



## 7. Moduly

### 7.1 Funkce

*Visual Basic  
for  
Applications*

Pro řešení složitějších aplikací nabízí Access programovací jazyk *Visual Basic for Applications* (VBA). VBA je strukturovaný programovací jazyk pro tvorbu aplikací v prostředí všech programů kancelářského balíku Microsoft Office. VBA je zjednodušenou verzí samostatného programovacího jazyka *Visual Basic*. VBA se používá pro:

- Tvorbu uživatelských funkcí, které lze využívat ve všech objektech Accessu, tj. také v dotazech, formulářích a sestavách.
- Programování složitějších postupů, které již nelze zapsat pomocí maker. V makrech není možné nebo velmi těžkopádné zejména provádění opakovaných činností.
- Lepší ošetření možných chybových stavů a přesnější definování chybových zpráv.
- Práci s jednotlivými větami. VBA umožňuje procházet jednotlivé věty a zpracovávat je či provádět operace s daty z různých vět.
- Vytváření uživatelsky přívětivých aplikací, kdy složitější postupy mohou být naprogramovány a realizovány klepnutím do tlačítka či jinou událostí, např. otevřením formuláře.

*Moduly*

Text programu (*kód*) se ukládá v *modulech*. Moduly mohou být součástí formulářů či sestav (*lokální moduly*), mohou být uloženy také samostatně (*globální moduly*), aby byly využitelné ve více objektech (formulářích, sestavách, makrech, jiných modulech apod.).

*Procedury*

Zápis kódů v modulech je členěn do *procedur*. Existují dva druhy procedur:

- *funkce* (Function): Navrací výstupní hodnoty vypočítané na základě vstupních hodnot (argumentů).
- *podprogramy* (Sub): Provádí opakované činnosti na základě vstupních argumentů. Na rozdíl od funkcí nemohou být použity ve výrazech, protože nevrací výstupní hodnotu. V kódu je volání podprogram zastoupeno samostatným příkazem.

*Deklarace*

Kromě procedur obsahuje modul v úvodu *deklarace*. Deklarace obsahují nastavení platná pro modul a případně deklarace jednotlivých proměnných používaných v modulu. V *proměnné* je pod jejím názvem uložena hodnota, která se může měnit. Typ proměnné může být v modulu specifikován dvěma způsoby:

- deklarací proměnné

V případě, že použijeme nedeklarovanou proměnnou, Access ohlásí chybu. Vyhneme se tak překlepům v názvech proměnných. Nutnost deklarovat proměnné zajistíme v úvodu deklarací modulu příkazem *Option Explicit*.

- prvním použitím proměnné

Nemusíme deklarovat proměnné. V úvodu modulu neuvedeme příkaz *Option Explicit*.

Připravme svou první proceduru. Access nabízí širokou nabídku vestavěných funkcí, kterou můžeme doplnit vlastními funkcemi. Připravíme funkci, jejímiž vstupními argumenty jsou titul před jménem, křestní jméno, příjmení, titul za jménem. Funkce bude vracet celé jméno včetně titulů. V okně databáze klepneme na záložku **Moduly**.



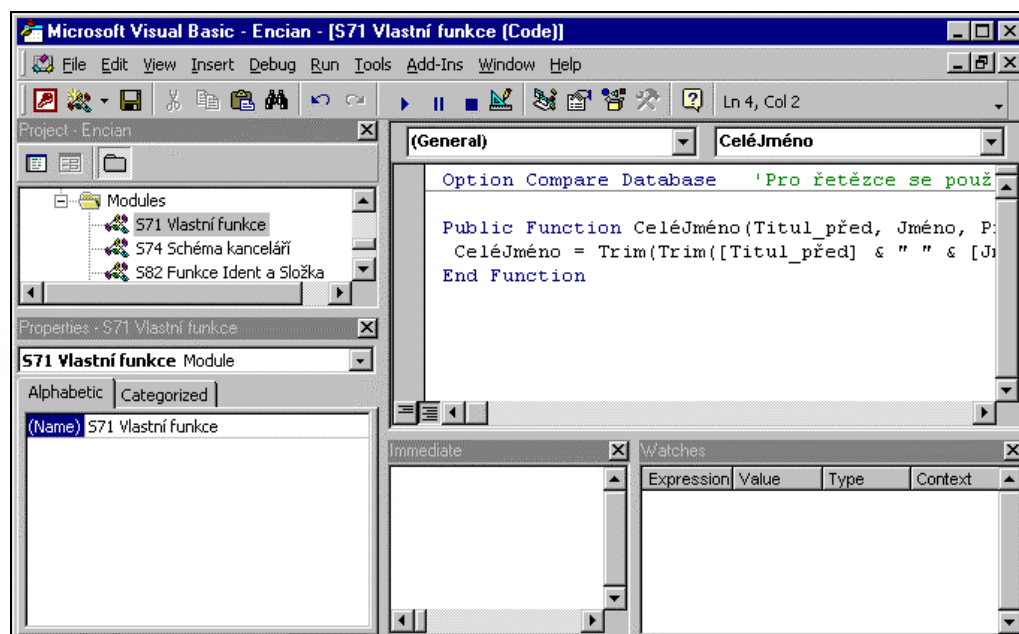
*Editor VBA*

V rámci karty modulů klepneme do tlačítka **Nový**. Jako samostatná aplikace se spustil *Microsoft Visual Basic*, v němž se otevřel kód modulu. Editor Visual Basicu se skládá z několika částí zobrazovaných v samostatných oknech (viz obr. 7-1):

- *Prohlížeč projektu* (Project): Ve stromové struktuře umožňuje vybrat lokální modul (např. z formuláře) či globální modul, který je zobrazen v okně kódu. Zpočátku je zde vybrán nový (dosud nenazvaný) modul, který jsme založili.
- *Okno vlastností* (Properties): Jednotlivé objekty Visual Basic mají své vlastnosti. Můžeme zde např. změnit název modulu.
- *Okno kódu* (Code): Zde budeme vytvářet funkce a podprogramy.
- *Okamžité okno* (Immediate): V okně můžeme zadat nebo vložit řádek kódu a odesláním klávesou **Enter** jej spustit.
- *Okno kukátek* (Watches): Umožňuje průběžně zkoumat hodnoty vybraných proměnných.

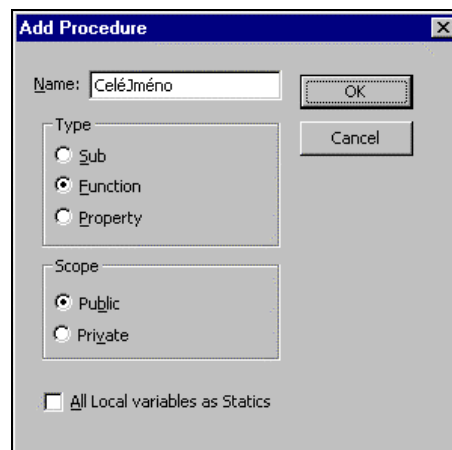


OBR. 7-1: EDITOR VISUAL BASICU



V úvodu okna kódu připravil Access příkaz Visual Basicu *Option Compare*, který upřesňuje způsob porovnávání řetězců.<sup>108</sup> V případě, že bychom chtěli vyžadovat deklarování proměnných před jejich použitím, dopsali bychom příkaz *Option Explicit*.<sup>109</sup>

Zadáme z menu VBA příkaz INSERT, PROCEDURE nebo klepneme do tlačítka **Insert Procedure** (vybereme z nabídky tři tlačítka, která se nabízí při klepnutí do šipky napravo od druhého tlačítka panelu nástrojů). V dialogovém okně **Add Procedure** (viz obr. 7-2) zadáme název funkce *CeléJméno*<sup>110</sup>, v poli *Type* vybereme *Function*, zbývající pole ponecháme beze změny.

OBR. 7-2: DIALOGOVÉ OKNO  
ADD PROCEDURE

Editace funkce

Do okna modulu se přidaly dva příkazy:  
– *Public Function*: Zahajuje definici funkce. Do závorek můžeme zadat argumenty funkce.  
– *End Function*: Ukončení funkce.

Do závorek prvního příkazu doplníme argumenty *Titul\_před*, *Jméno*, *Příjmení* a *Titul\_zá* oddělené čárkami. Ve funkci musíme přiřadit proměnné s názvem funkce hodnotu. Do volného řádku mezi zahajovacím a ukončovacím příkazem zapíšeme jediný příkaz, kterým sloučíme textové řetězce:

*CeléJméno* = Trim(Trim([*Titul\_před*] & " " & [*Jméno*]) & " " & [*Příjmení*]) & [*Titul\_zá*]  
Přiřazení hodnoty jsem provedl příkazem, který proměnné přiřazuje hodnotu danou výrazem. Funkce *Trim* zajišťuje, aby se mezera mezi titulem a jménem či jménem a příjmením nevypisovala, když jsou titul či jméno nevyplněné.

Přiřazovací  
příkaz

```
Trim(  
Trim(String)
```

Při psaní funkce *Trim* si všimneme, že se pod kurzorem objeví nápověda syntaxe funkce *Trim(String)*. Před a za operátorem & musíme zapsat mezery.

<sup>108</sup> Příkaz upřesňuje, kdy je jeden řetězec větší nebo roven jinému řetězci:

- *Option Compare Text*: při porovnání nejsou rozlišována velká a malá písmena,
- *Option Compare Binary*: při porovnání jsou rozlišována velká a malá písmena,
- *Option Compare Database*: metoda porovnávání je nastavena aktuální databázi.

<sup>109</sup> Příkazem z menu editoru VBA *Tools, Options* můžeme v kartě **Editor** zaškrtnutím pole *Require Variable Declaration* zajistit, aby VBA při vytvoření nového modulu vložil také příkaz *Option Explicit*.

<sup>110</sup> V ukázkové databázi *Encian* již je funkce přichystána, proto ji nazveme *PokusCeléJméno*.



Příkazy procedury začínáme mezerou, abychom naznačili strukturu procedury. (Všechny příkazy mezi *Public Function* a *End Function* jsou odrazeny mezerou od kraje.)

Za deklarační příkaz doplníme komentář *Pro řetězce se použije porovnávání nastavené v databázi*. Komentáře se od příkazu oddělují apostrofem. Komentáře mohou tvořit samostatný řádek, který dokumentuje proceduru. V kódu procedury můžeme zařadit pro přehlednost volné řádky.

Visual Basic doplňuje do příkazů vhodné mezery, popř. odstraňuje zbytečné. Modře zvýrazňuje klíčová slova, zeleně zobrazuje komentáře.<sup>111</sup> Pro přehlednost Visual Basic zobrazuje před začátkem procedury vodorovnou čáru<sup>112</sup>.

Uložení  
modulu



S71

Vlastní funkce

Uložíme vytvořený modul (či všechny moduly) příkazem FILE, SAVE ENCIAN nebo klávesami **Ctrl S** nebo klepnutím do tlačítka **Save**. Zobrazí se dialogové okno pro zadání názvu modulu s připraveným názvem *Module1*. Název přepíšeme na *S71 Vlastní funkce*.<sup>113</sup> Při pozdějších úpravách se již nebude Access na název modulu ptát a bude přepisovat původní modul. Kombinací kláves levý **Alt F11** se vrátíme do Accessu. Stejnou kombinací se později můžeme vrátit do okna Visual Basicu.

Moduly můžeme zaznamenat kdykoliv při jejich tvorbě příkazem z menu FILE, SAVE nebo kombinací kláves **Ctrl S** nebo klepnutím do tlačítka **Save**.<sup>114</sup> Moduly na rozdíl od jiných objektů nemůžeme uložit pod jiným názvem. Můžeme jej však přejmenovat stejným způsobem jako ostatní objekty.



S71

CeléJméno

Ověříme fungování připravené funkce. Připravíme jednoduchý dotaz vycházející z tabulky *Personal*. Vypíšeme příjmení, jméno a výraz:<sup>115</sup>

Celé jméno: CeléJméno(Titul1; Jméno; Příjmení; Titul2)

Ve Visual Basicu jsme argumenty funkce oddělovali čárkou, v aplikaci funkce v dotazu, formuláři či sestavě se argumenty oddělují středníkem. Spuštěním dotazu ověříme fungování funkce. Dotaz uložíme pod názvem *S71 CeléJméno*. Obdobně lze funkci použít ve formulářích a sestavách.



Dále připravíme funkci, která zbaví řetězec háčků a čárek a první písmeno povýší na velké. V seznamu objektů modulů vybereme *S71 Vlastní funkce* a klepneme do tlačítka **Návrh** či poklepeme na název modulu. (Tlačítko **Spustit** není pro moduly k dispozici.) Klepnutím do tlačítka **Insert Procedure** vložíme další funkci tentokrát s názvem *BezDiakritiky*.<sup>116</sup> Názvy funkcí nesmí obsahovat mezery. Funkce má jediný argument. Výsledný tvar funkce je uveden v obr. 7-3.

Nejprve si vyložíme způsob odstranění háčků a čárek:

- Vstupní argument funkce je nazván *Vstup*.
- V pomocné proměnné *Cestina* připravíme formou řetězce všechna písmena s háčky a čárkami.
- V pomocné proměnné *ASCII* budou stejná písmena ve stejném pořadí bez háčků a čárek.
- Do proměnné *Vystup* zkopírujeme hodnotu proměnné *Vstup*.
- Po jednotlivých znacích procházíme proměnnou *Vystup*. Každý jednotlivý znak hledáme v proměnné *Cestina*. Když se shoduje znak z proměnné *Vystup* se znakem v proměnné *Cestina*, nahradíme znak z proměnné *Vystup* znakem z proměnné *ASCII* ze stejné pozice, na níž jsme jej našli v proměnné *Cestina*.
- Na závěr proměnnou *Vystup* uložíme do proměnné s názvem funkce, tj. *BezDiakritiky*.

<sup>111</sup> Zadáme-li z menu příkaz TOOL, OPTIONS, můžeme v kartě **Editor Format** upravit řadu parametrů zobrazování kódu procedur v modulech.

<sup>112</sup> Příkazem z menu TOOLS, OPTIONS můžeme v kartě **Editor** zrušením zaškrtnutí pole *Procedure Separator* potlačit zobrazování čar před začátky procedur.

<sup>113</sup> Modul *S71 Vlastní funkce* je v ukázkové databázi *Encian* připraven. Připravujte proto svůj modul *71 Vlastní funkce*. Funkce se nesmí ve dvou modulech jmenovat stejně. Upravujte proto i názvy funkcí, např. *PokusCeléJméno*.

<sup>114</sup> Aby se nově editovaná verze procedur promítla do práce Accessu, nemusíme moduly ukládat. Uložení je však nutné před ukončení práce s databází, neboť moduly se ukládají do souboru databáze.

<sup>115</sup> Funkci ve výrazu můžeme připravit *Tvůrcem výrazů*. V levém dolní okně poklepeme na řádek *Funkce* a v rámci funkcí klepneme na řádek *Encian*. V prostředním okně se zobrazí seznam modulů obsahujících funkce. Klepneme na název modulu *S71 Vlastní funkce*. V pravém okně se zobrazí seznam funkcí modulu, tj. zatím pouze funkce *CeléJméno*. Poklepáním funkcí vložíme do výrazu včetně seznamu argumentů.

<sup>116</sup> V ukázkové databázi *Encian* již je funkce přichystána, proto ji nazveme *PokusBezDiakritiky*.



## OBR. 7-3: MODUL S71 VLASTNÍ FUNKCE



Funkce  
CeléJméno

Option Compare Database 'Pro řetězce se použije porovnávání nastavené v databázi

'Výpis celého jména  
Public Function **CeléJméno**(Titul\_před, Jméno, Příjmení, Titul\_zá)  
CeléJméno = Trim(Trim([Titul\_před] & " " & [Jméno]) & " " & [Příjmení]) & [Titul\_zá]  
End Function



Funkce  
BezDiakritiky

'Konverze textu do textu bez háčků a čárek  
Public Function **BezDiakritiky**(Vstup)  
Cestina = "áčďěíĺľňóřšťůúýžäöü"  
ASCII = "acdeeillnorstuuyzaeou"  
Vystup = Vstup  
For K = 1 To Len(Vystup)  
For J = 1 To Len(Cestina)  
If Mid(Vystup, K, 1) = Mid(Cestina, J, 1) Then  
P = Mid(ASCII, J, 1)  
If K = 1 Then P = UCase(P)  
Vystup = Left(Vystup, K - 1) & P & Right(Vystup, Len(Vystup) - K)  
End If  
Next  
Next  
BezDiakritiky = Vystup  
End Function

V proceduře jsou použity některé funkce Accessu:

- *Len(proměnná)*: vrací délku textového řetězce.
- *Mid(proměnná;začátek;délka)*: vrací část řetězce od zadané pozice o zadaném počtu znaků.
- *Ucase(proměnná)*: vrací proměnnou po transformaci jejich písmen na velká.
- *Left(proměnná;délka)*: vrací levou část řetězce o zadaném počtu znaků.
- *Right(proměnná;délka)*: vrací pravou část řetězce o zadaném počtu znaků.

V proceduře jsou použity dva příkazy pro řízení běhu programu.

*For – Next*

Příkaz **For** je příkazem cyklu<sup>117</sup>, umožňuje opakovat skupinu příkazů. Cyklus **For** je cyklus s čítačem. Začíná klíčovým slovem **For** a provádí pro stanovený počet opakování skupinu příkazů ukončených klíčovým slovem **Next**. Počet provedených opakování je zaznamenáván v čítači. Obecný zápis příkazu (syntaxe)<sup>118</sup>:

**For** čítač = začátek **To** konec [**Step** krok]

[příkazy]

**[Exit For]**

[příkazy]

**Next** [čítač]

Za klíčovým slovem **Step** můžeme uvést krok zvyšování čítače. Neuvedeme-li krok, bude roven jedné. Příkaz **Exit For** umožňuje předčasně ukončit cyklus, i když nebyla dosažena koncová hodnota čítače. Použití proměnné *čítač* v příkazu **Next** není povinné, pouze může zpřehlednit program, zejména v případě vnořených cyklů.

*If - Then - Else*

Podmíněný příkaz **If** umožňuje provést některý příkaz (popř. skupiny příkazů) jen při splnění (popř. nesplnění) podmínky. Syntaxe:

- jednořádková verze:

**If** podmínka **Then** [příkaz] [**Else** příkaz]

<sup>117</sup> Příkaz cyklu obsahuje několik dílčích příkazů.

<sup>118</sup> V syntaxi budeme klíčová slova zvýrazňovat tučným písmem, proměnné části kurzívou. Nepovinné části budou uvedeny v hranatých závorkách.



– víceřádková verze:

```
If podmínka Then
  [příkazy]
[ElseIf podmínka Then
  [elseifpříkazy]]...
[Else
  [elsepříkazy]]
End If
```

Příkaz **ElseIf** se používá zřídka, umožňuje zadat alternativní příkazy.

## 7.2 Lokální modul výpočtu ve větě



S72

Aktualizace

E\_mail

Funkci *BezDiakritiky* budeme aplikovat na tabulku *Personal*. Pole *E\_mail* je zde dáno příjmením bez háčků a čárek odděleným tečkou od křestního jména bez háčků a čárek. Abychom odstranili případné překlepy v poli *E\_mail*, vytvoříme obsah pole *E\_mail* znovu aktualizacním dotazem. Dotaz bude čerpat z tabulky *Personal*, pomocí výrazu s využitím dříve připravených funkcí sestavíme obsah pole *E\_mail*.<sup>119</sup>

OBR. 7-4: DOTAZ S72 AKTUALIZACE E\_MAIL

Pole:	E_mail
Tabulka:	Personal
Aktualizovat do:	BezDiakritiky([Příjmení]) & " " & BezDiakritiky([Jméno])
Kritéria:	



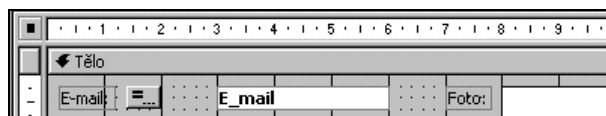
S71

Personal –

Identifikace

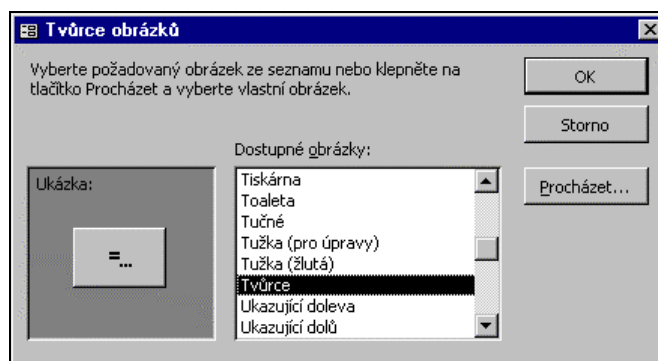
Chceme uživateli usnadnit vkládání nových zaměstnanců, při němž musí vložit pole *E\_mail*, neboť je primárním klíčem. Do formuláře *S41 Personal – Identifikace* chceme vložit tlačítko, které v aktuální větě sestaví *E\_mail* z polí *Příjmení* a *Jméno*. Nejprve vytvoříme kopii formuláře *S41* pod názvem *S71 Personal – Identifikace*. Otevřeme návrh formuláře.

OBR. 7-5: TLAČÍTKO TVORBA VE FORMULÁŘI S71 PERSONAL IDENTIFIKACE



V *Soupravě nástrojů* klepneme do tlačítka **Příkazové tlačítko** a vedle popisku *E\_mail* vytvoříme ve formuláři příkazové tlačítko. Vlastnosti *Název* a *Titulek* tlačítka změníme na *Tvorba*. V tlačítku se zobrazí text z *Titulku*. Tlačítko s textem musí však být vyšší, aby text byl dobře vidět. Proto raději zobrazíme na tlačítku obrázek. Klepneme do tlačítka se třemi tečkami na konci řádku vlastnosti *Obrázek*. Zobrazí se dialogové okno **Tvůrce obrázků** (viz obr. 7-6).

OBR. 7-6: DIALOGOVÉ OKNO TVŮRCE OBRÁZKU



<sup>119</sup> Aktualizaci pole, které je součástí referenční integrity, si můžeme dovolit jen díky aktivní vlastnosti *Aktualizace souvisejících polí* v kaskádě relací *Personal – Faktury* a *Personal – Výjezdy*.



Tlačítko  
s obrázkem

Výpočet ve větě

V okně vybereme z řady dostupných obrázků *Tvůrce*. V levé části vidíme náhled vybraného obrázku. Tlačítkem **Procházet** bychom mohli vybrat i námi připravený obrázek typu rastr (bmp) či ikony (ico). Klepnutím do tlačítka **OK** se obrázek přiřadí tlačítku.

Chceme, aby se při klepnutí na tlačítko provedl programový kód sestavující z polí *Příjmení* a *Jméno* pole *E\_mail*. V řádce vlastnosti *Při klepnutí* tlačítka *Tvorba* klepneme do tlačítka se třemi tečkami. V dialogovém okně **Vybrat tvůrce** zvolíme *Tvůrce kódu*. Otevře se editor Visual Basicu. V projektu (okno vlevo nahoře) přibyl modul ve skupině *Microsoft Access Class Objects* s názvem podle názvu formuláře. V modulu se založila nová procedura (přesněji podprogram) s názvem *Tvorba\_Click()*. Název procedury obsahuje název objektu (*Tvorba*) a událost objektu, při níž se procedura provádí (při klepnutí – *Click*).<sup>120</sup> Mezi příkazy začátku a konce procedury dopíšeme jediný příkaz, kterým do pole *E\_mail* vložíme spojení textových polí *Příjmení* a *Jméno* bez diakritiky oddělených tečkou (obr. 7-7).

OBR. 7-7: PROCEDURA TVORBA\_CLICK



Podprogram  
Tvorba\_Click

Option Compare Database	'Pro řetězců se použije porovnávání nastavené v databázi
Private Sub <b>Tvorba_Click()</b>	
	E_mail = BezDiakritiky(Příjmení) & "." & BezDiakritiky(Jméno)
End Sub	

Všimněme si, že procedura nemá žádný vstupní argument.

Kombinací **Alt F11** se vrátíme do Accessu a vyzkoušíme funkci tlačítka. Změníme křestní jméno prvního zaměstnance z *Miloše* na *Františka* a klepneme do tlačítka **Tvorba**. E-mail se upravil.

Chceme, aby se pole *E\_mail* upravilo vždy, když upravíme pole *Jméno* nebo *Příjmení* a přejdeme na jiné pole (klávesou **Enter**, tabulátorem nebo klepnutím myši). Do vlastnosti *Při výstupu* (Exit) polí *Jméno* i *Příjmení* vložíme kód shodný s předchozí procedurou (viz obr. 7-8).

OBR. 7-8: PROCEDURY JMÉNO\_EXIT A PŘÍJMENÍ\_EXIT



Podprogram  
Jméno\_Exit



Podprogram  
Příjmení\_Exit

Private Sub <b>Jméno_Exit</b> (Cancel As Integer)	
	E_mail = BezDiakritiky(Příjmení) & "." & BezDiakritiky(Jméno)
End Sub	
Private Sub <b>Příjmení_Exit</b> (Cancel As Integer)	
	E_mail = BezDiakritiky(Příjmení) & "." & BezDiakritiky(Jméno)
End Sub	



Aby stejný kód nebyl obsažen ve třech procedurách, připravíme společný podprogram. Zůstaneme stále v modulu *Form\_S71 Personal – Identifikace*. Klepneme do tlačítka **Insert Procedure** a vytvoříme podprogram s názvem *TvorbaEMail* s jediným příkazem dle předchozích procedur. V předchozích procedurách potom upravíme jediný příkaz, který nahradíme voláním společné procedury (viz obr. 7-9).

OBR. 7-9: ZAVEDENÍ SPOLEČNÉ PROCEDURY

Option Compare Database	
Private Sub <b>Jméno_Exit</b> (Cancel As Integer)	
	TvorbaEMail
End Sub	
Private Sub <b>Příjmení_Exit</b> (Cancel As Integer)	
	TvorbaEMail
End Sub	
Private Sub <b>Tvorba_Click</b> (Cancel As Integer)	
	TvorbaEMail
End Sub	

<sup>120</sup> Když bychom změnili název tlačítka, nezměnil by se automaticky název procedury.





```
Public Sub TvorbaEMail()  
    E_mail = BezDiakritiky(Příjmení) & "." & BezDiakritiky(Jméno)  
End Sub
```



Accessem vkládané událostní procedury jsou seřazeny dle abecedy. V okně kódu jsou zobrazeny všechny procedury. Klepnutím do tlačítka **Procedure View** v levém dolním rohu okna kódu můžeme zobrazit jen proceduru, v níž byl kurzor. Mezi procedurami se potom můžeme pohybovat klávesami **Ctrl** **PageDown** a **Ctrl** **PageUp**. Tlačítkem **Full Module View** zobrazíme opět celý modul se všemi procedurami.

### 7.3 Lokální modul pro výpočet mezi větami

Chceme nyní provést kontrolu vykazování služebních cest. Prověříme, zda se u jednotlivých zaměstnanců nepřekrývají vykázané služební cesty, tj. když seřadíme služební cesty podle pracovníků a podle zahájení, služební cesta nesmí začínat dříve, než skončila předchozí cesta stejného pracovníka.



S73

Kontrola cest

Vytvářecím dotazem nejprve připravíme pomocnou tabulku, v níž budou služební cesty seřazeny podle zaměstnanců a podle doby zahájení a v níž bude doplněno nové pole *Odstup* zatím vyplněné nulami.

Vytvářecí dotaz bude vycházet z tabulky *Cesty*, z níž převezme všechna pole, a z tabulky *Výjezdy*, z níž převezme e-mail zaměstnance. Navíc doplní pole *Odstup*. Dotaz vytvoří tabulku *Kontrola\_cest*.

OBR. 7-10: DOTAZ S73 KONTROLA CEST

Cesty  
Výjezdy  
(30 vět)

Pole:	E_mail	Cesty.*	Zahájení	Odstup: 0
Tabulka:	Výjezdy	Cesty	Cesty	
Řadit:	vzestupně		vzestupně	
Zobrazit:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Kritéria:				



S73

Kontrola cest

Data z tabulky *Kontrola\_cest* zobrazíme ve formuláři. Věta tabulky není složitá, vejde se na jeden řádek. Tvorbu formuláře urychlíme použitím průvodce *Automatický formulář: tabelární*. (Data pocházejí z tabulky *Kontrola\_cest*.) Formulář uložíme pod názvem *S73 Kontrola cest* (viz obr. 7-11). Do zápatí formuláře umístíme příkazové tlačítko s názvem i titulkem *Kontrola1* (na konci názvu tlačítka je číslice *1*).

OBR. 7-11: FORMULÁŘ S73 KONTROLA CEST

Vlastnosti *Při klepnutí* tlačítka **Kontrola1** přiřadíme programový kód. V proceduře budeme postupně procházet věty tabulky *Kontrola\_cest* a do pole *Odstup* budeme počítat rozdíl hodnoty pole *Zahájení* a hodnoty pole *Ukončení* zapamatované z předchozí věty (v proměnné *Konec\_predchozi*, viz obr. 7-12).



## OBR. 7-12: PROCEDURA KONTROLA1\_CLICK



Podprogram  
Kontrola1  
\_Click

```
Option Compare Database
Private Sub Kontrola1_Click()
Set Kontrola = CurrentDb.OpenRecordset("Kontrola_cest")
Konec_predchozi = 0
Do Until Kontrola.EOF
    Kontrola.Edit
    Kontrola!Odstup = Kontrola!Zahájení - Konec_predchozi
    Konec_predchozi = Kontrola!Ukončení
    Kontrola.Update
    Kontrola.MoveNext
Loop
MsgBox "Hotovo!"
End Sub
```

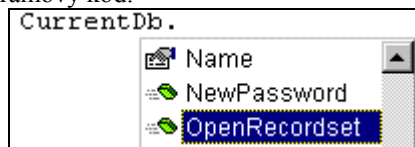
Objekty,  
metody,  
vlastnosti

Access obsahuje řadu objektů. *Objektem* je např. formulář, který obsahuje objekty ovládacích prvků.<sup>121</sup> Některé objekty mají svou vizuální reprezentaci (např. formulář), jiné jsou přístupné jen v kódu Visual Basicu. Objekty mají *vlastnosti*, které popisují objekt, např. formulář má vlastnost *Výchozí zobrazení*, v níž můžeme zadat *nekonečný formulář*. *Metody* jsou akce, které můžeme zadat objektu k provedení.

Poznali jsme již funkce, jejichž výsledkem byla hodnota (celé jméno či vstup bez háčeků a čárek). Kromě těchto obvyklých funkcí existují objektové funkce, jejichž výsledkem je objekt. Objekt můžeme také přiřadit proměnné, musíme tak však učinit příkazem *Set*. Přiřazením objektu proměnné můžeme zpřehlednit programový kód.

Objektová  
funkce  
*CurrentDb*

Objektová funkce *CurrentDb* vrací objektovou proměnnou typu *Database*, která představuje aktuálně otevřenou databázi v okně programu Microsoft Access. Když zapíšeme název funkce do kódu a tečku, Access nám nabízí vlastnosti (např. *Name*) a metody (např. *OpenRecordset*), které přísluší k objektu. Vlastnosti a metody jsou odlišeny ikonkami<sup>122</sup>. Vybereme metodu *OpenRecordset*, která vytvoří nový objekt *Recordset* – sadu záznamů z tabulky *Kontrola\_cest* do proměnné *Kontrola*.<sup>123</sup>



Nyní se můžeme na objekt datové sady odkazovat názvem *Kontrola*. S objektem provádíme akce – metody (*Edit*, *Update*, *MoveNext*) a využíváme jeho vlastnosti (*EOF*).

Před zpracováním jednotlivých vět datové sady (tabulky) *Kontrola\_cest* uložíme do pomocné proměnné *Konec\_predchozi* hodnotu 0, kterou využijeme při zpracování první větě datové sady. Proměnná bude vždy obsahovat konec předchozí služební cesty, abychom jej mohli porovnávat se začátkem následující cesty. Zpočátku bude proměnná vynulována, teprve po zpracování první větě jí přiřadíme ukončení první cesty.

Do – Loop

Ve funkci *BezDiakritiky* jsme se seznámili s cyklem *For – Next* s počítadlem. Zde použijeme příkaz cyklu *Do – Loop*. Provádí příkazy, které jsou zapsány mezi příkazy **Do** a **Loop**, dokud není splněna podmínka zapsaná v řádku příkazu **Do** nebo v řádku příkazu **Loop**. Syntaxe:<sup>124</sup>

– verze s podmínkou na začátku:

```
Do [{While|Until}podmínka]
    [příkazy]
[Exit Do]
    [příkazy]
Loop
```

<sup>121</sup> Mnoho objektů má kolekce, které definují nastavení několika objektů v daném objektu, např. objekt formuláře má kolekci *Controls*, která nastavuje všechny objekty definované ve formuláři.

<sup>122</sup> Access pomáhá i s laděním programů. Po odeslání špatně zapsaného příkazu je takový příkaz zobrazen červeně. Další chyby můžeme odhalit příkazem *DEBUG, COMPILE ENCIAN*. Zbývající formální chyby jsou odhaleny až při provozu modulu. Access při chybě automaticky zobrazí modul a zvýrazní řádek, v němž došlo k chybě.

<sup>123</sup> Tento objekt připojí do kolekce *Recordsets*.

<sup>124</sup> Závorky {} značí, že vybíráme některou z položek uvedených v seznamu a oddělených od sebe znakem |.





– verze s podmínkou na konci:

**Do**

[příkazy]

**[Exit Do]**

[příkazy]

**Loop** [{**While**|**Until**}podmínka]

Operace provedené pro větu z datové sady:

- Metoda *Edit* zkopíruje aktuální záznam z aktualizovatelného objektu *Recordset* do vyrovnávací paměti kopírování pro další úpravu. Tuto metodu je nutné provést vždy před programovou aktualizací jednotlivé věty datové sady.
- Do pole *Odstup* spočteme rozdíl zahájení cesty a ukončení předchozí cesty (v proměnné *Konec\_predchozi*). Při zpracování první věty výsledkem bude sériové číslo data zahájení cesty (pořadí od roku 1900), neboť v proměnné *Konec\_predchozi* byla uložena nula.
- Pro zpracování další věty si do proměnné *Konec\_predchozi* zapamatujeme ukončení zpracovávané služební cesty.
- Metoda *Update* uloží obsah vyrovnávací paměti pro kopírování do aktualizovatelného objektu *Recordset* a ukončí tak aktualizaci věty datové sady.
- Metoda *MoveNext* posune ukazovátka zpracování na další větu datové sady.
- Příkaz **Loop** vrátí program na předchozí příkaz **Do**. Vlastnost datové sady *EOF* (end of file) má hodnotu *False* (nepravda), dokud jsme nevyčerpali všechny věty datové sady. Až je ukazovátka datové sady za poslední větou, vrátí funkce *EOF* hodnotu *True* (pravda) a program pokračuje příkazy za příkazem **Loop**.<sup>125</sup>

Po zpracování všech vět zobrazíme hlášení *Hotovo!* v okně s titulkem *Microsoft Access*.<sup>126</sup>

Po zapsání programového kódu se vrátíme do Accessu, kde po vytvoření tabulky *Kontrola\_cest* zobrazíme formulář a klepnutím do tlačítka **Kontrola1** otestujeme náš programový kód. Po skončení připravené procedury se objeví hlášení *Hotovo!*, pole *Odstup* však je pro všechna pole formuláře nulové. Jakmile se myši přiblížíme k jednotlivým hodnotám, aktualizují se. Program funguje, pouze se neaktualizovalo zobrazení ve formuláři.

Ve formuláři připravíme napravo od prvního tlačítka nové tlačítko s názvem a titulkem *Kontrola2*, kterému přiřadíme zdokonalenou proceduru. K němu připojíme při klepnutí programový kód dle obr. 7-13. Programový kód je v mnohém podobný předchozímu kódu, proto jej v editoru VBA zkopírujeme pomocí schránky z procedury *Kontrola1\_Click()* do procedury *Kontrola2\_Click()*.

OBR. 7-13: PROCEDURA **KONTROLA2\_CLICK**<sup>127</sup>



Podprogram  
Kontrola2  
\_Click

```
Private Sub Kontrola2_Click()  
Set Kontrola = CurrentDb.OpenRecordset("Kontrola_cest")  
With Kontrola  
Konec_predchozi = 0  
Do Until .EOF  
.Edit  
!Odstup = !Zahájení - Konec_predchozi  
Konec_predchozi = !Ukončení  
.Update  
.MoveNext  
Loop  
End With  
Repaint  
MsgBox "Hotovo!"  
End Sub
```

With

V proceduře jsme použili příkaz *With*, kterým můžeme zjednodušit odkazy na složité objekty. Syntaxe:

<sup>125</sup> Podobně existuje vlastnost *BOF* (bottom of file), která má hodnotu *False*, pokud není ukazovátka před první větou.

<sup>126</sup> Příkaz *MsgBox* je analogií akce *OknoSeZprávou* v makrech.

<sup>127</sup> Příkazy shodné s předchozí procedurou jsou vypisovány kurzívou.



**With** odkaz na objekt

[příkazy procedury]

**End With**

Díky příkazu *With* nemusíme opakovaně specifikovat objekt *Kontrola*, pouze tečkou (při oddělování vlastností a metod) a vykřičníkem (při oddělování dílčích objektů) signalizujeme návaznost na objekt. Názvy proměnných (*Konec\_predchozi*) zůstávají nedotknuty.

Překreslení  
formuláře

Na závěr procedury jsme doplnili spuštění metody objektu formuláře. Objekt formuláře bychom mohli zapsat *Forms![S73 Kontrola cest]*<sup>128</sup>, pro aktuálně otevřený objekt můžeme použít zjednodušeného tvaru *Me* nebo použít neupřesněný název metody. Metoda *Repaint* překreslí formulář a zobrazí v něm tak aktuální hodnoty tabulky *Kontrola\_cest*.

V první větě zatím uvádíme nesmyslné sériové číslo data zahájení první služební cesty. V prvních větách každého zaměstnance je nesmyslné (většinou záporné) číslo dané rozdílem zahájení první cesty zaměstnance a ukončení poslední cesty předchozího zaměstnance. Tyto nepřesnosti nyní opravíme. Do nevyhodnotitelných polí *Odstup* v první větě každého zaměstnance vložíme prázdnou hodnotu (obr. 7-14). Procedura bude spouštěna klepnutím do nového tlačítka.

OBR. 7-14: PROCEDURA KONTROLA3\_CLICK



Podprogram  
Kontrola3  
\_Click

```
Private Sub Kontrola3_Click()  
Set Kontrola = CurrentDb.OpenRecordset("Kontrola_cest")  
With Kontrola  
.Edit  
!Odstup = Null  
E_mail_predchozi = !E_mail  
Konec_predchozi = !Ukončení  
.Update  
.MoveNext  
Do Until .EOF  
.Edit  
If !E_mail = E_mail_predchozi Then  
!Odstup = !Zahájení - Konec_predchozi  
Else  
!Odstup = Null  
E_mail_predchozi = Kontrola!E_mail  
End If  
Konec_predchozi = !Ukončení  
.Update  
.MoveNext  
Loop  
End With  
Repaint  
MsgBox "Hotovo!"  
End Sub
```

Poznámky k proceduře:

- Před cyklem zpracování jednotlivých vět datové sady zpracujeme první větu. Do pole *Odstup* vložíme prázdnou hodnotu. Do proměnné *E\_mail\_predchozi* si zapamatujeme *E\_mail* z první věty, do proměnné *Konec\_predchozi* si zapamatujeme *Ukončení*. Ukazovátko přesuneme na druhou větu.
- V cyklu jednotlivých vět potom diferencujeme mezi dvěma případy:
  - Jestliže dále zpracováváme stejného zaměstnance (*E\_mail* zůstává stejný), vypočteme pro něj pole *Odstup*.
  - Jestliže jsme přešli na dalšího zaměstnance, vložíme do pole *Odstup* prázdnou hodnotu a zapamatujeme si *E\_mail* zaměstnance (*E\_mail\_predchozi*), abychom mohli v příští větě zjistit, zda zpracováváme stejného zaměstnance.

<sup>128</sup> Hranaté závorky jsou nutné v případě, že se název objektu skládá z více slov.



Do formuláře přidáme ještě jedno tlačítko s názvem a titulkem *Kontrola4*. Proceduru doplníme o vypisování mezivýsledků formou dialogového okna se zprávou (viz horní část obr. 7-17).

**OBR. 7-15: PROCEDURA KONTROLA4\_CLICK (ZJEDNODUŠENÁ VERZE VIZ STR. 190)**



Podprogram  
Kontrola4  
\_Click

```
Private Sub Kontrola4_Click()  
Set Kontrola = CurrentDb.OpenRecordset("Kontrola_cest")  
With Kontrola  
    .Edit  
    K = 1  
    !Odstup = Null  
    T = K & ". věta" & Chr(13)  
    T = T & "E-mail předchozí: žádný" & Chr(13)  
    T = T & "Konec předchozí: žádný" & Chr(13)  
    T = T & "E-mail aktuální: " & !E_mail & Chr(13)  
    T = T & "Začátek aktuální: " & !Zahájení & Chr(13)  
    T = T & "Odstup začátku od konce: " & !Odstup  
    MsgBox (T)  
    E_mail_predchozi = !E_mail  
    Konec_predchozi = !Ukončení  
    .Update  
    .MoveNext  
    K = K + 1  
    Do Until .EOF  
        .Edit  
        T = K & ". věta" & Chr(13)  
        T = T & "E-mail předchozí: " & E_mail_predchozi & Chr(13)  
        T = T & "Konec předchozí: " & Konec_predchozi & Chr(13)  
        T = T & "E-mail aktuální: " & !E_mail & Chr(13)  
        T = T & "Začátek aktuální: " & !Zahájení & Chr(13)  
        If !E_mail = E_mail_predchozi Then  
            !Odstup = !Zahájení - Konec_predchozi  
        Else  
            !Odstup = Null  
            E_mail_predchozi = !E_mail  
        End If  
        Konec_predchozi = !Ukončení  
        T = T & "Odstup začátku od konce: " & !Odstup  
        MsgBox (T)  
        .Update  
        .MoveNext  
        K = K + 1  
    Loop  
End With  
Repaint  
MsgBox "Hotovo!"  
End Sub
```

Poznámky k proceduře:

- V počítadle *K* si počítáme pořadí věty, abychom ho mohli zobrazit v dialogovém okně mezivýsledků.
- Text zobrazený v dialogovém okně připravujeme do proměnné *T*. Funkce *Chr(13)* odřádkuje.
- Po každé větě se zobrazuje dialogové okno mezivýsledků. Program pokračuje v provádění příkazů po klepnutí do tlačítka **OK**.

Proceduru můžeme předčasně ukončit stisknutím kombinace kláves **Ctrl** **Break** a klepnutím do tlačítka **End** v dialogovém okně **Microsoft Visual Basic**.



V poslední verzi procedury provedeme následující změny:

- Jedno tlačítko **Kontrola5a** bude sloužit k provedení procedury bez vypisování mezivýsledků. Druhé tlačítko **Kontrola5b** provede proceduru s vypisováním mezivýsledků.
- Také vypisování mezivýsledků bude možné předčasně ukončit klepnutím do tlačítka, nikoliv jen kombinací kláves **Ctrl** **Break**.

Procedury v obr. 7-16 budou komentovány na následující stránce.

**OBR. 7-16: PROCEDURY KONTROLA5, KONTROLA5A\_CLICK A KONTROLA5B\_CLICK**



Podprogramy  
Kontrola5  
Kontrola5a  
\_Click  
Kontrola5b  
\_Click

```
Private Sub Kontrola5(Prepinac)
Set Kontrola = CurrentDb.OpenRecordset("Kontrola_cest")
With Kontrola
.Edit
K = 1
!Odstup = Null
If Prepinac = "ano" Then
T = K & ". věta" & Chr(13)
T = T & "E-mail předchozí: žádný" & Chr(13)
T = T & "Konec předchozí: žádný" & Chr(13)
T = T & "E-mail aktuální: " & !E_mail & Chr(13)
T = T & "Začátek aktuální: " & !Zahájení & Chr(13)
T = T & "Odstup začátku od konce: " & !Odstup
MsgBox (T)
End If
E_mail_predchozi = !E_mail
Konec_predchozi = !Ukončení
.Update
.MoveNext
K = K + 1
Do Until .EOF
.Edit
T = "E-mail předchozí: " & E_mail_predchozi & Chr(13)
T = T & "Konec předchozí: " & Konec_predchozi & Chr(13)
T = T & "E-mail aktuální: " & !E_mail & Chr(13)
T = T & "Začátek aktuální: " & !Zahájení & Chr(13)
If !E_mail = E_mail_predchozi Then
!Odstup = !Zahájení - Konec_predchozi
Else
!Odstup = Null
E_mail_predchozi = !E_mail
End If
Konec_predchozi = !Ukončení
T = T & "Odstup začátku od konce: " & !Odstup & Chr(13) & Chr(13)
T = T & "Chcete pokračovat ve vypisování mezivýsledků?"
If Prepinac = "ano" Then
V = MsgBox(T, vbYesNo, K & ". věta")
If V = vbNo Then Prepinac = "ne"
End If
.Update
.MoveNext
K = K + 1
Loop
End With
Repaint
MsgBox "Hotovo!"
End Sub
```



```
Private Sub Kontrola5a_Click()
    Kontrola5 ("ano")
End Sub

Private Sub Kontrola5b_Click()
    Kontrola5 ("ne")
End Sub
```

Nejprve připravíme proceduru *Kontrola5*, která není událostní. Proceduru potom budou využívat dvě událostní procedury pro dvě tlačítka **Kontrola5a** a **Kontrola5b**. Událostní procedury by mohly být řešeny samostatně, byly by však velmi podobné, proto je budeme diferencovat vstupním parametrem *Prepinac*.

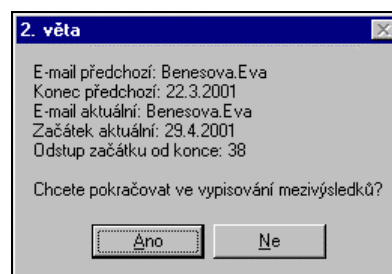


Z formuláře se kombinací kláves **[Alt][F11]** přesuneme do editoru VBA, kde příkazem INSERT, PROCEDURE či klepnutím do tlačítka **Insert Procedure** založíme novou proceduru *Kontrola5*. Procedura může být soukromá (*Private*) na rozdíl od dříve připravovaných funkcí, které byly veřejné (*Public*).<sup>129</sup>

Do procedury *Kontrola5* zkopírujeme kód z procedury *Kontrola4\_Click* a upravíme jej dle obr. 7-16. Poznámky k proceduře:

- Procedura bude spouštěna z procedury tlačítka **Kontrola5a** (se vstupní proměnnou *Prepinac* o hodnotě *ano*, tj. uživatel chce zobrazit mezivýsledky) nebo z procedury tlačítka **Kontrola5b** (se vstupní proměnnou *Prepinac* o hodnotě *ne*, tj. uživatel nechce zobrazit mezivýsledky).
- Dialogové okno mezivýsledků pro 1. větu se zobrazí jen, když *Prepinac* nabude hodnotu *ano*.
- Obdobně se zobrazí mezivýsledky i po dalších větách jen, když *Prepinac* nabývá hodnoty *ano*. Pro další věty je však příkaz *MsgBox* nahrazen funkcí *MsgBox*. Funkce *MsgBox* také zobrazuje dialogové okno se zprávou, avšak okno může obsahovat více tlačítek. Počet, uspořádání a názvy tlačítek určuje druhý argument funkce. Jako argument můžeme zadat číslo nebo použít systémovou (předem definovanou) hodnotu Visual Basicu, např. *vbYesNo*. Nabývá-li druhý argument hodnotu *vbYesNo* (tj. 4), zobrazuje se dialogové okno s tlačítky **Ano** a **Ne** (viz obr. 7-17). Po klepnutí do tlačítka **Ano** funkce *MsgBox* vrací hodnotu 6 (tj. *vbYes*), po klepnutí do tlačítka **Ne** funkce vrací hodnotu 7 (tj. *vbNo*).
- Jestliže uživatel klepne v dialogovém okně se zprávou do tlačítka **Ne**, uložíme do proměnné *Prepinac* hodnotu *ne*, čímž předčasně ukončíme vypisování mezivýsledků. (Zde je aplikována jednořádková verze příkazu *If–Then*.)
- Třetí argument funkce *MsgBox* je zobrazen v titulku dialogového okna se zprávou. Do titulku zobrazujeme pořadí věty.

OBR. 7-17: MEZIVÝSLEDKY



Do formuláře doplníme další dvě tlačítka s názvy a titulky **Kontrola5a** a **Kontrola5b**. Po klepnutí do tlačítka **Kontrola5a** se spustí procedura *Kontrola5* s argumentem *ano*. Po klepnutí do tlačítka **Kontrola5b** se spustí procedura *Kontrola5* s argumentem *ne*.

Závěrem uložíme formulář *S73 Kontrola cest*, čímž uložíme také lokální procedury.

## 7.4 Globální modul modifikující formulář a sestavu

Úkolem nového formuláře bude zobrazit schéma tří podlaží firmy Encián včetně zobrazení umístění zaměstnanců do kanceláří. Cílový stav schématu je zobrazen v obr. 7-18. Vytvoříme nejprve nový formulář *S74 Schéma kanceláří*. Ve formuláři chceme zobrazit všechny věty z tabulky *Personal*, přesto však nevychází z tabulky ani podkladového dotazu. Data jsou totiž uspořádána ve formuláři neobvykle. Neplatí, že by každá věta měla stejné uspořádání. Zobrazujeme více vět najednou.

Dialogové  
okno  
se zprávou

<sup>129</sup> U veřejných procedur se předpokládá jejich využití i v jiných modulech.



OBR. 7-18: SCHÉMA KANCELÁŘÍ ZAMĚSTNANCŮ

1. podlaží	<b>11</b> JUDr. Eva Benešová <i>Lukáš Škoda</i>	<b>12</b> Bc. Miloš Adamec	<b>13</b> Jiří Hanák Petra Kalousková
	<b>16</b> RNDr. Milan Smetana	<b>15</b> Milan Kos Dana Drobná	<b>14</b> Ing. Jan Dvořák
2. podlaží	<b>21</b> Ing. Pavel Beneš Ing. František Janda	<b>22</b> Petr Škoda PhDr. Martin Zeman	<b>23</b> RNDr. Eva Kolínská
	<b>25</b> Ing. Alena Pospíšilová Alice Sládková	<b>24</b> Jana Klímová	
3. podlaží	<b>31</b> Ing. Petr Novák, PhD.		
	<b>32</b> Marie Sladká		

Do formuláře umístíme jen popisky, které budeme později naplňovat programem. Popisky uspořádáme dle rozložení kanceláří v podlažích (viz obr. 7-19). Tučně zobrazíme popisky, které programem nebudeme měnit (označení podlaží a čísla kanceláří). Předpokládáme, že v každé kanceláři sedí maximálně dva zaměstnanci, pro něž přichystáme popis s názvem i titulkem složeným z písmene K, čísla kanceláře, podtržítka a pořadí zaměstnance v kanceláři. Titulky budeme programem měnit. Kanceláře ohraničíme obdélníky (viz obr. 7-18).

OBR. 7-19: FORMULÁŘ S74 SCHÉMA KANCELÁŘÍ – NÁVRH



S74  
Schéma  
kanceláří

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	Tělo													
1	1. podlaží	<b>11</b>	K11_1 K11_2										<b>13</b>	K13_1 K13_2
2		<b>16</b>	K16_1 K16_2										<b>14</b>	K14_1 K14_2
3	2. podlaží	<b>21</b>	K21_1 K21_2										<b>23</b>	K23_1 K23_2
4		<b>25</b>	K25_1 K25_2											
5														
6	3. podlaží												<b>31</b>	K31_1 K31_2
7													<b>32</b>	K32_1 K32_2



S74  
Schéma  
kanceláří

V programu budeme čerpat z datové sady zaměstnanců s přiřazenou kanceláří seřazených dle čísla kanceláře a příjmení. Dotaz připravíme pod názvem *S74 Schéma kanceláří* (viz obr. 7-20).

OBR. 7-20: DOTAZ S74 SCHÉMA KANCELÁŘÍ

Personal  
(19 vět)

Pole:	Personal.*	Kancelář	Příjmení
Tabulka:	Personal	Personal	Personal
Řadit:		vzestupně	vzestupně
Zobrazit:	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Kritéria:		Is Not Null	

Naplnění popisků formuláře hodnotami z datové sady dotazu *S74* provedeme v globálním modulu *S74 Schéma kanceláří*. Jeho proceduru *Schema* (viz obr. 7-21, procedura se nesmí jmenovat stejně jako modul) bude využívat formulář a později také sestava.





Podprogram  
Schema  
v modulu  
S74  
Schéma  
kanceláři

OBR. 7-21: PROCEDURA SCHEMA

```
Public Sub Schema(Objekt)
'Naplnění netučných popisků mezerami
For K = 0 To Objekt.Count - 1
Set Prvek = Objekt(K)
If TypeOf Prvek Is Label Then
If Prvek.FontWeight <> 700 Then
Prvek.Caption = " "
End If
End If
Next
'Naplnění netučných popisků daty s datové sady
Set Zam = CurrentDb.OpenRecordset("S74 Schéma kanceláři")
Kancelář_Předchozí = 0
With Zam
Do Until .EOF
If !Kancelář = Kancelář_Předchozí Then L = L + 1 Else L = 1
Prvek = "K" & Trim(Str(!Kancelář)) & "_" & Trim(Str(L))
Objekt(Prvek).Caption = CeléJméno(!Titul1, !Jméno, !Příjmení, !Titul2)
If !Pohlaví = "M" Then Objekt(Prvek).FontItalic = True
If !Úvazek <> 1 Then Objekt(Prvek).FontUnderline = True
If !Oprávnění Then Objekt(Prvek).FontWeight = 700
Select Case !Pracoviště
Case "DIS": Objekt(Prvek).ForeColor = vbRed
Case "KON": Objekt(Prvek).ForeColor = vbBlue
Case "REK": Objekt(Prvek).ForeColor = 8421440
Case "ANA": Objekt(Prvek).ForeColor = 32768
Case "PRG": Objekt(Prvek).ForeColor = 16512
Case "RED": Objekt(Prvek).ForeColor = 8388736
End Select
Kancelář_Předchozí = !Kancelář
.MoveNext
Loop
End With
End Sub
```

Poznámky k proceduře:

- Do argumentu *Objekt* budeme dosazovat odkaz na objekt (formulář nebo sestava).
- Procedura se skládá ze dvou částí. V první části naplníme popisky kanceláři mezerami místo titulků shodných s názvy popisků.<sup>130</sup> Naplnění mezerami je nutné, neboť existují kanceláře, v nichž nesedí dva zaměstnanci. V druhé části naplníme popisky kanceláři daty z podkladové datové sady.
- V první části obecně procházíme všechny objekty formuláře. Protože později budeme chtít proceduru použít obecně, vstupní argument *Objekt* zastupuje obecný objekt, který bude nabývat hodnot *Form*, *Report* či zjednodušeně *Me*.
- Vlastnost *.Count* udává počet objektů (ovládacích prvků) formuláře. Objekty jsou Accessem indexovány od nuly, proto je konečná hodnota počítadla cyklu nastavena na počet snížený o jedničku.
- Do objektové proměnné *Prvek* vložíme odkaz na K-tý objekt formuláře či sestavy, který lze obecně zapsat *Form(K)*, *Report(K)* či *Me(K)*.
- V podmínce dalšího příkazu testujeme, zda typ ovládacího prvku je popisek.
- V další podmíněném příkazu testujeme, zda vlastnost *Tloušťka písma* (*FontWeight*) prvku nemá hodnotu *tučné* (700). Jestliže se jedná o popisek nevypisovaný tučně, vložíme do jeho titulku (*Caption*) mezeru.

<sup>130</sup> Prvotní vyplnění titulků názvy má dokumentační význam. V návrhu formuláře (viz obr. 7-19) vidíme přehledně názvy popisků.



- V druhé části naplňujeme popisky kanceláří hodnotami z datové sady *S74 Schéma kanceláří*.
- Pro každou větu si do proměnné *Kancelář\_Předchozí* pamatujeme *Kancelář*, abychom ji v následující větě mohli porovnat s *Kanceláři* z další věty a zjistit tak, zda se *Kancelář* změnila a tudíž máme znovu nastavit na výchozí hodnotu 1 počítadlo zaměstnanců v kanceláři (*L*). Před zpracováním první věty vložíme do proměnné *Kancelář\_Předchozí* hodnotu 0, která jistě nebude rovna číslu první zpracovávané kanceláře.
- Další příkazy provádíme s datovou sadou *S74 Schéma kanceláří*, kterou zastupuje objektová proměnná *Zam* (zaměstnanci).
- Jestliže *Kancelář* je shodná s *Kanceláři* z předchozí věty, zvýšíme hodnotu počítadla *L* o 1.
- Do proměnné *Prvek* přichystáme název ovládacího prvku, s nímž budeme pracovat v dalších příkazech. Název se skládá z konstantního písmene *K* (kancelář), z čísla kanceláře (musíme funkcí *Str* převést číslo na text a funkcí *Trim* odstranit úvodní mezeru na nezobrazované kladné znaménko) a z pořadí zaměstnance v kanceláři (proměnná *L*).
- Do *Prvku* vložíme celé jméno zaměstnance vytvořené dříve připravenou funkcí *CeléJméno*.
- Jestliže se jedná o muže, přiřadíme vlastnosti prvku *Kurzíva* (*FontItalic*) hodnotu *ano* (*True*).
- Jestliže zaměstnanec není zaměstnán na plný úvazek, přiřadíme vlastnosti prvku *Podtržení písma* (*FontUnderline*) hodnotu *ano* (*True*).
- Jestliže zaměstnanec je oprávněn fakturovat, vypíšeme jeho jméno tučně (vlastnost *FontWeight* nabude hodnotu 700).
- V dalších příkazech přiřadíme prvku barvu diferencovaně dle *Pracoviště*. K větvení použijeme příkaz *Case* se syntaxí:

Select – Case

**Select** testovaný příkaz

[**Case** seznam hodnot-*n*

[příkazy-*n*]]...

[**Case Else**

příkazy]]

**End Select**

Barvu můžeme zadat konstantou Visual Basicu (např. *vbRed*, *vbBlue*) nebo číslem.

- Na jeden řádek můžeme psát více příkazů oddělených dvojtečkou.

Ve formuláři *S74* do vlastnosti *Při otevření* doplníme kód s jediným příkazem, který spustí proceduru *Schema* z globálního modulu s argumentem *Me*: *Schema(Me)*, popř. *Schema(Form)*.



S74  
Schéma  
kanceláří

Schéma chceme připravit pro účely tisku ve formě sestavy. Připravíme novou sestavu s názvem *S74 Schéma kanceláří*:

- Zdrojem sestavy není tabulka ani dotaz. Zrušíme zobrazení *Záhlaví a zápatí stránky*.
- Ve formuláři označíme všechny ovládací prvky a kombinací **Ctrl**+**C** je zkopírujeme do schránky, odkud je vložíme do sestavy kombinací **Ctrl**+**V**.
- Pro tělo sestavy definujeme vlastnost *Při formátování* prostřednictvím *Tvůrce kódu*. Jediným příkazem procedury *Tělo\_format* je příkaz *Schema(Me)*, popř. *Schema(Report)*.

## Shrnutí

1. *Moduly* obsahují procedury (funkce a podprogramy), které jsou vyšší formou automatizace operací Accessu než makra.
2. *Moduly* mohou být globální nebo lokální. Seznam *globálních* modulů je uveden v databázovém okně na kartě *Moduly*. Funkce a procedury z globálních modulů můžeme použít kdekoliv v databázi. *Lokální* moduly jsou součástí jiných objektů (formulářů, sestav). Vztahují se k událostním vlastnostem formulářů (např. *Při otevření*) či sestav (např. *Při formátování*) nebo k jednotlivým ovládacím prvkům (např. *Při klepnutí* do tlačítka).
3. Funkce i procedury mohou obsahovat argumenty, kterými modifikujeme jejich fungování. Argumenty jsou v modulu odděleny čárkami, ve výrazech v jiných objektech (dotazech, formulářích, sestavách) jsou odděleny středníky.
4. *Deklarace* modulu obsahují nastavení modulu a případně deklarace proměnných.
5. Objekty mají *vlastnosti*, které popisují objekt. *Metody* jsou akce, které můžeme zadat objektu k provedení.
6. V modulech lze používat řadu příkazů. Mezi základní patří přiřazovací příkaz *=*, objektový příkaz *Set*, příkaz *If – Then – Else*, příkazy cyklu *For – Next*, *Do – Loop*, příkaz pro práci s objektem *With*, příkaz větvení *Select Case*.