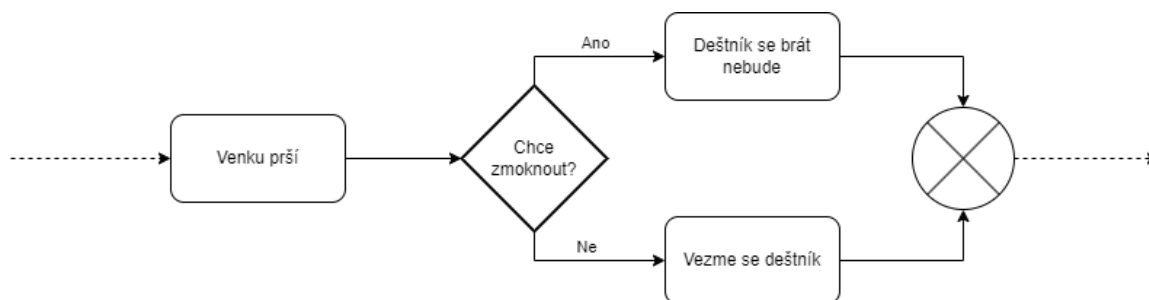


1.4 Lekce 3 – Větvení a podmíněné příkazy

Při tvorbě algoritmů vývojář narazí na místo v toku programu, kde je potřeba kód rozdělit na dvě a více možných cest. Rozdělení probíhá na základě určitých podmínek, které v rámci programu v dané situaci nastali. Pokud je podmínka pro danou větev splněna, program se rozhodne tuto větev následovat a ostatní větve jsou ignorovány, dokud se větve opět nespojí.



Obrázek 19 – Diagram příkladu více toků v algoritmu

Pro příklad, pokud bude venku pršet, uživatel si položí otázku, zda chce zmoknout? Pokud ne, musí si vzít deštník, což vyžaduje jinou činnost, než pokud by se deštník nevzal. V tomto bodě se kód větví dle informace na existující otázku.

Ve VBA existují dva hlavní způsoby pro vypořádání se s větvením algoritmu – Příkazy *If*, *Else* a *Else If* nebo příkaz *Select Case*.

1.4.1 Příkazy If, Else, Else If

Příkaz *If* lze použít samostatně, a to v případě, kdy je potřeba provést kód pouze pokud platí podmínka a není potřeba řešit nic jiného. Podmínka musí být datového typu *Boolean*, a pokud nabývá hodnoty *PRAVDA*, provede se příslušný kód. Tento blok kódu začíná na novém řádku po klíčovém slově *Then* a je ukončen klíčovými slovy *End If*.

```
' Kód
If podmínka Then
    ' Kód, který se spustí, pokud je podmínka PRAVDA (True)
End If
' Pokračování kódu
```

Obrázek 20 – Struktura jednoduchého podmíněného příkazu

Příkaz *ElseIf* přidává další možnost větvení kódu, pokud první podmínka (u příkazu *If*) obsahuje hodnotu *NEPRAVDA*. Blok pod *ElseIf* se spustí v případě, že jeho podmínka je opět *PRAVDA*. Tento příkaz s bloky kódu se může opakovat v libovolném množství a může se

vyzkoušet stejné libovolné množství podmínek. Konečný příkaz *Else* již nepotřebuje žádnou podmínku, protože kód se spustí vždy, pokud žádný z předchozích příkazů *If* a *ElseIf* nebyl *PRAVDA*.

```

If podmínka Then
    ' Kód pro první podmínku
ElseIf jinaPodminka Then
    ' Kód pro jinou podmínku
Else
    ' Kód pro všechny ostatní případy
End If

```

Obrázek 21 – Struktura příkazu *If*, *ElseIf* a *Else*, ukázka větvení

1.4.2 Porovnávací operátory

Kromě proměnných lze v příkazech *If*, *ElseIf* a *Else* použít porovnávací operátory, k dosažení logické (*Boolean*) hodnoty pro správné spuštění kódu. Porovnávací operátory lze použít, pokud se snažíme získat logickou hodnotu i na jiných místech, například při ukládání do proměnné nebo při získání návratové hodnoty z funkce.

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

Tabulka 6 – Tabulka porovnávacích operátorů a jejich popis činnosti

Pro příklad, uživatel má vstupní číslo, u kterého je potřeba zjistit, zda je záporné, kladné, nebo nula. Problém je možné vyřešit pomocí příkazu *If*, který rozdělí cestu algoritmu dle potřeby. Potřebné jsou k tomu ovšem podmínky, které splňují určená pravidla. K získání těchto podmínek je možné použít správně dosazené porovnávací operátory do porovnání čísla společně s číslem nula.

```

' Proměnná s číslem
cislo = 11

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

```

Obrázek 22 – Ukázka kódu použití porovnávacích operátorů

1.4.3 Logické operátory

Využití logických operátorů nastává v případě, kdy je potřeba použít více podmínek dohromady a přidat k nim další možnou rozšiřující logiku.

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

Tabulka 7 – Tabulka logických operátorů ve VBA

1.4.4 Příkaz Select Case

Alternativou k příkazům *If*, *ElseIf* a *Else* je příkaz *Select Case*, který jako vstup přijme hodnotu (většinou z proměnné), kterou se následně snaží přirovnat ke všem možnostem, které má nadefinované. Možnosti jsou nadefinované pomocí klíčového slova *Case* následované možnou hodnotou. Pokud žádná hodnota nevyhovuje, lze nadefinovat možnost *Case Else*, která bude výchozí možností, pokud nebudou odpovídat možnostem jiným.

```

' Proměnná obsahující nějakou hodnotu
moznost = 1
' Ukázka použití Select Case
Select Case moznost
    Case 1
        MsgBox ("Vybrána možnost číslo 1.")
    Case 2
        MsgBox ("Vybrána možnost číslo 2.")
    Case 3, 4
        MsgBox ("Vybrána možnost číslo 3 a/nebo 4.")
    Case Else
        MsgBox ("Vybrána jiná možnost.")
End Select

```

Obrázek 23 – Ukázka kódu použití příkazu *Select*

V rámci jedné možnosti lze nadefinovat i více hodnot, které stačí oddělit pomocí čárky. Blok kódu pod takovou možností se spustí, pokud bude platit jedna z nadefinovaných variant. V rámci možnosti *Case* lze použít i porovnávací operátor vůči porovnávané hodnotě. Takový případ vyžaduje použít klíčové slovo *Is* následované operátorem a hodnotou.

```

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

End Function

```

Obrázek 24 – Ukázka kódu použití příkazů *Select*, *To* a *Is*

Pokud je potřeba vytvořit možnost pro řadu čísel, lze použít příkaz *To*. Například, pokud chceme, aby možnost byla spustitelná pro čísla od 0 po 10, použije se *Case 0 To 10*, které zajistí tuto hranici.

1.4.5 Podmíněné funkce

Psaní příkazů *If* a *Switch* je poměrně textově rozsáhlé, a ne vždy vývojář potřebuje takové množství řádků. K dosažení jednoduchých podmíňovacích příkazů lze i za pomoci podmíňovacích funkcí *Switch*, *IIf* a *Choose*.

Funkce *IIf* vrací jednu ze dvou definovaných hodnot na základě podmínky, která je v prvním parametru funkce. Dochází ke zjednodušení zápisu příkazů *If* a *Else*, pokud vývojář

potřebuje určit pouze jednu hodnotu, na základě nějaké podmínky. Výstup funkce je ovšem omezen pouze na jednu ze dvou možností a více možností s touto funkcí nelze použít.

```
Function VysledekZBoduIIf(Optional pocetBodu = 100)
    VysledekZBoduIIf = IIf(pocetBodu > 50, "Uspěl", "Neuspěl")
End Function
```

Obrázek 25 – Ukázka kódu použití funkce *IIf*

Funkce *Switch* zjednodušuje formát příkazu *Select Case* a definici jeho možností, pokud je potřeba vrátit pouze jednu hodnotu a nepoužít celý blok kódu. Funkce má neomezené množství parametrů a vždy se střídá podmínka, která pro hodnotu musí platit, a hodnota, která se navrátí, pokud podmínka platí.

```
Function VysledekZBoduSwitch(Optional pocetBodu = 100)
    VysledekZBoduSwitch = Switch(pocetBodu > 100, "Exceloval", _
        pocetBodu > 50, "Uspěl", True, "Neuspěl")
End Function
```

Obrázek 26 – Ukázka kódu použití funkce *Switch*

Průchod podmínek probíhá od první k poslední a jakmile je určitá podmínka splněna, vrátí její návratovou hodnotu a dále nepokračuje. To znamená, že i když platí více podmínek, návratová hodnota funkce bude mít hodnotu první, která je *PRAVDA*.

Princip funkce *Choose* spočívá ve vybírání z možností. Prvním parametrem funkce je hodnota typu kladného čísla. Tato hodnota dle pořadí určuje, která z hodnot se použije. První hodnota (druhý parametr funkce) je pod indexem 1 a ostatní (další parametry funkce) se po jedné navyšují.

```
Function VysledekZBoduChoose(Optional pocetBodu = 100)
    ' Znamka musí nabývat hodnot 1, 2, 3 - ty určí, která možnost z Choose se vybere
    ' Stupnice je po 50, proto dělení, funkce RoundUp zaokrouhluje na celá čísla nahoru
    ' Zaokrouhlením vznikají čísla 1, 2, 3, 4, ...
    znamka = WorksheetFunction.RoundUp(pocetBodu / 50, 0)
    VysledekZBoduChoose = Choose(znamka, "Neuspěl", "Uspěl", "Exceloval")
End Function
```

Obrázek 27 – Ukázka kódu použití funkce *Choose*

1.4.6 Úkoly k procvičení

- 1) Vytvořte funkci *JePrestupnyRok* s číselným parametrem *rok*, která vrátí hodnotu *PRAVDA* pokud je rok přestupný a hodnotu *NEPRAVDA*, pokud rok přestupný není.
- 2) Vytvořte funkce *DenVTydmu* s parametrem *den*, který bude představovat pořadí dne v týdnu. Návrátová hodnota funkce bude vracet (text) řetězec s názvem daného dne anebo řetězec „*CHYBA*“, pokud vstup nebude odpovídat danému dni.
- 3) Vytvořte funkci *Prospel*, která bude vracet text „Prospěl“ nebo „Neprospěl“ v sešitě *zaci.xlsx*. Pro úspěšné hodnocení musí mít žák alespoň 250 bodů a maximálně 3 absence.
- 4) Vytvořte funkci *KvadratickaRovnice* pro výpočet kořenů kvadratické rovnice. Parametry funkce budou *a*, *b*, *c*. Funkce bude vracet hodnoty „*Nemá reálné kořeny.*“ nebo dvě hodnoty spojené v řetězci pomocí „*a*“.