu. Při cyklu *For* se ve výchozím stavu nekontroluje žádná ukončovací podmínka. Cyklus se vždy posune o další běh, dokud nedosáhne svého posledního běhu.

```
' Počítání do 20
For i = 0 To 20
    MsgBox (i)
Next i
```

Obrázek 35 – Ukázka kódu *For* cyklu

Při deklaraci cyklu *For* se definuje proměnná cyklu a přidělí se jí počáteční celočíselná hodnota. Hodnota je následována klíčovým slovem *To*, které označuje cílovou hodnotu, které

cyklus musí dosáhnout k jeho ukončení. Po konečné hodnotě může příkaz pokračovat klíčovým slovem *Step*, které určuje velikost kroku, o který se proměnná po doběhnutí cyklu zvýší.

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

' Zobrazí "0 2 4 6 8 10 12 14 16 18 20 "
MsgBox (zprava)
```

Obrázek 36 – Ukázka kódu *For* cyklu se zvýšeným krokem

Krok cyklu je zakončen klíčovým slovem *Next*, které označuje konec bloku kódu, který se opakuje. Příkaz *Next* zároveň provede potřebný posun proměnné cyklu dle určené velikosti kroku. Tento příkaz by měl vždy být následován názvem proměnné cyklu, aby bylo v kódu určeno, kterou proměnnou má posunovat – více v podkapitole „Vnořené cykly.“

1.6.3 Cyklus Do

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

```
Dim i As Integer
i = 0

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

Obrázek 37 – Ukázka kódu *Do While* cyklu

Druhou variantou je cyklus *Do Until*, který na první pohled vypadá obdobně jako cyklus *Do While*. Rozdíl je v tom, že cyklus pokračuje, když je podmínka hodnoty *NEPRAVDA* a ukončí se v momentě, kdy je kontrolní podmínka hodnoty *PRAVDA*.

```
Dim i As Integer
i = 0

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

Obrázek 38 – Ukázka kódu *Do Until* cyklu

1.6.4 Vnořené cykly

V rámci programování ve VBA je možné použít i více cyklů vnořených do sebe, klidně i pomocí jiných příkazů. Jednotlivé cykly lze předčasně ukončit pomocí klíčového slova *Exit* následované názvem příkazu cyklu k ukončení (*Exit For*, *Exit Do*, *Exit While*). Opuštění cyklu bývá doprovázeno určitou podmínkou, která určuje situaci, ve které se má cyklus ukončit.

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

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

    i = i + 1
Loop
```

Obrázek 39 – Ukázka kódu s předčasným ukončením cyklu

Při použití více cyklů příkazu *For* je nutné přiřadit správné proměnné cyklu pomocí příkazu *Next*. Jinak by mohlo dojít v chybě logiky nebo kompilátoru při spuštění špatně napsaného příkazu.

```
zprava = ""

' Vytvoří 4 řádky s hvězdičkami
For radky = 1 To 4 Step 1
    For hvezdciky = 1 To 10
        zprava = zprava & "*"
    Next hvezdciky

    ' Přidá nový řádek na konec
    zprava = zprava & vbNewLine
Next radky

' Zobrazí:
' *****
' *****
' *****
' *****
MsgBox (zprava)
```

Obrázek 40 – Ukázka kódu s vnořeným cyklem *For*

Ve VB.NET je v rámci cyklů používán příkaz *Continue* (*Do* | *While* | *For*), který pokračuje s další iterací cyklu v momentě, kde byl použit. Jinými slovy příkaz ukončuje běh aktuálního

bloku kódu a spouští cyklus znovu, jako by byl celý blok kódu dokončen. VBA tuto funkci postrádá a vývojář musí psát logiku svých cyklů bez potřeby této funkce.

1.6.5 Rekurzivní funkce

Dalším způsobem opakování jsou rekurzivní volání funkce. V principu jde o volání dané funkce uvnitř bloku kódu této funkce. Volání by vždy mělo mít určitou ukončovací podmínku, která zamezí nekonečnému volání funkce. Kdyby volání nebylo ukončeno, program by vyčerpal veškerou paměť, kterou by měl k dispozici, a nakonec by spadl.

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

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

End Function
```

Obrázek 41 – Ukázka kódu rekurzivní funkce (faktoriál) rekurzivním způsobem

1.6.6 Pole a matice

Občas je potřeba zapsat hodnoty podobného charakteru do několika proměnných. Například řekněme, že existují data, která určují vzdálenost od jednotlivých míst trasy. Museli by se tvořit číselné proměnné *vzdalenost12*, *vzdalenost23*, *vzdalenost34* atd. Vytvářet tolik proměnných by ovšem bylo nepraktické, z tohoto důvodu existují pole (*array*).

Pole představuje jednu proměnnou určitého datového typu, která ukládá více hodnot. Hodnoty jsou indexovány od 0 do hodnoty o jedna menší, než je velikost pole. Pomocí indexu lze k určité hodnotě v poli přistoupit a přečíst ji nebo přepsat.

Ukázka indexování pole o velikosti 10 prvků										
Index prvku	0	1	2	3	4	5	6	7	8	9
Pořadí prvku	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.

Tabulka 9 – Ukázka indexování pole

V našem případě by se pole mohlo jmenovat *vzdalenosti* a pod indexem 0, by mohla být vzdálenost ze startu k prvnímu bodu, v indexu 1 by mohla být vzdálenost od prvního bodu k druhému bodu atd.

```
' Dynamické pole
Dim vzdalenosti() As Variant
vzdalenosti = Array(3.14, 3.2, 4, 5.2, 6.8, 2.7, 0.5, 13, 0.2, 0.8)

' Statické pole s omezenou velikostí
Dim sVzdalenosti(10) As Double

For i = 0 To 10
    sVzdalenosti(i) = vzdalenosti(0)
Next i

' Vypíše první vzdálenost
MsgBox (vzdalenosti(0))
```

Obrázek 42 – Ukázka deklarace statického a dynamického pole

Pole se převážně zapisují a čtou pomocí *For* a jiných cyklů, protože čtení z pole je opakující se činnost, která potřebuje iterační proměnnou. Statické pole má omezené množství prvků podle jeho deklarace. Množství je omezeno celý číslem v závorce deklarovaného pole. Dynamické pole se deklaruje bez čísel a nemá omezené množství.

```
' Dynamické pole
vzdalenosti = Array(3.14, 3.2, 4, 5.2, 6.8, 2.7, 0.5, 13, 0.2, 0.8)

soucetVzdalenosti = 0
' Index má maximum 9, protože pole má 10 prvků
For i = 0 To (10 - 1)
    soucetVzdalenosti = soucetVzdalenosti + vzdalenosti(i)
Next i

' Zobrazí 39,54
MsgBox (soucetVzdalenosti)
```

Obrázek 43 – Ukázka součtu všech prvků v poli pomocí cyklu *For*

Kromě jedno rozměrových polí existují vícedimenzionální pole. Pro jednoduchou představu lze říct, že dvourozměrné pole (matice) má pro každý svůj prvek další vlastní jednodimenzionální pole. Dvoudimenzionální pole mají dva indexy pro určení hodnoty, představující souřadnice – podobně jako je řádek a sloupec v tabulce. Pro procházení matic je potřeba použít vnořené cykly.

```
' První číslo představuje souřadnici rok, druhé číslo pak měsíc
Dim ziskMesic(2001 To 2023, 1 To 12) As Currency

ziskMesic(2001, 12) = 1235485.35

' Zobrazí zisk 1235485 Kč v lednu 2001
MsgBox (ziskMesic(2001, 12))
```

Obrázek 44 – Ukázka kódu vytvoření matice

1.6.7 Cyklus For Each

V poli a jiných kolekcích, které lze procházet, lze použít *For* cyklus s příkazem *Each*, které místo počítání s proměnou cyklu postupně přebere prvky v určité řadě a přiřadí je do proměnné pro dané kolo cyklu. Tímto způsobem projde všechny prvky v kolekci a aplikuje pro ně naprogramovaný blok kódu. Vývojář se nemusí starat o indexy jednotlivých prvků a všechny je může projít a použít dle vhodného uvážení.

```
' Dynamické pole
vzdalenosti = Array(3.14, 3.2, 4, 5.2, 6.8, 2.7, 0.5, 13, 0.2, 0.8)

soucetVzdalenosti = 0
' Index se neřeší, protože jej cyklus postupně prochází
For Each vzdalenost In vzdalenosti
    soucetVzdalenosti = soucetVzdalenosti + vzdalenost
Next vzdalenost

' Zobrazí 39,54
MsgBox (soucetVzdalenosti)
```

Obrázek 45 – Ukázka kódu použití cyklu *For Each*

1.6.8 Úkoly k procvičení

- 1) Vytvořte funkci *Pyramida* s vnořeným cyklem *For*, které pomocí znaku #, vytvoří ze symbolů 5 patrovou pyramidu v buňce sešitu.

```
#  
###  
#####  
#####  
#####
```

- 2) Vytvořte makro *Nasobilka* s dvoudimenzionálním polem, o rozměrech 10x10, které bude obsahovat hodnoty násobilky malých čísel od 1x1 až 10x10. Obsah matice vypište pomocí jedné zprávy.
- 3) Zkopírujte makro *Nasobilka* a přejmenujte ho na *NasobilkaObracene*. Upravte cyklus makra, tak aby vznikla nová matice otočená přes stoupající diagonálu s využitím cyklů.