Slovo autora

Protože jste sáhli po této brožuře, patrně jste se už rozhodli, že vaším vývojářským nástrojem bude (i nadále) Visual Basic. Jestliže o tom stále ještě přemítáte, neváhejte. Úsilí a náklady, které věnujete jemu a nové vývojářské platformě .NET, budou patrně vaší nejlepší investicí a na rozdíl od nejistých akciových trhů se vám vložené prostředky brzy vrátí, navíc s vysokými úroky.

Od svého vzniku se Visual Basic postupně stal nejpopulárnějším programovacím nástrojem. Odemkl a široce otevřel bránu do světa programování Windows "obyčejným lidem", protože už nevyžaduje dlouhodobé a náročné předběžné studium a přestává být výsadou vysoce kvalifikovaných specialistů.

Převratné schopnosti nového vývojového prostředí, jehož integrovanou součástí Visual Basic.NET je, umožňují inteligentním lidem začít téměř okamžitě vyrábět plnohodnotné aplikace Windows i Web, služby Web a další druhy specializovaných řešení, která splňují potřeby skutečného světa a jejichž prototypy se sestaví velmi rychle.

Charakteristickými rysy Visual Basic.NET jsou především snadnost práce kombinovaná s neobyčejně mocnými a všestrannými schopnostmi, jeho univerzálnost a využitelnost v mnohých, často velmi odlišných kontextech. Navíc je soběstačný, takže se nebudete muset nic dalšího učit, ani dokupovat.

Zdravím všechny čtenáře a přeji mnoho úspěchů na palubě .NET. Máte-li k obsahu brožury nebo k jejímu vzhledu či uspořádání jakékoli připomínky či náměty, buďte tak laskaví a pošlete mi zprávu na adresu

janpokorny@volny.cz

O B S A H

ISBN 80-86097-73-0

PCWorld Edition – Úvod do Visual Basic.NET

Vyšlo v srpnu 2001

Než začnete

Vážená čtenářko, vážený čtenáři. V ruce držíte brožuru, která by vám měla pomoci hladce vplout do vod Visual Basic.NET (plánují se i další, specializované brožury). Soustřeďuje se na seznámení s uživatelským rozhraním nové verze, na základy programovacího jazyka a tvorby formulářů Windows. Je určena především těm z vás, kdo máte první kroky s Visual Basicem za sebou nebo přicházíte odjinud a chcete se seznámit s tímto převratným vývojovým prostředím a populárním programovacím jazykem.

U nás je jen velmi málo lidí, kteří by už někde "nepřičichli" k nějakému Visual Basicu (například napsali nějaký skript nebo jednoduché procedury při práci v některé z aplikací Office). Tito potenciální čtenáři budou v brožuře asi také hledat hlavní novinky a rozdíly, které je v nové verzi čekají. Pro ně jsem zařadil jako součást výkladu konkrétní ukázkové aplikace připomínky označené „**VB6**“ všude tam, kde přichází v úvahu uživatelský postup, programovací prvek či konstrukce, které se považují za zastaralé, z hlediska doporučovaných programátorských zvyklostí nevhodné, v nové verzi se mají dělat jinak nebo se už nepodporují.

Z důvodu omezeného rozsahu brožury se některé témata probírají velmi stručně a rychle, proto to není příručka pro úplné programátorské elévy, kteří nevědí vůbec nic o tom, co jsou programy, proměnné, běžné příkazy, programové konstrukce (podmínky, cykly apod.) a neznají principy objektově orientovaného programování. Jste-li úplní začátečníci, měli byste si asi nejprve přečíst nějakou úvodní příručku. Typické publikace tohoto druhu vydává například *Microsoft Press* v edici *Step By Step*).

Co budete potřebovat

Abyste si mohli vyzkoušet vše, co je v brožuře uvedeno, musíte mít kopii aplikace Visual Basic.NET, která tvoří součást nové verze Visual Studio.NET. Než ji budete moci využívat, budete možná muset předběžně něco stáhnout a nainstalovat z Internetových stránek Windows Update (Internet Explorer 5.5, Nástroje Internetu, aktualizace samotného systému Windows, balíčky aktualizací týkající se problému roku 2000 apod.). Nasměruje vás na ně instalační disk sloužící pro aktualizaci komponent Windows (v průběhu instalace můžete přímo přejít na patřičné Internetové adresy). Teprve pak se nainstalují další potřebné komponenty a zahájí vlastní instalace Visual Studia .NET.

Budete také možná muset modernizovat svůj počítač. Celé Visual Studio zabírá přes 3 GB (nemusíte ale zdaleka instalovat všechno, například celou MSDN, čímž ušetříte spoustu místa). Hlavně ale budete potřebovat rychlý počítač. Například Pentium 120 s pamětí 32 je v podstatě za hranicí únosnosti, protože se budete drtivou většinu času dívat na přesýpací hodiny a poslouchat cvrkání pevného disku.

Co se naučíte

Po přečtení brožury byste se měli vyznat v části vývojového prostředí Visual Basic.NET, která se týká tvorby základních jednovrstevných aplikací Windows a zvládnout ty prvky programovacího jazyka a formulářů Windows, které k takovým aplikacím potřebujete. Čtenářům, kteří mají nějaké předběžné zkušenostmi s tvorbou projektů typu standard EXE ve Visual Basicu 6.0, by brožura měla pomoci v tom, jak se nyní řeší běžné akce na formulářích a v kódu, aby je nemuseli pracně vyhledávat v obrovité a ne vždy přehledné elektronické dokumentaci.

Svazek o rozsahu přibližně 80 stran nemůže pochopitelně ani vyjmenovat, tím méně pokrýt všechny aspekty všech schopností vývojového prostředí Visual Basic.NET. Chcete-li si znalosti týkající se Visual Basicu.NET resp. samotného Visual Studia.NET prohloubit, poohlédněte se po nějaké objemnější publikaci věnované tomuto tématu.

Brožura také existuje v elektronické podobě-na CD. Disk obsahuje také ukázkový projekt Visual Basic.NET zde popisovaný.

Plnohodnotné aplikace rychle

Při budování aplikačních programů v současném, rychle se rozvíjejícím a měnícím se světě, vystupuje stále více do popředí požadavek na rychlost vývoje těchto programů (včera bylo pozdě). Co největší vývojářská rychlost by však neměla nepříznivě ovlivňovat kvalitu aplikačních programů, které by i při "bleskovém" vývoji měly být vyspělé, spolehlivé, plnohodnotné a měly by to být aplikace skutečného světa, které vyhovují i potřebám obrovitých globálních korporací. Nejlepším nástrojem, který plně vyhovuje těmto požadavkům (rozhodně je to cíl společnosti Microsoft), by mělo být nové Visual Studio.NET, jehož integrovanou součástí je i nový Visual Basic.NET.

Nepodceňujte předběžnou analýzu

Hned na začátku bych ale chtěl připomenout, že sebelepší nástroje samy o sobě nic hodnotného nevytvoří. Chcete-li úspěšně budovat aplikační programy, musíte především "umět řešit úlohy". Vlastnímu programování, vizuálnímu i ručnímu psaní kódu, by měla předcházet fundovaná analýza předložené úlohy. Pominu-li obecné otázky související se systémem, sítí, zabezpečením, distribucí apod., měli byste provést alespoň rozbor toku dat (jaká vstupní data a v jaké struktuře bude aplikace potřebovat, jak se budou data zpracovávat a jaké mají poskytovat výstupy) nebo toku úloh (jaké dílčí úlohy má aplikace řešit, v jakém pořadí, v jakých návaznostech a jaká vstupní data jednotlivé úlohy potřebují) nebo obojího. Teprve na podkladě předběžných analýz byste měli navrhnout (nebo vymyslet) algoritmy řešení, volbu komponent atd.

Při řešení mnoha obecných problémů i pro potřeby předběžných analýz ostatně také asistuje řada vhodných vizuálních nástrojů (namátkou: různí průvodci distribucí, instalací, FrontPage pro vytvoření struktury a správu stránek Web nebo Microsoft Projekt pro přípravu řešení složitých projektů).

I když ale předběžné práce hodláte dělat převážně ve vyspělých vizuálních nástrojích, ale nemáte přitom žádné nebo jen malé zkušenosti s věcnou či programovou analýzou předložených úloh, doporučuji vám, abyste si obstarali nějakou knihu věnovanou tomuto okruhu problémů. Protože ve většině aplikací skutečného světa budou vstupní data v podobně nějaké relační databáze, možná byste si také měli přečíst nějakou knihu o tom, jaké zásady se mají dodržovat při navrhování a modifikacích struktury relačních databází a jaké prostředky jsou k dispozici pro údržbu a správu dat, které jsou v těchto databázích uloženy.

Vývojový cyklus řešení

Teprve na samém konci celkového procesu tvorby aplikačního programu se nachází vývojový cyklus konkrétního řešení. Vytvoření prototypu, jeho odladění a ověření, modifikace prototypu na základě požadavků vyplývajících z testování prototypu a dodatečných nebo změněných požadavků zadavatelů, odladění a ověření "překopaného" prototypu, další změny a stále kolem dokola, dokud aplikační program nefunguje po všech stránkách uspokojivě.

Jak se učit nový produkt

Jak asi víte, je objem současných produktů, (ty, které tvoří součást Visual Studia.NET jsou toho typickým příkladem), tak obrovský, že vyvolává v mnoha lidech pocit malomyslnosti, protože se prostě nedá v rozumné době ani všechno *přečíst*, natož tomu porozumět, tím méně umět vše aktivně využívat.

Proto se při seznamování s daným produktem nesnažte vstřebávat všechno do posledního detailu. Trvalo by vám to možná tak dlouho, že aniž byste vytvořili něco, za co vám někdo zaplatí, přešly by v tichosti okolo vás další verze a většinu svých, v potu tváře těžce nabytých znalostí, byste si mohli strčit za klobouk. Je třeba zvolit jinou strategii. Vyjděte z toho, že v dnešní době v podstatě žádný produkt stoprocentně zvládnout nejde. Informací je prostě příliš mnoho a příliš rychle zastarávají, takže je třeba se jejich přílivu spíše *bránit*, ne se je snažit za každou cenu absorbovat.

Zahájení prací

Dost řečí, je nejvyšší čas dělat něco konkrétního. S vývojovým prostředím Visual Basic.NET a s jeho programovacím jazykem se seznámíte při řešení jednoduché úlohy, která vás bude provázet celou brožurou.. Protože každou dílčí úlohu lze řešit obvykle více způsoby a nedá se vždy jednoznačně říci, který je nejlepší, ukážeme si několik variant jejich řešení. Zabijeme tím přinejmenším tři mouchy jednou ranou: seznámíte se s více algoritmy, s více programovacími prvky a s postupy, které už používat nejde.

Specifikace řešené úlohy

Dostali jste za úkol sestrojit jednoduchý hrací automat v podobě formuláře, který má mít tyto schopnosti:

- Umožní uživateli zadat počáteční podmínky, jako jsou počáteční částka, kterou chce prohrát a výše sázky. Až uživatel všechno prohraje, bude moci dát v plen další libovolnou částku (obecně stanovit nové počáteční podmínky). Při zadávání vstupních hodnot poskytnete uživateli nápovědu v podobě vysvětlivek.
- Uživatel si zahraje tak, že klepne na tlačítko Roztočit (nebo zvolí ekvivalentní příkaz z nabídky Akce formuláře). Automat vygeneruje trojčíslí od 000 do 999. Uživatel obdrží malou výhru, budou-li dvě číslice stejné (například 225), podstatně větší výhru při postupce (například 234 nebo 765) a nejvyšší výhru při všech třech stejných číslicích (například 555).

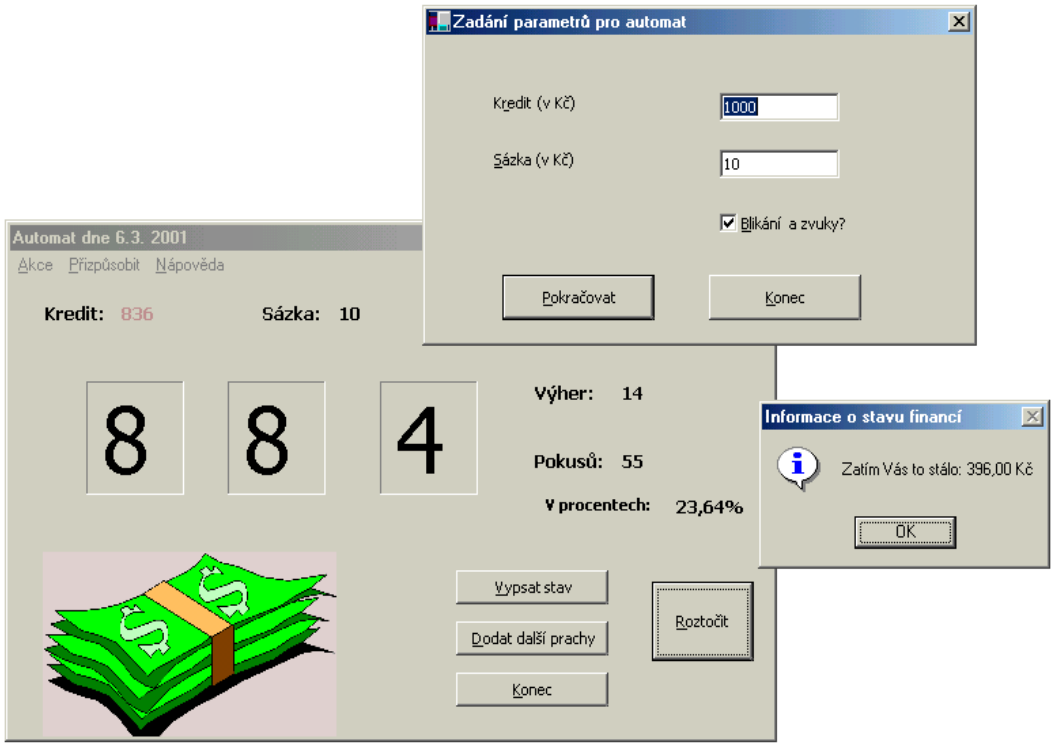
  Ukážeme si několik variant, jak je možné losovaná čísla získat.
- Podle výše výhry se zobrazí odpovídající obrázek s balíkem prachů. Pro každý druh výhry se zobrazí jiný obrázek. Jestliže uživatel vyhraje, bude dále odměněn "cingrlátky". Ozve se potlesk a formulář bude blikat. Tyto efekty bude moci uživatel zastavit v daném kole hry nebo je úplně vypnout jako součást počátečních podmínek.

  Při řešení této části aplikace se seznámíte s tím, jak se volají funkce API a jak se na formulářích pracuje s časovačem, který patří mezi tzv. neviditelné komponenty.
- Na formuláři se bude uživateli zobrazovat průběžný stav od poslední vložené částky (bude-li prohrávat, bude částka červeně), případně další statistické údaje. Celkový stav se ale zobrazí uživateli jen na přání (aby ho od dalších her stále neodrazovala celková výše částky, kterou doposud prohrál).
- Výše výher bude interní záležitost aplikace, včetně způsobu, jakým se generuje „náhodné“ trojčíslí.
- Uživatel bude moci aplikaci také ovládat pomocí nabídek. Na hlavním formuláři bude mít k dispozici pruh nabídek s vysouvacími nabídkami. Na formuláři parametrů bude mít jako ekvivalent rozhodovacích ovládacích prvků k dispozici místní nabídku.
- Jako součást systému nabídek bude mít uživatel aplikace možnost, aby si ji mohl jistým způsobem přizpůsobit. Bude moci změnit pozadí formuláře a některé atributy písma, jímž se zobrazují vylosované číslice.

  V této části úlohy se seznámíte s dalšími dvěma neviditelnými komponentami – pro zobrazení společných dialogových oken *Otevřít* a Písmo.

Poznámka. Duchovním otcem tohoto druhu hracího automatu je patrně Michael Halvorson, který ho uvedl v knize „*Microsoft Visual Basic 4 - Step By Step*“. V brožuře jsem variace na toto téma zvolil proto, že je tento typ aplikace u posluchačů na školeních Visual Basicu, která jsem měl tu čest lektorovat, zdaleka nejpopulárnější aplikací.

Takhle bude aplikace Automat vypadat, až bude zhruba hotová (uživatel v tomto tahu získal malou výhru a vypsal si právě, jak si stojí):



Spuštění Visual Basic.NET

Za předpokladu, že máte nové "studio" nainstalované, spustíte ho příkazem Start > Programy > Microsoft Visual Studio.Net 7.0 > Microsoft Visual Studio.Net 7.0. Nejprve se zobrazí úvodní obrazovka. Při její levé dolní straně uvidíte ikony nainstalovaných vývojářských nástrojů z rodiny Visual Studia. Tato obrazovka po chvíli sama zmizí a na pracovní ploše se objeví primární okno vývojového prostředí se svou domovskou stránkou.

Hypertextové odkazy v levém panelu úvodní stránky poskytují informace o novinkách a různé další odkazy. V pravé části jsou uvedeny hypertextové odkazy, s jejichž pomocí můžete otevírat projekty, s nimiž jste pracovali naposled (při úplně prvním spuštění zde nebude nic), otevřít nějaký existující projekt či založit nový projekt. Zrušíte-li zaškrtnutí políčka vpravo dole, při příštím spuštění se úvodní stránka už nezobrazí. Úvodní operace můžete urychlit, vytvoříte-li si svůj *profil*.

1. Klepněte na hypertextový odkaz My Profile.
2. Na domovské stránce (VS Home Page) se zobrazí pět rozevíracích seznamů, v nichž určíte o jaký profil se jedná, rozložení klávesnice, rozvržení okna, jak se má omezit rozsah nápovědy (tento seznam vidíte rozevřený na obrázku dále) a co se má zobrazovat při příštích spuštěních.

Prototyp aplikace Windows

S prostředím Visual Basic.NET se seznámíte tak, že sestrojíte prototyp aplikace typu "formulář Windows", který postupně nabalíte o další schopnosti tak, aby vykonával činnosti zmíněné na začátku této části brožury. Při tvorbě počátečního prototypu se nejprve naučíte:

- Nastavovat vlastnosti formuláře v návrhovém režimu
- Nastavit nápis v titulkovém pruhu formuláře až při běhu
- Sestrojit na formuláři příkazové tlačítko, jímž uživatel ukončí běh aplikace.

Přizpůsobení prostředí projektu

Jak vidíte na obrázku na příští straně, skládá se prostředí projektu VB především z mnoha oken (některá z nich nejsou vidět). Patří mezi ně především *návrhář formuláře*, v němž se zobrazí prázdný výchozí formulář. Jeho vzhled si postupně upravíte a nakladete na něj ovládací prvky ze soupravy nástrojů. Vpravo nahoře se nachází okno *Průzkumníka řešení* (Solution Explorer), v němž se zobrazuje stromová struktura řešení (co vše je jeho součástí). Pod ním vidíte *okno vlastností* (Properties) objektu vybraného v průzkumníkovi řešení. Pod ním se nachází okno *dynamické nápovědy* (Dynamic Help). Tato okna jsou *připoutána v doku*, který tvoří pravý okraj primárního okna.

Nechcete-li pracovat s připoutanými okny, klepněte na titulkovém pruhu okna pravým tlačítkem myši a v místní nabídce odstraňte zaškrtnutí položky Dockable. Místní nabídka je vidět na obrázku v pravém dolním rohu. Další příkazy umožňují pracovat s okny plujícími volně po obrazovce (Floating) nebo zvolit dvojí způsob *skrývání* oken (Hide, Auto Hide). Nevidíte-li některé z oken, máte k němu vždy přístup z nabídky View. Otevřete nabídku a vyhledejte příkaz s názvem okna nebo rozviňte kaskádovou nabídku příkazu ostatních oken (Other Windows). Nevidíte-li okno dynamické nápovědy, zobrazíte je příkazem Help > Dynamic Help. Možných variant zobrazení je velmi mnoho a možná vám zpočátku bude připadat, že se okna chovají poněkud divně. Vyplatí se věnovat manipulaci s okny trochu času, aby vám přešla do krve.

Místní nabídkou je vybaven téměř každý objekt, s nímž budete ve vývojovém prostředí pracovat. Například, na obrázku vidíte další dvě místní nabídky. První z nich se zobrazí, když klepnete pravým tlačítkem myši na ploše formuláře. Umožňuje rychle zobrazit okno kódu (View code), okno vlastností (Properties) a také *uzamknout ovládací prvky* (Lock Controls). Když totiž umístíte nějaké objekty na formulář a zdá se vám, že je to jejich finální pozice, můžete jejich pozici fixovat (zároveň fixujete i formulář, protože ten je také objektem). Budete-li to považovat za vhodné, můžete pak později stejným příkazem objekty (i jednotlivě) odemykat.

Důležitým oknem, bez něhož se při práci s formuláři neobejdete, je *souprava nástrojů* (Toolbox). Obsahuje mnohem více ovládacích prvků než v předchozích verzích Visual Basicu a její výchozí pozice je na levé straně (má svislou záložku), odkud se automaticky vysouvá a kam se automaticky zasouvá. Pokud si na to nemůžete zvyknout (jako já), klepněte pravým tlačítkem myši na titulkovém pruhu okna soupravy nástrojů a „odškrtněte“ položku Auto Hide v místní nabídce. Okno soupravy nástrojů můžete také přetáhnout jinam a pracovat s ní jako s plovoucím oknem.

Místní nabídka, která se zobrazí, když klepnete pravým tlačítkem myši na některém nástroji soupravy, umožňuje s nástroji v soupravě manipulovat, přizpůsobit vzhled soupravy, přidávat do ní další skupiny prvků apod.

Zobrazování zpráv (o stavu financí)

Protože už teď hráč do automatu něco dává a máme jakous takous evidenci o tom, jak si v daném okamžiku stojí, můžeme mu poskytnout možnost, aby si kdykoli zobrazil, kolik vyhrává, resp. kolik prohrává. Naprogramujeme proto událostní proceduru `Click` tlačítka s nápisem Vypsat stav:

```
Public Sub btnStav_Click(ByVal sender As Object, _
   ByVal e As System.EventArgs) Handles btnStav.Click

   If StavCelkem > 0 Then
      MessageBox.Show("Vyhráváte: " & FormatCurrency(StavCelkem), _
        "Informace o stavu financí", _
        Messagebox.IconExclamation + Messagebox.OK)
   Else
      MessageBox.Show("Zatím Vás to stálo: " & FormatCurrency(-StavCelkem), _
         "Informace o stavu financí", _
          Messagebox.IconInformation)
   End If

End Sub
```

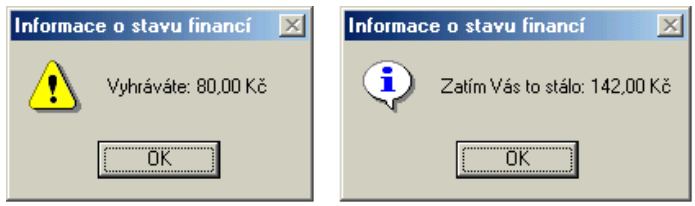
Pro výpis jednoduchých zpráv uživateli v dialogovém okně se využívá třída `MessageBox` pojmenovaného prostoru `System.WinForms`. Zpráva se zobrazuje její metodou `Show`, v níž první parametr udává nápis v okně, druhý nápis v titulkovém pruhu a třetí různé informace o tom, jaká tlačítka v dialogovém okně budou, jaká tam bude ikona a jaké tlačítko bude výchozí. Tyto informace se specifikují pomocí *polí* třídy `MessageBox`.

V souvislosti se zobrazováním zpráv si můžete vyzkoušet další rys Visual Basicu, totiž podporu tzv. *pojmenovaných parametrů*. Má-li metoda hodně parametrů, a vy přitom ve volání chcete uvést jen několik (pro ostatní chcete ponechat výchozí hodnoty), můžete parametry specifikovat tak, že uvedete název parametru, speciální operátor `:=` (dvojtečka a rovná se) a hodnotu parametru. Výhody jsou zřejmé. Parametry můžete uvést v libovolném pořadí a jen ty, které opravdu potřebujete. Například, obrázek o stavu financí byste mohli zobrazit také tímto příkazem:

```
MessageBox.Show(caption:="Informace o stavu financí", _
   text:="Vyhráváte: " & FormatCurrency(StavCelkem))
```

Názvy parametrů se dozvíte přímo při zápisu volání metody z pomocného okénka, v němž se zobrazuje kompletní syntax metody.

Až budete později řešit formulář parametrů, ukážeme si, jak pracovat v situacích, kdy je v dialogovém okně více než jedno tlačítko a je třeba zjistit, na kterém z nich uživatel klepl (a podle toho rozvětvit další běh programu). Obě okna se zprávou z procedury `btnStav_Click` vidíte na obrázku:



VB6. Ptáte se možná: „Ani taková základní věc jako funkce `MsgBox` ve Visual Basicu nezůstala?“ Pokud ji chcete volat i nadále, najdete ji spolu se spoustou dalších programovacích prvků (včetně `Beep` nebo `InputBox`) v pojmenovaném prostoru `Microsoft.VisualBasic.Interaction`. Chcete-li ji volat, udělejte to takto.Přidejte k příkazům `Imports` tento:

```
Imports Microsoft.VisualBasic.MsgBoxStyle
```

Chcete-li zobrazit okno s tlačítky Ano, Ne a s informační ikonou:

dá se to zapsat příkazem:

```
MsgBox("Chceš si ještě vsadit?", Information BitOr YesNo, "Dotaz")
```

Zatímco ve Visual Basicu 6 byste to napsali takto:

```
MsgBox "Chcete ještě sázet?", vbInformation + vbYesNo, "Dotaz"
```

nebo takto, pokud byste chtěli uložit návratovou hodnotu

```
Výsledek = MsgBox("Chcete ještě sázet?", vbInformation + vbYesNo, "Dotaz")
```

Tyto ukázky neuvádím proto, že bych se chtěl vyžívat na oknech se zprávami, ale protože jejich prostřednictvím mohu upozornit na další dva závažné rozdíly mezi VB6 a VB.NET. Především vidíte, že vnitřní konstanty Visual Basicu už nemají předponu `vb`.

Druhý rozdíl spočívá v tom, že se při volání procedur `Sub`, které mají neprázdný seznam parametrů, *musejí vždy uvádět závorky*, takže typ zápisu bez nich (druhý zdola) už povolen není! Nemá-li procedura parametry, lze závorky vynechat. Příkaz `Call` je nepovinný, na tvar zápisu nemá vliv, z čehož plyne, že nemá smysl ho používat, takže s ním přestaňte, pokud snad tak dosud činíte.

Tleskání a blikání

Jaký by to byl hrací automat, kdyby neobsahoval alespoň primitivní multimediální efekty. Když uživatel vyhraje, zatleskáme mu a rozblikáme okno automatu. Poskytneme hráči také možnost tyto efekty předčasně ukončit. Později (na pomocném formuláři parametrů) mu také umožníme, aby mohl tyto efekty vypnout natrvalo.

Úlohu zde řešíme pomocí funkcí API Windows. Podrobnější výklad práce s funkcemi API přesahuje rámec i rozsah této brožury. Bude-li se vám kód zdát nesrozumitelný, použijte ho tak, jak je, nebo efekty do aplikace nedávejte a prostě tento oddíl přeskočte.

Chcete-li volat v kódu Visual Basicu nějaké procedury z API Windows, musíte je nejprve deklarovat příkazem `Declare`. Obvykle se také zároveň deklarují potřebné *symbolické konstanty*-ve Visual Basicu příkazem `Const`. (Deklarace lze opsat pomocí různých prohlížečů API.) Do třídy formuláře přidejte nad první proceduru kód vypsaný na příští stránce.

Konstanta `2800` v konstrukci `If` určuje horní mez, kdy časovač sám sebe vypne a blikání tedy skončí.

> **VB6**. Dříve se neprogramovala událostní procedura `Tick`, ale procedura s názvem stejným, jako je název ovládacího prvku, tedy `Timer`. Další rozdíl spočívá v tom, že se dříve dal časovač vypínat nastavením jeho vlastnosti *Interval* na nulu. Nyní to musíte dělat jen tak, že nastavíte vlastnost `Enabled` časovače na `False` (nastavíte-li *Interval* na nulu, přepne se automaticky na hodnotu 1).

Přidání komponenty bez ovládacího prvku na podnosu

Komponenty lze také do aplikace přidávat přímo v kódu, aniž byste použili soupravu nástrojů. Například, náš časovač můžete zařadit do kódu tímto postupem:

1. Přidejte do třídy formuláře příkaz

```
Private WithEvents Timer1 As System.WinForms.Timer
```

2. Do událostní procedury `Form1_Activated` přidejte příkazy

```
Timer1 = New System.WinForms.timer()
Timer1.Enabled = False
Timer1.Interval = 100
```

Další příkazy, v nichž se časovač využívá, zůstanou stejné, jako při vizuálním řešení.

Námět na procvičování. Některé procedury zařazené do formuláře mají obecnější charakter a možná byste je rádi využívali i v jiných projektech. Zkuste z nich vyrobit samostatnou třídu nebo je učiňte za základ svého vlastního pojmenovaného prostoru.

Zpracování chyb

Až do této chvíle jsem se v brožuře vůbec nedotkl jedné oblasti programovacího jazyka, bez níž se neobejdete v žádné aplikaci – zachycování chyb a jejich odstraňování. Při programování se setkáváte se třemi zásadními druhy chyb:

- *Syntaktické* chyby se ve vývojovém prostředí Visual Basicu už dávno snadno odstraňují přímo při psaní a VB.NET dále rozšířil aparát pomůcek, které pomáhají k tomu, aby kód, který napíšete, byl syntakticky správný.
- *Logické* chyby – nejhorší druh – způsobují, že aplikace sice běží, ale dělá něco jiného, než jste od ní očekávali. Při odstraňování těchto chyb se využívají ladicí prostředky vývojového prostředí, které jsou také velmi vyspělé.
- Zbývají *chyby při běhu* – které vznikají proto, že aplikace sice projde kompilací, ale při běhu nastane nějaká výjimečná situace, která způsobí, že aplikace dál běžet nemůže.

My se v brožuře krátce zastavíme u tzv. *zachytitelných* chyb (většinou se jedná o chyby při běhu), tj. takových, které můžete (a měli byste) ve své aplikaci zpracovat a rozhodnout, co se má dít dál (zda může aplikace pokračovat, že musí skončit apod.). Velmi nenáročnou (jen trochu nudnou a někdy přehlíženou) součástí zpracování chyb je zobrazování srozumitelných a přívětivých zpráv uživatelům. Přitom tato zdánlivá drobnost může mít enormní vliv na úspěšnost vašich aplikací (=jak se budou dobře prodávat).

Rozšíření prototypu (2) – další formulář

Hlavní formulář aplikace už v podstatě dělá všechno, co jsme si předsevzali, ovšem hodnoty kreditu a velikosti sázky pro jedno roztočení automatu jsou nastaveny fixně a netestuje se, aby hráč nemohl hrát dál, pokud počáteční kredit vyčerpá. Nyní proto rozšíříme prototyp automatu o další formulář, v němž bude moci uživatel zadat hodnotu kreditu a velikosti sázky.

Prostřednictvím pomocného formuláře jakýchsi vstupních parametrů aplikace se seznámíte s dalším typickým ovládacím prvkem uživatelských rozhraní aplikací. Na formulář parametrů přidáte *zaškrtávací políčko* (CheckBox). Jeho vyčištěním bude moci uživatel úplně potlačit multimediální efekty, které jsme na hlavní formulář přidali v předchozích oddílech.

Seznámíme se s tím, jak se do projektu Visual Basicu přidá nový formulář, jak se určí, který formulář bude spouštěcí, co to jsou modální a nemodální formuláře a jak je možno pracovat s hodnotami vlastností jednoho formuláře na jiném formuláři. Naučíte se také, jak si paletu ovládacích prvků můžete obohatit vlastními ovládacími prvky. Podrobný postup obsahují příští oddíly.

Přidání formuláře parametrů

Nové prvky se do projektu přidávají hlavně prostřednictvím nabídky Project:

1. Zvolte Project > Add Windows Form.
2. V dialogovém okně *Add New Item – Automat* vyberte v levé části složku *Local Project Items* a v pravém dvojitě klepněte na ikonu Windows Form.
3. Do textového pole *Name* napište název formuláře, například `frmZadáníParametrů` a klepněte na OK.

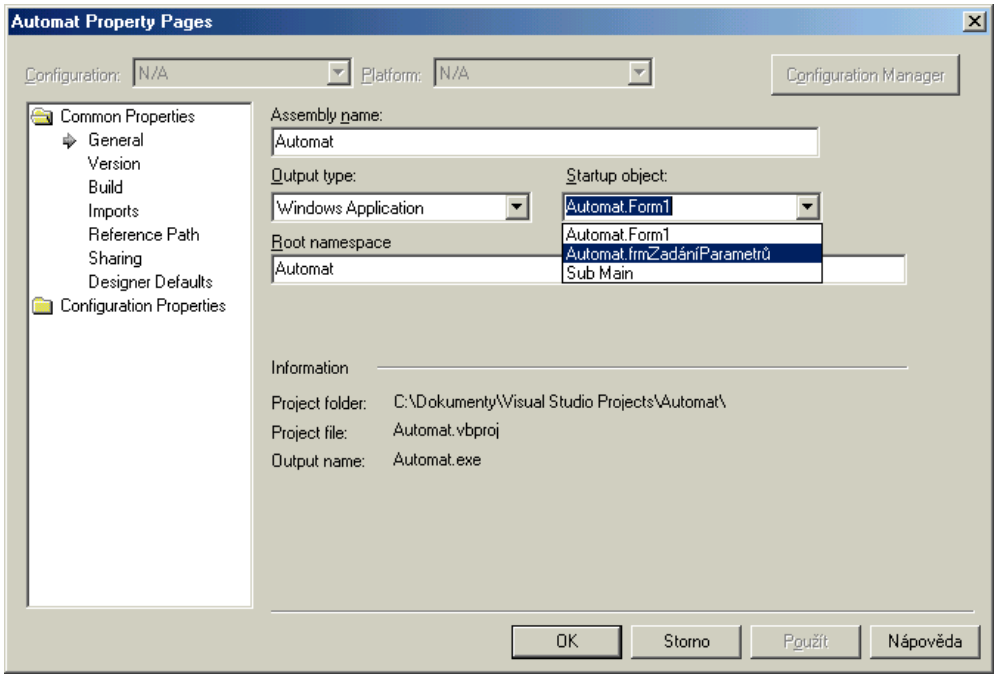
Nový formulář se zařadí do projektu a jeho název uvidíte ve složce projektu v průzkumníkovi řešení.

Nastavení startovacího objektu

Když sestrojíte pro projekt nový formulář a chcete ho ladit, usnadníte si jeho testování tím, že ho nejprve prověříte samostatně, bez případných vazeb na jiné části budované aplikace. Pohodlně se to udělá tak, že se formulář prohlásí za *startovací objekt projektu*, takže když pak klepnete na tlačítko Start (nebo stisknete F5), spustí se formulář, který jste prohlásili za spouštěcí a můžete ho vyzkoušet:

1. Nevidíte-li průzkumníka řešení, zvolte View > Solution Explorer.
2. Klepněte pravým tlačítkem myši na název projektu-v našem případě na položku *Automat (název projektu)* a z místní nabídky zvolte Properties.
3. V dialogovém okně stránek vlastností projektu (v našem případě *Automat Property Pages*) vyberte ze seznamu *Startup object* název formuláře, který chcete prohlásit za spouštěcí a klepněte na OK.

Dialogové okno stránek vlastností projektu se zobrazenou stránkou obecných vlastností (General) ukazuje následující obrázek. Jak vidíte v levé části dialogového okna, je vlastností projektu poměrně hodně:



Přidání ovládacích prvků na formulář

Teď začnete přidávat na formulář potřebné ovládací prvky (viz obrázek běžícího formuláře parametrů dále v textu).

1. Nastavte tyto vlastnosti formuláře: *Text* na `Zadání parametrů pro automat`, *BorderStyle* na `Fixed Dialog`, *MinimizeBox* a *MaximizeBox* na `False`.
2. Na formulář přidejte popisek (ovládací prvek Label) a nastavte jeho vlastnosti *Text* na `K&redit }v Kč)`. Znak ampersand (`&`) v popisku poslouží jako přístupová klávesa pro sdružené textové pole, do něhož bude hráč zadávat kredit.
3. Přidejte na formulář textové pole pro kredit. Nazvěte ho `txtKredit`.

Protože jste textové pole umístili na formulář těsně po popisku, který se k němu vztahuje, bude v tabelátorovém řazení prvků bezprostředně následovat za popiskem. Proto přístupová klávesa popisku bude vlastně simulovat přístupovou klávesu textového pole (popisek je totiž jeden z ovládacích prvků, který nemůže uživatel aktivovat (prvek nemůže „získat fokus“).

4. Obdobně sestrojte popisek a textové pole pro velikost sázky. Nazvěte textové pole `txtSázka`.
5. Přidejte na formulář dvě příkazová tlačítka, nastavte jejich vlastnosti *Name* na `btnPokračovat`, resp. `btnKonec` a hodnotu vlastností *Text* na `&Pokračovat`, resp. `&Konec`.
6. Nastavte vlastnost *AcceptButton* formuláře na `btnPokračovat`. Tlačítko bude mít stejnou funkci, jako když uživatel stiskne klávesu Enter.

9. Sestrojte si druhou variantu formuláře parametrů – stejnou jako původní, ale bez textových polí (nebo z původního formuláře parametrů textová pole odstraňte).
10. Zvolte Build > Rebuild.
11. Otevřete návrh formuláře parametrů, klepněte v soupravě nástrojů na svůj nástroj *ČíselnýTextBox* a klepněte na formuláři. Opakujte pro druhé textové pole.

Poznámka. Chcete-li modifikovat vlastnosti ovládacích prvků odvozených z vlastního ovládacího prvku, nastavte jeho vlastnost *Modifiers* na `Public`.

Náměty k procvičování. Logickým pokračováním při práci s vlastními ovládacími prvky je, že si vytvoříte vlastní knihovnu ovládacích prvků. Opět to jde vizuálně a v podstatě stačí dva kroky. Přidáte do řešení nový projekt patřičného typu a nastavíte odkazy.

V řadě úloh se hodí, když se vlastní ovládací prvek skládá z více prvků. Například, v našem případě by byl jednoduchým zobecněním vlastní ovládací prvek složený ze dvou textových polí, do nichž se zadává nějaký interval hodnot (od – do). Součástí ovládacího prvku by pak mohly být i veškeré potřebné kontroly platnosti a souvztažnosti hodnot.

Propojení formulářů

Formuláře Windows, které v této brožuře vytváříte, mohou být *modální* nebo *nemodální*. Co to znamená? Když pracujete s *modálním* formulářem a chcete dělat něco jiného, musíte jej nejprve *uzavřít* (skrýt nebo uvolnit z paměti), teprve pak můžete pracovat s jinou částí aplikace. Mnoho modálních formulářů znáte. Chová se tak valná většina dialogových oken běžných aplikací Windows, s nimiž pracujete. Modální formuláře mají pro programátora jednu podstatnou výhodu, nemusí ošetřovat situace, že uživatel z čista jasna práce s formulářem přeruší a začne dělat něco jiného.

Takové jsou právě *nemodální* formuláře. Typickým příkladem nemodálních formulářů jsou panely nástrojů. Chcete-li formulář zobrazit jako modální, zavolejte metodu `ShowDialog`, jinak zavolejte metodu `Show`.

Hlavní formulář a formulář parametrů, které jsme zatím testovali izolovaně, nyní propojíme tak, že spouštěcím formulářem bude opět hlavní formulář, při jehož startu (nebo později kdykoli uživatel klepne na tlačítko s nápisem Dodat další prachy) se zobrazí formulář parametrů jako dialogové okno. Postup:

1. Přejděte do okna kódu hlavního formuláře a do třídy `Form1` přidejte nad všechny procedury deklaraci:

   ```
   Private f As frmZadáníParametrů = New frmZadáníParametrů()
   ```

2. Na konec konstruktoru `New()` formuláře `Form1` přidejte příkazy:

   ```
   f.showdialog()
   lblKredit.Text = f.txtKredit.Text
   lblSázka.Text = f.txtSázka.Text
   ```

   Jak předchozí výpis ukazuje, hodnoty, které uživatel zapsal do textových polí, jsou přístupné jako hodnoty vlastností formuláře, jehož instanci jste vytvořili.

3. Do procedury `Click` tlačítka s nápisem Roztočit přidejte konstrukci `If…End If`, v níž otestujete, zda hráč nevyčerpal kredit, který zadal na počátku aplikace. Jestliže ano, zobrazí se mu dotaz, zda chce hrát dál. Výpis konstrukce je uveden na příští stránce.

```
If CInt(lblKredit.Text) <= 0 Or _
        CInt(lblKredit.Text) < CInt(lblSázka.Text) Then
    ' Me.Hide()

    If DialogResult.Yes = MessageBox.Show _
      ("Je mi líto, musíte dodat další prachy. Chcete hrát dál?", _
      "Zpráva o stavu financí", _
       MessageBox.IconQuestion + Messagebox.YesNo) Then
       f.ShowDialog()
       lblKredit.Text = f.txtKredit.text
       lblSázka.Text = f.txtSázka.Text
       ' Me.Show()

    Else
        btnKonec_click(sender, e)
    End If

End If
```

Považujete-li to za účelné, můžete pomocí metody `Hide` dočasně hlavní formulář skrýt, abyste zdůraznili, že je nutné dodat další peníze. Po dokončení prací na formuláři parametrů byste pak hlavní formulář opět zobrazili metodou `Show`.

V příkazu, který zobrazuje zprávu, si všimněte, jak se v třetím parametru specifikuje více informací současně (zde se konkrétně má zobrazit ikona otazníku a tlačítka s nápisy Ano a Ne. Jedná se o pole třídy `MessageBox`.

Dále si všimněte, jak se testuje návratová hodnota metody `MessageBox.Show` (tj. na kterém tlačítku uživatel klepl v okně zprávy). Jedná se o výčtový typ `DialogResult` (opět z pojmenovaného prostoru `System.WinForms`).

Jestliže uživatel nechce pokračovat, ukončení aplikace nemusíte programovat znovu, ale simulujete situaci, že uživatel klepl na tlačítko Konec. Zavoláte prostě tuto událostní proceduru.

4. Napište událostní proceduru `Click` tlačítka s nápisem Dodat další prachy:

```
Public Sub btnPrachy_Click(ByVal sender As Object, _
          ByVal e As System.EventArgs) Handles btnPrachy.Click

   ' Me.Hide()
     f.ShowDialog()
     lblKredit.Text = f.txtKredit.text
     lblSázka.Text = f.txtSázka.Text
     ' Me.show()

 End Sub
```

Rozšíření prototypu (3) – nabídky

Naše aplikace v podstatě žádné nabídky nepotřebuje, protože pro její řízení stačí uživateli příkazová tlačítka na formulářích. Nabídky však patří mezi základní a často vyžadované prvky uživatelského rozhraní., proto jsem považoval za vhodné, zařadit do této brožury také úvodní seznámení s tím, jak se ve Visual Basic.NET tvoří nabídky a místní nabídky. Panely nástrojů, stavové řádky a další uživatelský komfort už do aplikace dávat nebudeme, už tak toho bude v podstatě příliš. Ukážeme si vizuální sestrojení nabídek i vytvoření nabídek přímo v kódu.

Visual Basic.NET poskytuje v soupravě nástrojů pro tvorbu nabídek dva ovládací prvky: *MainMenu* a *ContextMenu*. Pomocí prvního z nich se tvoří obvyklý systém nabídek složený z pruhu nabídek, pod nímž se rozvinují podřízené nabídky. Druhý slouží k tvorbě místních nabídek.

> **VB6**. Dříve se nabídky tvořily nástrojem *Menu Editor*. Místní nabídky se sestrojovaly jako neviditelné součásti systému běžných nabídek. Místní nabídka se pak obvykle aktivovala v událostní proceduře zachycující stisk tlačítka myši, kde se pomocí prvního parametru událostní procedury odlišil stisk pravého tlačítka myši a nabídka se zobrazila metodou `Popup` formuláře.

Místní nabídka na formuláři parametrů

Seznámení s nabídkami začneme cvičnou místní nabídkou na formuláři parametrů, protože se jedná o jednodušší úlohu. Držte se následující ho postupu.

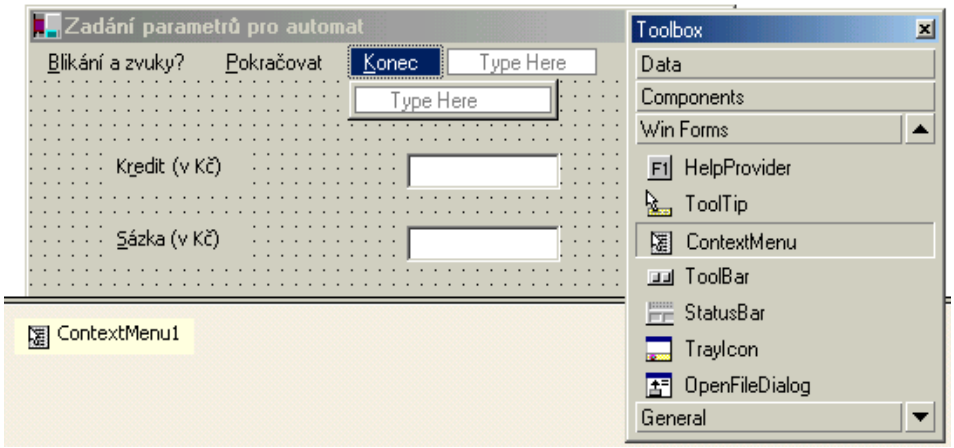
1. Abyste mohli formulář parametrů pohodlně ladit, prohlásíte ho opět dočasně za startovací objekt projektu. Nevidíte-li průzkumníka řešení, zvolte View > Solution Explorer, klepněte pravým tlačítkem myši na název projektu, z místní nabídky zvolte Properties, vyberte ze seznamu *Startup object* formulář frmZadáníParametrů a klepněte na OK.
2. V návrhovém zobrazení formuláře parametrů klepněte v soupravě nástrojů na tlačítko WinForms, pak na nástroj ContextMenu a klepněte na formuláři.

Pod formulářem se objeví podnos a v něm ikona ovládacího prvku s výchozím názvem `ContextMenu1`. Zároveň se pod titulkovým pruhem formuláře budou objevovat chlívky s napovídajícím titulkem „Type Here“ (Pište sem).

3. Do těchto chlívků napište texty, které se mají zobrazovat v nabídce. Jednotlivé příkazy budoucí místní nabídky pište *jako prvky pruhu nabídek*, ne pod sebe. Chcete-li rovnou v textech určit přístupové klávesy, napište před zvolené písmeno znak & (na české klávesnici pravý Alt+C).

Nastavujete vlastně hodnoty vlastnosti *Text* jednotlivých prvků v nabídce. Kdybyste psali texty pod sebe, vytvářeli byste systém místních nabídek, což je také možné, ale v této úloze chceme sestrojit prostou místní nabídku tří příkazů. Viz obrázek na další straně.

4. Klepněte pravým tlačítkem myši na formuláři, klepněte na Properties a nastavte vlastnost ContextMenu formuláře na `ContextMenu1`.

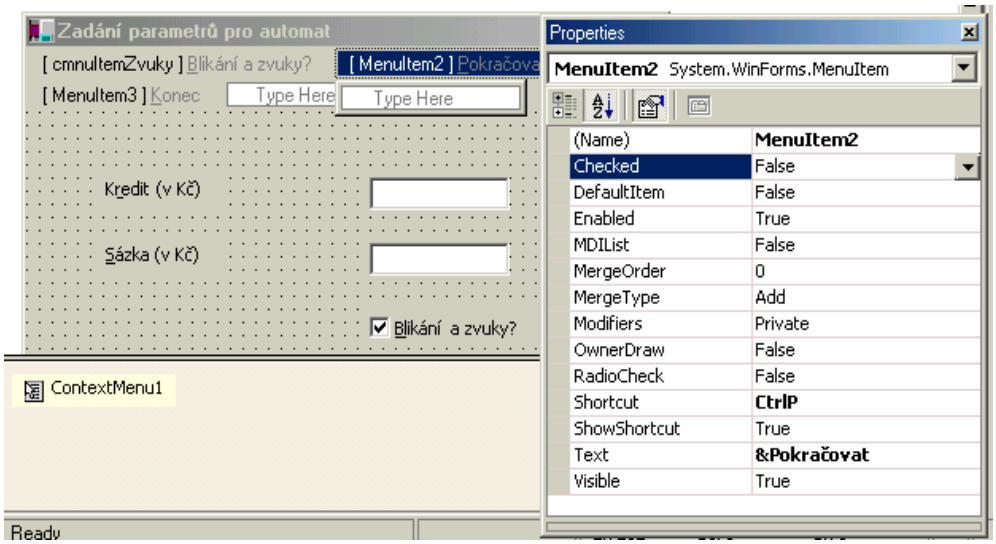


Jednotlivé prvky nabídky tvoří objekty `MenuItem` a dostávají výchozí názvy `MenuItem1`, `MenuItem2` atd. Protože budete později programovat událostní procedury těchto prvků (abyste příkazu v nabídce přiřadili nějakou konkrétní činnost), je žádoucí změnit tyto výchozí názvy (jako se to ostatně má dělat u všech ovládacích prvků, s nimiž hodláte nějak pracovat) na lépe vypovídající.

5. Klepněte na některém prvku v nabídce, klepněte pravým tlačítkem myši a z místní nabídky zvolte Edit Names. (Obrázek této místní nabídky viz příští oddíl *„Pruh nabídek na hlavním formuláři“*.)

Na formuláři se objeví výchozí názvy v hranatých závorkách vlevo od textů prvků nabídek.

6. Přejmenujte názvy prvků nabídek podle vašich dohodnutých konvencí. Lze to udělat přímo na formuláři nebo v okně vlastností, podobně jako u jiných ovládacích prvků. U prvku, který je nabídkou, se obvykle používá předpona `mnu`, `cmnu` apod., u ostatních prvků nabídky se přidává slovo `Item` (prvek), pak vlastní název. Viz obrázek.



```
' Nabídka Nápověda:
 Dim mnu2ItemObsah As New MenuItem()
 Dim mnu2ItemOAplikaci As New MenuItem()
 mnu2ItemObsah.Text = "&Obsah"
 mnu2ItemOAplikaci.Text = "&Informace o aplikaci"
 mnu2Nápověda.MenuItems.Add(mnu2ItemObsah)
 mnu2Nápověda.MenuItems.Add(mnu2ItemOAplikaci)
```

Náměty na procvičování

- Ve skutečných aplikacích se často používá obecnější model, kdy se uživatel zobrazuje na formuláři několik variant pruhu nabídek podle aktuální situace (podle stavu, v jakém se aplikace právě nachází). Zkuste vytvořit různé varianty buď pomocí více objektů `MainMenu` nebo změnami obsahu systému nabídek při běhu, abyste mohli varianty dynamicky přepínat.
- Jak uvidíte, provádějí se manipulační činnosti s nabídkami docela jednoduše (protože prvky nabídek tvoří kolekce). Proto můžete nabídky poměrně jednoduše přesouvat či odstraňovat (metodou `Remove`).
- Chcete-li nabídky kopírovat, vyzkoušejte metodu `CloneMenu`. Metodou `MergeMenu` můžete ze dvou nabídek udělat jedinou.
- Až budete programovat aplikaci s více dokumentovým rozhraním (MDI), určitě využijete to, že nabídky podporují také vytvoření seznamu dceřiných oken pro formulář MDI. (Vlastnost *IsMDIContainer* mateřského formuláře se nastaví na `True`, objekt hlavní nabídky se připojí k mateřskému formuláři a vlastnost *MDIList* jednoho prvku pruhu nabídky (její text by mohl být například `&Okno`) se nastaví na `True`).

Využití společných dialogových oken Windows

Tři příkazy, které jsme zařadili do prostřední nabídky, otvírají jednu z dalších oblastí práce s formuláři Windows, které se trochu dotkneme, protože je to záležitost, jíž se v řadě aplikacích nevyhnete a bylo by dost pracné a neefektivní, kdybyste se tyto typy úloh pokoušeli řešit sami.

Když uživatel pracuje s nějakou aplikací, není vždy vhodné nebo žádoucí (nebo to ani nejde), aby všechny akce probíhaly automaticky, bez aktivní účasti uživatele. Často je třeba uživateli umožnit, aby si mohl v rámci práce s aplikací otevřít nějaký soubor, uložit nějaký soubor, něco přebarvit či upravit atributy písma. Nebo z aplikace něco tiskne a chcete mu dám možnost, aby si mohl předem nastavit nějaké charakteristiky týkající se tištěné stránky, tisku samotného nebo tiskárny.

Když pracujete se standardními aplikacemi Windows (například ve Wordu), zobrazují se vám v těchto situacích tzv. *společná dialogová okna Windows* (Otevřít, Uložit jako, Barva, Písmo, Tisk apod.), která vypadají ve všech aplikacích stejně nebo velmi podobně.

Výhody jsou zřejmé. Uživatelé jsou na ně zvyklí a všechny aplikace s v tomto ohledu tváří“ na uživatele stejně. Proto by bylo dobré, kdyby uživatelé mohli s těmito okny pracovat i ve vašich vlastních aplikacích. Ve VB.NET to jde zařídit poměrně pohodlně, všechny potřebné nástroje najdete v soupravě nástrojů. Brožuru ukončíme ukázkou využití tří společných dialogových oken: *Otevřít*, *Písmo* a *Barva*.