

## 1.3 Lekce 2 – Proměnné, datové typy a výpočty

### 1.3.1 Proměnné

Při vytváření procedur je občas zapotřebí ukládat určité hodnoty, výsledky výpočtů a jiná data, které algoritmus zapsaný v kódu potřebuje. K tomuto účelu slouží proměnné, které alokují místo v paměti počítače a uloží do něj tyto potřebné hodnoty.

Deklarace proměnných ve VBA proběhne, pokud se dříve nepoužitému názvu (názvu proměnné) přiřadí požadovaná hodnota. Přiřazení hodnoty probíhá napsáním názvu proměnné následovaného znakem rovná se a hodnotou, kterou chce programátor do proměnné uložit.

Pro přečtení hodnoty z proměnné se na místo v kódu napíše název proměnné. Při běhu kódu pak počítač na toto místo vloží hodnotu proměnné, kterou má uloženou v paměti. Jedná se o podobný princip jako při používání parametrů ve funkcích.

Proměnná a její hodnota je v paměti počítače, dokud běh programu nedosáhne klíčového slova *End*. Pokud je proměnná deklarována mimo procedury, její platnost vyprší až s uzavřením celého programu.

### 1.3.2 Datové typy

Kontext hodnot v proměnných je udržován pomocí datových typů. V paměti počítače jsou tyto hodnoty zapsány pomocí jedniček a nul a bez kontextu se nedá určit, jestli je tato kombinace číslo, text, datum nebo něco kompletně odlišného. Pro vývojáře i počítač zároveň značí, jak s těmito hodnotami zacházet (alokovat místo v paměti, používání určitých funkcí, ...).

Binární číslo v paměti	0100 0001 0100 0000
Číslo (desítková soustava)	16 704
Text (UTF-16; znak)	A ( <i>U+0041</i> )
Datum	24.09.1945

Tabulka 1 – Ukázka příkladu reprezentace dat a kontextu

Pokud je proměnná definována implicitně bez datového typu, přiřadí se jí automaticky datový typ *Variant*, který se přizpůsobuje na základě vložené hodnoty. To vyžaduje vyšší náklady na paměť a řízení práce s proměnou. V dnešní době jsou tyto vyšší náklady při vyšších výkonech počítačů a větších kapacit pamětí zanedbatelné, ale v minulosti mohly značit

problémy. Datový typ *Variant* zároveň umožňuje vývojářům představit v chodu programu chyby v podobě načtení dat jiného datového typu do proměnné, než je potřeba.

Z těchto důvodů se většinou vývojáři v jazyce VBA snaží implicitní deklaraci vyvarovat a místo ní použít deklaraci explicitní, která zaručí, že v deklarované proměnné může být pouze jeden datový typ. Při přidělení jiného datového typu do takto deklarované proměnné program ohlásí chybu.

Vývojář si může nastavit tvrdé vyžadování explicitní deklarace pomocí klíčového spojení *Option Explicit*, které by se mělo uvádět na začátku každého modulu. Toto nastavení omezí implicitní deklaraci pro celý modul a nutí vývojáře uvádět datové typy. Při spuštění procedury s implicitní deklarací se zobrazí chyba kompilátoru s hláškou „*Proměnná není definována.*“

Datový typ	Význam	Hodnoty
<b>Boolean</b>	Logická hodnota	True / False (Pravda / Nepravda)
<b>Integer</b>	Celé číslo	-32 768 až 32 767
<b>String</b>	Text (řetězec znaků)	0 až 2 <sup>32</sup>
<b>Double</b>	Desetinné číslo	-1,79769313486231E308 až -4,94065645841247E-324 pro záporná čísla 4,94065645841247E-324 až 1.79769313486232E308 pro kladná čísla
<b>Date</b>	Datum	1. ledna 100 až 31. prosince 9999
<b>Currency</b>	Měna	-922 337 203 685 477,5808 až 922 337 203 685 477,5808
Long	Celé číslo	-2 147 483 648 až 2 147 483 647
LongLong	Celé číslo	-9 223 372 036 854 775 808 až 9 223 372 036 854 775 807
Byte	Bajt	0 až 255
Single	Desetinné číslo	-3.402823E38 až -1.401298E-45
<b>Variant</b>	Automaticky přizpůsobitelný datový typ	Podle přiřazené hodnoty
<b>Object</b>	Objekt	Podle vlastností objektu
Type	Uživatelsky definovaný datový typ	Libovolné (dle použitých typů)

Tabulka 2 – Tabulka důležitých datových typů a jejich význam a hodnoty

Datový typ *Boolean* je vhodné použít v případě, pokud je potřeba vyjádřit v logice programu stav o nějakém předmětu – například zapnuto a vypnuto, nebo platí a neplatí. Ostatní použití

datových typů (*Integer, Double, Date, String, Currency, ...*) v logice algoritmu je poměrně odvoditelné od jejich významu.

### 1.3.3 Operátory při práci s proměnnými

Do proměnných lze ukládat i výsledky výpočtu, které lze provést pomocí aritmetických operátorů, jiných proměnných a konstant. Číselné konstanty mohou být celá nebo desetinná čísla, která se ovšem musí vždy psát s desetinnou tečkou a nikoliv čárkou.

Operátor	Popis
+	Sečte dvě čísla (na levé a pravé straně)
-	Odečte od levého čísla pravé číslo
*	Vynásobí dvě čísla (na levé a pravé straně)
/	Podělí číslo na levé straně pravým číslem
Mod	Modulo, zjistí celočíselný zbytek po dělení levého čísla pravým číslem
^	Umocní levé číslo pravým číslem

Tabulka 3 – Tabulka aritmetických operátorů ve VBA

Výpočet může obsahovat závorky pro nejvyšší přednost ve výpočtu. Počítač nadále vypočítá veškerá umocnění ve výpočtu. Pokud jich je v příkladu více, postupuje z levé strany na stranu pravou. Po umocnění následuje násobení, dělení a modulo vše v pořadí zleva doprava. Na závěr počítač provede veškerá sčítání a odečítání opět ve stejném pořadí z levé strany.

```
Dim x As Integer
x = 10 + 5 * 2 ^ 3 / 4
'Výsledek x bude 20
```

Obrázek 17 – Ukázka kódu s příkladem priorit výpočtu

Pro práci s řetězcem znaků v proměnných existují pouze dva operátory, které řetězce spojují. Ostatní funkce pro práci s textem nelze provádět pomocí operátorů a musí být použity specializované funkce o kterých bude zmínka později.

Operátor	Popis
+	Spojí řetězec znaků na levé straně s řetězcem na straně pravé
&	Spojí řetězec znaků na levé straně s řetězcem na straně pravé

Tabulka 4 – Tabulka spojovacích operátorů ve VBA

```
levaStrana = "ABE"
pravaStrana = "CEDA"
dohromady = levaStrana + pravaStrana
```

Obrázek 18 – Ukázka kódu s použitím spojovacího operátoru

### 1.3.4 Převody mezi datovými typy

Pro případy, kdy je například potřeba převést řetězec obsahující číslo na číselnou hodnotu, se kterou lze počítat existují tzv. Funkce převodu typů (anglicky *Type conversion functions*). Tyto funkce se ze vstupního argumentu (přijímané hodnoty) pokusí vytvořit hodnotu s požadovaným datovým typem. Název funkce tvoří velké písmeno C (z anglického *convert*) následované zkratkou potřebného návratového datového typu.

Název funkce	Návratový datový typ	Přijímané hodnoty
<b>CBool()</b>	Boolean	Validní textová nebo číselná hodnota.
<b>CByte()</b>	Byte	0 až 255.
<b>CCur()</b>	Currency	-922 337 203 685 477,5808 až 922 337 203 685 477,5808.
<b>CDate()</b>	Date	Validní datum.
<b>CDbl()</b>	Double	-1,79769313486231E308 až -4,94065645841247E-324 pro záporná čísla 4,94065645841247E-324 až 1.79769313486232E308 pro kladná čísla.
<b>CInt()</b>	Integer	-32 768 až 32 767 (zaokrouhluje desetinná).
<b>CLng()</b>	Long	-2 147 483 648 až 2 147 483 647.
<b>CLngLng()</b>	LongLong	-9 223 372 036 854 775 808 až 9 223 372 036 854 775 807.
<b>CSng()</b>	Single	-3.402823E38 až -1.401298E-45.
<b>CStr()</b>	String	Různé, většinou z nich vyjadřuje vstup převedený na text.
<b>CVar()</b>	Variant	Podle typu.

Tabulka 5 – Tabulka funkcí převodu datových typů pro proměnné

### 1.3.5 Úkoly k procvičení

- 1) Vytvořte funkce *palecNaCm* a *cmNaPalec*, které budou převádět mezi jednotkami palec a centimetry. (1 palec = 2,54 cm).
- 2) Vytvořte funkci *obvodObdelniku*, která vypočítá obvod obdélníku na základě vložených parametrů *stranaA* a *stranaB*.
- 3) Vytvořte funkci *prevodNaCela*, která bude převádět desetinná čísla na celá.
- 4) Vytvořte funkci *jeSude*, která vrátí *PRAVDA* nebo *NEPRAVDA* podle toho, jestli je vstupní číslo sudé nebo ne. Použijte operátory pro výpočet a funkce pro převod datových typů. Otestujte funkci v sešitě Excel na několika číslech.
- 5) Vytvořte funkci *obsahKruhu*, která vypočítá obsah kruhu na základě argumentu poloměru  $r$ . Pro přesnější  $\pi$  použijte funkci *WorksheetFunction.Pi()*.