ÚVOD DO JAZYKA VBA

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OBSAH PŘEDNÁŠKY

Lexikální struktura

Datové typy

Konstanty, proměnné a pole

Operátory

Příkazy pro řízení běhu kódu

Uznávaná praxe doporučuje

LEXIKÁLNÍ STRUKTURA

- kód VBA mohou tvořit
  - příkazy Option
    příkazy, které upřesňují vlastnosti kódu na úrovni daného modulu
  - deklarace
    výčet identifikátorů, datových typů, příp. hodnot a rozměrů konstant, proměnných nebo polí, které programátor předává VBA k užívání
  - procedury Function
    bloky příkazů, které provádějí výpočty a vrací hodnotu nebo pole hodnot
  - procedury Sub (též makra)
    bloky příkazů, které provádějí automatickou činnost
  - procedury vlastností
    speciální procedury používané v modulech tříd objektů
  - komentáře
    poznámky programátora, které popisují výkonný kód

LEXIKÁLNÍ STRUKTURA

- příkazy na témže řádku musí být odděleny dvojtečkou, příkazy na samostatných řádcích se neoddělují
- příkazy mohou být libovolně dlouhé; velmi dlouhé příkazy lze zalomit na více řádků vložením mezery a podtržítka do místa zalomení
- VBA nerozlišuje velká a malá písmena
  - editor automaticky upravuje velikost písmen u klíčových slov, funkcí, příkazů, objektů, vlastností a metod tak, jak je definována ve VBA
  - editor automaticky upravuje velikost písmen v názvech proměnných, konstant, polí a objektů tak, aby odpovídala té, kterou uživatel uvedl v jejich deklaraci, anebo té, kterou uživatel uvedl naposledy
- VBA ignoruje mezery, tabulátory a nové řádky kromě těch, které jsou součástí řetězcových konstant nebo zalomení příkazu
  - editor automaticky vkládá mezery mezi operátory

LEXIKÁLNÍ STRUKTURA

- komentáře se uvozují apostrofem a mohou se vkládat za příkaz nebo na nový řádek
  - editor barevně odlišuje komentáře od různých prvků výkonného kódu
- datovými hodnotami v kódu (též literály) mohou být
  - čísla (např. 12; -34.78; +3.2e-4; -5E+9.7; &FF00AA)
  - textové řetězce (např. "Příliš žluťoučký kůň")
  - datumy* a časy (např. #12/31/2009#; #10:41:30#; #10:41:30 12/31/2009#)
  - logické hodnoty (TRUE nebo FALSE)
- vlastní identifikátory (= názvy proměnných, funkcí, maker atd.)
  - musí začínat písmenem a nesmí obsahovat *mezeru* . @ ! # $ & %
  - nesmí překročit délku 254 znaků
  - nesmí označovat názvy rezervovaných slov VBA

* tvar data je měsíc/den/rok bez ohledu na nastavení prostředí Windows

DATOVÉ TYPY

- datové typy určují způsob, jakým jsou hodnoty ukládány v paměti
- VBA má vestavěno 14 datový typů:
  - Boolean pro logické hodnoty TRUE a FALSE
  - Byte, Integer a Long pro celá čísla
  - Single a Double pro desetinná čísla
  - Currency pro celá čísla s posunutým řádem
  - Decimal pro celá i desetinná čísla
  - Date pro kalendářní data
  - Object pro objekty
  - String pro řetězce s pevnou délkou
  - String pro řetězce s proměnnou délkou
  - Variant* pro libovolné číselné hodnoty až do rozsahu typu Double
  - Variant* pro řetězce libovolné délky

* paměťově nejnáročnější s nejpomalejším přístupem k hodnotě

DATOVÉ TYPY

- datový typ se definuje pomocí klíčového slova As
  - v deklaraci za identifikátorem každé proměnné, konstanty, pole
  - v hlavičce procedury za identifikátorem každého vstupního argumentu
  - v hlavičce funkce za výčtem vstupních argumentů (pro návratovou hodnotu funkce)
- datové typy lze měnit pomocí konverzních funkcí CBool, CByt, CCur, CDate, CDbl, CDec, CInt, CLng, CSng, CStr, CVar
- datové typy lze zjišťovat pomocí funkcí VarType nebo TypeName

KONSTANTY, PROMĚNNÉ A POLE

- konstanta resp. proměnná je pojmenovaná pozice v paměti určená pro uložení konstantní resp. proměnné hodnoty
- pole je pojmenovaná pozice v paměti určená pro uložení jedné nebo několika množin hodnot, které se v poli navzájem odlišují pomocí speciálního identifikátoru – klíče, který
  - může být pouze celočíselný
  - začíná implicitně od 0, ale pokud se uvede na začátku modulu příkaz Option Base 1, bude začínat od 1 pro všechna pole v modulu
- deklarace konstant, proměnných a polí včetně datových typů před jejich použitím nejsou povinné, ale… pokud programátor deklaraci a/nebo datový typ neuvede, VBA provede deklaraci automaticky a použije datový typ Variant
- pokud se uvede na začátek modulu příkaz Option Explicit, deklarace bude vyžadována, jinak kód modulu nebude zkompilován a vykonán

KONSTANTY, PROMĚNNÉ A POLE

- způsob deklarace závisí na oboru platnosti hodnot:
  - lokální konstanty, proměnné a pole
    - platí uvnitř procedury a po jejím ukončení ztrácí svou hodnotu
    - deklarují se uvnitř procedury před jejich prvním použitím zápisem

      Const *Konstanta* As *DatovyTyp* = *Hodnota*

      Dim *Promenna* As *DatovyTyp*

      Dim *Pole*()*|*Pole*(*klic*$_{max}$)**|*Pole*(*klic*$_{min}$ To *klic*$_{max}$)** As *DatovyTyp*
  - statické proměnné a pole
    - platí uvnitř procedury, ale po jejím ukončení uchovávají svou hodnotu
    - deklarují se uvnitř procedury před jejich prvním použitím zápisem

      Static *Promenna* As *DatovyTyp*

      Static *Pole*()*|*Pole*(*klic*$_{max}$)**|*Pole*(*klic*$_{min}$ To *klic*$_{max}$)** As *DatovyTyp*

* prázdné závorky značí dynamické pole, které nemá daný počet prvků

** v případě vícerozměrných polí se další rozměry v závorkách oddělují čárkou

KONSTANTY, PROMĚNNÉ A POLE

- způsob deklarace závisí na oboru platnosti hodnot:
  - modulové konstanty, proměnné a pole
    - platí uvnitř všech procedur modulu
    - deklarují se v modulu před první procedurou zápisem

      Const *Konstanta* As *DatovyTyp* = *hodnota*

      Dim *Promenna* As *DatovyTyp*

      Dim *Pole*()*|*Pole*(*klic*$_{max}$)**|*Pole*(*klic*$_{min}$ To *klic*$_{max}$)** As *DatovyTyp*
  - veřejné (též globální) konstanty, proměnné a pole
    - platí uvnitř všech procedur všech modulů (= v celém projektu)
    - deklarují se v libovolném modulu VBA před první procedurou zápisem

      Public Const *Konstanta* As *DatovyTyp* = *hodnota*

      Public *Promenna* As *DatovyTyp*

      Public *Pole*()*|*Pole*(*klic*$_{max}$)**|*Pole*(*klic*$_{min}$ To *klic*$_{max}$)** As *DatovyTyp*

* ** viz předcházející snímek

KONSTANTY, PROMĚNNÉ A POLE

- POZOR na skupinovou deklaraci v řádku – VBA automaticky použije datový typ Variant pro všechny konstanty, proměnné a pole, které nemají uveden datový typ
  - kód Dim *Promenna1*, *Promenna2* As *DatovyTyp* je chybný
  - kód Dim *Promenna1* As *DatovyTyp*, *Promenna2* As *DatovyTyp* je správný
- je-li třeba deklarovat řetězec s pevnou délkou (s maximálním počtem znaků),
  je nutné za datovým typem uvést symbol * a poté počet znaků
- deklarováním číselné proměnné nebo pole číselných proměnných VBA automaticky proměnné nuluje
- k hodnotám konstant, proměnných resp. polí se přistupuje přes jejich identifikátory resp. identifikátory a klíče zápisem *Konstanta*, *Promenna* resp. *Pole(klic1, klic2, …)*

KONSTANTY, PROMĚNNÉ A POLE

- dynamické pole je před prvním přístupem nutné a později kdykoliv v kódu možné redeklarovat = nastavit/změnit počet prvků pole zápisem ReDim Preserve* *Pole*(*klic*$_{max}$)**|*Pole*(*klic*$_{min}$ To *klic*$_{max}$)** As *DatovyTyp*
- do proměnných nebo polí lze ukládat hodnoty pomocí přiřazovacího operátoru = nebo přiřazovacího příkazu (příkazu, který provede vyhodnocení výrazu za rovnítkem a výsledek přiřadí do proměnné)
- do pole lze hodnoty ukládat i pomocí funkce Array podle vzoru *Pole* = Array(*hodnota*,...) nebo *Pole* = Array(Array(*hodnota*,...),...), ale... návratová hodnota funkce je vždy datového typu Variant
- VBA disponuje stovkami vestavěných konstant: názvy excelovských konstant začínají na xl, názvy konstant jazyka VBA na vb

* zachovává stávající uložené hodnoty v poli

** v případě vícerozměrných polí se další rozměry v závorkách oddělují čárkou

KONSTANTY, PROMĚNNÉ A POLE

Ukázky deklarací:

```
Dim a
Dim a As Variant
Dim b As Boolean
Const e As Double = 2,71828182845904523536028747135 2
Public Const DnesniDatum As Date = #10/9/2009#
Public Popisek As String * 50
Dim Text As String
Dim Prijmeni(10) As String *20
Dim Bydliste(1 To 10) As String
Dim Cislo As Integer, Poradi As Integer
Dim DynamickePole1() As Byte
ReDim DynamickePole2(0 To 5, 0 To 3) As Long
ReDim Preserve DynamickePole2(0 To 10, 0 To 10) As Long
```

OPERÁTORY

- operátory jsou symboly, které specifikují typ operace mezi operandy
- VBA má vestavěno 22 operátorů:
  - přiřazovací operátor =
  - aritmetické operátory

    sčítání **+**, odčítání **–**, násobení *****, dělení **/**, celočíselné dělení ****, umocňování **^**, modulo (= zbytek po dělení) Mod
  - řetězcové operátory

    spojování řetězců **&**, porovnávání řetězce se vzorem Like
  - relační operátory

    je rovno **=**, není rovno **<>**, větší než **>**, větší nebo rovno **>=**, menší než **<**, menší nebo rovno **<=**
  - logické operátory

    negace Not, součet Or, součin And, excluze XOr, ekvivalence Eqv, implikace Imp

PŘÍKAZY PRO ŘÍZENÍ BĚHU KÓDU

- Příkaz If…Then směřuje vykonávání kódu jednou z několika možných cest podle toho, zda byly nebo nebyly splněny logické podmínky

Ukázka použití If…Then

```
If Time < 0.5 Then MsgBox "Dobré ráno"

If Time < 0.5 Then MsgBox "Dobré ráno" Else "Dobré odpoledne"

If Time < 0.5 Then
    MsgBox "Dobré ráno"
ElseIf Time>=0.5 And Time<0.75 Then
    MsgBox "Dobré odpoledne"
Else
    MsgBox "Dobrý večer"
End If
```

PŘÍKAZY PRO ŘÍZENÍ BĚHU KÓDU

- Příkaz Select…Case směřuje vykonávání kódu jednou z několika možných cest podle výsledku výrazu nebo hodnoty proměnné

Ukázka použití Select…Case

```
Select Case Time
    Case Is < 0.5
        MsgBox "Dobré ráno"
    Case 0.5 To 0.75
        MsgBox "Dobré odpoledne"
    Case Else
        MsgBox "Dobrý večer"
End Select
```

PŘÍKAZY PRO ŘÍZENÍ BĚHU KÓDU

- Příkaz For…Next opakovaně vykonává blok příkazů až do okamžiku, kdy počet opakování (hodnota počítadla) dosáhne stanovené meze (konečné hodnoty)

Ukázka použití For…Next

```
Soucet = 0
For  Pocitadlo = 0 To 10 Step 1
    If Pole[Pocitadlo] < 0 Then Exit For
    Soucet = Soucet + Pole[Pocitadlo]
Next Pocitadlo
```

PŘÍKAZY PRO ŘÍZENÍ BĚHU KÓDU

- Příkaz For Each…Next opakovaně vykonává stejný blok příkazů pro každý prvek resp. objekt uvnitř pole resp. kolekce objektů

Ukázka použití For Each…Next

```
Soucet = 0
For  Each Prvek In Pole
    If Prvek < 0 Then Exit For
    Soucet = Soucet + Prvek
Next Prvek
```

PŘÍKAZY PRO ŘÍZENÍ BĚHU KÓDU

- Příkaz Do…While opakovaně vykonává blok příkazů po dobu, po kterou platí logická podmínka (cyklus skončí, jakmile podmínka přestane platit)
- testování podmínky může být umístěno na začátku i na konci cyklu (s podmínkou na konci cyklus projde alespoň jednou)

Ukázka použití Do…While

```
Pocitadlo = 0
Soucet = 0
Do While (Pocitadlo<=10)     'opakuj, dokud platí…
    If Pole[Pocitadlo] < 0 Then Exit Do
    Soucet = Soucet + Pole[Pocitadlo]
    Pocitadlo = Pocitadlo + 1
Loop
```

PŘÍKAZY PRO ŘÍZENÍ BĚHU KÓDU

- Příkaz Do…Until opakovaně vykonává blok příkazů po dobu, po kterou neplatí logická podmínka (cyklus skončí, jakmile podmínka začne platit)
- testování podmínky může být umístěno na začátku i na konci cyklu (s podmínkou na konci cyklus projde alespoň jednou)

Ukázka použití Do…Until

```
Pocitadlo = 0
Soucet = 0
Do Until (Pocitadlo >10 Or Pole[Pocitadlo] < 0)    'opakuj, než začne platit…
    Soucet = Soucet + Pole[Pocitadlo]
    Pocitadlo = Pocitadlo + 1
Loop
```

UZNÁVANÁ PRAXE DOPORUČUJE

- kód jednotně vizuálně strukturovat a zapisovat na jeden řádek jeden příkaz
- používat komentáře
  - pro popis proměnných
  - pro popis činnosti každé procedury
  - pro popis provedených změn
  - pro popis kódu použitého neobvyklým nebo nestandardním způsobem
  - pro popis částí kódu (záplat), které slouží pro obcházení omezení Excelu nebo chyb vzniklých za běhu kódu (*run-time errors*)
  - ihned během psaní kódu nikoliv až dodatečně…
- vytvářet identifikátory podle jednotné jmenné konvence, např. používat předpony b, i, l, s, d, c, dt, str, o, v, u symbolizující jednotlivé datové typy, dílčí slova zapisovat velkými počátečními písmeny anebo oddělovat zvoleným znakem apod.

UZNÁVANÁ PRAXE DOPORUČUJE

- explicitně deklarovat všechny použité proměnné, konstanty, pole, vstupní argumenty procedur a návratové hodnoty funkcí včetně datového typu
- deklarace uvádět na začátku modulu nebo ihned za hlavičkou procedury
- používat takový datový typ, který zabírá co nejmenší počet bajtů, ale zároveň je schopen »pojmout« všechny hodnoty, které mu budou přiřazeny
- řetězcové proměnné deklarovat pokud možno o pevné, nikoliv proměnlivé délce
- explicitně inicializovat (nastavit) hodnoty všech proměnných před jejich prvním použitím ve výrazu
- není-li nutné pracovat s polem proměnných různých datových typů, nepoužívat pokud možno funkci Array k uložení hodnot do pole