08. HODINA

8.1 Funkce pro práci s řetězci:

Funkce	Popis	Příklad	Hodnota
Len(řetězec)	Délka řetězce	*Len("Visual Basic")*	12
Replace(ř1, ř2, ř3)	Nahradí v řetězci ř1, řetězec ř2 řetězcem ř3	*Replace (Ahoj,“A“,““)*	"hoj"
Split(ř1,ř2, ,metoda)	Zjistí počet výskytů podřetězce ř2 ve zdrojovém řetězci ř1, podle zvolené metody	*Split(„AAAhoj“,“A“)*	III (je nutné převést na celé číslo- UBound *)
LCase$(řetězec)	Mění velká písmena na malá, ostatní znaky zůstávají beze změny	*LCase$("Visual Basic")*	"visual basic"
UCase$(řetězec)	Mění malá písmena na velká	*UCase$("Visual Basic")*	"VISUAL BASIC"
LTrim$(řetězec)	Odstraní mezery na začátku řetězce	*LTrim$(" VB ")*	"VB "
RTrim$(řetězec)	Odstraní mezery na konci řetězce	*LTrim$(" VB ")*	" VB"
Trim$(řetězec)	Odstraní mezery na začátku i na konci řetězce	*Trim$(" VB ")*	"VB"
Left$(řetězec,Délka)	Levá část řetězce o dané délce	*Left$("Visual Basic",4)*	"Visu"
Right$(řetězec,Délka)	Pravá část řetězce o dané délce	*Right$("Visual Basic",8)*	"al Basic"
Mid$(řetězec,pozice, Délka)	Část řetězce od dané pozice v dané délce	*Mid$("Visual Basic",3,2)*	"su"

cvičný 1
cvičný 2
cvičný 3
cvičný 4

Funkce můžeme psát i bez znaku *$* na konci jejich názvu, například *Trim(" VB ")*. Pak jejich funkční hodnota je typu *Variant* místo *String*. Výpočet s typem *Variant* je jen poněkud méně efektivní, jiný vliv to na aplikaci nemá.

8.2 Pole

- umožňuje odkazovat se na množinu proměnných,
- používá číslo – index k jejich odlišení,
- všechny prvky pole jsou stejného typu (není-li to pole hodnot typu Object).

8.2.1 Deklarace polí s pevnou délkou

- pole deklarujeme nastavením horní hranice v kulatých závorkách:

```
Dim Pole(18) As String
```

  Toto pole má 19 prvků s indexy 0-18,
- můžeme také definovat spodní hranici indexu:

```
Dim Pole (1 To 10) As String
```

  Toto pole má indexy v rozmezí 1 až 10,

Pole obsahující jiná pole

- můžeme deklarovat pole typu Object a zaplnit je jinými poli různých typů.

8.2.2 Vícerozměrná pole

- deklarace:

```
Dim Pole(18,18) As String
Dim Pole(1 To 10, 2 To 5) As Integer
Dim Pole(18,10,5) As String
```

8.2.3 Dynamická pole

- dynamické pole může být kdykoliv zvětšeno nebo zmenšeno,
- deklarace:

  1.Nadeklarujeme pole:

```
Dim Pole( )
```

  2.Pomocí příkazu ReDim alokujeme aktuální počet prvků:

```
ReDim Pole(18)
```

- příkaz ReDim se může objevit jen v proceduře a používá stejnou syntaxi jako pole s pevnou velikostí (lze měnit dolní a horní hranici).

Ochrana obsahu dynamického pole

- vždy když se použije ReDim je obsah pole ztracen,
- změnit horní hranici bez ztráty hodnot lze pomocí klíčového slova Preserve a funkce Ubound (lze však měniy pouze horní mez posledního rozměru):

```
ReDim Preserve Pole(Ubound(Pole)+1) As String
ReDim Preserve Pole(10, Ubound(Pole,2)+1) As
String
```

Příklad procedury *Nastav*, která všechny prvky číselného pole nastaví na zvolenou hodnotu (1, str, 321):

```
Sub Nastav (Pole() As Integer, ByVal Hodnota As Integer)
    Dim i As Integer
    For i =LBound(Pole) To UBound(Pole)
        Pole(i) = Hodnota
    Next i
End Sub
```

Je-li parametrem pole, jeho meze se v deklaraci formálního parametru v proceduře neuvádějí. Funkce *LBound* a *UBound* vrací dolní a horní hranici dimenze pole *.

Při volání procedury se parametry nedávají do závorek, jen se oddělí od jména procedury mezerou:

8.3 Cykly

Do…Loop

- použijeme tehdy, jestliže neznáme definitivní počet opakování cyklu.

Do While

```
Do While podmínka
     příkazy
Loop
```

- provádí se tehdy, jestliže je podmínka splněna.

Do Loop While

```
Do
     příkazy
Loop While podmínka
```

- cyklus se alespoň jednou provede.

Do Until

```
Do Until podmínka
     příkazy
Loop
```

- provádí se tehdy, jestliže podmínka není splněna.

Do Loop Until

```
Do
     příkazy
Loop Until podmínka
```

- cyklus se alespoň jednou provede.

For…Next

- použije se, pokud víme, kolikrát cyklus chceme provést.
-

```
For počítadlo = start To konec [Step přírůstek]
     příkazy
Next [počítadlo]
```

- provádění cyklu postupuje takto:
  1. Nastavení *počítadla* na hodnotu *start*.
  2. Otestuje se, zda je *počítadlo* větší než *konec*. Pokud ano, opustí se cyklus. (Je-li přírůstek záporný, testuje, je-li *počítadlo* menší než *konec*.)
  3. Provede *příkazy*.
  4. Zvětší hodnotu počítadla o 1 nebo o *přírůstek*, je-li nastaven.
  5. Opakuje kroky 2-4.

Novinkou ve Visual Basic (verze NET) je možnost deklarace proměnné přímo součástí cyklu pro zpřehlednění.

```
For a As Integer = 1 To 10 [Step přírůstek]
    příkazy
Next [počítadlo]
```

For Each…Next

- cyklus se opakuje pro každý prvek ve skupině objektů nebo poli.

```
For Each prvek In skupina
    příkazy
Next prvek
```

- pro použití platí tato omezení:
  - pro skupiny (kolekce) může být *prvek* pouze typu Object (dříve Variant), obecného typu Object, nebo objekt uvedený v prohlížeči objektů,
  - pro pole může být *prvek* pouze proměnná typu Object,
  - nelze použít pro pole uživatelského typu.

Opuštění řídící struktury

- opuštění řídící struktury lze pomocí příkazu **Exit**,
- podle toho, jakou řídící strukturu opouštíme, napíšeme příslušný exit:
  - Exit For,
  - Exit Do,
  - Exit Sub (opuštění procedury),
  - Exit Function (opuštění funkce).