

# Obsah

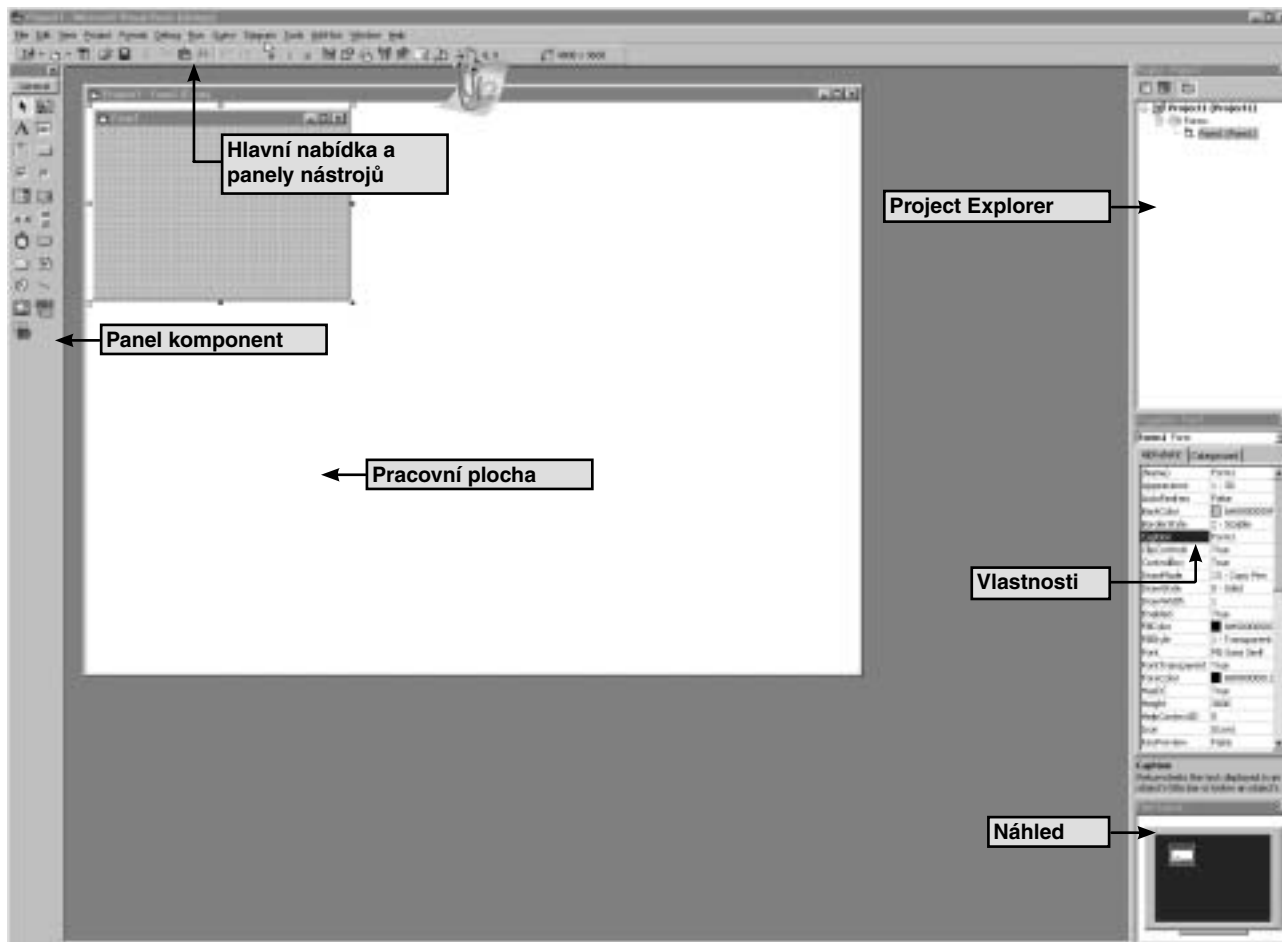
<b>Vysvětlivky k prvkům použitým v knize .....</b>	<b>6</b>
<b>Slovo autora .....</b>	<b>7</b>
<b>Prostředí jazyka Visual Basic .....</b>	<b>8</b>
CO JE PROJEKT .....	9
Vytvoření nového projektu .....	9
OTEVŘENÍ EXISTUJÍCÍHO PROJEKTU .....	10
<i>Přidání existujícího formuláře do projektu a odstranění formuláře z projektu .....</i>	<i>10</i>
ULOŽENÍ PROJEKTU .....	11
<b>Základy objektového programování.....</b>	<b>12</b>
OBJEKTY .....	12
VLASTNOSTI .....	12
UDÁLOSTI .....	13
METODY .....	13
<b>Identifikátory.....</b>	<b>13</b>
PROCEDURY A FUNKCE .....	14
KLÍČOVÁ SLOVA .....	14
ROZSAH PLATNOSTI IDENTIFIKÁTORŮ .....	15
<b>Vkládání ovládacích prvků a psaní kódu.....</b>	<b>16</b>
VKLÁDÁNÍ OVLÁDACÍCH PRVKŮ .....	16
<i>Práce s ovládacím prvkem .....</i>	<i>16</i>
PSANÍ ZDROJOVÉHO KÓDU .....	16
Pravidla a tipy pro psaní zdrojového kódu .....	18
SPOUŠTĚNÍ PROGRAMU .....	21
VYTVOŘENÍ SPUSTITELNÉHO SOUBORU (EXE).....	21
<b>Komponenty a jejich použití.....</b>	<b>22</b>
FORMULÁŘ (FORM) .....	22
MODUL (MODULE) .....	24
TEXTOVÉ POLE (TEXTBOX).....	24
POPISEK (LABEL) .....	25
TLAČÍTKO (COMMANDBUTTON) .....	26
ZATRŽÍTKO (CHECKBOX) .....	26
PŘEPÍNAČ (OPTION) .....	27
SEZNAM (LIST) A ROLETKA (COMBO).....	28
RÁMEČEK (FRAME).....	29
DALŠÍ KOMPONENTY .....	30
<b>Proměnné a konstanty .....</b>	<b>30</b>
PROMĚNNÉ.....	30
<i>Datové typy .....</i>	<i>31</i>
<i>Deklarace lokálních proměnných .....</i>	<i>32</i>
<i>Deklarace pole .....</i>	<i>32</i>
<i>Použití pole a odkaz na jeho položky.....</i>	<i>34</i>
<i>Deklarace globálních proměnných.....</i>	<i>34</i>
KONSTANTY .....	35
<b>Operátory a výrazy.....</b>	<b>36</b>
PŘÍKAZ PŘIŘAZENÍ .....	36
OPERÁTORY .....	36
<i>Aritmetické operátory .....</i>	<i>37</i>
<i>Logické a bitové operátory .....</i>	<i>37</i>
<i>Relační operátory.....</i>	<i>38</i>
<i>Řetězcové operátory.....</i>	<i>40</i>
PRIORITA OPERÁTORŮ .....	41

PŘÍKLADY .....	43
<b>Příklad 1</b> .....	<b>43</b>
<b>Příklad 2</b> .....	<b>43</b>
<b>Příklad 3</b> .....	<b>43</b>
<b>Souhrnný příklad 1</b> .....	<b>45</b>
VYTVOŘENÍ NOVÉ APLIKACE .....	45
VYTVOŘENÍ VZHLEDU APLIKACE .....	45
VYTVOŘENÍ ZDROJOVÉHO KÓDU .....	46
<b>Podmíněné příkazy</b> .....	<b>48</b>
PODMÍNĚNÝ PŘÍKAZ IF... THEN .....	48
PODMÍNĚNÝ PŘÍKAZ SELECT CASE .....	50
VOLBA PODMÍNĚNÉHO PŘÍKAZU .....	53
PŘÍKLADY .....	54
<b>Příklad 1</b> .....	<b>54</b>
<b>Příklad 2</b> .....	<b>55</b>
<b>Příklad 3</b> .....	<b>55</b>
<b>Příkazy cyklu</b> .....	<b>56</b>
CYKLUS FOR... NEXT .....	56
CYKLUS DO... LOOP .....	58
VNOŘENÉ CYKLY .....	60
PŘÍKLADY .....	61
<b>Příklad 1</b> .....	<b>61</b>
<b>Příklad 2</b> .....	<b>62</b>
<b>Příklad 3</b> .....	<b>62</b>
<b>Příklad 4</b> .....	<b>63</b>
<b>Příklad 5</b> .....	<b>64</b>
<b>Příklad 6</b> .....	<b>64</b>
<b>Příklad 7</b> .....	<b>65</b>
<b>Příklad 8</b> .....	<b>66</b>
<b>Souhrnný příklad 2</b> .....	<b>68</b>
VYTVOŘENÍ VZHLEDU APLIKACE .....	68
VYTVOŘENÍ ZDROJOVÉHO KÓDU .....	68
<b>Vestavěné funkce Visual Basicu</b> .....	<b>70</b>
MATEMATICKÉ FUNKCE .....	71
PŘÍKLADY NA MATEMATICKÉ FUNKCE .....	72
<b>Příklad 1</b> .....	<b>72</b>
<b>Příklad 2</b> .....	<b>73</b>
<b>Příklad 3</b> .....	<b>73</b>
<b>Příklad 4</b> .....	<b>74</b>
ŘETĚZCOVÉ FUNKCE .....	74
PŘÍKLADY NA ŘETĚZCOVÉ FUNKCE .....	76
<b>Příklad 1</b> .....	<b>76</b>
<b>Příklad 2</b> .....	<b>76</b>
<b>Příklad 3</b> .....	<b>76</b>
FUNKCE PRO PRÁCI S DATEM A ČASEM .....	77
PŘÍKLADY NA FUNKCE DATA A ČASU .....	79
<b>Příklad 1</b> .....	<b>79</b>
<b>Příklad 2</b> .....	<b>80</b>
<b>Příklad 3</b> .....	<b>80</b>
<b>Příklad 4</b> .....	<b>80</b>
<b>Příklad 5</b> .....	<b>81</b>
PŘEVODNÍ FUNKCE .....	82
PŘÍKLADY NA PŘEVODNÍ FUNKCE .....	84
<b>Příklad 1</b> .....	<b>84</b>
<b>Příklad 2</b> .....	<b>85</b>

ZJIŠŤOVACÍ A OVĚŘOVACÍ FUNKCE .....	86
PŘÍKLADY NA ZJIŠŤOVACÍ A OVĚŘOVACÍ FUNKCE .....	87
Příklad 1 .....	87
Příklad 2 .....	88
<b>Uživatelské funkce a procedury .....</b>	<b>88</b>
VYTVÁŘENÍ FUNKCÍ A PROCEDUR .....	88
<i>Umístění zdrojového kódu funkcí a procedur</i> .....	89
<i>Předčasné opuštění těla procedury či funkce</i> .....	89
<i>Lokální proměnné</i> .....	89
<i>Volání podprogramů</i> .....	89
PŘÍKLADY .....	90
Příklad 1 .....	90
Příklad 2 .....	90
Příklad 3 .....	91
Příklad 4 .....	92
Příklad 5 .....	92
PARAMETRY VOLANÉ ODKAZEM A HODNOTOU .....	93
PŘÍKLADY .....	94
Příklad 1 .....	94
Příklad 2 .....	95
REKURZIVNÍ FUNKCE .....	96
PŘÍKLADY .....	97
Příklad 1 .....	97
Příklad 2 .....	97
<b>Aplikace s více formuláři .....</b>	<b>98</b>
SDI APLIKACE .....	98
<i>Předávání informací mezi formuláři</i> .....	100
Příklad 1 .....	100
Příklad 2 .....	101
MDI APLIKACE .....	102
<i>Tvorba nabídky v MDI formuláři</i> .....	103
<b>Souhrnný příklad 3 .....</b>	<b>104</b>
VYTVOŘENÍ VZHLEDU APLIKACE .....	104
VYTVOŘENÍ ZDROJOVÉHO KÓDU .....	106
<b>Práce se soubory .....</b>	<b>109</b>
PŘÍKAZY PRO MANIPULACI SE SOUBORY .....	109
PŘÍKAZY A FUNKCE PRO MANIPULACI S DATY V SOUBORECH .....	111
<i>Otevření a uzavření souboru</i> .....	111
<i>Funkce sloužící ke zjišťování vlastností souboru</i> .....	113
<i>Čtení dat ze souboru</i> .....	113
<i>Zápis dat do souboru</i> .....	117
PŘÍKLADY NA PRÁCI SE SOUBORY .....	119
Příklad 1 .....	119
Příklad 2 .....	119
Příklad 3 .....	120
<b>Souhrnný příklad 3 - pokračování .....</b>	<b>121</b>
<b>Ošetřování chybových stavů .....</b>	<b>122</b>
<b>Testování aplikací a hledání chyb .....</b>	<b>123</b>
KROKOVÁNÍ .....	124
SLEDOVÁNÍ PROMĚNNÝCH (WATCHES) .....	124
BREAKPOINTY .....	125
PŘÍMÉ VSTUPY .....	126
PŘÍKLAD LADĚNÍ APLIKACE .....	126

## Prostředí jazyka Visual Basic

Než se budete věnovat samotnému programování, musíte se seznámit s prostředím Visual Basicu. Jedná se o typický program pro operační systém Windows. Novější verze, kterou se budete učit, je určena výhradně pro 32bitovou verzi Windows, tj. Windows 95/98/2000. Hardwarové požadavky jsou na dnešní dobu velmi skromné. Program se dá úspěšně využívat na starším stroji s Pentiem na frekvenci 200 MHz, 64 MB paměti RAM a několika sty MB volného místa na pevném disku (závisí na typu instalace). Visual Basic je dodáván jako kompletní vývojové prostředí. To znamená, že se uživateli dostává do rukou kompletní sada nástrojů pro vývoj nových aplikací. S jednotlivými částmi prostředí se postupně seznámíte v jednotlivých kapitolách této knihy. Program komunikuje v angličtině, takže je nanejvýš vhodné znát alespoň základy tohoto světového jazyka. V opačném případě budete odkázáni na to, že si vše zapamatujete. Základní podoba prostředí Visual Basicu je vidět na obrázku.



Na obrázku vidíte hlavní součásti prostředí Visual Basicu, které budete při své práci využívat. Je to jednak hlavní nabídka a panely nástrojů. Ty představují cestu ke všem příkazům a funkcím prostředí. Dále je to panel komponent, který budete využívat pro výběr komponent, ze kterých se bude skládat váš program. Máte na výběr například textová pole, popisky, rozbalovací seznamy, zatržítka, přepínače či obrázky.



**Poznámka:** Toto jsou základní komponenty, které se vyskytují snad v každém programu. Visual Basic je velice modulární a umožňuje přidat další komponenty. Ty mohou plnit často velmi specifické úkoly, jako je například prohlížení html kódu, spojení s jiným PC, přehrávání MP3 apod. Některé komponenty jsou součástí Visual Basicu, další lze stáhnout bezplatně z internetu. Ostatní jsou běžné komerční zboží, za které je nutné zaplatit. Snažte se mít v projektu vždy jen ty komponenty, které jsou v něm skutečně využity. Ostatní jen zvyšují prostorové nároky výsledného instalátoru aplikace.

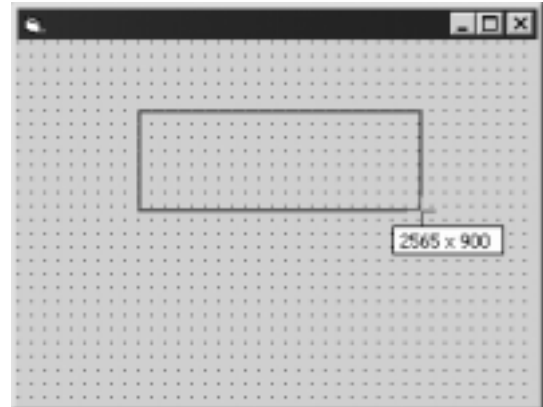
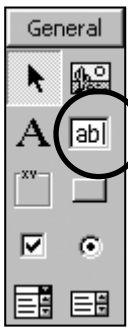
Nyní zaměřte svůj zrak na pravou stranu obrazovky. Tam se nachází Project Explorer, který slouží k rychlé orientaci v projektu. Projekt se může skládat z mnoha souborů různých typů a právě díky Project Exploreru v nich budete mít pořádek. Velice důležitá je i další součást, a to vlastnosti. Zde máte možnost nastavit vlastnosti jednotlivých komponent, které umístíte do svého projektu. Pro vizuální kontrolu, jak budou jednotlivá okna

## Vkládání ovládacích prvků a psaní kódu

Programování ve Visual Basicu se skládá ze dvou hlavních činností. První je návrh vzhledu aplikace, tedy umístění zvolených komponent na plochu formuláře tak, aby vzhled odpovídal požadavkům programátora i potenciálního uživatele výsledné aplikace. Druhou je psaní zdrojového kódu, tedy samotné programování, kdy definujeme činnosti, které se mají provést při daných událostech. Vývoj probíhá v tomto pořadí, proto se naučíte nejprve tvořit vzhled aplikace, tedy „kreslit“.

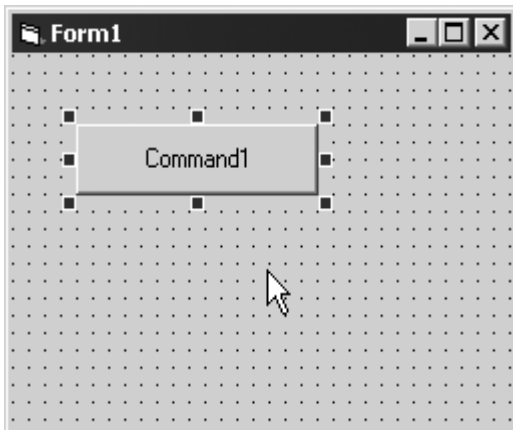
### VKLÁDÁNÍ OVLÁDACÍCH PRVKŮ

Chcete-li vložit jakýkoliv ovládací prvek na plochu formuláře, musíte jej nejprve vybrat v panelu ovládacích prvků. Poté zvolíte místo na formuláři, stisknete levé tlačítko myši a za současného držení tlačítka určíte tažením velikost objektu. Pustíte-li tlačítko myši, máte objekt vytvořený. Druhou možností je dvojklik na zvoleném ovládacím prvku, přičemž se daný ovládací prvek vytvoří uprostřed formuláře. Poté můžete změnit umístění a velikost tohoto prvku. Pokud chcete vytvořit další ovládací prvek, byť stejného typu, musíte jej opět vybrat v panelu ovládacích prvků.

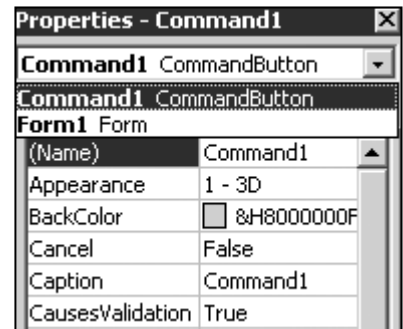


### Práce s ovládacím prvkem

Vytvořením ovládacích prvků však nekončí možnost manipulovat s tímto prvkem. Pokud s objektem chcete manipulovat, musíte jej nejprve vybrat. Výběr můžete provést více způsoby – tím nejjednodušším je klepnutí na daný prvek. Někdy však tento způsob není možný – pokud je objekt zakryt jiným objektem. V tomto případě



je nutno použít druhý způsob – v okně **Properties** (vlastnosti) vyberete zvolený objekt ze seznamu. Vybraný objekt poznáte podle modrých čtverců, které jej obklopují. Pomocí těchto čtverců můžete měnit velikost objektu. Klepnutím a následným tažením se současným držetím tlačítka myši je možné nastavit velikost objektu. Touž činností, ale klepnutím mimo čtverce přímo na objekt, můžete objekt přemístit. Všechny tyto činnosti lze samozřejmě provést



i číselně změnou příslušných vlastností (**Top**, **Left**, **Width**, **Height**).

### PSANÍ ZDROJOVÉHO KÓDU

Psaní zdrojového kódu je nejdůležitější částí programování (jedná se vlastně o samotné programování). V této kapitole se naučíte, jak máte zdrojový kód psát, ale také poznáte základní principy, které se při programování využívají. Prozatím jste se zabývali pouze jednotlivými komponentami a návrhem vzhledu aplikace. Nyní se tedy přepněte do režimu pro psaní zdrojového kódu. To můžete provést dvěma způsoby:

- Pomocí nabídky **View / Code**
- Dvojitým klepnutím na objekt, jehož kód chcete měnit či zapisovat

V tomto okně budete zapisovat veškerý zdrojový kód vašich programů. Okno má tři důležité součásti – dva seznamy a jedno textové pole. Seznamy slouží k výběru objektu a události, kterou chcete programovat. Pokud jste se do okna přesunuli prvním způsobem, pak máte okno přesně jako na obrázku – tedy objekt **General** (obecné) a událost **Declaration** (deklarace). Zde se nejedná o žádný konkrétní objekt, ale o společnou část pro celý formulář. Do této sekce budete zapisovat deklarace proměnných a vlastní funkce a procedury, které si v rámci formuláře vytvoříte. Pokud jste použili druhý způsob, pak bude objektem ten prvek,

Nejprve kód neupravený:

```
Dim A As Integer
    Dim B As Integer
    For A = 1 To 10
    For B = 1 To 10
    If (A * B) Mod 2 = 0 Then
    Print A & " * " & B & " = " & (A * B)
    End If
    Next B
Next A
```

A nyní kód upravený pomocí odsazení:

```
Dim A As Integer
    Dim B As Integer
    For A = 1 To 10
        For B = 1 To 10
            If (A * B) Mod 2 = 0 Then
                Print A & " * " & B & " = " & (A * B)
            End If
        Next B
    Next A
```

Je vidět zcela jasný rozdíl (byť se jedná o funkčně stejný kód) v přehlednosti. Zatímco v prvním případě nám kód splývá v jednom proudu, v druhém je jasně vidět, která část kódu je na jaké úrovni. V případě, že budete mít ještě více podúrovní, je tento způsob jedinou možností jak si udržet přehled o svém vlastním kódu v době tvorby, natož o kódu cizím či vlastním, který čtete po delší době. Ono je vůbec důležité uvědomit si, že se k vlastnímu kódu budete často vracet s odstupem několika měsíců či let, a i po takové době je nutné, aby pro vás kód zůstal čitelný a srozumitelný.

- Využívejte poznámky** – poznámka je část zdrojového textu, která se nevykonává, a plní tak skutečně jen roli poznámky. Poznámka je uvozena klíčovým slovem **Rem** nebo znakem apostrofu ('). Oba způsoby vloží do poznámky celý řádek. Není možné „zakomentovat“ jen část řádku. Poznámka se využívá ve dvou případech. Jedním z nich je případ, kdy programátor chce skutečně vytvořit poznámku ke zdrojovému kódu. Druhou je situace, kdy v programu je chyba – pomocí příkazu Rem je možné učinit z části zdrojového kódu poznámku, čímž jej vyřadíte z činnosti, aniž byste jej museli mazat. Podívejte se na předchozí kód, okomentovaný poznámkami.

```
Dim A As Integer 'deklarace proměnné vnějšího cyklu
    Dim B As Integer 'deklarace proměnné vnitřního cyklu
    For A = 1 To 10 'vnější cyklus
        For B = 1 To 10 'vnitřní cyklus
            If (A * B) Mod 2 = 0 Then 'rozhodnutí, zda je číslo sudé
                Print A & " * " & B & " = " & (A * B) 'výpis výsledku
            End If
        Next B 'zvýšení hodnoty proměnné B
    Next A 'zvýšení hodnoty proměnné A
```

Samozřejmě není nutné komentovat tak důsledně jako v tomto případě, ale je účelné vložit komentář k obtížnějším částem kódu, voláním funkcí (zejména vlastních) apod. či obecně tam, kde nám to jako programátorům vyhovuje. Poznámky nejsou povinné, slouží především pro pohodlí programátora. Představte si, že chcete otestovat, jak by se program choval, kdyby netestoval sudost výsledku – vložíme tedy podmínku do komentáře.

```
Dim A As Integer 'deklarace proměnné vnějšího cyklu
    Dim B As Integer 'deklarace proměnné vnitřního cyklu
    For A = 1 To 10 'vnější cyklus
        For B = 1 To 10 'vnitřní cyklus
            'If (A * B) Mod 2 = 0 Then 'rozhodnutí, zda je číslo sudé
                Print A & " * " & B & " = " & (A * B) 'výpis výsledku
            'End If
```

## Deklarace lokálních proměnných

Deklarace proměnné je její vytvoření, tedy sdělení překladači, že existuje nová proměnná daného jména a daného datového typu. Dále je pro tuto proměnnou vyhrazeno místo v paměti. Deklaraci provádíme pomocí příkazu **Dim**. (Visual Basic sice nevyžaduje deklaraci proměnných, což znamená, že je možné použít i proměnnou, která nebyla deklarována pomocí příkazu **Dim**, nicméně doporučuji deklarovat proměnné z důvodu přehlednosti a jasného určení typu proměnné). Příkaz má následující syntaxi:

```
Dim promenna [As datovytyp],
```

kde *promenna* je název nově vytvořené proměnné a *datovytyp* je určení datového typu proměnné. Všimněte si však, že část příkazu s *As* není povinná, a nemusí tedy být použita. Přesto doporučuji sdělit překladači typ proměnné. V opačném případě sice bude proměnná vytvořena, ale její typ bude tzv. **Variant** – neurčitý typ, se kterým mohou být problémy u některých funkcí nebo s porovnávacím příkazem.



**Poznámka:** *Ve skutečnosti není deklarace proměnných nutná vůbec. Visual Basic totiž nevyžaduje deklaraci proměnných. V praxi to tedy znamená, že proměnnou nedeklarujete pomocí DIM, ale přímo ji použijete v kódu. Problém je zcela stejný – proměnná nemá přesně určený typ. Proto je doporučeno proměnné deklarovat a tím přesně definovat jejich datový typ.*

Nyní tedy malý příklad, který by měl osvětlit situaci. Chcete vytvořit tři proměnné – do jedné chcete uložit jméno, do druhé plat a do třetí věk zaměstnance. Nejprve tedy rozmyslete datové typy. U jména je to jasné, tam použijete typ **String**. Plat? Musíte vědět, zda může přesáhnout hranici typu **Integer**. Pokud si jisti nejste, pak zvolte typ **Long**. Dnešní paměti jsou natolik velké, že nemusíte počítat každý byte. Věk je opět jasný. Pokud nebudete zapisovat biblické postavy, bohatě stačí typ **Byte**. Názvy proměnných zvolíte logicky česky bez diakritiky.

Deklarace tedy bude vypadat následovně:

```
Dim Jmeno As String
Dim Plat As Long
Dim Vek As Byte
```



**Upozornění:** *Chcete-li si usnadnit práci při deklaraci více proměnných najednou, můžete použít příkaz Dim ve tvaru Dim promenna1 As datovytyp1, promenna2 As datovytyp2..., takže příkaz bude vypadat například takto: Dim jmeno As String, Plat As Long, Vek As Byte. Jestliže však deklarujete více proměnných stejného typu, musíte tento typ u všech uvést. Pokud napíšete příkaz ve tvaru Dim A, B, C As Integer, počítač to pochopí tak, že chcete deklarovat proměnnou C jako proměnnou typu Integer a ostatní proměnné jako proměnné bez uvedeného typu, tedy typ Variant.*

## Deklarace pole

Prozatím jsme deklarovali pouze jednoduché proměnné. Nyní si ukážeme, jak deklarovat pole. Co je vlastně pole? Je to skupina položek stejného typu, které navenek vystupují pod jediným jménem. K jednotlivým položkám přistupujeme pomocí indexů (jednoho nebo více), které určují polohu prvku v poli. Podle toho, kolik indexů určuje jednotlivé položky pole, pak toto pole označujeme jako jedno či vícerozměrné. Jednorozměrné pole představuje například vektor, dvojrozměrné pak matici. Jak tedy nadeklarovat pole a jak k němu přistupovat? Deklarace používá stejný příkaz, tedy **Dim**. V případě pole má následující tvar:

```
Dim promenna(dolni To horni) [As datovytyp],
```

kde *promenna* a *datovytyp* mají stejný smysl jako u deklarace jednoduché proměnné. **Dolni** a **horni** jsou celá čísla, která určují rozsah přípustných indexů. Pokud chcete, aby indexování začalo od nuly, stačí zadat horní mez. Příkaz pak vypadá takto:

```
Dim promenna(horni) [As datovytyp].
```

Kde tedy můžete pole využít? Všude tam, kde máte mnoho hodnot stejného datového typu se stejným významem. Například byste chtěli spočítat průměrný počet žáků ve vaší škole v jednotlivých ročnících a potřebujete samozřejmě někde uložit tato data. V druhém případě zjišťujete, kolik dní v roce byly naměřeny určité teploty – zjišťujete tedy jejich četnost. V prvním případě tedy vytvoříte pole, kde indexem budou čísla jednotlivých ročníků a hodnotami jednotlivých položek počty žáků v jednotlivých ročnících. Vzhledem k tomu, že ročník je od prvního do devátého, pak stačí definovat pole s indexem začínajícím od nuly s tím, že prvek

## Operátory a výrazy

Výrazy jsou jedním ze základních stavebních kamenů Visual Basicu a obecně jakéhokoliv programovacího jazyka. Právě ony umožňují výpočty, zobrazování a vůbec jakékoliv přesuny informací z jednoho místa na druhé. Co vlastně výrazy jsou? Jsou to posloupnosti operandů a operátorů. Operandy jsou jakékoliv objekty, proměnné či hodnoty, se kterými ve výrazu pracujete, operátory jsou symboly zastupující jednotlivé operace, které s operandy provádíte. Výrazy se běžně používají například v matematice, příklady naleznete v tabulce.

$$2x + 7y$$

$$\frac{1+x}{y-7} \geq x+y^2$$

$$(x+y)^2$$

Ve Visual Basicu můžete tyto výrazy používat také, nicméně způsob jejich zápisů je odlišný – nemáte totiž možnost použít zlomky, indexy a různé symboly. Výše uvedené výrazy by ve Visual Basicu byly zapsány následovně:

```
2*x+7*y  
(1+x)/(y-7)>=x+y^2  
(x+y)^2
```

Jak vidíte, jedná se opravdu o posloupnosti operandů a operátorů. Místo čísel a proměnných mohou být součástí výrazu také objekty, konstanty apod. Nyní se tedy zaměříme na jednotlivé operátory, které můžete při své práci využívat.

## PŘÍKAZ PŘÍŘAZENÍ

Abyste vůbec mohli pracovat s hodnotami proměnných a vlastností objektů, musíte se nejprve naučit jak tyto hodnoty nastavit a měnit. K tomu slouží příkaz přiřazení. Ten má následující tvar:

Cíl = Výraz

Cílem může být jakákoliv proměnná či vlastnost, do které máte právo zapisovat. Dále následuje samotný příkaz přiřazení (znak =). Poslední částí příkazu je výraz, jehož hodnotu chcete zapsat do proměnné označené jako Cíl. Příklady přiřazení máte uvedeny zde. Nemusíte se trápit tím, že prozatím neznáte význam některých prvků v kódu. Všimněte si pouze příkazu přiřazení.

```
C = A + B  
Pokus = Sin(A) * B - Cos(X)  
Text1.Text = Val(Text2.Text) + Val(Text3.Text)
```



**Upozornění:** Cílem může být opravdu jen samotná proměnná nebo vlastnost některého z použitých objektů. Přes naprostou nesmyslnost tohoto konání velmi často vidím situaci, kdy je jako cíl použit rovněž výraz. Příkaz funguje tak, že vyhodnotí výraz a jeho výsledek uloží do paměťového prostoru, který je pojmenován cíl.



**Poznámka:** Příkaz, tj. znak „=“, je naprosto stejný jako níže uvedený operátor rovnosti. Jak tedy Visual Basic a programátor poznají, kdy se jedná o příkaz přiřazení, a kdy o operátor rovnosti? Je to jednoduché – příkaz přiřazení je samostatný příkaz, který není spojen s ničím jiným. Je-li výraz se znakem „=“ spojen s výstupem, podmínkou či cyklem, jedná se o relační výraz a znak „=“ plní funkci operátoru.

## OPERÁTORY

Jak již bylo řečeno, operátory jsou symboly, které zastupují jednotlivé operace. Použitím operátoru dáváte programu najevo, že chcete použít tu operaci, jejíž symbol byl použit. Každý operátor ke své činnosti vyžaduje předem daný počet operandů. Podle toho rozlišujeme operátory

## Podmíněné příkazy

Až dosud jste psali programy, jejichž zdrojový kód byl lineární. To znamená, že se příkazy vykonávaly v tom pořadí, v jakém jste je napsali, a jejich vykonávání nebylo nijak závislé na hodnotách proměnných či na uživatelských vstupech. A právě podmíněné příkazy (zkráceně podmínky) vám umožňují to změnit. V závislosti na hodnotě zvoleného výrazu umožňují větvit kód. Podmíněné příkazy se využívají velice často, a je dokonce možné říci, že žádný pokročilejší program není možné napsat bez použití podmínek.



**Poznámka:** Podmínky v programovacím jazyce plní stejnou úlohu jako v běžném životě, kde jsou rovněž velice často používány. Mají dokonce velice podobnou syntaxi – když je něco splněno, pak se provede daná činnost. Typickým příkladem je například věta „Bude-li pršet, vezmu si deštník“. Běžně však využíváme mnohem složitější podmínky – „Přijdu na oslavu, pokud tam bude Tomáš, a nebude Aleš“. Kdybychom tuto větu přepsali do matematictějšího zápisu – „(bude Tomáš) and not(bude Aleš => přijdu na oslavu“ – zjistili bychom, že běžně využíváme logické výrazy s logickými operátory, které byly dříve popsány a často o tom ani nevíme. Protože jsou podmíněné příkazy a jejich podmínky velice podobné realitě, jsou snadno pochopitelné.

Ve Visual Basicu máte k dispozici dva základní příkazy, které plní funkci podmíněných příkazů – příkaz **If** a příkaz **Select Case**. Jak je použít a který se hodí v dané situaci, se dozvíte z následujících odkazů.

### PODMÍNĚNÝ PŘÍKAZ IF... THEN

Základním příkazem podmíněného větvení je příkaz **If ... Then**. Ten se používá ve dvou základních variantách. V té jednodušší má tvar

```
If podmínka Then prikaz1 [Else prikaz2],
```

kde podmínka je logický výraz, při jehož splnění se provede příkaz (nebo příkazy) uvedené za klíčovým slovem **Then**. V opačném případě se provede příkaz (nebo příkazy) uvedené za klíčovým slovem **Else**. Hranaté závorky udávají, že část **Else** je nepovinná, a programátor tedy může vytvořit takový podmíněný příkaz, který bude reagovat jen na splnění podmínky. Jako příklad je uveden kód, který rozhodne, zda je zadané číslo větší než nula, či nikoliv.

```
Dim A As Integer
```

```
A = 26
```

```
If A > 0 Then Print „Číslo je větší než 0“ Else Print „Číslo je menší než 0“
```

Do proměnné **A** je přiřazeno námi zvolené číslo. Podmínka, která rozhoduje o výsledku, je  $A > 0$ . Pokud je výsledek tohoto výrazu **True**, pak se provede příkaz za slovem **Then**, tedy vypíše se hlášení o skutečnosti, že číslo je větší než 0. V opačném případě je i tento fakt sdělen uživateli. O něco složitější situace nastane, chcete-li rozhodnout, zda číslo patří do intervalu 0 až 100 včetně. Kód bude téměř stejný, ale podmínka bude odpovídat změněné situaci.

```
Dim A As Integer
```

```
A = 144
```

```
If A >= 0 And A <= 100 Then Print „Patří“ Else Print „Nepatří“
```

Aby číslo patřilo do intervalu, musí současně splnit dvě podmínky – musí být větší nebo rovno dolní hranici intervalu a musí být menší nebo rovno horní hranici. Požadavek rovnosti je nutný, protože dle zadání je číslo součástí intervalu, pokud je rovno některé z hranic. Požadavek současného splnění podmínek je vyjádřen logickou spojkou **And**.

Další příklad se bude týkat situace, kdy dvě čísla vyjadřují naše příjmy a výdaje. Převyšují-li příjmy výdaje, je vše v pořádku. Opačná situace, tedy výdaje převyšující příjmy, vyvolá hlášení.

```
Dim Prijmy As Integer
```

```
Dim Vydaje As Integer
```

```
Prijmy = 150
```

```
Vydaje = 190
```

```
If Vydaje > Prijmy Then Print „Máte dluh“
```

Zde je podmínka velice jednoduchá – porovnájí se příjmy a výdaje relačním operátorem **>**, a pokud je výsledek **True**, vypíše se text „Máte dluh“. Prozatím jsme se zabývali jen jednoduchými podmínkami, ve kterých mohly nastat jen dvě situace – tedy podmínka buď platí, nebo neplatí. Nebylo možné vytvářet složitější konstrukce,



**Poznámka:** Nyní, když znáte cykly a podmínky, můžete již vytvářet složitější programy. Ano, spoustu věcí ještě neznáte – zejména funkce a práci se soubory, ale pro některé úlohy to ani není třeba. Proto si zkoušejte sami vymyslet různé náměty, které poté zpracujete. A už by se nemělo jednat jen o jednoduché programky, ale o programy na úrovni posledních příkladů. Nemusí to být jen hry - můžete si také zpracovat program, který vám pomůže při řešení školních úloh.

## Souhrnný příklad 2

Dostali jste k dalšímu souhrnnému příkladu, ve kterém máte možnost procvičit si své znalosti. I tentokrát budete mít za úkol vytvořit program, stejně jako v prvním souhrnném příkladu. Ale protože jste již pokročili, bude váš úkol těžší, ale také zábavnější. Uživatel zadá dva rozsahy a operátor, který vybere ze seznamu. Váš program vytvoří tabulku matematických příkladů se zadanými rozsahy čísel a příklady také vypočte. Výslednou tabulku příkladů s výsledky uloží program do seznamu.

### VYTVOŘENÍ VZHLEDU APLIKACE

Protože jsou vaše znalosti programování na lepší úrovni, není nutné popisovat komplexně celý postup při vytváření nového projektu a umístování jednotlivých komponent. Komu by tato problematika stále nebyla jasná, nechť se vrátí ke kapitole o projektech nebo k prvním souhrnnému příkladu. Vytvořte si tedy nový projekt a umístěte na formulář ovládací prvky tak, aby vypadal jako na obrázku. Text uvnitř polí a vedle tlačítka označuje, jak mají být prvky pojmenovány. Jak vidíte, na formuláři chybí ještě dva ovládací prvky. A sice seznam, ze kterého budeme vybírat operaci, a seznam, do kterého budeme vypisovat výsledky. Pro výběr zvolíme roletku (**Combo**), jejíž vlastnosti budou následující:

- **Name:** cmbOperace
- **Style:** 2 - Dropdown List

Do samotného seznamu hodnot, tedy vlastnosti **List**, zadáme jednotlivé operace, které budeme

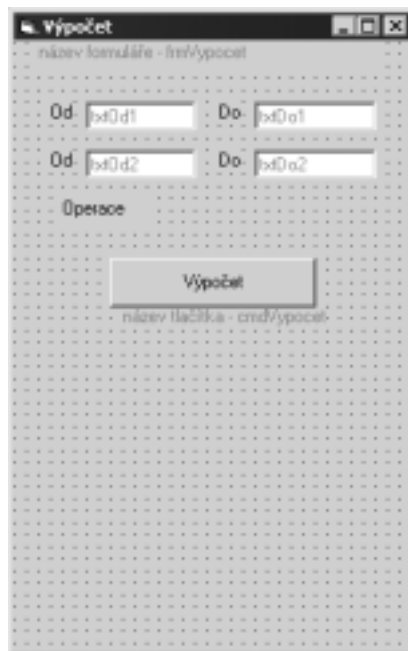
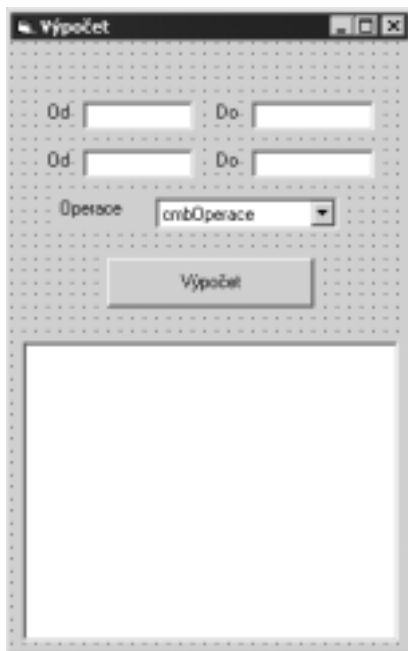
chtít v programu mít (+, -, \*). Dělení vynecháme, protože by bylo komplikované vyřešit problém možného dělení nulou. Poté umístíme klasický seznam **List** pod tlačítko **Výpočet**. Tento seznam bude sloužit k výpisu výsledků, které bude program poskytovat. Vlastnosti, jejichž hodnotu změním, jsou tyto:

- **Name:** lstVysledek

Výsledný formulář by měl vypadat přibližně jako na obrázku vlevo. Již byly odstraněny červené nápisy, které vám jen sdělovaly pojmenování objektů. Nyní máte kompletně vytvořený vzhled programu. Přejdeme tedy ke druhé fázi, tedy samotnému programování – psaní zdrojového kódu.

### VYTVOŘENÍ ZDROJOVÉHO KÓDU

Program bude vykonávat veškerou svou činnost tehdy, jestliže stiskneme tlačítko **Výpočet**. Žádný jiný objekt v programu nebude vykonávat činnost. Budeme se tedy soustředit na toto tlačítko, resp. na ošetření jeho události **Click**. Otevřeme okno pro psaní kódu a zvolíme v seznámech název tlačítka a poté událost **Click**. Jak bude vlastně program pracovat? Je třeba, aby vzal v úvahu všechny kombinace mezi prvky v rozsahu, který je určen hodnotami v polích **txtOd1** a **txtOd2**, a rozsahu, který je určen hodnotami v polích **txtDo2** a **txtDo1**. Tento požadavek splní vnořené cykly **For... Next**, z nichž jeden bude probíhat první rozsah a druhý bude probíhat druhý rozsah. Kód bude tedy následující:



Nejprve jsou deklarovány potřebné proměnné pro počet hodnot, počet číselných hodnot, hodnotu aktuálně načtenou, mezisoučet a proměnná A, která slouží jako řídicí proměnná cyklu. Dále je načten počet hodnot. Poté probíhá cyklus od 1 do tohoto počtu. V každém průchodu je načtena hodnota, která je vzápětí otestována, zda se jedná o číslo. Pokud ano, je toto číslo přičteno k součtu a je zvýšena hodnota proměnné **PocetCisel** o jedna. Na konci je vypsán součet a součet dělený počtem čísel, tzn. průměr.

## Příklad 2

Napište program, který ze vstupu (InputBox) přečte datum a uživateli sdělí, kolik dní uplynulo od tohoto data do dnešního dne. Pokud uživatel zadá hodnotu, která není datem a nelze ji na datum převést, je toto sdělení uživateli vypsáno na obrazovku a následně je běh programu ukončen.

```
Dim Datum As String
Dim PocetDni As Long
Datum = InputBox(„Zadejte datum“)
If IsDate(Datum) Then
    PocetDni = DateDiff(„d“, Datum, Now)
Else
    MsgBox „Nezadal jste správnou hodnotu data“
End
End If
Print PocetDni
```

Jak je vidět, program je velice jednoduchý – po zadání data testuje, zda zadaná hodnota opravdu odpovídá předpisům pro datum. Pokud ano, vypočítá pomocí funkce **DateDiff** počet dní mezi tímto datem a dnešním datem. V případě, že zadáte pozdější datum než dnešní, dostanete zápornou hodnotu. Pokud byste chtěli pouze kladné hodnoty, stačí změnit tento řádek na

```
PocetDni = Abs(DateDiff(„d“, Datum, Now)).
```

V opačném případě se vypíše hlášení o nepřipustné hodnotě a následně se celý program ukončí pomocí příkazu **End**. Toto řešení přesně odpovídá zadání – program se má skutečně ukončit.

## Uživatelské funkce a procedury

Až dosud jsme se zabývali vestavěnými funkcemi, tedy takovými, které jsou standardní součástí Visual Basicu. Nyní pokročíme a začneme se zabývat uživatelskými podprogramy (jak funkcemi, tak procedurami), tedy takovými, které si programátor vytvoří sám. Ačkoliv jsem se o tom již zmínil v jedné z úvodních kapitol, zopakují základní rozdíl mezi funkcemi a procedurami.

- Uživatelské funkce jsou ekvivalentem vestavěných funkcí, stejně jako ony mají návratovou hodnotu.
- Uživatelské procedury jsou ekvivalentem vestavěných příkazů, stejně jako ony provádějí určitou činnost bez návratové hodnoty.

Programátor může vytvářet jak funkce, tak procedury. Princip jejich vytváření je velice podobný. Jediný rozdíl spočívá v tom, že u funkcí je nutné pracovat s určením návratové hodnoty a jejím vrácením volajícímu programu.

## VYTVÁŘENÍ FUNKCÍ A PROCEDUR

A jak si tedy můžete vytvořit vlastní funkce a procedury? Pro tvorbu funkcí, resp. procedur, jsou určeny příkazy **Function** a **Sub**. Ty mají následující tvar:

```
Function Nazev([Par1 [as Typ1], Par2 [As Typ2], ... , ParN [as TypN]]) [As TypVystupu]
    ...
    [Nazev=NavratovaHodnota]
End Function

a

Sub Nazev([Par1 [as Typ1], Par2 [As Typ2], ... , ParN [as TypN]])
    ...
End Sub
```