Programování ve 4th Dimension

Obsah

4th Dimension

Školící materiál

Programování ve 4th Dimension

Verze 6.0.5

Kurs vytvořen:

Jméno	E-Mail
Bart Scott	bart@acius.com
Kent Wilbur	kent@acius.com

Kurz upraven:

Jméno	E-Mail
Jaroslav Macháček	inforce@mbox.vol.cz

ACI University

1. Úvod

1.1. Vítejte

Tento kurz je zaměřen na nováčky v programování a ty, kteří začínají programovat ve 4thDimension ("4D"). Kurz zahrnuje základí zásady programování a rovněž specifické techniky programování ve 4D.

1.2. Jak budete v kursu pracovat

Délka trvání kursu je tři dny. Způsob práce je následující:

- Instruktor diskutuje některé rysy a demonstruje je s pomocí promítání na plátno, vy provádíte cvičení a příklady z této příručky, které procvičují diskutované rysy.
- Vy provádíte cvičení a příklady z této příručky, které procvičují diskutované rysy.
- Instruktor při cvičeních odpovídá na dotazy a poskytuje pomoc těm, kteří ji potřebují.

1.3. Scénář

Při práci v tomto kurzu budete přidávat nové rysy k zjednodušené databázi pro distributora videokazet. Je to tatáž databáze, kterou jste vytvořili v kurzu „Úvod do 4thDimension". Budete sledovat zákazníky, jejich faktury, položky těchto faktur a seznam prodejních produktů.

Databáze je pojmenována po vaší fiktivní wspolečnosti "ACI Video." Na konci kurzu bude databáze zahrnovat tabulky a pole ze seznamu níže.

[zDialogy]

Název pole	Typ	Vlastnosti
Povinnné	Logické	

[Zákazníci]

Název pole	Typ	Vlastnosti
IDzákazníka	Alfa 10	Nutný vstup; Pouze zobrazit; Indexované; Jedinečné
Jméno	Alfa 20	Indexované
Příjmení	Alfa 20	Indexované
Firma	Alfa 25	Indexované; Nutný vstup
Adresa	Alfa 25	
Město	Alfa 20	Indexované
Stát	Alfa 2	Indexované
PSČ	Alfa 10	Indexované
Telefon	Alfa 10	
PlátceDaně	Logické	
Důležitý	Logické	
CelkovéProdeje	Real	
DatumVytvoření	Datum	Pouze zobrazit; Neviditelné
ČasVytvoření	Čas	Pouze zobrazit; Neviditelné
DatumÚpravy	Datum	Pouze zobrazit; Neviditelné
ČasÚpravy	Čas	Pouze zobrazit; Neviditelné

[Faktury]

Název pole	Typ	Vlastnosti
IDZákazníka	Alfa 10	Nutný vstup, Neměnné; Indexované; Jedinečné
ČísloFaktury	Long Integer	Indexované; Jedinečné
DatumFaktury	Datum	Indexované
FakturaCelkem	Real	
Placeno	Logické	
ZpůsobPlatby	Alfa 15	
DatumVytvoření	Datum	Pouze zobrazit; Neviditelné
ČasVytvoření	Čas	Pouze zobrazit; Neviditelné
DatumUpravení	Datum	Pouze zobrazit; Neviditelné
ČasUpravení	Čas	Pouze zobrazit; Neviditelné

[PoložkyFaktur]

Název pole	Typ	Vlastnosti
ČísloFaktury	Long Integer	Indexované
IDZbožíProdukty	Long Integer	Indexované
Neužito	Logické	Neviditelné
Cena	Real	
Množství	Integer	
JednCena	Real	
CenaCelkem	Real	

[Produkty]

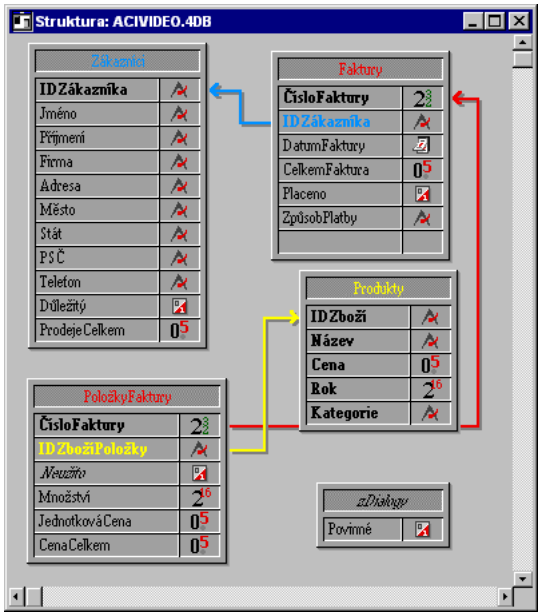
Název pole	Typ	Vlastnosti
IDZboží	Alfa 10	Nutný vstup; Neměnné; Indexované; Jedinečné
Název	Alfa 30	Indexované
Cena	Real	Indexované
Rok	Integer	Indexované
Kategorie	Alfa 15	Indexované
DatumVytvoření	Datum	Pouze zobrazit; Neviditelné
ČasVytvoření	Čas	Pouze zobrazit; Neviditelné
DatumUpravení	Datum	Pouze zobrazit; Neviditelné
ČasUpravení	Čas	Pouze zobrazit; Neviditelné

[zNavracenáČísla]

Název pole	Typ	Vlastnosti
NázevSekvence	Alfa 20	Indexované
Hodnota	Long integer	

[zSekvence]

Název pole	Typ	Vlastnosti
NázevSekvence	Alfa 20	Indexované
Hodnota	Long integer	



1.4. Co dostanete domů…

Na konci kurzu obdržíte disketu s počáteční verzí databáze a vaší konečnou verzí

1.5. Co je programování

Programování je, zcela jednoduše řečeno, způsob jak sdělit počítači co má dělat. Programováním můžete říci 4th Dimension, aby hrála zvuky, zobrazovala informace uživateli, sčítala položky. Můžete učinit svou databázi jednodušeji užívatelnou sloučením několika kroků do jedné metody, automatizováním některých funkcí, setříděním výsledku v abecedním pořadí a případným tiskem výsledku ve formě zprávy.

Jeden z důvodů proč je 4D oblíbená je, že pro základní databázové funkce potřebuje minimální programování. 4D je nazývána nástroj pro rychlé programování aplikací, protože za vás provede spoustu kroků automaticky: zobrazeí nabídky, oken a záznamů. V tomto kurzu se naučíte přidat tu a tam několik programovacích instrukcí k lepšímu přizpůsobení aplikace vašim požadavkům. Platí samozřejmě čím více chcete přizpůsobit databázi konkrétním požadavkům, tím více musíte provést programování.

Jestliže nemáte žádné zkušenosti s programovacími jazyky, vše co potřebujete je :

• nápad jak vylepšit databázi

• trochu trpělivosti k nalezení správných příkazů a experimentování s jazykem 4D

Při programování nemůžete poškodit váš počítač (pokud nevhodně nepoužijete některé příkazy ovládání zdrojů – ty jsou však zcela mimo rámec tohoto kurzu), jediné co můžete poškodit jsou vaše data. Jestliže jso vaše data důležitá, experimentujte pouze s kopií dat.

Je pouze jeden způsob jak se naučit programovat: metoda pokusů a omylů. Pohrajte si s příkazy jazyka, nebojte se chyb. To vše je součástí procesu učení.

1.6. Konvence

- "Programování" a "tvorba kódu" jsou synonyma pro instrukce, které budete vkládat.
- To co budete zapisovat je v tomto tvaru.

  Příklad: Napište Johnson do pole Příjmení.
- Speciální klávesy na klávesnici jsou uvedeny následovně:

  Stiskněte Enter.
- Jestliže je požadováno, abyste stiskli více než jednu klávesu současně je tento požadavek zapsán následovně:

  Stiskněte CTRL + Shift + 3.
  Vysvětlení: Stiskněte současně klávesy CTRL, Shift a 3.

  Macintosh™ ekvivalent bude okamžitě následovat po Windows™ v kurzívě, jak je ukázáno níže.
  Stiskněte: Ctrl + Shift + 3 *(Comma + Shift + 3)*
- Volba položky z nabídky je popsána následovně:

  Zvolte Soubor ¬ Otevřít (Ctrl + O) *(_ + O)*
  Vysvětlení: Z nabídky Soubor, zvolte položku Otevřít. Na Windows™ stiskněte Control + O.
  Na Macintosh™ stiskněte Comma + O.
- Položky 4D kódu jsou zobrazovány stejně jako jsou v 4D Editoru metod. Například:

[Zákazníci]Stát	Pole
ALERT	Příkaz
MyProcedure	Metoda (procedura)
- Sekce označené "Extra Credit" jsou výběrové. Jesliže skončíte svůj příklad dříve můžete pracovat v sekci "Extra Credit" a naučit se více.

1.7. Copyright

20883 Steven Creek Blvd.

Cupertino,. CA 95014

(408) 252-4444

Příklady kódu používané v této příkladu jsou považovány za veřejné, mohou být použity v libovolné databázi, kterou vytvoříte.

1.8. Obchodní značky

Macintosh™ je registrovaná obchodní značka Apple Computer, Inc.

Windows™ a Windows 95™ jsou registrované obchodní značky Microsoft Corporation.

4th Dimension. ACI, ACIUS jsou registrované obchodní značky ACI/ACI US, Inc.

4D je registrovaná obchodní značka ACI/ACI US, Inc.

1.9. Příkazy a konstanty

V tomto kurz se naučíte následující příkazy z jazyka 4D. Některé z těchto příkazů jsou podrobně vysvětleny jen v dalších rozšiřujících materiálech a jsou určeny pro samostudium.

- ->
- :=
- •…•
- Abs
- ACCEPT
- ACCUMULATE
- ADD RECORD
- ADD TO SET
- ALERT
- ALL RECORDS
- APPLY TO SELECTION
- ARRAY STRING
- BREAK LEVEL
- CANCEL
- Carriage return
- Case of…End case
- Char
- CLEAR NAMED SELECTION
- CLEAR SET
- CLOSE WINDOW
- CONFIRM
- Count parameters
- CREATE EMPTY SET
- CREATE RECORD
- CREATE SET
- Current date
- Current form table
- Current process
- Current time
- CUT NAMED SELECTION
- C_BOOLEAN
- C_DATE
- C_LONGINT
- C_POINTER
- C_REAL
- C_STRING
- Database event
- DELETE RECORD
- DELETE SELECTION
- DIALOG
- DIFFERENCE
- DISABLE BUTTON
- DISPLAY SELECTION
- DISTINCT VALUES
- ENABLE BUTTON
- EXECUTE
- EXPORT TEXT
- False
- Table name
- FIRST RECORD
- For…End for
- Form event
- GET WINDOW RECT
- GOTO AREA
- GOTO XY
- GRAPH TABLE
- If…Else…End if
- INTERSECTION
- IMPORT TEXT
- INPUT FORM
- Length
- Lowercase
- MENU BAR
- Menu bar height
- MESSAGE
- MESSAGES OFF
- MESSAGES ON
- MODIFY RECORD
- MODIFY SELECTION
- On Close Detail
- On Data Change
- On Load
- On Printing Footer
- On Validate
- Open window
- ORDER BY
- OUTPUT FORM
- PAGE SETUP
- PLATFORM PROPERTIES
- Position
- PRINT LABEL
- PRINT SELECTION
- Printing page
- QUERY
- QUERY BY EXAMPLE
- QUERY SELECTION
- QUIT 4D
- Record number
- Records in selection
- Records in set
- Records in table
- RELATE MANY
- RELATE MANY SELECTION
- RELATE ONE SELECTION
- Replace string Repeat…Until
- Replace string
- REPORT
- Request
- Round
- Save Existing Record Event
- SAVE RECORD
- Screen height
- Screen width
- Selected record number
- Self
- SEND PACKET
- Sequence Number
- SET ABOUT
- SET CHANNEL
- SET VISIBLE
- SET WINDOW TITLE
- Short
- String
- Substring

- Subtotal
- Sum
- Table name
- TRACE
- True
- UNION
- Uppercase
- USE NAMED SELECTION
- USE SET
- Windows

Q. QuickCode Pro™

QuickCode Pro™ (“QCP”) je produkt třetí strany, který může být zakoupen a „zasunut do 4D“. Tento produkt přidává k 4D mnoho dalších rysů, které urychlují psaní procedur a metod. QCP vám poskytne úplnou kontrolu editoru metod pomocí klávesových zkratek.

Natural Intelligence, Inc. věnovatla kopii QCP ACI US, Inc pro účely školení. Tato kopie může být používána pouze pro školící centra. V našem kurzu použijeme několik rysů k urychlení vývoje databáze.

Začneme importem maker při vývoji databáze, umožní nám to nezdržovat se samotným psaním a budeme mít více času na komentáře.

Q.1. Import maker do QCP

1. Vyberte Import/Export Macros… z nabídky the QCP Macros  .
2. Importujte tabulku nazvanou QCP Macros umístěnou ve složce Start With.
3. Uzavřete dialog Import/Export.
4. Vyberte Configure Macros… z nabídky QCP Macros  .
5. Otevřete makro Designer Name a nehraďte, “Your name here” here” vaším jménem.
6. Otevřete makro Designer Identifier. Nahraďte "J" iniciálou vašeho jména. Nahraďte slovo name vaším příjmením
   <> + Iniciála+ _ + vaše příjmení + := False
   Příklad: <>fJ_Steinman := True.

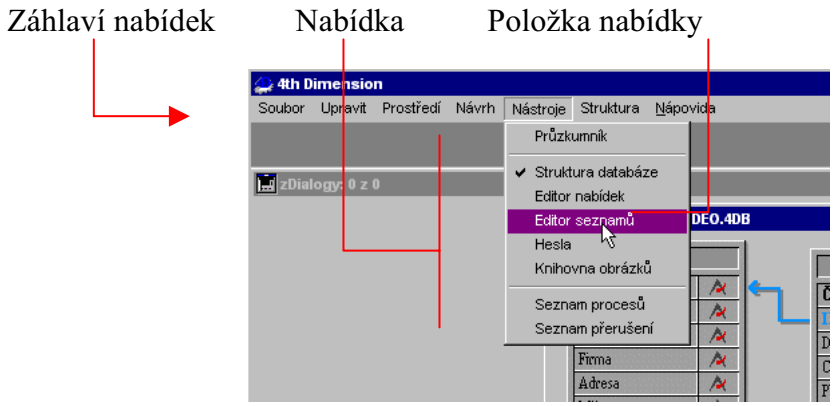
2. Vytvoření prostředí vlastních nabídek

2.1. Prostředí Vlastní nabídky

Prostředí uživatele poskytuje obecný soubor nabídek pro vkládání dat, úpravy, ukládání, mazání, import, export, tisk, dotazy, třídění, vytváření zpráv a prohlížení záznamů. Jestiže chcete můžete si vytvořit svůj vlastní soubor nabídek. Umožní vám to plně přizpůsobit rozhraní uživatele vlastním požadavkům. Jestliže nechcete, aby uživatel mazal zázmnamy, nedejte mu k dispozici položku nabídky pro mazání. Vytváření nabídek pro prosředí aplikace závisí pouze na Vás.

Po vytvoření první nabídky se prostředí aplikace stane dostupným. Vybrání Prostředí¬Aplikace způsobí, že 4D zobrazí vaše záhlaví nabídek místo své vlastní. Jediné akce, keré uživatel může provádět jsou ty, keré jste mu umožnili ve vytvořených nabídkách.

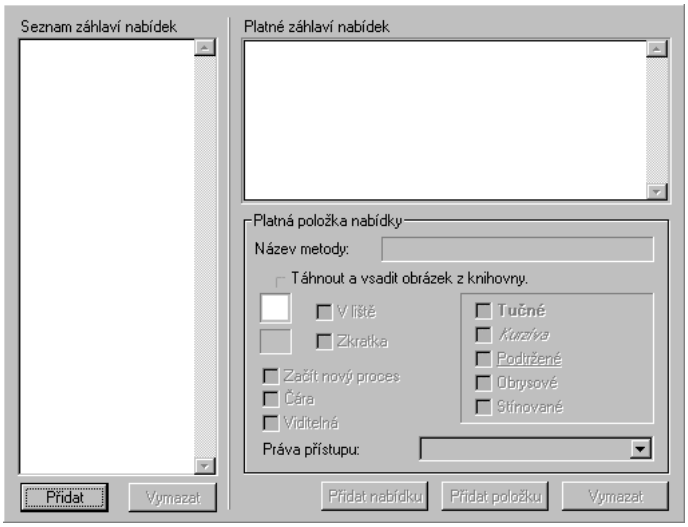
Nejdříve zopakování termínů nabídek. Záhlaví nabídek je soubor nabídek. Nabídka je sobor položek nabídek. Položka nabídky je jeden určitý řádek nabídky, který má jednu určitou funkci.



2.2. Vytváření nabídek

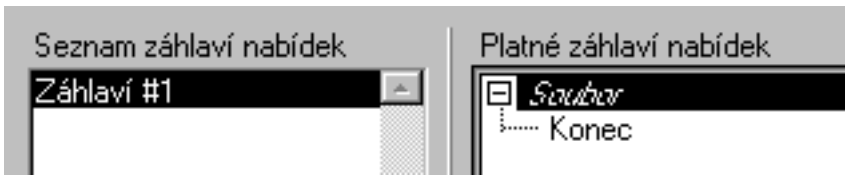
V prostředí návrháře můžete vytvářet nabídky a záhlaví nabídek. Soubor nabídek, keré vidíte v záhlaví okna (obrazovky) je nazývána záhlaví nabídek. Ve 4D vytvoříte nabídku tak, že ji vložíte do záhlaví nabídek.

Když vyberete Nástroje ¬ Editor nabídek, zobrazí 4D dialog záhlaví nabídek. Pro vytvoření Vašeho prvního záhlaví kepněte na tlačítko Přidat. 4D automaticky čísluje každé záhlaví nabídek, keré vytvoříte, začínajíc u Záhlaví #1.



2.3. Použití editoru nabídek

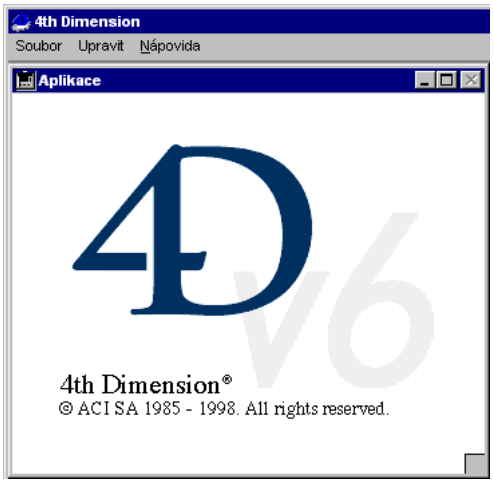
Když otevřete editor nabídek uvidíte, že obsahuje tři části. První v levé části je seznam záhlaví nabídek. Druhá v pravé části je seznam nabídek vybraného záhlaví nabídek. Třetí část pomáhá instruovat 4D, kerou metodu spustit, jestliže uživatel vybere položku nabídky. Tato část je v pravé spodní části okna a obsahuje rovněž nástroje pro vytváření vzhledu každé položky nabídky. Pro usnadnění vytváření obsahuje každé nové záhlaví nabídku Soubor s položkou Konec.



Když jste vytvořili záhlaví nabídek můžete již použít prostředí aplikace vyvráním Prostředí¬ Aplikace. Na rozdíl od prostředí uživatele prostředí aplikace nezobrazí ihned záznamy tabulek ve formulářích. Místo toho 4D zobrazí logo a záhlaví nabídek Záhlaví #1. S použitím programování můžete rozhodnout co dále. Protože není zobrazen žádný formulář, je toto okno nazýváno úvodní obrazovka. Do prostředí uživatele se můžete přepnout z prostředí aplikace třemi způsoby:

- Stiskněte CTRL (Jablko) + f Ujistěte, že je vypnut Caps Lock!)
- Stiskněte Alt + F4 na PC.
- Vyberte položku nabídky, kerá nemá přiřazenu žádnou metodu (jako původní položka Konec).

Poznámka: Tyto tři způsoby pracují pouze jste-li v úvodní obrazovce.


4th Dimension
Soubor Upravit Nápovida
Aplikace
v6
4th Dimension®
© ACI SA 1985 - 1998. All rights reserved.

2.3.1. Vytvoření záhlaví nabídky

1. Přejděte do prostředí Návrháře.
2. Vyberte Nástroje ¬ Editor nabídek...
3. V dialogu Záhlaví nabídek klepněte na tlačítko Přidat.

2.3.2. Vyzkoušení prostředí aplikace

1. Vyberte Prostředí ¬ Uživatele.(_ + U) (Ctrl + U)
2. Vyberte Prostředí ¬ Aplikace (_ + I) (Ctrl + I).

2.3.3. Opuštění prostředí aplikace

1. Stiskněte CTRL + f (Alt + F4) nebo vyberte Soubor ¬ Konec (_ + Q) (Ctrl + Q).

2.4. Užití první tabulky pro dialogy a další různé formuláře

V tomto dalším cvičení vytvoříte nový formulář. Každý formulář musí náležet do určité tabulky. Nezáleží na tom do které tabulky bude náležet tento formulář, protože dialogy, okno „O aplikaci..“, úvodní obratovka atd. nejsou používána k upravám záznamů tabulek. Uživatel do těchto formulářů nemůže vkládat data náležející sloupcům tabulek, proto v těchto formulářích nejsou záznamy ani zobrazovány. Formulář diologu můžete tedy vložit do libovolné tabulky.

Protože si nemusíte pamatovat ke které tabulce jste dialog a další formuláře vytvořili je zvykem vytvořit si ve struktuře tabulku složící pouze pro ukládání dialogových formulářů. Tato specielní tabulka neslouží k ukládání žádných záznamů. Neměli by jste do ní vkládat žádné záznamy ani jestliže je později plánujete vymazat.

Když je 4D zaváděna do paměti první z úloh, která je prováděna je zavedení tabulek adres první tabulky databáze. Jestliže je touto první tabulkou databáze, tabulka s velkým množsvím záznamů, zabere otevření databáze podstatně delší čas. Jestliže je první tabulka databáze prázdná a neobsahuje ani neobsahovala nikdy žádné záznamy spuštění aplikace se podstaltně zkrátí. Rozdíl mezi první prázdnou tabulkou a tabulkou obsahující jeden záznam je 32K paměti a odpovídající čas.

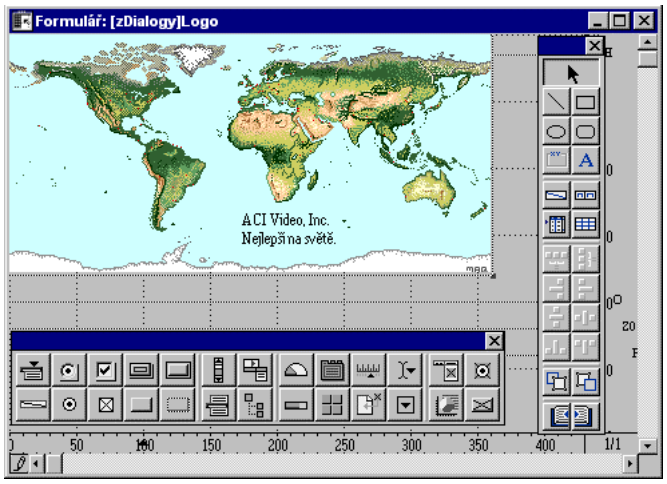
V naší databázi používáme jako první tabulku tabulku [zDialogy]. Tuto tabulku budeme používat nejen k vylepšení chování databáze, ale i jako tabulku pro uložení dialogových formulářů, jejichž zobrazovaný obsah nezáleží na tabulce. Konvence „z“ v počátku názvu je volena pro tabulky, keré nejsou uživateli dostupné.

2.5. Umístění vlastního loga do úvodní obrazovky

Můžete nahradit 4D logo z úvodní obrazovky vlastní grafikou. K zobrazení vlastní grafiky přejděte do prostředí návrháře a zobrazte editor nabídek. V editoru nabídek Vyberte Nabídka ¬ Zobrazit nabídky uživatele. 4D zobrazí grafiku úvodní obrazovky, nespleťte si ji se samotnou úvodní obrazovkou. Nejste v prostředí aplikace. V tomto módu můžete provést dfva úkony:

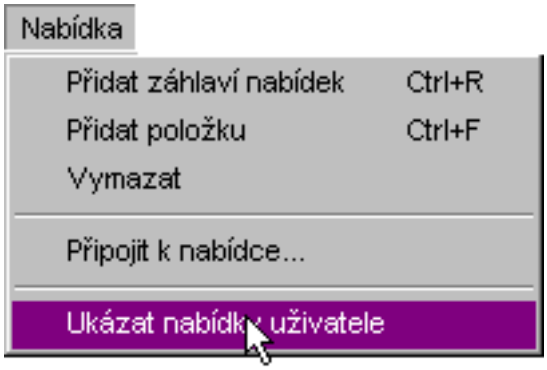
- Testovat vzhled vašeho záhlaví nabídek.
- Změnit grafiku v úvodní obratzovce.

K vytvoření vaší vlastní grafiky úvodní obrazovky můžete vložit obrázek z libovolného grafického programu s použitím staardního PICT formátu, nebo můžete použít Editor formulářů 4D, jak je ukázáno na obrázku níže.



Aby jste zobrazili své logo na úvodní obrazovku, musíte svou grafiku přiřadit k záhlaví nabídek. V tomto cvičení používáte Záhlaví #1 v Editoru nabídek a volbu Nabídka ¬ Ukázat nabídky uživatele

To že nejste v skutečné úvodní obrazovce dokumentuje poznámka v spodní části obrazovky. Přiřazení vaší grafiky provedete výběrem z nabídky Upravit ¬ Vložit, když jste předtím provedli kopírování obrázku do schránky v jiné aplikaci, nobo editoru formulářů 4D.



K opuštění módu Ukázat nabídky uživatele klepněte myší kamkoliv v obrazovce. Po vložení vaší grafiky, bude tato grafika přiřazena patřičnému záhlaví nabídek a bude s ním zobrazována, kdekoliv se toto záhlaví objeví. Jestliže chcete obrázek opět ze záhlaví nabídek vyjmout zvolte v módu Ukázat nabídky uživatele položku Upravit ¬ Vyjmout. V dalším cvičení si rychle vytvoříme logo a použijeme jej jako úvodní obrazovku.

2.5.1. Změna obrázku úvodní obrazovky

1. Vytvořte návrhářem formulářů v tabulce [zDialogy] formulář pojmenovaný Logo.
2. V novém formuláři nakreslete své logo.
3. Vyberte Upravit ¬ Vybrat vše (_ + A) (Ctrl + A).
4. Vyberte Upravit ¬ Kopírovat (_ + C) (Ctrl + C).
5. Vyberte Nástroje ¬ Editor nabídek….
6. Veberte Nabídka ¬ Ukázat nabídky uživatele.
7. Vyberte Upravit ¬ Vložit (_ + V) (Ctrl + V).
8. Klepněte myší kamkoliv v okně.
9. Přejděte do prostředí aplikace a zkontrolujte svou novou úvodní obrazovku.

2.6. Emulace vhodných částí prostředí uživatele

Cíl našich nabídek bude emulovat vhodné části prostředí uživatele a zamezit přístupu k nechtěným rysům, které mohou být pro naši aplikaci při přímém přístupu uživatelem riskantní.

3. Co by mělo být v našem systému nabídek?

Je několik věcí, které si musíme při navrhování nabídek pamatovat.

Vyvíjíte-li pro Macintosh musíte brát v úvahu doporučení publikace The Apple Macintosh Human Interface Guidelines (Addison-Wesley).

Vyvíjíte-li pro Windows musíte brát v úvahu doporučení publikace The Windows Interface: An Application Design Guide (Microsoft Stisknout) ("TWI"), kerá obsahuje jak navrhovat aplikace pro operační systém Windows 95.

Ve všech hlavních rysech se tato doporučení shodují a mají podobné cíle a směry. Cílem je vytvořit konzistentní a jednoduché rozraní pro uživatele.

Následující jsou výňatky z těchto doporučení týkající se nabídek.

> "Názvy nabídek by měli zůstat neměnné. Tato neměnnost dá uživateli pocit stability rozhraní a pomůže uživateli identifikovat aplikaci při přepnutí z jedné aplikace do druhé."
>
> "Logické slučování položek nabídek do skupin je nejdůležitější aspekt vytváření nabídek. Umístěte nejčastěji používané položky do vrchní části nabídky. Nejméně používané položky umístěte do spodní části nabídky. Spíše než jen jednoduché umisťování nejčastěji používaných položek na vrch nabídek, vytvářejte skupiny položek, které mají pro uživatele smysl."
>
> "Když je položka nabídky nedostupná (nic neprovádí), má být zobrazena šedými písmeny."

Můžete si povšimnout, že některé položky nabídek mají za názvem tři tečky. Existuje mnoho definic a vysvětlení co to znamená. Jednoduše lze říci, že to jsou položky, které vyžadují před provedením akce více informací. Pouhé zobrazení dialogu (jako O aplikaci) není důvodem pro vložení teček. Nepište přímo tři tečky, ale zadejte Option + ; na Mac nebo Alt+0133 (nikoliv Ctrl+z) na Windows. Poznámka: Napište s Num lock vypnutým a stisknutým Alt: 0133 na numerické klávesnici.

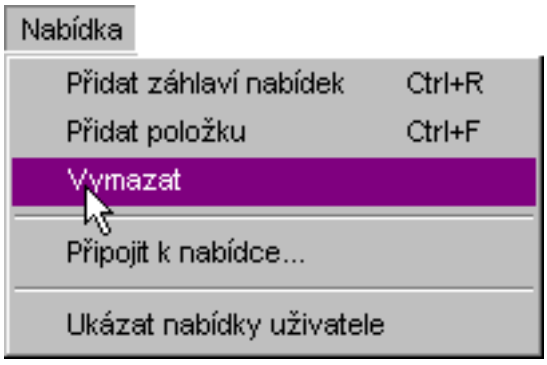
3.1. Přidání položky do nabídky Soubor a vytvoření nabídek zakládajících záznamy a zprávy

1. Přejděte do prostředí návrháře.
2. Přejděte do Editoru nabídek.
3. V oblast Platné záhlaví nabídek klepněte na trojúhelník před Soubor.
4. Klepněte na Konec.
5. Zvolte Nabídka ¬ Přidat položku nebo klepněte na tlačítko Přidat položku.
6. Zatrhněte políčko Čára .
   Tip: Místo odělení čárou můžete do položky napsat mezeru.

7. Zvolte Nabídka ¬ Add Item or click the Add Item button.
8. In the new menu item, type `Products`.
9. Zvolte Menu ¬ Přidat položku nebo klepněte na tlačítko Přidat položku.
10. Do nové položky napište `Faktury`.
11. Zvolte Menu ¬ Přidat položku nebo klepněte na tlačítko Přidat položku.
12. Do nové položky napište `Zákazníci`.
13. Uchopte a potáhněte položky a přesu%nte je tak do správného pořadí.
14. Poklračujte jako v předchozím a přidejte nabídky Záznamy a Zprávy podle odsouhlaseného schématu. (pokud procházíte pouze text seznam najdete níže)
15. Přejděte do prostředí aplikace a podívejte se na nabídky.

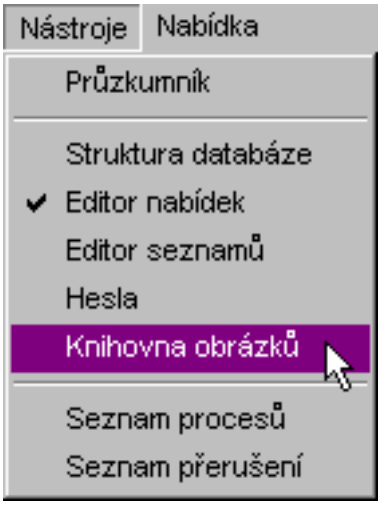
3.2. Upozornění na používání nabídky Nabídka

Buďte pozorní při vybírání položek nabídky. Nabídka obsahuje dvě podobné položky pracující s nabídkami a položkami. Přidat nabídku, přidává nabídky v sloupci 1 a Přidat položku, přidává položky vybrané nabídky v sloupci 2, obě na konec seznamu. Můžete je uchopit a potáhnout a změnit tak pořadí. Vymazat lze nabídku, nebo položku, když na ně klepnete a vyberete Vymazat z Nabídka.

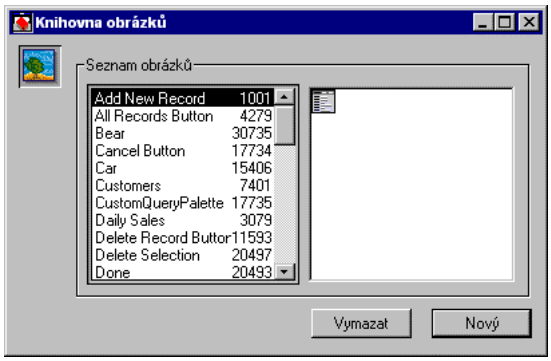


3.3. Přidání ikon k položkám nebídek

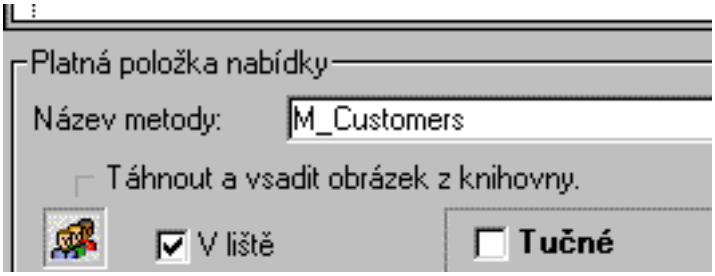
Uživatelé velice často dávají přednost tlačítkům před nabídkami. Proto jsou tak populární palety nástrojů. Ve verzi 6 4D můžete mít své vlastní palety nástrojů. Položky v paletě nástrojů musí odpovídat položkám nabídky, včetně obrázků. Aby byla položka zahrnuta do palety nástrojů musí být obrázek v Knihovně obrázků.



V této databázi jsme pro vás vytvořili obrázky pro použití s nejčastějšími položkami.



K zahrnutí obrázku ze seznamu do nabídky jednoduše obrázek, nebo jeho název, přetáhněte do patřičného políčka v Editoru nabídek.



Jestliže chcete vytvářet své vlastní obrázky do palet nástrojů, měli by mít velikost 18 bodů a šířku 19 bodů.

Poznámka: Nepřetahujte do palety zádný obrázek s nápisem, pro nabídky to jsou nevhodné obrázky.

3.4. Jak položky nabídek učinit akceschopnými

Když jste vytvořili záhlaví nabídek a přidali položky, 4D vaši nabídku již zobrazí. Jak však provést, aby prováděli správné akce, pokud je uživatel vybere ?

Provedete to tak, že napíšete metody projektu a přiřadíte je k nabídkám tak, že jména metod napíšete do oblasti Název metody v části Platná položka nabídky.

3.5. Co jsou metody projektu?

Metody projektu jsou části kódu, které nepatří výslovně k žádnému objektu nebo formuláři. Z tohoto důvodu musí být metody projektu pojmenovány již při jejich vytvoření. Toto jméno je používáno když chcete metodu spustit (Metoda je volána). Metoda projektu může být volána z nabídky, metody objektu, nebo virtuálně odkudkoliv.

V editoru nabídek můžete přiřadit metodu, tak že bude volána při výběru položky.

3.5.1. Přiřazení metod v Záhlaví #1

1. Vyberte Nástroje ¬ Editor nabídek….
2. Klepněte na Záhlaví #1.
3. Přidejte následující položky nabídek pokud jste tak neučinili dříve, ALE nepište (nepřiřazujte) zatím tyto metody položkám nabídek!
4. Přidejte z Knihovny obrázků obrázky ke keždé položce nabídky.

Soubor

Položka	Názvy "přiřazených" metod
Zákazníci…/1	M_Customers
Faktury…/2	M_Invoices
Produkty…/3	M_Products
-	
Konec /Q	M_GEN_Quit

Záznamy

Položka	Metoda
Ukázat vše /G	M_GEN_ShowAllRecords
Ukázat vybrané /H	M_GEN_ShowSubset
Vynechat vybrané	M_GEN_OmitSubset
-	
Přidat nový… /N	M_GEN_AddRecords
Vymazat vybrané…	M_GEN_DeleteUserSet
-	
Editor dotazů… /S	M_GEN_QueryEditor
Dotaz dle příkadu… /L	M_GEN_QueryByForm
Dotaz ve výběru…	M_GEN_QuerySelection
-	
Třídit…/T	M_GEN_OrderByEditor
-	
Importovat data…	M_IMPEXP_Import
Exportovat data…	M_IMPEXP_Export
-	
Hotovo /.	M_GEN_Done

Zprávy

Položka	Metoda
Rychlé…/R	M_GEN_QuickReport
Štítky…/J	M_GEN_Labels
Diagramy…/K	M_GEN_Chart
-	
Speciální zprávy …	M_SpecialReports

3.6. Zobrazení záznamů na obrazovku

V prostředí aplikace, uživatel vidí okno aplikace s úvodní obrazovkou a vaší vlastní grafikou. K zobrazení záznamů zákazníků v tomto okně jste přidali položku Zákazníci do nabídky Soubor. Tato položka nabídky bude spouštět metodu s příkazem MODIFY SELECTION k zobrazení seznamu záznamů ve výstupním formuláři/Seznamu.

3.7. Příkaz MODIFY SELECTION

Tento příkaz ukáže na obrazovku platný výběr záznamů tabulky v platném výstupním formuláři. K tomu, aby jste mohli vkládat záznamy ve vstupním formuláři musí uživatel poklepat na jeden řádek. Když uživatel skončí se zadáváním záznamu, po stisknutí tlačítka Přijmout se 4D, automaticky vrátí do výstupního formuláře.

Seznam záznamů zůstává na na obrazovce dokud uživatel neklepne na tlačítko Hotovo v zápatí výstupního formuláře. Příkaz MODIFY SELECTION je stále v běhu po celou dobu, kdy jsou záznamy zobrazovány na obrazovce. Tento příkaz je příkaz stálý a po svém provedení je aktivní dokud uživatel neklepne na tlačítko Hotovo. Uživatel může strávit hodiny uvnitř příkazu MODIFY SELECTION, prohlížením záznamů, prováděním změn nebo mazáním záznamů nebo přidáváním záznamů, předtím než klepne na tlačítko Hotovo.

Poznámka: Příkaz DISPLAY SELECTION je podobný příkazu MODIFY SELECTION. Oba příkazy ukazují platný výběr záznamu v platném výstupním formuláři. Rozdíl je v tom, že při DISPLAY není uživateli povoleno provádět změny v již existujících záznamech. Když uživatel poklepe na záznam a vstoupí do záznamu ve vstupním formuláři, zobrazený záznam je uzamčen. Uživatel si může záznam prohlížet, ale nemůže provádět žádné trvalé změny. Příkaz DISPLAY SELECTION je vhodný pro uživatele na úrovni Host, kterým nechcete povolit žádné změny.

3.8. Platný výběr

Příkaz MODIFY SELECTION zobrazuje na obrazovku platný výběr záznamů. 4D udržuje jeden výběr záznamů pro každou tabulku (platný výběr). Platný výběr může být buď všechny záznamy tabulky, žádný záznam z tabulky nebo několik záznamů z tabulky. Platný výběr pro tabulku je vždy definován pro každou tabulku v databázi, i když je tato tabulka třeba prázdná.

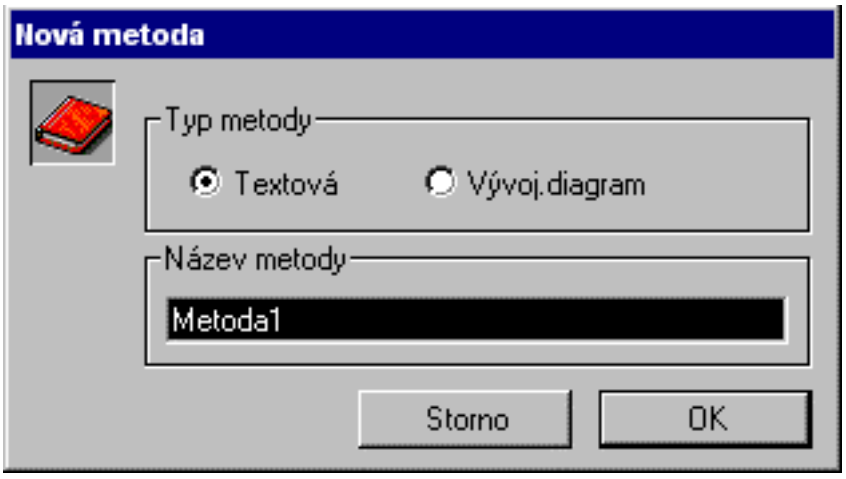
3.9. Příkaz ALL RECORDS

Příkaz ALL RECORDS umístí všechny záznamy tabulky do platného výběru této tabulky. Při provádění příkazu MODIFY SELECTION nemusíte explicitně vědět, které záznamy tabulky jsou obsaženy v platném výběru. Aby jste se ujistili, že v platném zobrazeném na obrazovce budou všechny záznamy tabulky, musíte před provedením příkazu MODIFY SELECTION, provést příkaz ALL RECORDS.

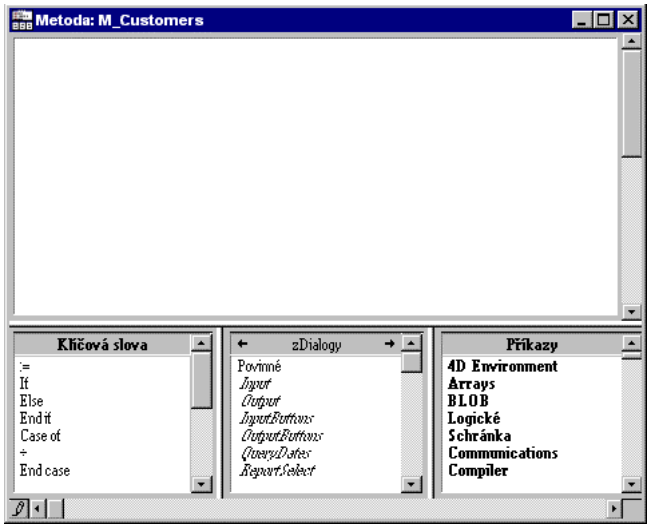
V následujícím cvičení vytvoříte metodu pro zobrazení záznamu v okně a pak přidáte tuto metodu do nabídky Soubor, aby byla spuštěna při vybrání položky nabídky.

3.9.1. Vytvoření metody pro položku nabídky Soubor ¬ Zákazníci

1. Vyberte Návrh ¬ Nová metoda…(_ + M) (Ctrl + M).



2. Napište `M_Customers`.
3. Klepněte na tlačítko OK.



Textový Editor metod

3.10. Použití textového Editoru metod

Textový Editor metod je v podstatě textovým editorem. Můžete zde psát instrukce pro 4D, jeden řádek po druhém a klepnutím na tlačítko Return (Enter) vkládat nové řádky. Tento editor může být nazýván podle místa svého použití: Textový Editor metod, Editor metod, Editor metod objektů atd.

Tento textový editor má několik rysů, které vám zjednodušší a zpřehlední programování. Oddělovací čára ve spodní části svislého posuvníku se nazývá dělič. Potažení děliče myší uschovává nebo zobrazuje tři sloupce ve spodní části okna. Tyto sloupce zobrazují nejčastěji používané položky. Klepnutím na některou položku v libovolném sloupci ji vkládáte do textového editoru. Jestliže nejste dobře obeznámeni s programovacím jazykem 4D a nebo nejste dobrým písařem, ušetří vám tyto sloupce mnoho času a zvýší přesnost vašeho programování.

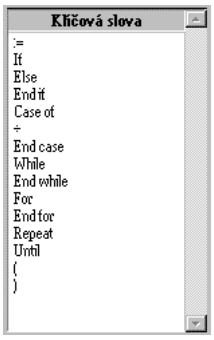
3.11. Klávesové zkratky

K rychlému otevření metody objektu ve formuláři zobrazeném Editorem formulářů existuje klávesová zkratka Option + Click (Alt + Click). Pokud se stisknutou klávesou Option (Alt) klepnete myší na objekt, otevře se automaticky metoda objektu a textový Editor metod.

V dalším cvičení napíšete metodu objektu pro pole [Zákazníci]Stát. Pamatujte si, že používání tří sloupců ve spodu okna Editoru metod vám usnadní programování a zabrání chybám v psaní.

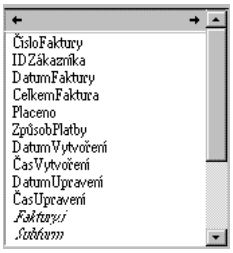
3.12. První sloupec – Klíčová slova

Sloupec klíčových slov je krátký seznam nejběžněji používaných operátorů. Tyto operátory jsou používány k řízení vašich metod a provádění rozhodování.



3.13. Druhý sloupec — Pole

Sloupec Pole zobrazuje seznam polí. Jako výchozí pokud jste v metodě objektu zobrazuje pole tabulky, do které patří formulář objektu. Všimněte si, že titul tohoto sloupce je prázdný. Tento sloupec rovněž obsahuje seznam formulářů navržených pro tuto tabulku (uvedeno kurzívou pod seznamem polí).



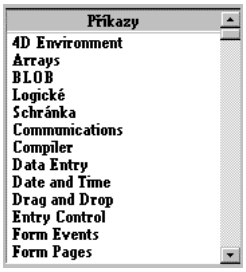
V tomto prostředním sloupci se můžete přepnout do dalších tabulek, klepnutím na šipky vlevo nebo vpravo v titulu sloupce. Nebo tím, že klepnete do titulu sloupce a podržíte myš (zobrazí se seznam tabulek ze kterého si můžete vybrat). Jestliže klepnete na pole ve sloupci, kde je v titulu uveden název tabulky, 4D vloží do Editoru metod úplnný název pole včetně názvu tabulky. Např [Zákazníci]Stát

Jestliže klepnete na pole pod prázdným titulem 4D, vloží do Editoru metod pouze název pole bez názvu tabulky. Např. Stát

Tip: Vždy se snažte o to, aby název pole byl zadáván úplnně tj. i s názvem tabulky. Tento způsob zpřehlední váš kód metody objektu a pokud budete jednou z metody objekty vytvářet metodu projektu, nemusíte již kód doplňovat o názvy tabulek.

3.14. Třetí sloupec — Příkazy

Sloupec příkazy je seznamem všech příkazů jazyka 4D. Tyto příkazy jsou sloučeny do skupin, ve kterých jsou uváděny v Příručce jazyka. Okno Editoru metod se vždy otevře se skupinami příkazů, které jsou zobrazeny tučně. Každá z těchto skupin má přiřazenu nabídku příkazů. Jeden konkrétní příkaz vyberete tak, že klepnete myší na skupinu příkazů, myš podržíte a vyberete příkaz ze zobrazené nabídky. Tento seznam je natolik dlouhý, že se nevejde do okna a musíte v něm listovat posuvníkem.



3.15. Abecední seznam

Klepnete-li na slovo Příkazy v hlavičce sloupce, změní se zobrazený seznam tak, že jsou zobrazeny přímo všechny příkazy a to v abecedním pořadí.

3.16. Rychlé listování

K rychlému nalezení příkazu v seznamu příkazů, aniž by jste museli listovat posuvníkem, klepněte na klávesy Comma + Shift (Ctrl + Shift) a počáteční písmeno názvu příkazu. Tímto způsobem přelistujete ve sloupci na první příkaz začínající tímto písmenem. Pokud se vám v rychlém sledu podaří zadat více písmen budete ještě blíže vašemu hledanému příkazu. Tento způsob nalezení příkazu může být použit ať je zobrazen abecední seznam příkazů nebo seznam skupin.

3.17. Vyhodnocení řádky kódu

Jestliže správně napíšete název příkazu jazyka 4D. Zobrazí jej po kontrole textový Editor metod tučně. Přinutit 4D ke kontrole řádku příkazu, můžete těmito způsoby:

- Klepněte na Enter (Enter numerické klávesnice).
- Klepněte na jinou řádku kódu.
- Klepněte na konec řádky a stiskněte klávesu Return (nebo Enter).

- Uzavřete metodu objektu a znovu ji otevřete.
- Klepněte na Comma + Enter (Ctrl + Enter) a vyhodnoťte tak všechny řádky.

Tomuto způsobu se také říká “tokenizing” řádku kódu.

3.18. Automatické uložení změn

4D automaticky uloží všechny změny ve formulářích a metodách. Je několik způsobů, jak toto automatické uložení všech změn v prostředí návrháře iniciovataut :

- Volbou Prostředí ¬ Uživatele (_ + U) (Ctrl + U) a přejitím tak do prostředí uživatele.
- Klepnutím do okna prostředí uživatele a přejitím tak do prostředí uživatele.
- Klepnete-li na uzavírací políčko v okně Editoru metod (uloží pouze obsah tohoto okna)
- Vyberete –li you choose Soubor ¬ Uložit xxxx (_ + S) (Ctrl + S). (Uloží pouze okno na popředí.)
- Vyberete-li Soubor ¬ Uzavřít xxx (_ + W) (Ctrl + W) . (). (Uloží pouze okno na popředí)
- Když ukončíte 4D.

Takže není nutné ukládat jedno okno po druhém pomocí xxx (_ + S) (Ctrl + S). 4D uloží všechny změny automaticky přejdete-li do prostředí uživatele (aplikace), nebo když uzavřete okna.

3.19. Terčíky v editoru metod indikují chybu

Jestliž uvidíte terčík (velkou tečku)(“•”) ve svém programu, znamená to, že 4D objevila chybu v názvu tabulky nebo pole. Používáním seznamu polí a zadáváním klenutím se vyhnete těmto chybám a budete mít jistotu, že pole a tabulky byly vloženy správně.

3.19.1. Oprava terčíků v Editoru metod

1. V metodě projektu M_Customers napište následující kód.

```
ALL RECORDS ([Zákazníci])
```

2. Upravte název [Zákazníci] tak, že bude uveden nesprávně.
3. Klepněte na další řádek nebo stiskněte Enter.
4. Všimněte si, že název tabulky je nyní ohraničen terčíky.
5. Vyberte celý nesprávný název a v seznamu tabulek klepněte na Zákazníci.
6. Klepněte na další řádek nebo stiskněte Enter.

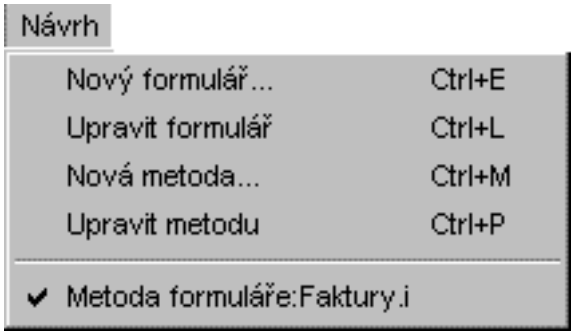
Povšimněte si, že terčíky samy zmizely a nemuseli jste je mazat.

3.20. Zkrácení zadávání

Zkrácení zadávání příkazu. Napište all r@ a stiskněte Enter. 4D prohledá svůj seznam příkazů a nahradí řetězec all r@ příkazem ALL RECORDS. Jestliže budete používat toto zkrácené zadávání ujistěte se, že při zadání řetězce již 4D nebude mít pochyby, který příkaz vybrat. Např. jestliže napíšete arr@, existuje několik různých příkazů, které začínají ARRAY jako ARRAY INTEGER, ARRAY STRING atd. V tomto případě 4D, vybere první příkaz v abecedním pořadí nalezených příkazů.

3.21. Rychlé přecházení mezi otevřenými okny

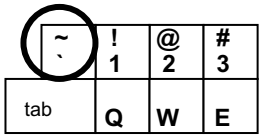
Jestliže neuzavřete okno metody (nebo formuláře), ve kterém pracujete a místo toho otevřete další okno další metody, 4D si pamatuje, která okna jsou stále otevřená a umístí je do seznamu otevřených oken v nabídce Návrh. Mezi těmito otevřenými okny lze pak přecházet pomocí této nabídky.



3.22. Psaní poznámek (komentářů)

Při programování budete chtít psát ke svému kódu poznámky, aby jste si kód zpřehlednili a mohli jej případně předat jiným programátorům. Tyto poznámky se nazývají komentáře.

Aby jste vytvořili komentář musíte v Editoru metod napsat znak (`) vypadá jako obrácený apostrof. Tento znak zadáte na klávesnici Windows Ctrl – Alt – ý a na klávesnici Mac vedle klávesy Return. Tento znak říká 4D, aby ignorovala libovolný text, který za tímto znakem následuje, takže se jej 4D nebude snažit interpretovat a nebude tento text chápat jako programovací příkazy. Můžete napsat cokoliv chcete do délky řetězce 80 znaků.



3.22.1. Přidání komentáře do metody projektu

1. Přejděte do Prostředí návrháře.
2. Otevřete metodu projektu M_Customers.
3. Přidejte komentáře podobné následujícím řádkům:

```
` Method: M_Customers
` Kurs ACI Univerzity
` Vytvořeno: Jim Steinman
` Datum: 15/1/97

` Účel: Zobrazí seznam záznamů [Zákazníci] ve výstupním formuláři.

ALL RECORDS ([Zákazníci])
`  Změna platného výběru.
MODIFY SELECTION Chyba! Nenalezen zdroj odkazů.([Zákazníci]; *)
`  Ukáže záznamy na obrazovku.
` Konec metody
```

3.23. Použití If (False) k přeskočení kódu

Komentáře jsou velmi důležité a pomohou vám rozeznat co se stane v této metodě s databází. Pokud však je databáze spuštěna v interpretačním módu, je každý řádek komentáře vyhodnocován, aby se určilo jestli tento řádek obsahuje nějaký kód ke spuštění. Toto vyhodnocování zpomaluje proces provádění a když metodu krokujete musíte projít přes všechny komentáře a zdržujete se. Proto je dobré umístit větší bloky samostatných komentářů uvnitř bloku If (False). Tyto řádky kódu (komentáře mezi If a End if) nebudou nikdy v interpretačním módu vyhodnocovány. Při kompilaci všechny řádky kódu obsažené mezi If (False) a End if jsou kompilátorem přeskočeny a ignorovány. Můžete použít Case of pak : (False) a obdržíte tentýž výsledek jako s If (False) a to účinněji pokud je již blok Case použit. Jediný rozdíl je, že kompilátor neignoruje tuto část kódu.

3.23.1. Zahrnutí bloku If (False).

1. V Editoru metod upravte metodu projektu M_Customers následujícím způsobem:

```
   If (False)
` Method: M_Customers
      ` Kurs ACI Univerzity
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Zobrazí seznam záznamů [Zákazníci] ve výstupním formuláři.
   <>f_Version6x10 := True
   <>fJ_Steinman :=  True
End if

ALL RECORDS ([Zákazníci])
`  Změna platného výběru.
MODIFY SELECTION([Zákazníci]; *)
`  Ukáže záznamy na obrazovku.
` Konec metody
```

2. Přejděte do Editoru nabídek, přesvědčte se, že název této metody je zadán k položce nabídky Zákazníci .
3. Přejděte do prostředí Vlastní nabídky a vyzkoušejte novou metodu vybraním položky Soubor ¬ Zákazníci.

3.24. Přinucení příkazu MODIFY SELECTION, aby vždy zobrazoval výstupní formulář

Jestliže platný výběr obsahuje pouze jeden záznam pak po provedení metody M_Customers, 4D použije vstupní formulář místo výstupního formuláře.

```
   ` 4D použije vstupní formulář při jednom záznamu v platném výběru.
MODIFY SELECTION ([Zákazníci])
   ` Hvězdička znamená, že 4D použije vždy výstupní formulář.
MODIFY SELECTION ([Zákazníci]; *)
```

Důvodem pro toto chování je, že pokud je zobrazován pouze jeden záznam co jiného může uživatel dělat, než poklepat na záznam a upravovat jej. Takže 4D v některých situacích šetří počet kroků, které uživatel musí provést, aby mohl záznam upravovat. Uživatelé však mohou být zmateni tím, že se aplikace nechová vždy stejně a někdy vidí záznamy ve výstupním formuláři a někdy vidí záznam ve vstupním formuláři.Tomuto se můžete vyhnout tím, že do příkazu MODIFY SELECTION přidáte středník a hvězdičku.

3.25. Doporučení pro názvy metod

Názvy metod mohou být dlouhé do 31 znaků. Můžete používat i mezery. V Editoru metod vám však názvy s mezerami neumožní vybrat celý název poklepáním myší na název metody. Z tohoto důvodu se většina programátorů ve 4D vyhýbá mezerám v názvech metod (a rovněž používání českých znaků). Totéž platí pro názvy polí, tabulek, proměnných a formulářů.

Místo toho většina programátorů používá velká počáteční písmena pro oddělená slova v názvech. Příklad: MojeMetoda.

Zkuste si vytvořit svoji konvenci názvů a vždy ji používejte. Např. v tomto příkladě jsme pojmenovali metodu použitou v nabídce (menu) tak, že začíná písmeny „M_“. Dodržení této konvence způsobí, že všechny metody nabídky budou v okně Průzkumníka uvedeny společně a za sebou.

V tomto kurzu budeme pomocí konvence názvů sdružovat dohromady metody podle způsobů použití. Na počátku názvu metody budeme vždy uvádět písmena, která nám pomohou určit do které skupiny metod tato metoda patří.

3.26. Nepoužívejte příkazy nebo konstanty jako názvy polí a metod.

Nikdy nenazývejte pole nebo metodu stejným názvem jako příkaz nebo konstantu programovacího jazyka 4D. Na konci Příručky jazyka naleznete přílohu se seznamem všech příkazů. Prostudujte si tento seznam a vyhněte se těmto slovům. Dále je uveden stručný výtah nejpoužívanějších příkazů:

- Abort
- Aborted
- Abs
- Accept
- Accumulate
- Activated
- After
- Alert
- April
- Arctan
- Ascii
- August
- Average
- Backspace
- Beep
- Before
- Black
- BLOB
- Blue
- Bold
- Boolean
- Brown
- Cancel
- Char
- Condensed
- Confirm
- Cos
- Date
- Deactivated
- Dec
- Dec
- December
- Delayed
- Dialog
- Difference
- During
- Enter
- Escape
- Execute
- Executing
- Exp
- Extended
- False
- February

- Field
- Font
- For
- Friday
- Gestalt
- Graph
- Green
- Gray
- Idle
- Int
- Integer
- Intersection
- Italic
- January
- July
- June
- Keystroke
- Length
- Level
- Locked
- Log
- Long
- Lowercase
- March
- Max
- May
- Message
- Milliseconds
- Min
- Mod
- Modified
- Monday
- Nil
- Not
- November
- Num
- October
- Old
- Orange
- Outline
- Paused
- Pentium
- Picture
- Plain

- Play
- Pointer
- Position
- Purple
- Query
- Raom
- Real
- Red
- Redraw
- Reject
- Report
- Request
- Round
- Saturday
- Self
- Semaphore
- September
- Shadow
- Short
- Sin
- String
- Subfile
- Substring
- Subtotal
- Sum
- Sunday
- Tab
- Table
- Tan
- Text
- Thursday
- Tickcount
- Time
- Trace
- True
- Trunc
- Tuesday
- Type
- Undefined
- Underline
- Union
- Uppercase
- Variance
- Wednesday

- White
- Windows
- Yello

3.27. Konvence názvů pro funkce

Vestavěné příkazy 4D jsou psány dvěma způsoby:

- Všechna písmena velká – pro příkazy, které provádějí nějakou akci jako např. CANCEL.

- Smíšeným způsobem – pro příkazy, které vracejí nějaké hodnoty, jako např. Uppercase.

Příkazy psané smíšeným způsobem se vždy uvádějí na pravé straně operátoru přiřazení (:=).

V programátorském žargonu se metoda, která navrací hodnotu nazývá funkce (smíšený způsob psaní ve 4D).

Jedna ze škol názvů doporučuje dodržovat tytéž konvence pro názvy vlastních metod. Použití všech písmen velkých však znepřehledňuje kód, který se také mnohem hůře čte. Většina programátorů proto pojmenovává své metody tak, že velká písmena jsou uvedena pouze pro počáteční písmena oddělených slov v názvu. Metody, které mají speciální funkce jsou pak často uvedeny zvláštními znaky podle vytvořené konvence názvu. Nejčastěji upravované metody jako např. metody s konstantami by měly začínat písmeny „aa“ takže budou uvedeny vždy na vrchu seznamu metod. Dále je uvedena jedna z konvencí názvů:

- “M_” pro metody nabídek
- “P_” pro metody, které spouštějí nový proces
- “E_” pro metody, které jsou navrženy k použití s příkazem Execute
- “WIN_” pro metody ovládající okna
- “GEN_” pro generické metody spouštěné odkudkoliv
- “IMPEXP” pro metody, které se zabývají importem a exportem

Příklady:
- M_Customers
- P_Customers
- aaConstants
- GEN_MojeMetoda

Když píšete vlastní metodu, která je funkcí je běžnou praxí začínat název funkce písmenem, které určuje typ hodnoty, kterou tato funkce vrací. Příklad je uveden v následujícím seznamu:

Předpona	Typ dat	Příklad

a	Array	aCategory
b	Button/Tlačítko	bOK
d	Datum	dDueDate
f	Logické (Flag)/	fJ_Steinman
g	Grafika	gOKPicture
h	Hodiny	hCurrentTime
i	čítač smyčky (Interakce)	$i
j	čítač smyčky (Interakce)	$j
k	Konstanta	k_Accept
L	Long Integer	LACIProfits
o	BLOB (Objekt)	oClipboard
p	Pointer/Ukazatel	pTable
r	Real	rPrice
s	Alfa (String)/Řetězec	sFirstName
t	Text	tComments

Kvíz

- Proč, když vyberete Soubor ¬ Konec, přejdete z prostředí Vlastní nabídky do Prostředí uživatele?

3.28. Jednoduchý způsob sledování verzí a autorů

Často budete chtít vědět, ve které verzi vašeho programu byla daná metoda vytvořena. Jinou častou otázkou je, kdo tuto metodu vytvořil. Pro tyto účely můžete použít jednoduchý seznam proměnných. Zde používáme:

```
<>f_Version6x10
```

Pro metody vytvořené v tomto kurzu. Kromě toho můžete do metody uložit své jméno tak jak je ukázáno níže. Vaši práci bude určovat proměnná, která se skládá z iniciály vašeho jména a příjmení.

```
<>fJ_Steinman :=  True
```

Tyto proměnné by měly být umístěny uvnitř bloku If(False). Na tomto místě 4D o těchto proměnných neví, nebudou nikdy prováděny a nebudou zpomalovat databázi a rovněž nebudou nikdy kompilovány. Důvodem pro použití proměnných je, že tyto proměnné mohou být velmi jednoduše sledovány pomocí nástroje vývojáře 4D Insider.

3.28.1. Použití jednoduchého způsobu sledování systému.

1.V Editoru metodu upravte metodu projeku M_Customers následujícím způsobem:

```
If (False)
` Method: M_Customers
```

```
      ` Kurs ACI Univerzity
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Zobrazí seznam záznamů [Zákazníci] ve výstupním formuláři.
   <>f_Version6x10 := True
   <>fJ_Steinman :=  True
End if

ALL RECORDS ([Zákazníci])
`  Změna platného výběru.
MODIFY SELECTION([Zákazníci]; *)
`  Ukáže záznamy na obrazovku.
` Konec metody
```

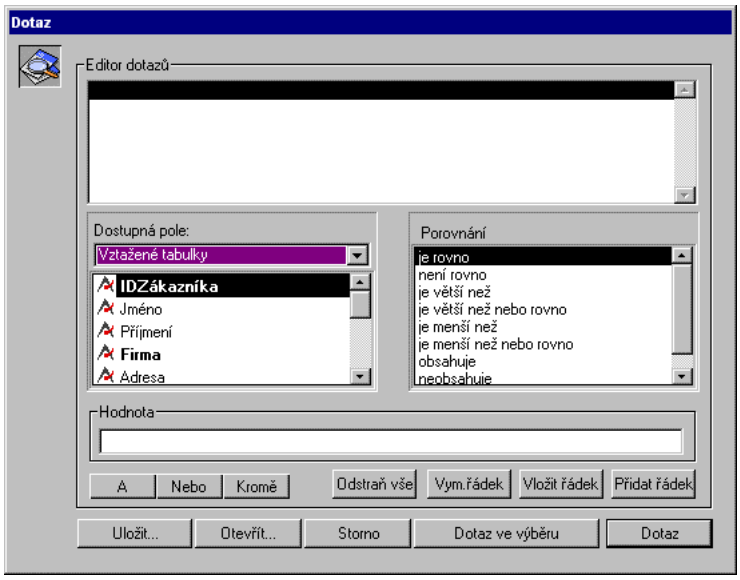
4. Přidání rysů pro práci se záznamy

4.1. Práce se záznamy v prostředí Vlastní nabídky

Vytvořili jste položku nabídky, která ukazuje zánamy tabulky [Zákazníci] na obrazovku v prostředí Vlastní nabídky. Nyní potřebujeme přidat takové rysy, které nám umožní stěmito záznamy pracovat. Uživatel by měl být schopen přidávat, mazat, hledat, třídit a tisknout záznamy. V tomto oddíle přidáme do vaší databáze tyto rysy a to pomocí nabídek a metod projektů.

4.2. Přidání metody pro hledání, dotazování

Přidejme metodu, která nám umožní vyhledávat (dotazovat se na) určité záznamy. Využijeme k tomu jeden z vestavěných nástrojů v Prostředí uživatele a to Editor dotazů:



4.21. Vytvoření metody pro položku nabídky Záznamy ¬ Editor dotazů…

1. Vytvořte novou metodu projektu nazvanou M_GEN_QueryEditor.
2. Do textu metody napište následující:

```
If (False)
    ` Metoda: M_GEN_QueryEditor
    ` Kurz ACI University
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97
    ` Účel: Zobrazit Editor dotazů pro  [Zákazníci].

  <>f_Verze6x10 := True
  <>fJ_Steinman := True
End if

QUERY ([Zákazníci])

  ` Konec metody
```

3. Přejděte do Editoru nabídek a přesvědčte se, že název této metody je zadán v položce Editor dotazů.
4. Přejděte do prostředí Vlastní nabídky a vyzkoušejte tuto metodu z úvodní obrazovky.

4.3. Řízení vašeho systému záhlaví nabídek

Všimněte si, že když provedete dotaz, nejsou záznamy, které byly vyhledány zobrazeny. To je proto, že nebyl proveden příkaz DISPLAY SELECTION nebo MODIFY SELECTION. Můžeme říci, že v tomto místě aplikace by uživatel neměl být schopen tuto položku nabídky vybrat, pokud není jasné, v které tabulce konkrétně chce vyhledávat a co chce dále se záznamy provádět. Je zřejmé, že musíme do aplikace zavést určitý řád a řízení zobrazování a použití jednotlivých položek nabídek. Nejjednodušší způsob jak to provést bude vytvořit druhé záhlaví nabídek pro práci se záznamy uvnitř oken zobrazujících záznamy jednotlivých tabulek a první záhlaví nabídek ponecháme pouze pro používání s úvodní obrazovkou.

4.4. Případ nefunkčního záhlaví nabídek

Máme ještě druhý, závažný problém. Při zobrazení záznamů z tabulky Zákazníci pomocí položky Soubor ¬ Zákazníci, nám nově vytvořená metoda s Editorem dotazů zatím nepracuje. 4D zpřístupňuje záhlaví nabídek na základě jednotlivých formulářů. Jestliže chceme, aby záhlaví nabídek pracovalo správně v konkrétním formuláři musíme 4D říci, aby toto záhlaví nabídek danému formuláři přiřadila. Poznamenejme, že úvodní obrazovka není formulář, takže 4D automaticky podporuje celé zobrazené Záhlaví #1.

4.4.1. Vyzkoušení položky nabídky Záznamy ¬ Editor dotazů… v zákaznících

1. Přejděte do prostředí Vlastní nabídky

2. Vyberte položku nabídky Soubor ¬ Zákazníci.
3. Vyzkoušejte položku nabídky Záznamy ¬ Editor dotazů...

Vidíme, že Editor dotazů zatím nefunguje.

5. Správné použití záhlaví nabídek

Předtím než vytvoříme plnně funkční a uživateli přívětivý systém nabídek, musíme si říci několik věcí.

Již jste vytvořili Záhlaví #1 4D automaticky zobrazuje jako druhou nabídku, nabídku Upravit. Pořadí nabídek Soubor a Upravit je řízeno automaticky 4D.

5.1. Rozdělené použití nabídek

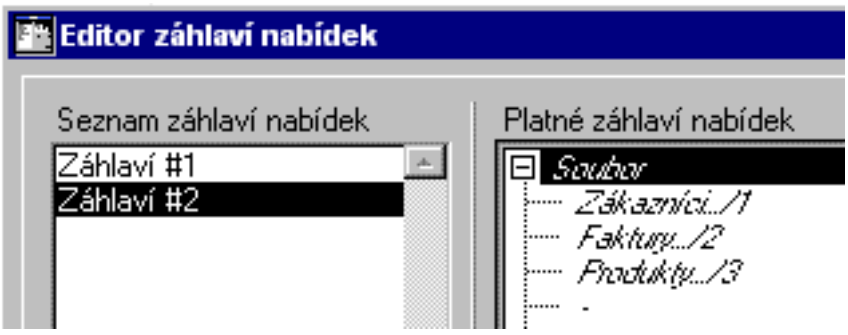
Již máte vytvořeno Záhlaví #1. V úvodíní obrazovce jsou všechny nabídky zobrazeny. Některé položky však nepotřebujeme, aby zde pracovali a pokud bude toto záhlaví funkční ve výstupních formátech opět jiné nebudeme chtít mít činné zde.

Jak tedy rozdělit činnost? Během tohoto kurzu použijeme dvě různé metody. První jednodušší je použita v následující části.

Začneme použitím a vytvořením Záhlaví #3 (nestarejme se zatím o Záhlaví #2), které bude vhodné pro výstupní formuláře. Pokud uživatel zvolí Soubor ¬ Zákazníci z úvodní obrazovky, přinutíme 4D přepnout na Záhlaví #3 a použijeme tak správné nabídky.

5.2. Vždypřítomná nabídka Soubor

Až vytvoříte Záhlaví #2 uvidíte, že obsahuje tutéž nabídku Soubor jako Záhlaví #1. 4D vždy přidává automaticky do každého nového záhlaví nabídek tuto nabídku.



5.2.1. Vytvoření Záhlaví #2

1. Vyberte Nástroje ¬ Editor nabídek….
2. Klepněte na tlačítko Přidat v levém spodním rohu.

5.2.2. Vytvoření Záhlaví #3

1. Klepněte na tlačítko Přidat v levém spodním rohu

5.2.3. Přidání nabídky Záznamy

1. V Záhlaví #3 s vybranou nabídkou Soubor, klepněte na tlačítko Přidat nabídku v pravé spodní části.
2. Pojmenujte novou nabídku záznamy.
3. Klepněte na tlačítko Přidat položku
4. Napište název položky Ukázat vše/G
5. Přidejte správný obrázek z knihovny obrázků.
6. Opakujte kroky 3-5 dokud nezadáte následující:

```
Vybrat vše/G
Vybrat ozna•ené/H
Vynechat ozna•ené
-
P•idat nový…/N
Vymazat vybrané…
-
Editor dotaz•…/S
Dotaz dle p•íkladu…/L
Dotaz ve výb•ru…
-
T•ídit…/T
-
Importovat data…
Exportovat data…
-
Hotovo/.
```

7. Vložte metodu `M_GEN_QueryEditor` do položky nabídky Editor dotazů…/S.

5.2.4. Přidání nabídky Zprávy

1. V Záhlaví #3 s vybranou nabídkou Záznamy, klepněte na tlačítko Přidat nabídku.
2. Pojmenujte novou nabídku Zprávy.
3. Klepněte na tlačítko Přidat položku
4. Pojmenujte položku Rychlé.../R
5. Z knihovny obrázků přidejte odpovídající obrázek.
6. Opakujte kroky 3-5 dokud nezadáte následující:

```
Zprávy…/R
Štítky…/L
Diagramy…/K
-
```

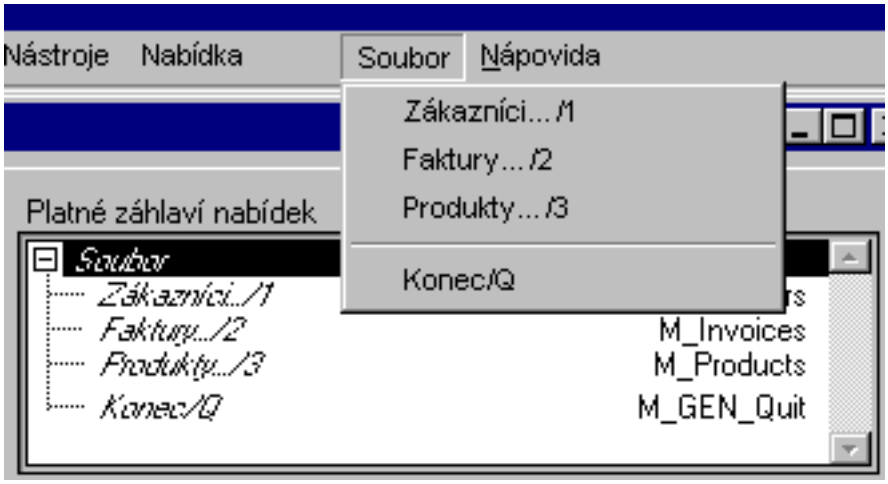
```
Speciální zprávy…
```

5.3. Propojené nabídky

Chceme, aby nabídka Soubor byla tatáž v Záhlaví #1 i Záhlaví #3, abychom dosáhli konzistentnosti nabídek. Můžete to provést pomocí kopírovat a vložit z jednoho záhlaví do druhého, ale naštěstí 4D obsahuje silný rys pro sdílení stejných nabídek mezi jednotlivými záhlavími: Propojení nabídek. Propojené nabídky jsou jednoduše řečeno jedna nabídka, která se objeví ve více než jednom záhlaví. Jestliže provedete změny v nabídce Soubor v Záhlaví #3 tak se tato nová verze nabídky Soubor objeví automaticky i v Záhlaví #1. Propojené nabídky nejsou kopírované a vložené do jiných nabídek, jsou totožné a mohou se objevit ve více záhlavích současně. Jako výchozí rys 4D, propojuje nabídku Soubor ze Záhlaví #1 do všech nově vytvořených záhlaví.

5.4. Náhled vašich nabídek

Ještě v Prostředí návrháře můžete vidět jak budou vaše nabídky vypadat graficky. Podívejte se na pravou stranu záhlaví nabídek ve vrchní části obrazovky a uvidíte, že nabídka Zprávy je zde dostupná pro vaši kontrolu.



5.5. Zaktivnění položky nabídky Záznamy ¬ Editor dotazů…

Při zobrazení výstupního formuláře by se vaše záhlaví nabídek mělo skládat z nabídek Soubor, Upravit, Záznamy a Zprávy. Zaktivnění položky nabídky spočívá v přiřazení správného záhlaví formuláři a přiřazení správné metody projektu položce nabídky.

5.6. Znepřístupnění položek nabídek v úvodní obrazovce

Nechceme, aby v úvodní obrazovce měl uživatel přístup k položkám nabídek z nabídek Záznamy a Zprávy. Přístup k těmto položkám zakážeme jednu po druhé v Editoru nabídek v Záhlaví #1.

5.6.1 Úpravy Záhlaví #1 k znepřístupnění položek

1. Vybírejte v Záhlaví #1 v nabídkách Záznamy a Zprávy jednu položku po druhé a odškrtněte políčko Viditelná v oblasti Platná položka nabídky.

5.6.2. Úpravy Záhlaví #3 k zpřístupnění položek

1. Vybírejte v Záhlaví #3 v nabídkách Záznamy a Zprávy jednu položku po druhé a zaškrtněte políčko Viditelná v oblasti Platná položka nabídky.

5.7. Přiřazení nabídek k formuláři

Pokud nyní zvolíte v úvodní obrazovce Soubor ¬ Zákazníci, uvidíte, že je stále zobrazováno Záhlaví #1. Je to proto, že jsem 4D neinstruovali ke změně platného záhlaví nabídek a přiřazení záhlaví formuláři. V dalším cvičení přiřadíme Záhlaví #3 výstupnímu formuláři tabulky Zákazníci.

5.7.1. Přiřazení záhlaví formuláři

1. Otevřete formulář [Zákazníci];"Výstupní2".
2. Vyberte Formulář ¬ Přiřadit záhlaví nabídek….
3. Do zadávací oblasti dialogového okna napište 3.
4. Klepněte na tlačítko OK v dialogovém okně.
5. Přejděte do prostředí Vlastní nabídky a vyzkoušejte toto přiřazení v agendě Zákazníci.

5.8. Přiřazená nabídka připojená k existujícím nabídkám

Když jste vstoupili do agendy Zákazníci, změnilo se záhlaví nabídek proti předchozím vstupům. Jsme nyní asi trochu zmateni, protože jsou zobrazeny dvě kopie téměř každé nabídky. Jediná nabídka, která nebyla zdvojená je nabídka Soubor (pamatujeme si, že to je propojená nabídka). Stalo se tak proto, že libovolné číslo Záhlaví vložené v dialogovém okně formuláře, připojí toto záhlaví k existujícímu systému nabídek. My samozřejmě v tomto případě, kdy máme záhlaví nabídek takřka identické, budeme chtít v některých místech a to ve výstupních formulářích, zobrazit pouze Záhlaví #3 a nikoliv Záhlaví #1. Všimli jsme si rovněž, že položky nabídek Záhlaví #1 nepracují a položky nabídek Záhlaví #3 jsou činné. Pokud bychom chtěli, aby obě záhlaví, která zde připojíme pracovala současně, musíme v dialogovém okně přiřadit záhlaví nabídek, zadat přiřazované záhlaví se znakem „-„ (v našem případě –3). Toto mínus instruuje 4D, aby připojila tuto nabídku již existující nabídce a zpřístupnila i položky předchozího záhlaví

nabídek. V našem případě to však nepřichází do úvahy, protože máme dvě duplikovaná záhlaví nabídek.

Abychom mohli změnit záhlaví nabídek úplnně, musíme říci metodě projektu, aby 4D kompletně vyměnila záhlaví nabídek a používala Záhlaví #3 místo Záhlaví #1.

5.9. Úpravy záhlaví nabídek v průběhu jeho používání

Jestliže přejdete do Prostředí návrháře zatímco v prostředí Vlastní nabídky je zobrazeno v agendě Zákazníci Záhlaví #3, je toto záhlaví dále používáno. Pokud provedete úpravy v Záhlaví #3 a přepnete se zpět do prostředí Vlastní nabídky, klepnutím do otevřeného okna agendy Zákazníci, změny v záhlaví neuvidíte. Je to proto, že si 4D při prvním použití tohoto záhlaví načte záhlaví do paměti a pokud chceme vidět provedené změny, musíme ji přinutit k obnovení a novému načtení tohoto záhlaví. V tomto případě je jediným způsobem opustit okno agendy Zákazníci, klepnutím na tlačítko Hotovo a znovu do ní vstoupit vybráním Soubor ¬ Zákazníci a novým načtením výstupního formuláře s novým načtením Záhlaví #3.

5.10. Zavedení způsobu výměny záhlaví

Způsob úplnné výměny záhlaví nabídek je používán mnoha programátory, při ovládání nabídek. Tato strategie je založena na jednoduché myšlence, kompletním přepínáním mezi podobně vypadajícími záhlavími nabídek. Je to poměrně jednoduchý způsob na programování a používá pouze jeden příkaz jazyka k výměně záhlaví, tento příkaz je MENU BAR.

Když uživatel vybere Soubor ¬ Zákazníci z úvodní obrazovky, budete muset 4D říci, aby přepnula kompletně do Záhlaví #3, které jsme vytvořili pro používání ve výstupních formulářích.

5.11. Výměna záhlaví při MODIFY SELECTION

Nyní, když jsme dokončili naše záhlaví nabídek, potřebujeme ještě, aby je 4D zobrazila ve správný čas na správném místě. Potřebujeme tedy přepnout do Záhlaví #3 před zobrazením záznamů ve výstupním formuláři a navrátit se zpět k Záhlaví #1 po opuštění agendy před zobrazením úvodní obrazovky. Záznamy ve výstupním formuláři se zobrazují jako výsledek použití příkazu MODIFY SELECTION po volbě Soubor ¬ Zákazníci a zpět se nevracíte ukončením tohoto příkazu klepnutím na tlačítko Hotovo ve výstupním formuláři. Proto přepnutí záhlaví nabídek pomocí příkazu MENU BAR musí být těsně před a těsně za příkazem MODIFY SELECTION.

5.11.1. Úprava metody projektu M_Customers k přepnutí záhlaví nabídek

1. Otevřete metodu projektu M_Customers.
2. Upravte metodu následovně:

```
If (False)
` Method: M_Customers
      ` Kurs ACI Univerzity
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Zobrazí seznam záznamů [Zákazníci] ve výstupním formuláři.
   <>f_Version6x10 := True
   <>fJ_Steinman :=  True
End if

ALL RECORDS ([Zákazníci])
`  Změna platného výběru.
MENU BAR (3) `přepne do záhlaví 3
MODIFY SELECTION([Zákazníci]; *)
`  Ukáže záznamy na obrazovku.
MENU BAR (1) `přepne do záhlaví 1

` Konec metody
```

3. Přejděte do prostředí Vlastní nabídky a zvolte Soubor ¬ Zákazníci a podívejte se jak nabídka vypadá .
4. Otevřete formulář [Zákazníci];"Výstupní2".
5. Vyberte Formulář ¬ Přiřadit záhlaví nabídek.
6. V dialogovém okně napište 0.
7. V dialogovém okně klepněte na tlačítko OK.
8. Přejděte do prostředí Vlastní nabídky a vyzkoušejte.

5.12. Zaktivnění záhlaví ve zobrazeném formuláři

Překvapení! Položka nabídky Záznamy ¬ Editor dotazů... stále nepracuje. Je to tomu tak, protože při zobrazení formuláře na obrazovku zobrazené záhlaví nabídek nepracuje automaticky.

Řešení tohoto problému je např. ve využití výhod připojených nabídek poněkud kuriózním způsobem.

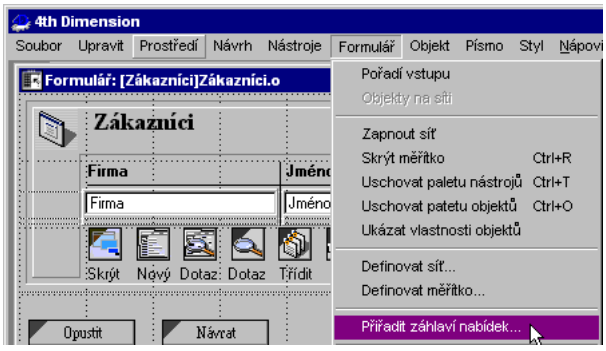
5.13. Zaktivnění nabídek ve verzi 3 (do verze 3.5)

Co musí provést přiřazené záhlaví nabídek, aby již zobrazené záhlaví nabídek pracovalo? Pokud pouze přiřadíme záhlaví nabídek ve formuláři, 4D nezaktivní již zobrazené záhlaví nabídek, pouze přiřadí přiřazované záhlaví nabídek a pouze toto záhlaví nabídek bude aktivní. Ale my nechceme mít zobrazeno ještě další záhlaví nabídek. Řešením tedy ve starších verzích bylo přiřadit nějaké neexistující záhlaví nabídek. Např. záhlaví 99, protože počet záhlaví jednoho dne může přesáhnout 99, bylo nejčastěji voleno nějaké větší číslo, protože může existovat až 32 000 záhlaví nabídek, většina programátorů jednoduše přiřazovala 32 000.

5.14. Zaktivnění nabídek ve verzi 6 (3.5.1. a vyšší)

Protože předchozí metoda byla v minulosti tradičně používaná je to rovněž druhý způsob, který může být použit. V této verzi 4D byly nabídky upgradovány a 32 000 nabídek již nepracuje (toto číslo funguje ve 4D pouze jako trik). Další metoda může být použita v obou verzích:

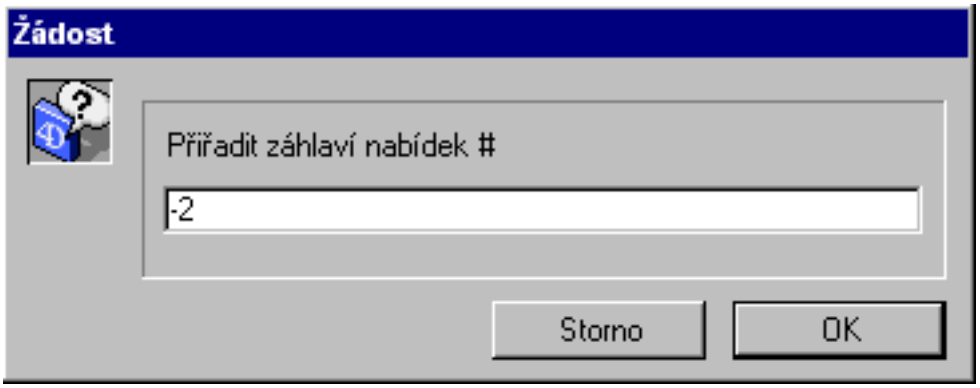
Tato metoda je rovněž přiřazení jiného záhlaví nabídek, ale místo neexistujícího záhlaví se přiřazuje prázdné záhlaví (připomeňme si námi vytvoření Záhlaví #2). Ačkoliv Záhlaví #2 není ve skutečnosti prázdné je jeho nabídka Soubor připojenou nabídkou, stejnou ve všech dalších záhlavích nabídek, takže 4D toto záhlaví považuje za prázdné.



5.15. Záporné hodnoty záhlaví nabídek

Potřebujete přiřadit záhlaví nabídek se zápornou hodnotou. Jak již bylo uvedeno, znaménko minus instruuje 4D, aby zpřístupnila předchozí zobrazené záhlaví nabídek, takže použití znaménka minus nám zaktivní námi změněné záhlaví nabídek pomocí přikazu Menu Bar. Záhlaví

#2 bude fiktivně přiřazeno na konec Záhlaví #3. Tento způsob je jeden ze způsobů jak zaktivnit záhlaví nabídek a vyvarovat se většiny možných chyb při dalším vývoji systému nabídek.



K zaktivnění zobrazeného záhlaví s formulářem zobrazeným na obrazovku:

- Přiřaďte Záhlaví#2 (-2) k formuláři.

5.15.1. Přiřazení záporné hodnoty záhlaví k výstupnímu formuláři

1. Otevřete formulář [Zákazníci];"Výstupní2".
2. Otevřete Formulář ¬ Přiřadit záhlaví nabídek.
3. V dialogovém okně napište -2.

6. Přidání více rysů nabídkám

Nyní když základní páteř systému vašich nabídek již funguje správně, je čas přidat nabídkám více rysů. V této části přidáme nové položky/ metody projektu, které vám umožní zobrazovat všechny záznamy a třídit záznamy. Rovněž si zobrazíme počet záznamů v platném výběru, velice podobně jak to dělá Prostředí uživatele..

6.1. Přidání položky nabídky Zobrazení všech záznamů

Potom, kdy uživatel provede dotaz, může chtít změnit platný výběr zpět ke všem záznamům v tabulce. Napíšeme si metodu projektu, která nahradí existující platný výběr všemi záznamy tabulky Zákazníci.

6.1.1. Vytvoření metody M_GEN_ShowAllRecords

1. Vytvořte metodu projektu nazvanou M_GEN_ShowAllRecords.
2. Vložte následující kód:

```
If (False)
    ` Method: M_GEN_ShowAllRecords
    ` Kurs ACI Universita
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Změní platný výběr na všechny záznamy.

  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if

ALL RECORDS  ([Zákazníci])

  ` Konec metody
```

3. Klepněte na uzavírací políčko.
4. Přejděte zpět do okna Průzkumníka, zmenšete jeho okno tak, aby seznam metod byl uveden zcela vlevo a přesvědčte se, že je otevřen seznam metod projektu.
5. Otevřete Editor nabídek a umístěte okno vedle okna Průzkumníka.
6. Vyberte Záhlaví #3, otevřete nabídku Záznamy a klepněte na položku nabídky Ukázat vše/G.
7. V okně Průzkumníka klepněte na metodu M_GEN_ShowAllRecords a táhněte ji do okna Editoru nabídek.
8. V zadávací oblasti Název metody v okně Editoru nabídek uvolněte myš.
9. Přejděte do prostředí Vlastní nabídky a vyzkoušejte novou položku nabídky (po provedení dotazu).

6.2. Třídění záznamu

Třídění záznamu v platném výběru je běžná úloha pro databáze. Pro třídění využijeme vestavěný Editor třídění z Prostředí uživatele.

6.2.1. Vytvoření položky nabídky pro Editor třídění

1. Vytvořte metodu projektu M_GEN_OrderByEditor.
2. Napište následující metodu:

```
If (False)
      ` Metoda: M_GEN_OrderByEditor
      ` Kurs ACI Universita
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

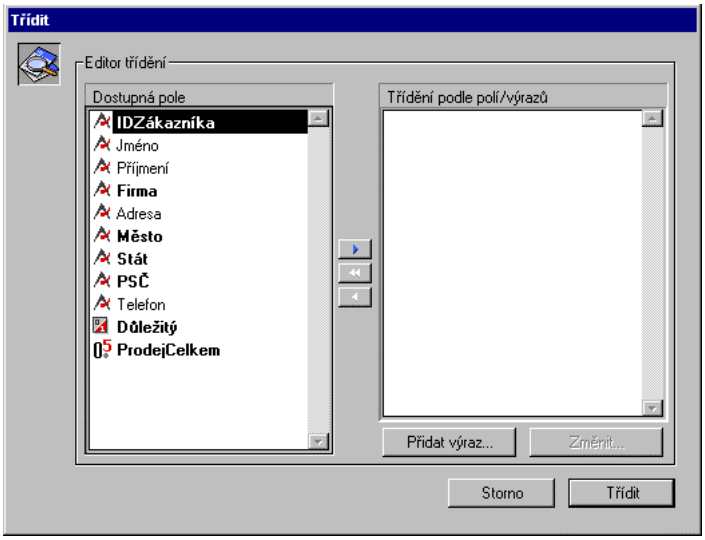
      ` Účel: Zobrazí vestavěný Editor 4D.


   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

ORDER BY([Zákazníci])

   ` Konec metody
```

3. Klepněte na uzavírací políčko.
4. Přejděte zpět do okna Průzkumníka a umístěte toto okno v levém horním rohu, přesvědčte se, že je rozevřen seznam metod projektu.
5. Otevřete Editor nabídek a umístěte jeho okno vedle okna Průzkumníka.
6. Vyberte Záhlaví #3, otevřete nabídku Záznamy a klepněte na položku nabídky Třídit.../T.
7. Klepněte na metodu M_GEN_OrderByEditor v okně Exploreru a táhněte ji do okna Editoru nabídek.
8. V Editoru nabídek nad zadávací oblastí Název metody, uvolněte tlačítko myši.
9. Přejděte do prostředí Vlastní nabídky a vyzkoušejte je.



6.3. Vícenásobné použití kódu

V Prostředí uživatele 4D automaticky obnovuje titulek v záhlaví okna. Pokaždé, když se změní platný výběr, změní se v Prostředí uživatele nadpis okna. V prostředí Vlastní nabídky je v názvu okna napsáno pouze Aplikace, abychom mohli zobrazovat změny v platném výběru, musíme měnit nadpis okna pomocí příkazu SET WINDOW TITLE a zobrazovat jím patřičný text.

Obecně je dobrá myšlenka použít podobný formát užívaný v Prostředí uživatele; Název tabulky:# z #, kde první # je počet záznamů v platném výběru a druhý znak # je celkový počet záznamů v tabulce. Později se pak budete moci rozhodnout a zvolit si případně svůj vlastní formát závisející na vašich potřebách či potřebách uživatele.

K zjištění počtu záznamů v platném výběru použijete přikaz Records in selection a pro zjištění celkového počtu záznamů v tabulce příkaz Records in table.

Můžete napsat kód, který požadované úlohy splní poměrně jednodušše, ale důležitější otázka je, kde musíme tento kód spouštět?

Prvním a nejzřejmějším umístěním tohoto kódu je uvnitř metody projektu M_Customers, protože to je místo, kde je tabulka poprvé vybírána a vytváří se počáteční platný výběr. Existuje však mnoho dalších míst (mysleme na ně zatím jako na možnosti), kde budete potřebovat tento kód spouštět. Dále je uveden jeden z takovýchto příkladů možností, které mohou změnit platný výběr a kde je potřeba změnit název okna:

Po volbě Zákazníci... z úvodní obrazovky, provede uživatel dotaz, který nenalezne žádný záznam v tabulce (žádný záznam tabulky nesplňuje kritéria dotazu). V tomto případě bude v titulu okna uveden nesprávný text.

Platný výběr je prázdný! Takže název okna by pak neodrážel správnou informaci a uživatel by asi nemohl věřit těmto údajům.

Jsou tři místa, kde váš kód bude muset být spouštěn a bude obnovovat název okna a každé z těchto míst odpovídá určitému typu metod.

1) Kdekoliv se změní platná tabulka (zatím se tímto místem nebudeme zabývat).

2) Kdekoliv se změní platný výběr.

3) Kdekoliv se změní celkový počet záznamů tabulky.

To znamená, že potřebujeme přidat kód do metod projektu M_Customers a M_GEN_QueryEditor. Nepotřebujeme jej přidat do metody M_GEN_OrderByEditor protože tato nemění nic co by vyhovovalo našim třem předchozím podmínkám. Znamená to však přidat tentýž kód zatím do dvou rozdílných míst (později při naplnění celé nabídky by jich bylo ještě více).

Jeden způsob jak vyřešit tento problém je zkopírovat tentýž kód do všech potřebných metod projektu. Ale nesmíte to zapomenout provést. Kopírování kódu je nudné jestliže duplikujete kód do více než několika míst. Je samozřejmě nepříjemné, že toto budete muset udělat pokaždé, když provedete změnu formátu názvu okna. Nejlepší způsob je vytvořit oddělenou metodu projektu. Tato metoda může být volána (spouštěna) z jiných metod projektu.

6.3.1. Vytvoření metody projektu WIN_OutputWindowTitle

1. V Prostředí návrháře vyberte Návrh ¬ Nová metoda….
2. V dialogovém okně metody napište název WIN_OutputWindowTitle.
3. Klepněte na tlačítko OK a vytvořte tak novou metodu.
4. Napište následující metodu:

```
If (False)
      ` Metoda: WIN_OutputWindowTitle
      ` Kurs ACI Universita
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Dle potřeby obnovuje název okna.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

SET WINDOW TITLE ("Zákazníci: "+ Records In Selection ([Zákazníci]) + " z  " +
        Records in table ([Zákazníci]))

   ` Konec metody
```

6.4. Použití metody projektu

Metodu projektu použijete v jiných metodách projektu nebo metodách objektu jednoduše tak, že napíšete její název. Když 4D provádí jeden řádek kódu po druhém a narazí na název metody, je právě prováděná metoda odložena zatímco uvedená metoda je spuštěna. Po dokončení běhu nově spuštěné metody pokračuje běh původní metody za řádkem volané metody.

6.4.1. Použití metody projektu WIN_OutputWindowTitle

1. Upravte metodu M_GEN_QueryEditor následujícím způsobem:

```
If (False)
      ` Metoda: M_GEN_QueryEditor
      ` Kurz ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97
      ` Účel: Zobrazit Editor dotazů pro  [Zákazníci].

   <>f_Verze6x10 := True
   <>fJ_Steinman := True
End if

QUERY ([Zákazníci])
WIN_OutputWindowTitle

   ` Konec metody
```

2. Upravte metodu projektu M_GEN_ShowAllRecords následovně:

```
If (False)
      ` Metoda: M_GEN_ShowAllRecords
      ` Kurs ACI Universita
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Změní platný výběr na všechny záznamy.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

ALL RECORDS  ([Zákazníci])
WIN_OutputWindowTitle

   ` Konec metody
```

3. Upravte metodu projektu M_Customers následovně:

```
If (False)
` Metoda: M_Customers
      ` Kurs ACI Univerzity
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

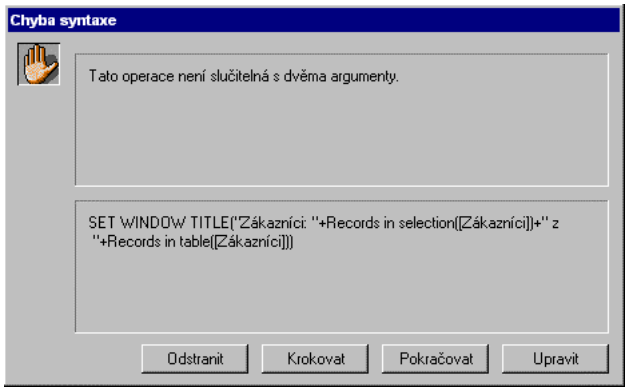
      ` Účel: Zobrazí seznam záznamů [Zákazníci] ve výstupním formuláři.
   <>f_Version6x10 := True
   <>fJ_Steinman :=  True
End if

ALL RECORDS ([Zákazníci])
`  Změna platného výběru.
MENU BAR (3) `přepne do záhlaví 3
WIN_OutputWindowTitle
MODIFY SELECTION([Zákazníci]; *)
`  Ukáže záznamy na obrazovku.
MENU BAR (1) `přepne do záhlaví 1

` Konec metody
```

6.5. Přidání hodnot a konverze typů dat

Vyzkoušejte si provedené změny na přepnutím do agendy Zákazníků. Překvapení! Kód, který jsmesi napsali ve výše uvedeném cvičení nepracuje. Místo toho jste obdrželi chybové hlášení, které vás upozorňuje, že použité typy daj nejsou slučitelné.



Když skládáte dohromady dvě čísla, výsledkem je součet čísel. Když skládáte dohromady dva texty (nazývané rovněž řetězce) výsledkem je spojení těchto dvou hodnot. Např. složením „kolo“ a „běžka“ je výsledkem koloběžka. Nebo složením čísel „1“ + „2“ = „3“ ale pokud je používáme jako řetězce „1“ + „2“ = „12“.

Některé hodnoty nemohou být přidány k jiným hodnotám. Čísla nemohou být přidána k řetězcům. A to je náš problém, protože příkaz SET WINDOW TITLE, očekává jako hodnotu řetězec a zatímco funkce Records in selection navrací číslo. Abychom tento problém mohli vyřešit, musí být hodnota navracená příkazem Records in selection, konvertována do řetězce. Tuto úlohu můžeme provést pomocí funkce String.

Funkce String konvertuje čísla, datumy a čas do textu do řetězce. Takže tato funkce překonvertuje číslo navracené funkcí Records in selection do řetězce, který již může být spojen s textovou zprávou v příkaze SET WINDOW TITLE.

4D vygenerovala chybu, protože váš řádek programu prostě nedával smysl. Pamatujte si, že počítače zacházejí s různými typy dat různě. Čísla nejsou totéž jako obrázky, které zase nejsou totéž co řetězce. Když uvidíte zprávu podobnou této, která říká něco jako neslučitelné, není kompatibilní nebo zmatení typů dat. Znamená to nejčastěji, že jste se pokusili sloučit dva různé typy dat dohromady. V tomto případě funkce Records in selection navrací číslo (long integer) a příkaz SET WINDOW TITLE odmítl pracovat s číslem. Takže jsme potřebovali změnit čísla na řetězec pomocí příkazu String.

Jestliže budete někdy potřebovat změnit řetězec na číslo opačným příkazem k příkazu String je Num.

6.5.1. Úprava metody WIN_OutputWindowTitle

1. V Prostředí návrháře vyberte Návrh ¬ Upravit metodu….
2. V okně Průzkumníka klepněte do seznamu metod projektu na WIN_OutputWindowTitle.
3. Klepněte na tlačítko Upravit nebo poklepejte myší:

```
If (False)
    ` Metoda: WIN_OutputWindowTitle
    ` Kurs ACI Universita
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Dle potřeby obnovuje název okna.

  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if

SET WINDOW TITLE ("Zákazníci: "+ String (Records In Selection ([Zákazníci])) + " z  " +
String (Records in table ([Zákazníci])))

  ` Konec metody
```

4. Přejděte do prostředí Vlastní nabídky a vyzkoušejte ji.

7. Generické programování

Mohli by jsme pokračovat dále v přidávání metod projektu v navídce Záznamy. Každá metoda bude vyžadovat programování pro tabulku [Zákazníci], jako mazání záznamů z agendy zákazníků, export dat do agendy zákazníků. Problém je zde následující, pokud budete chtít tuto nabídku použít i pro jiné agendy, museli by jste vytvořit novou s novými metodami pro tuto jinou tabulku jako jsou [Faktury],[Produkty] atd. jak se vám bude databáze rozrůstat. Znamenalo by to vždy překopírovat Záhlaví #3, vytvořit nové a přiřadit mu nově napsané metody. Poněkud nudné, nešikovné a co teprve testování takovéhoto systému.

7.1. Použití ukazatelů k vytvoření generických metod

Nejlepší způsob jak znovu použít váš kód metod a nabídku cyklicky pro všechny další tabulky je vytvořit tyto metody, tak že budou moci být beze změn (genericky) použity i v dalších tabulkách.

Co kdyby jste mohli nahradit každý výskyt [Zákazníci], určitým zástupcem, který bude reprezentovat název tabulky (v našem případě Zákazníci). Pak pokud budete chtít použít tutéž metodu v jiné tabulce, změníte pouze v první metodě, která tabulku volá tohoto zástupce na zástupce jiné tabulky jako např. [Faktury]. 4D zná způsob jak vytvořit takovéto zástupce, kteří odpovídají názvu tabulky: ukazatele. Dále si pamatujte, že ukazatel tabulky je jednoduše zástupcem za tuto tabulku.

7.2. Přiřazení „:=“ versus „=“

4D používá := (dvojtečka je rovno) a = (je rovno) pro porovnání. Všimněte si, že v seznamu klíčových slov v Editoru metod není znak je rovno. Jako začátečníkům vám doporučujeme, aby jste pro zadávání operátorů přiřazení používali sloupec klíčových slov. Je to jednodušší a spíše se vyhnete běžné chybě v přepsání = místo :=.

7.3. Ukazatele a symboly

Použití ukazatelů je spojeno s použitím symbolu ukazatele „->“. Na platformách Macintosh i Windows se tento symbol vytvoří jednoduše napsáním znaků „-„ (pomlčka) a „>“ (větší než). Ve starších verzích 4D byl používán symbol » Tento symbol bude automaticky nahrazen při konverzi do verze 6. Symbol pro ukazatel se používá ve dvou místech:

- Vytvoření či přiřazení ukazatelů
- Užití ukazatelů.

Existují dvě pravidla, která je potřeba si pamatovat:

- Symbol ukazatele vlevo od názvu tabulky, vytváří ukazatel na tuto tabulku.
- Symbol ukazatele na pravé straně proměnné obsahující ukazatel používá tento ukazatel zpětně jako název tabulky.

Příklad	Umístění	Akce
->[Zákazníci]	Vlevo	Vytváří ukazatel
pTable->	Vpravo	Užívá ukazatel

Umístění symbolu ukazatele na levé straně názvu tabulky, vytváří ukazatel odpovídající této tabulce. Zde je příklad vytvoření ukazatele k tabulce [Zákazníci]:

```
->[Zákazníci]
```

To co potřebujete je vzít tento ukazatel a uložit jej v nějaké proměnné pro pozdější použití. Dále je uveden příklad, jak ukazatel zachytit a uložit v proměnné:

```
pTable := ->[Zákazníci]
```

Proměnná pTable nyní obsahuje tento ukazatel. Nyní můžete tuto proměnnou ukazatele použít v libovolném místě, kde by jste jinak použili název tabulky. Např. podívejme se na řádek kódu v metodě M_GEN_QueryEditor a pak ji upravme s použitím proměnné ukazatele a symbolu ukazatele použitého vpravo od proměnné ukazatele.

Staré použití:

```
QUERY ([Zákazníci])
```

Nové použití:

```
QUERY (pTable->)
```

Abychom novou verzi této metody pracující s proměnnou ukazatele mohli použít, musíme nejdřív danou proměnnou ukazatele vytvořit. Takže prvním krokem v generizování vašeho kódu je vytvoření ukazatele v některé úvodní části práce uživatele. Umístěme tento řádek programu těsně před příkaz MODIFY SELECTION v metodě M_Customers.

7.3.1. Vytvoření proměnné ukazatele odkazující na tabulku

1. Otevřete metodu M_Customers.
2. Upravte metodu následujícím způsobem:

```
If (False)
` Metoda: M_Customers
` Kurs ACI Univerzity
` Vytvořeno: Jim Steinman
` Datum: 15/1/97

    ` Účel: Zobrazí seznam záznamů [Zákazníci] ve výstupním formuláři.
  <>f_Version6x10 := True
  <>fJ_Steinman :=  True
End if
```

```
pTable := -> [Zákazníci]

ALL RECORDS (pTable->)
`  Změna platného výběru.
MENU BAR (3) `přepne do záhlaví 3
WIN_OutputWindowTitle
MODIFY SELECTION (pTable->; *)
`  Ukáže záznamy na obrazovku.
MENU BAR (1) `přepne do záhlaví 1

` Konec metody
```

7.3.2. Úprava metody M_GEN_QueryEditor s užitím ukazatele

1. Otevřete metodu M_GEN_QueryEditor.
2. Upravte metodu následujícím způsobem:

```
If (False)
      ` Metoda: M_GEN_QueryEditor
      ` Kurs ACI University
      ` Generická metoda
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97
      ` Účel: Zobrazit Editor dotazů pro pTable

   <>fGeneric := True
   <>f_Verze6x10 := True
   <>fJ_Steinman := True
End if

QUERY (pTable->)
WIN_OutputWindowTitle

   ` Konec metody
```

3. Všimněte si, že tato metoda může nyní pracovat s libovolnou tabulkou, která byla předtím vložena jako ukazatel do proměnné ukazatele. Jako důsledek jsme nyní upravili i komentáře tak, že odrážejí toto generické použití s libovolnou tabulkou. Takže ani v komentářích se neodráží název agendy. K tomu, abychom nalezli tyto generické procedury jsme zavedli další speciální logickou proměnnou a nastavili ji na true.

7.3.3. Úprava metody M_GEN_ShowAllRecords s použitím ukazatelů

1. Otevřete metodu M_GEN_ShowAllRecords.

2. Upravte ji následujícím způsobem:

```
If (False)
      ` Metoda: M_GEN_ShowAllRecords
      ` Kurs ACI Universita
      ` Generická metoda
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Změní platný výběr na všechny záznamy pro pTable

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

ALL RECORDS  (pTable->)
WIN_OutputWindowTitle

   ` Konec metody
```

7.3.4. Úprava metody M_GEN_OrderByEditor s použitím ukazatelů

1. Otevřete metodu M_GEN_OrderByEditor.
2. Upravte ji následujícím způsobem:

```
If (False)
      ` Metoda: M_GEN_OrderByEditor
      ` Kurs ACI Universita
      ` Generická metoda
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97
      ` Účel: Zobrazí vestavěný Editor 4D pro pTable

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

ORDER BY(pTable->)
   ` Konec metody
```

7.4. Zobrazení WIN_OutputWindowTitle s názvem tabulky

Nyní, protože váš program používá ukazatele v proměnné pTable, můžete tento ukazatel použít i k určení názvu tabulky. Pokud máte k dispozici název tabulky, můžete upravit i název zobrazovaný v okně nezávisle na přímém vepsání názvu agendy.

K zjištění názvu tabulky použijeme příkaz Table name. Funkce Table name přijímá jako parametr přímo ukazatel a vrací název tabulky, na který tento ukazatel odkazuje, jako řetězec.

Poznamenejme, že příkaz Table name je jedním z několika, který používá přímo ukazatel.
V těchto několika příkazech (Table, Field, Table name, Field name) vkládáte jako parametr samotný ukazatel, bez symbolu ukazatele (->).

Poznámka: v následujícím cvičení, kód následující se Set Window title patří do jednoho řádku. Zde v textu to vypadá jako kdyby byl rozdělen na několik řádků

7.4.1. Úprava metody WIN_OutputWindowTitle k zobrazení názvu tabulky

1. Otevřete metodu WIN_OutputWindowTitle
2. Upravte metodu následovně:

```
If (False)
      ` Metoda: WIN_OutputWindowTitle
      ` generická metoda
      ` Kurs ACI Universita
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Dle potřeby obnovuje název okna.
   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

SET WINDOW TITLE ( Table name (pTable) +
String (Records In Selection (pTable->)) + " z  " + String (Records in table (pTable->)))

   ` Konec metody
```

7.5. Ukončení

Položka nabídky Soubor ¬ Konec v Záhlaví #1, neprovádí nic, protože k této položce není přiřazena žádná metoda. Ke skutečnému ukončení 4D musíte po přepnutí do Prostředí uživatele nebo návrháře, použít vestavěný příkaz Soubor ¬ Konec.

K tomu, abychom zaktivnili vaši vlastní položku Soubor ¬ Konec, musíme napsat metodu používající příkaz Quit 4D. Quit 4D okamžitě ukončuje 4D a přeruší všechny aktivity, které jsou v běhu včetně tisku, importu nebo exportu.

7.5.1. Zaktivnění položky nabídky Soubor ¬ Konec

1. Vytvořte metodu projektu M_GEN_Quit
2. Upravte metodu následujícím způsobem:

```
If (False)
      ` Metoda: M_GEN_Quit
      ` Kurs ACI Universita
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Přeruší všechny operace a ukončí 4D, návrat do OS

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

QUIT 4D

   ` Konec metody
```

7.6. Prostředí uživatele

Tím, že jsme zaktivnili položku Soubor ¬ Konec, znepřístupnili jsme poslední prázdnou položku nabídky, kterou jsme se přepínali z úvodní obrazovky do Prostředí uživatele. Do nabídky Soubor si můžete přidat vlastní prázdnou položku Prostředí uživatele, která nebude mít přiřazenu metodu a bude provádět totéž co předtím položka Konec.

Extra Kredit Cvičení

7.6.1. Přidání položky nabídky pro Prostředí uživatele

1. Přejděte do Záhlaví #1 v Prostředí návrháře.
2. Vložte položku nazvanou Prostředí uživatele a nepřiřazujte ji žádnou metodu.

3. Nezapomeňte vymazat tuto položku nabídky vždy předtím, když budete svou aplikaci poskytovat konečným uživatelům. Tato položka nabídky je zde pouze pro vaše pohodlí.

7.7. Vytvoření metody Přidat záznam

7.7.1. Přiřazení metody k položce nabídky Záznamy ¬ Přidat nový…

1. Vytvořte novou metodu projektu nazvanou M_GEN_AddRecords.
2. Napište následující metodu:

```
If (False)
      ` Metoda: M_GEN_AddRecords
      ` Kurs ACI Universita
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Přidává záznamy do tabulky.

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

ADD RECORD (pTable->)
WIN_OutputWindowTitle

   ` Konec metody
```

3 Potáhněte a vložte novou metodu do příslušného místa v Editoru nabídek.

7.8. Opakované přidávání záznamů

Nyní jsme schopni přidat najednou pouze jeden záznam. V Prostředí uživatele, když přidáváte záznamy, vytváří 4D nový záznam dokud není poslední záznam zrušen. Jestliže chcete můžete vaši metodu M_GEN_AddRecords upravit tak, aby přidávala záznamy v cyklu.

Nyní musíme přinutit 4D opakovat část programu dokud ji nedáme jiný příkaz. Příkazy Repeat...Until, opakují libovolné řádky kódu umístěné mezi ně. Kód je opakován ve smyčce dokud není podmínka následující za výrazem Until pravdivá.

7.9. Systémová proměnná „OK“

Metoda projektu sleduje, které tlačítko bylo klepnuto testování proměnné OK. Proměnná je jednoduše dočasné místo pro uložení určité hodnoty jako např. počtu částí textu. Systémová proměnná je speciální proměnná řízená automaticky 4D (může být pokládána za stálou proměnnou, rezervované slovo). Proměnná OK je jednou z těchto systémových proměnných. Jestliže uživatel klepnul na tlačítko ACCEPT/PŘIJMOUT je tato proměnná nastavena na hodnotu 1, OK=1. Jestliže uživatel klepnul na tlačítko CANCEL/STORNO pak OK=0. Více informací o systémových proměnných naleznete v příloze Příručky jazyka.

Potom, kdy uživatel přidá záznam změní se platný výběr na tento záznam, který byl naposledy přidán (pamatujte si, že jiný způsob jak si představit platný výběr je, že je to výsledek poslední uživatelem provedené operace). Pokud budete potřebovat změnit platný výběr tak, aby po přidání záznamu ukazoval všechny záznamy v tabulce, včetně právě přidaného záznamu, přidejte do následující procedury jeden řádek kódu těsně před volání metody projektu WIN_OutputWindowTitle.

7.9.1. Přidávání záznamů ve smyčce

1. Otevřete metodu M_GEN_AddRecords.
2. Upravte ji následujícím způsobem:

```
If (False)
      ` Metoda: M_GEN_AddRecords
      ` Kurs ACI Universita
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Přidává záznamy do tabulky.

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

•  Repeat
     ADD RECORD (pTable->)
•  Until (OK = 0) `Opakuje dokud uživatel neklepne na tlačítko Storno
   WIN_OutputWindowTitle

     ` Konec metody
```

3. Přejděte do prostředí Vlasní nabídky a vyzkoušejte položku nabídky.

7.10. Přenesení funkce tlačítka bDone do položky nabídky Záznamy ¬ Hotovo

Toto tlačítko je odlišné, protože neobsahuje žádnou metodu objektu, kterou bychom mohli překopírovat a přenést do metody projektu. Místo metody projektu provádí toto tlačítko automatickou akci. Položky nabídky nemohou provádět automatické akce, mohou provádět pouze metody projektu k nim přiřazené. Naštěstí programovací jazyk obsahuje příkazy ekvivalentní ke každé automatické akci. Příkazy ACCEPT či CANCEL fungují jako příkaz k uzavření výstupního formuláře.

7.10.1. Přiřazení metody k položce nabídky Záznamy ¬ Hotovo

1. Vytvořte novou metodu projektu s názvem M_GEN_Done.
2. Napište následující metodu:

```
If (False)
      ` Metoda: M_GEN_Done
      ` Kurs ACI Universita
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Uzavře výstupní formulář z  MODIFY SELECTION

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

CANCEL

   ` Konec metody
```

3 Potáhněte a vložte novou metodu do příslušného místa v Editoru nabídek.
4. Přejděte do Prostředí uživatele a vyzkoušejte.

7.11. Dotaz dle příkladu

Editor dotazů je silný nástroj, ale někdy může být pro některé uživatele složitý a nebo jim může poskytovat více možností než jim chcete dát. Prostředí uživatele poskytuje jednoduchý nástroj dotazů: DOTAZ DLE PŘÍKLADU. Tento nástroj zobrazí platný vstupní formulář a dovolí uživateli vyplnit pole hodnotami na které se chce dotázat. 4D se pak dotazuje na záznamy, které vyhovují všem polím, které uživatel vyplnil (dotaz a současně). NEMŮŽETE použít DOTAZ DLE PŘÍKLADU pro vyhledání záznamů u kterých chcete vyhledat všechny výskyty (dotaz „nebo“).

7.11.1. Vytvoření metody pro Dotaz dle příkladu...

1. Vytvořte novou metodu projektu: M_GEN_QueryByForm
2. Napište následující metodu:

```
If (False)
    ` Metoda: M_GEN_QueryByForm
    ` Kurs ACI Universita
    ` Generická procedura
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Zobrazí platný vstupní formulář a použije jej jako nástroj Dotazu.

  <>fGeneric := True
  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if

QUERY BY EXAMPLE(pTable->)
WIN_OutputWindowTitle

  ` Konec metody
```

3. Potáhněte a vložte novou metodu do patřičného místa v Editoru nabídek.

7.12. Speciální formulář pro příkaz QUERY BY EXAMPLE

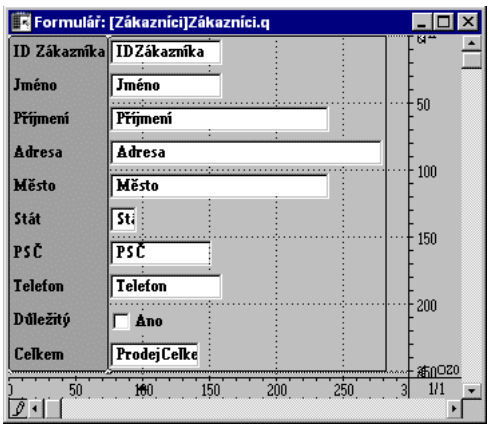
Příkaz QUERY BY EXAMPLE používá platný vstupní formulář. Všechna tlačítka automatických akcí jsou nepřístupná, kromě tlačítek Přijmout, Storno a tlačítek listování stránkami formuláře.

Abychom učinili dotazování co nejjednodušší pro uživatele, budeme pravděpodobně chtít za tímto účelem vytvořit speciální vstupní formulář. V tomto formuláři můžeme vyloučit všechny speciální tlačítka a rovněž všechna pole podle kterých nechceme, aby uživatel prováděl dotazy (např. neindexovaná pole).

V další cvičení použijeme příkaz INPUT FORM k přepínání mezi staartním vstupní formulářem a formulářem, který jsme vytvořili pouze pro dotazy. Příkaz INPUT FORM nezpůsobuje zobrazení formuláře. Tento příkaz řekne 4D pouze, který formulář bude použit příště v příkazu, který zobrazuje vstupní formulář. Tento příkaz je platný dokud není zrušen, takže musíme zajistit změnu vstupního formuláře zpět na staartní formulář na konci metody.

7.12.1 Vytvoření speciálního formuláře pro QUERY BY EXAMPLE

1. Vyberte Návrh ¬ Nový formulář…
2. Ze seznamu tabulek vyberte Zákazníci.
3. Nazvěte tento formulář Dotazy.
4. Jako typ formuláře vyberte S obsahem vstupní.
5. Užitý vzor vyberte Simple – žádná tlačítka.
6. Klepněte na dvojitou šipku ukazující vpravo a přidejte tak všechna pole do nového formuláře.
7. Jestliže nechcete pole v abecedním pořadí, můžete je pomocí myši potáhnout a vložit na nové místo dokud nejste s novým pořadím spokojeni.
8. 4D automaticky vytvoří dvě tlačítka Hledat a Zrušit, takže nepotřebujete již provádět nic dalšího.



7.12.2. Úprava metody M_GEN_QueryByForm

1. Upravte metodu M_GEN_QueryByForm následujícím způsobem:

```
If (False)
    ` Metoda: M_GEN_QueryByForm
    ` Kurs ACI Universita
    ` Generická procedura
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

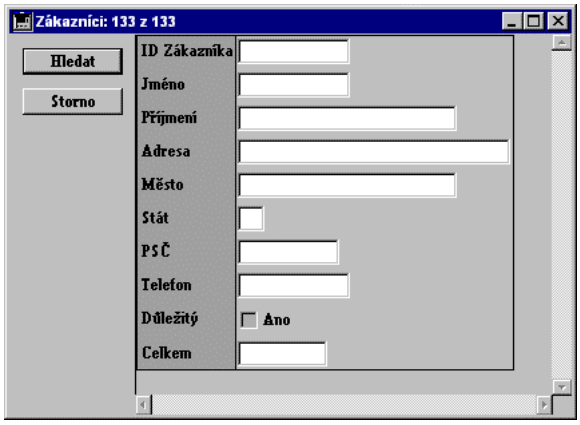
    ` Účel: Změní platný vstupní formulář na formulář Dotazy.
    ` Použije tento formulář jako nástroj pro Dotaz.

  <>fGeneric := True
  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if
```

- ```
  INPUT FORM (pTable->; "Dotazy")
  QUERY BY EXAMPLE (pTable->)
  WIN_OutputWindowTitle
    ` Přepne zpět staartní vstupní formulář.
  ```
- ```
  INPUT FORM (pTable->; "Vstupní")

    ` Konec metody
  ```

2. Přejděte do prostředí Vlastní nabídky a otestujte ji.

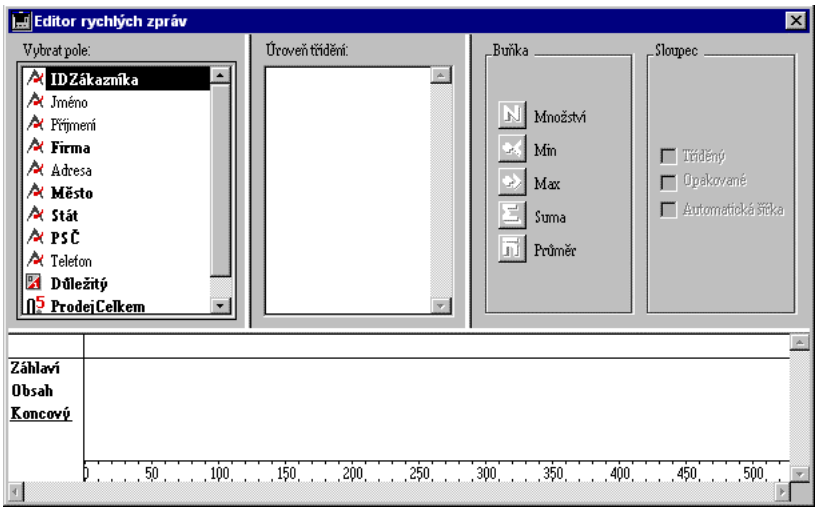


7.12.3 Vytvoření formuláře Dotazy pro M_GEN_QueryByForm a ostatní tabulky

1. Vytvořte nové vstupní formuláře nazvané Dotazy pro tabulky [Produkty] a [Faktury] a vložte do nich pole podle, kterých si myslíte, že je nejvhodnější vyhledávat.

7.13. Zobrazení Editoru rychlých zpráv

Jestliže chcete, aby uživatel byl schopen sestavovat své vlastní sloupcové zprávy můžete mu zobrazit Editor rychlých zpráv. Toto můžete provést pomocí příkazu Report.



Příkaz Report je používán k vytvoření a tisku zpráv pomocí Editoru rychlých zpráv. K tisku určité již existující rychlé zprávy, která byla uložena na disk, předáte příkazu jako parametry název tabulky a název rychlé zprávy podobně jak je uvedeno dále:

```
REPORT (pTable->; "Nová zpráva")
```

Jestliže chcete, aby uživatel byl schopen vybrat si libovolnou rychlou zprávu, kterou uložil, předejte jako parametr v názvu zprávy prázdný řetězec:

```
REPORT (pTable->; "")
```

K zobrazení samotného Editoru rychlých zpráv musíte předat jako název zprávy název, který neexistuje a nebyl nikdy na disk uložen. Musíte použít takový název, který určitě nikdo nikdy nepoužije jako např. „a*b*xx“ Jestliže by uživatel použil tento název k uložení své zprávy na disk již nikdy (pokud tento návrh z disku nevyhodí do koše) se v dané agendě nedostane k Editoru rychlých zpráv. Proto je potřeba použít název, který rozhodně nikdo nemůže vložit. Můžete použít například i znaku nový řádek, protože tento znak uživatel do názvu tabulky rozhodně není schopen z klávesnice napsat.

7.14. Užití funkce Char k vytváření znaků na základě kódu ASCII

Aby jste se ujistili, že v názvech zpráv budou znaky, které uživatel z klávesnice nemůže napsat, můžete použít i znaky, které jsou z klávesnice nedostupné.

V rámci prgramovacího jazyka můžete použít libovolný znak, aniž by jste jej zadávali přímo z klávesnice. Pro toto zadání lze použít funkce Char do které se jak argument vkládá číslo, které odpovídá číslu ASCII ve znakové sadě. Každá klávesa na vaší klávesnici má číselný kód, který je k ní přiřazen. Tato čísla byla přiřazena staardizační komisí a nazývají se American Staards Code for Information Interchange nebo zkráceně ASCII. Téměř každý počítač na této planetě používá nějaký systém kódování ASCII.

Celý seznam 256 kódů ASCII je uveden v příloze vaší Příručky návrháře, zde je několik příkladů:

Znak	ASCII Kód
A	65
B	66
Čárka	44
Return	13
Tab	9
0	48
1	49

Pomocí programovacího jazyka můžete zadat libovolný znak tak, že předáte jeho ASCII hodnotu funkci Char.

```
REPORT (pTable->; Char (13))
```

Poznámka: České klávesnice mají samozřejmě odlišné znaky a tedy i odlišné kódování, které závisí na některé z používaných sad znaků (Win 1250, ISO 8859-2, MAC CE atd.)

7.14.1. Přidání metody k položce Zprávy ¬ Rychlé

1. Vytvořte novou metodu M_GEN_QuickReport.
2. Napište následující metodu:

```
If (False)
      ` Metoda: M_GEN_QuickReport
      ` Kurz programování ACI Univerzita
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15.1.97


      ` Účel: Otevře Editor rychlých zpráv pro uživatele.

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

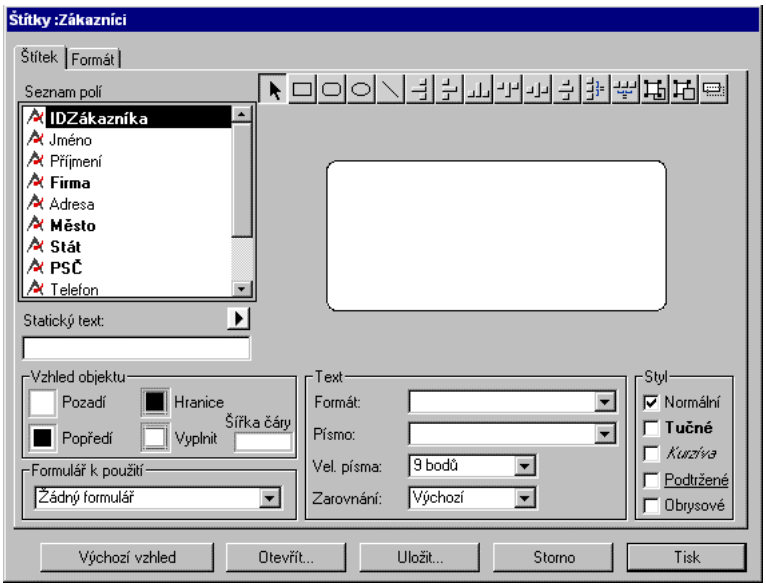
REPORT (pTable->; Char (Carriage Return))

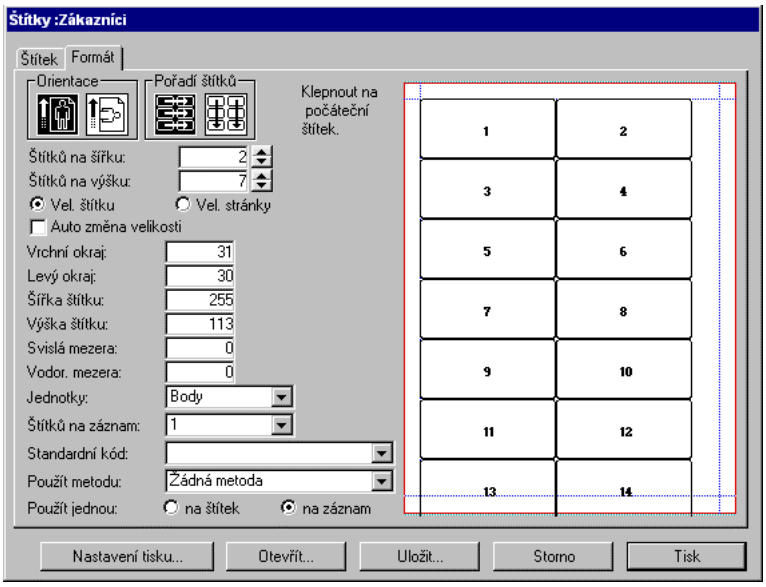
   ` Konec metody
```

3. Potáhněte a vložte novou metodu do patřičného místa v Editoru nabídek.

7.15. Přidání Editoru Štítků

Pro svého uživatele můžete rovněž přidat Editor štítků.





Editor štítků může být přidán obdobným způsobem jako editor rychlých zpáv. Jen místo příkazu REPORT uvedete příkaz PRINT LABEL. Tento příkaz rovněž očekává název vzoru uloženého na disku. Jestliže zadáte nesmyslný název bude opět zobrazen celý editor. Platí rovněž diskuze se znaky, keré nelze zadat z klávesnice.

7.15.1. Přidání metody pro položku nabídky Zprávy ¬ Štítky

1. Vytvořte metodu projektu nazvanou M_GEN_Labels.
2. Upravte metodu následujícím způsobem:

```
If (False)
    ` Metoda: M_GEN_Labels
    ` Kurz programování ACI Univerzita
    ` Generická procedura
    ` Vytvořeno: Jim Steinman
    ` Datum: 15.1.97

    ` Účel: Otevře Editor

  <>fGeneric := True
  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if

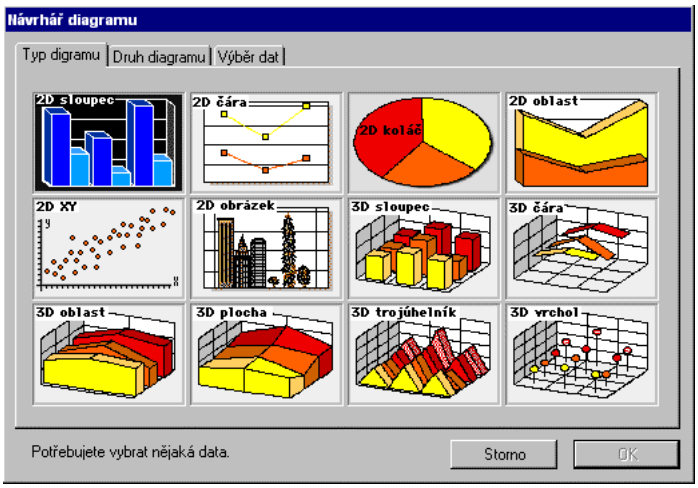
PRINT LABEL (pTable->; Char (Carriage Return))

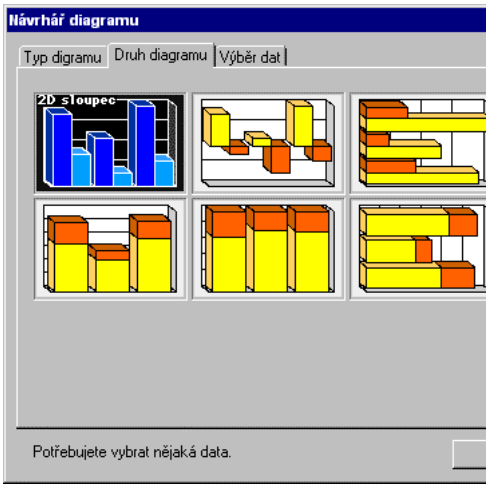
  ` Konec metody
```

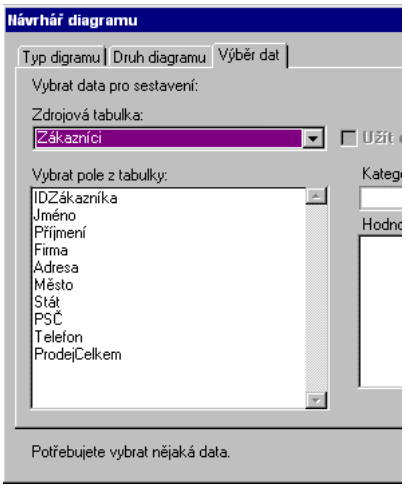
3. Potáhněte a vložte metodu projektu na patřičné místo v editoru nabídek.

7.16. Tisk diagramů

Rovněž můžete uživateli zobrazit vestavěný editor diagramů.







Editor diagramů může být zobrazen jen s názvem tabulky jako parametrem v příkazu GRAPH TABLE .

7.17. Přidání metody k položce nabídky Zprávy ¬ Diagramy

1. Vytvořte novou metodu projektu M_GEN_Chart
2. Vložte následující metodu:

```
If (False)
    ` Metoda: M_GEN_Chart
    ` Kurz ACI Univerzity
```

```
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15.1.97

      ` Účel: Otevře editor diagramů (Chart).

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

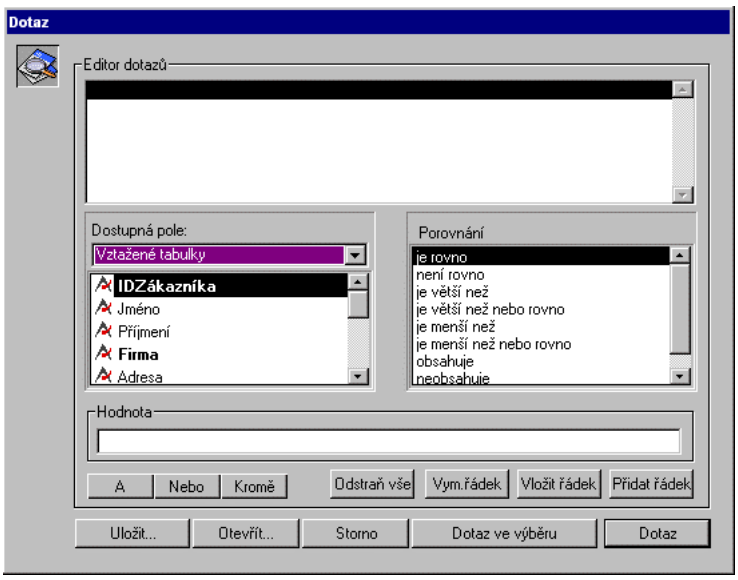
GRAPH TABLE (pTable->)

   ` Konec metody
```

3. Potáhněte a vložte novou metodu na patřičné místo v editoru nabídek
4. Přejděte do prostředí vlastních nabídek a vyzkoušejte.

7.18. Omezení dotazů na platný výběr místo dotazů v celé tabulce

Příkaz QUERY provádí dotaz v celé tabulce pro všechny záznamy. Můžete chtít omezit vyhledávání pouze na ty záznamy, které jsou obsaženy v platném výběru, použitím příkazu QUERY SELECTION.



Výsledný editor dorazů vypadá úplně stejně jako při příkazu QUERY. Rozdíl je v tom, že tlačítko Dotaz není přístupné a je přístupné pouze tlačítko Dotaz ve výběru. (Uživatel tedy nemůže provést hledání, keré si nepřejete)

7.18.1. Vytvoření položky nabídky Dotaz ve výběru

1. Vytvořte metodu M_GEN_QuerySelection následovně:

```
If (False)
      ` Metoda: M_GEN_QuerySelection
      ` Kurz ACI Univerzita
      ` Generická
      ` Vytvořeno: Jim Steinman
      ` Datum: 1/15/97

      ` Účel: Omezuje dotaz na dotaz ve výběru

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

QUERY SELECTION (pTable->)
WIN_OutputWindowTitle
```

```
` Konec metody
```

2. Potáhněte a vložte metodu na patřičné místo editoru nabídek
3. Přejděte do prostředí uživatele a vyzkoušejte.

Kvíz

- Proč existuje Editor štítků, když 4D již má Editor rychlých zpráv?
- Kromě adresových štítků na dopisy co ještě jiného lze tisknout pomocí Editoru štítků?
- Editor štítků ve svém výchozím nastavení neumí tisknout oddělovače jako čárka mezi poli (např. titul a jméno). Proč to není problém?

7.19. Přidání rysů pro import a export záznamů do nabídky Záznamy

Export vytváří nový diskový soubor a kopíruje data ze 4D do tohoto diskového souboru. Import pracuje v opačném směru a přináší data z diskového souboru do 4D.V této části si přidáme řadu nových metod do nabídky Záznamy a napíšeme generické metody pro import a export záznamů.

7.20. Přidání metod pro položky nabídky Záznamy ¬ Import a Export

Použití příkazu PRINT SELECTION umístí kopii vašich záznamů na papír. Někdy je lepší uložit záznamy místo na papír na disk. Tímto způsobem, můžete sdílet vaše data se spolupracovníky a jinými aplikacemi jako tabulkové procesory a textové procesory. Tento způsob vytažení dat ze 4D se nazývá export.

7.21. Exportování dat za pomoci formuláře jako filtru

Exportování ze 4D může být provedeno za pomoci výstupního formuláře jako filtru. V dalším příkladu, vytvoříte specielní formulář pouze pro účely exportu. Jakékoliv pole umístíte do tohoto formuláře, bude exportováno. Nemusíte tedy exportovat všechna pole tabulky. Navíc pořadí vstupu v tomto formuláři určuje pořadí v kterém budou pole exportována. Například pořadí vstupu určuje, že Jméno předchází Příjmení, nebo naopak.

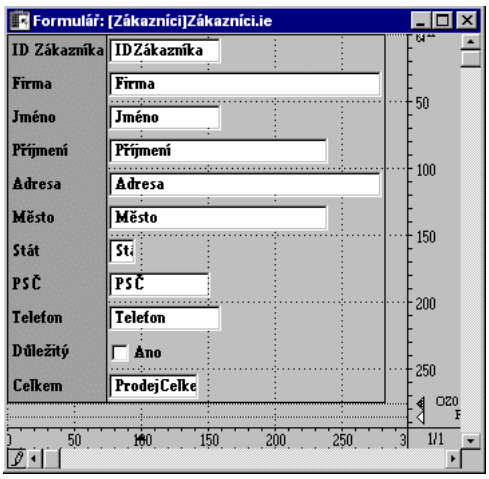
7.21.1 Vytvoření nového formuláře pro export a import

1. Vyberte Návrh ¬ Nový formulář...
2. Vyberte Zákazníci z nabídky.
3. Napište `ImportExport` jako název formuláře.
4. Zvolte jako vzor S obsahem (vstupní).
5. Jako vzor užijte Simple – žádná tlačítka.
6. Vložte pole v následujícím pořadí:

   IDzákazníka
   Jméno
   Příjmení
   Firma
   Adresa
   Město
   Stát
   PSČ
   Telefon
   Důležitý
   ProdejeCelkem

Toto zajistí, že pro export a import bude zvoleno správné pořadí polí.

Obě další tabulky Faktury a Produkty budou potřebovat formuláře “ImportExport“.



7.21.2. Přidání položky nabídky pro Záznamy ¬ Export

1. Vytvořte formulář nazvaný [Zákazníci];"ImportExport".
2. Změňte pořadí vstupu ve formuláři, který jste vytvořili a určete tak pořadí polí, které budou exportovány.
3. Vytvořte metodu projektu M_IMPEXP_Export následujícím způsobem:

```
If (False)
      ` Metoda: M_IMPEXP_Export
      ` Modul: Import a Export
      ` Kurs ACI Universita
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Exportuje všechny záznamy platného výběru a platné tabulky
      ` do textového souboru.
      ` Používá formulář ImportExport.

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

OUTPUT FORM (pTable->; "ImportExport")
EXPORT TEXT (pTable->; "") `     Zobrazuje dialog pro vložení názvu souboru OUTPUT FORM
(pTable->; "Output")
   ` Konec metody
```

4. Přejděte do prostředí Vlastní nabídky a otestujte tuto metodu.

7.22. Přidání metody pro položku nabídky Záznamy ¬ Import

Nyní můžete pro import použít stejné formuláře jako pro export. Je zde ale jeden rozdíl. Zatímco při exportu se používá platný výstupní formulář jako filtr, import používá za tímto účelem platný vstupní formulář.

7.22.1. Přidání metody pro položku nabídky Záznamy ¬ Import.

1. Vytvořte metodu projektu nazvanou M_IMPEXP_Import následujícím způsobem:

```
If (False)
      ` Metoda: M_IMPEXP_Import
      ` Modul: Import a Export
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Importuje záznamy z textového souboru s použitím formuláře ImportExport.

   <>fGeneric := False
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

INPUT FORM (pTable->; "ImportExport")
IMPORT TEXT (pTable->; "")  `    Zobrazuje okno pro výběr souboru z disku
INPUT FORM (pTable->; "Input")
WIN_OutputWindowTitle
   ` Konec metody
```

2. Přejděte do prostředí Vlastní nabídky a otestujte tuto metodu.

7.23.3 Většina tabulek bude pro export a import dat potřebovat formulář

1. Vytvořte nové formuláře nazvané ImportExport pro tabulku [Produkty] a [Faktury]. Pole zadejte dle vlastního uvážení .

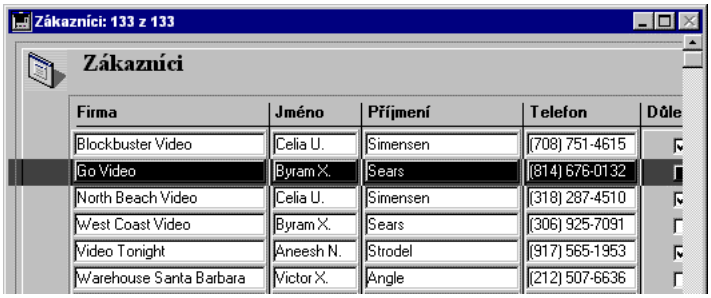
7.24.4. Ověření správného pojmenování všech

Pro správné fungování kódu je nyní nutné, aby byli všechny jednotlivé vstupní a výstupní formuláře pojmenovány „Vstupní“ a „Výstupní“

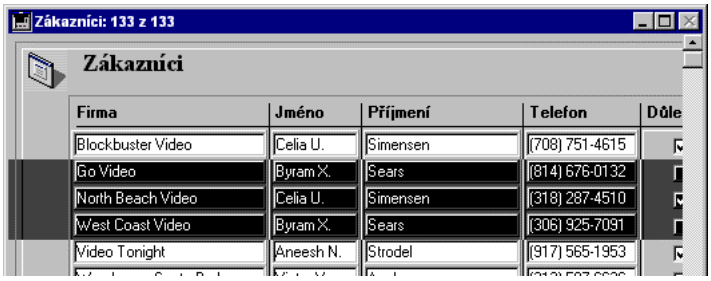
1. Přejmenujte formulář [Zákazníci];"Výstupní2" na [Zákazníci];"Výstupní"
2. Vymažte formulář nazvaný [Faktury];"Výstupní"
3. Přejmenujte formulář [Faktury];"Výstupní2" na [Faktury];"Výstupní"

7.25. Použití sady UserSet

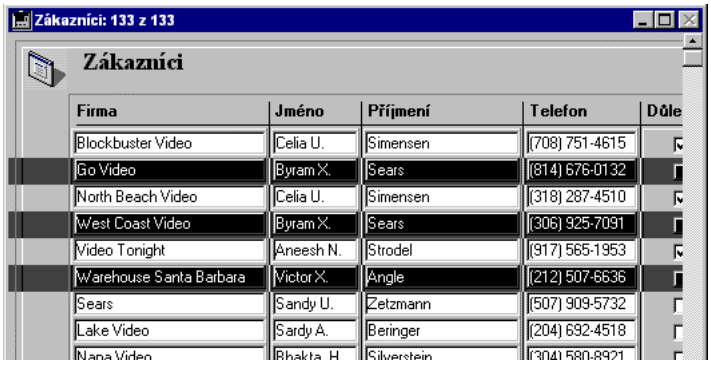
Sady (Set) jsou způsobem jak si zapamatovat, změnit či obnovit platný výběr. Sada je soubor bitů, který určuje jestli je některý záznam databáze ve výběru přítomen či ne. Pokud je hodnota bitu 0 znamená to, že záznam není přítomen, pokud je hodnota bitu 1 znamená to, že je přítomen. Sada obsahuje počet bitů, který se rovná počtu záznamů příslušné tabulky. Sady lze vytvářet programovacím jazykem. Ve 4D je používáno jako určité systémové sady několik speciálních sad. Jednou z těchto sad je sada nazvaná UserSet. Tato sada obsahuje ty záznamy, které uživatel vybral (vysvítil) ve výstupním formuláři. Uživatel může vybrat záznamy třemi rozdílnými způsoby:



Klepnutím na jeden záznam.



Shift + klepnutí na první a poslední záznam výběru.



Ctrl (apple) + klepnutí na několik oddělených záznamů.

Když sadu použijete (Use set), změníte platný výběr tabulky. Když použijete sadu nazvanou UserSet, změníte platný výběr na pouze ty záznamy, které byly uživatelem vybrány. Všimněte si dále, že v kroku Použití sady se nezadává název tabulky. UserSet odpovídá vždy té tabulce jejíž formulář je právě zobrazen na obrazovce.

Ve spojení s položkou nabídky Dotazy ¬ Vybrat označené, můžete s pomocí UserSet změnit platný výběr v Prostředí uživatele.

7.26. Proměnné lokální versus procesu

Proměnné můžete pojmenovat jakkoliv chcete, ale musíte dodržet tato pravidla.

- Maximální počet znaků v názvu proměnné je 31.
- Znak dolaru ($), kterým začíná název proměnné určuje lokální proměnnou.

Lokální proměnná je proměnná, která zmizí, když je metoda objektu či projektu, která tuto proměnnou používá ukončena. Při ukončení metody je vymazána jak hodnota proměnné tak i samotná proměnná a její definice. Paměť (RAM) používaná touto proměnnou je uvolněna pro další užití.

Vedle lokálních proměnných rovněž existují proměnné procesu, které přežijí dobu vykonání metody. Jakákoliv další metoda může být přístup do nebo provádět změny v těchto proměnných procesu.

Proměnné procesu ()

Proměnné lokální ($)

Typ proměnné	Začíná jsem	Životnost	Délka názvu
Proměnné procesu		Pracovní sekce	31 znaků
Lokální proměnné	$	Během metody	$ + 31 znaků

7.27. Konvence názvu pro proměnné

Zrovna tak jako pro metody a funkce, je pro proměnné důležité zavedení konvence v názvech. ACI explicitně nezavádí žádnou konvenci názvů, uvedeme zde však jednu, která je běžně používána v různých publikacích a při školeních. Všechna písmena jsou malá kromě L, které označuje Long integer, které by při zavedení malého l bylo zaměňováno s 1 (číslice 1).

Všechna array začínají „a“. Dvourozměrná array začínají „a2“ nebo „aa“.

Proměnné mají následující předpony, které jsou rovněž používány pro array po počátečním „a“.

Typ	Proměnná	1D Array	2D Array
Řetězec/String	s	as	a2s
Text	t	at	a2t

Integer	Ne	ai	a2i
Longint	L	aL	a2L
Real	r	ar	a2r
Datum	d	ad	a2d
Hodiny/Čas	h	Ne	Ne
Logické/Flag	f	af	a2f
Obrázek/Grafika	g	ag	a2g
Ukazatel/Pointer	p	ap	a2p

7.28. Aktivní objekty

Aktivní objekty jsou téměř vždy číselnými objekty. Pro odlišení se však všeobecně u těchto objektů používají vlastní předpony.

Objekt	Předpona
Tlačítko/Button	b
Zaškrtávací políčko/Checkbox	ck
Teploměr	th
Volič/Radio button	rb1; rb2; rb3; sb1; sb2; sb3, etc.

7.29. Deklarování proměnných

4D automaticky typuje všechny proměnné, které použijete. Proměnným je při automatickém typování přidělen nejvyšší možný typ dané skupiny. To znamená, že pro čísla bude použit typ Real, pro proměnné, které používají znaky bude použit typ Text. Použití automatického typování proměnných proto nemusí být vždy výhodné z hlediska paměti a rychlosti. Proto, i když neplánujete kompilovat váš program měli by jste určit jaký typ chcete, aby 4D přidělila vašim proměnným. Provádí se to speciálními deklaračními příkazy kompilátoru pro každou použitou proměnnou. Dokonce, i když databáze není zkompilovaná použije 4D pro proměnné typy, které byly deklarovány.

7.29.1. Přiřazení metody k položce nabídky Záznamy ¬ Vybrat označené

1. Vytvořte metodu projektu nazvanou M_GEN_ShowSubset následujícím způsobem:

```
If (False)
      ` Metoda: M_GEN_ShowSubset
      ` Kurs ACI University
      ` Generická procedura
      ` Vytovřeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Omezí platný výběr na záznamy vysvícené uživatelem.

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($LRecordsInSet)

$LRecordsInSet := Records in set ("UserSet")

If ($LRecordsInSet > 0)
   USE SET ("UserSet")
   WIN_OutputWindowTitle
Else
   ALERT ("Nejdřív musíte záznamy vybrat!")
End if
   ` Konec metody
```

2. Přejděte do prostředí Vlastní nabídky a otestujte tuto metodu.

7.30. Opačné použití sady UserSet

Dále použijete koncept sady UserSet k vytvoření položky nabídky nazvané Vynechat označené. Tato položka nabídky odečte z platného výběru vysvícené záznamy a ponechá v něm pouze záznamy nevysvícené.

Tato metoda bude vyžadovat trošku více programování než předchozí cvičení. Nejdříve musíte vytvořit sadu obsahující celý platný výběr, pak jej porovnat se sadou UserSet, aby jste zjistili, které záznamy nebyly uživatelem vybrány. Toto porovnání bude provedeno příkazem DIFFERENCE. Tento příkaz používá dvě sady. Zkontroluje, které záznamy jsou v druhé sadě společné s první sadou, tyto záznamy nalezené v druhé sadě vezme, vyjme je z první sady a z výsledku vytvoří třetí sadu pod názvem, který určíte.

Po skončení práce se sadami by jste měli všechny použité sady vymazat a uvolnit paměť. Nepotřebujete však mazat sadu UserSet, protože tuto sadu za vás kompletně ošetřuje 4D.

Pamatujte si, že v příkazu CREATE SET (vytvořit sadu) musíte určit pro, kterou tabulku je tato sada vytvářena.

7.30.1. Vytvoření metody pro položku nabídky Záznamy ¬ Vynechat označené

1. Vytvořte metodu projektu nazvanou M_GEN_OmitSubset následujícím způsobem:

```
If (False)
      ` Metoda: M_GEN_OmitSubset
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Omezí platný výběr na záznamy Nevybrané uživatelem.

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($LRecordsInSet)

$LRecordsInSet := Records in set ("UserSet")

If ($LRecordsInSet > 0)
   CREATE SET (pTable->; "SadaPuvodnihoVyberu")
   DIFFERENCE ("SadaPuvodnihoVyberu "; "UserSet"; "SadaNevybrana")
   USE SET ("SadaNevybrana ")
   CLEAR SET ("SadaNevybrana ")
   CLEAR SET ("SadaPuvodnihoVyberu ")
   WIN_OutputWindowTitle
Else
```

```
    ALERT ("K vynechání musíte záznamy nejdříve vybrat!")
End if
  ` Konec metody
```

2. Přejděte do prostředí Vlastní nabídky a otestujte tuto metodu.

Kvíz

- Jaké jsou klady a zápory pojmenování položek nabídky Vybrat označené a Vynechat označené?
- Kdy by jste měli mazat UserSet pomocí příkazu CLEAR SET?

7.31. Položka nabídky Vymazat vybrané

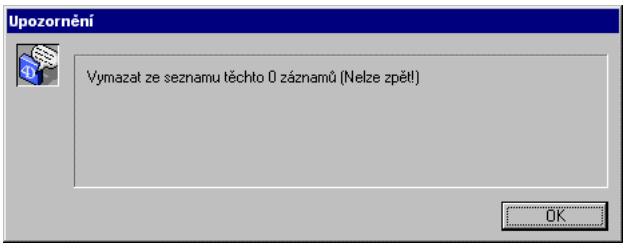
Stále ještě nemůžeme mazat jakýkoliv záznam z databáze. Vymazání záznamy by mělo být nejošetřenější úlohou v databázi, aby se zabránilo náhodnému vymazání záznamu. Nyní, když již znáte koncepci sad. Máte dost znalostí na to přidat pro uživatele rys, který bude provádět mazání pouze vybraných záznamů. Aby jste to mohli provést musíte sledovat platný výběr a sadu UserSet předtím než začnete mazání. Pak lze použít UserSet k omezení platného výběru na pouze vybrané záznamy. Po vymazání záznamů bude platný výběr prázdný. K zobrazení zbývajících záznamů z původního platného výběru použijete původní sadu.

7.32. Vymazání NĚKTERÝCH záznamů v sadě

Samozřejmě některé záznamy původní sady byly vymazány. Naštěstí užití sady s vymazanými záznamy není problém. 4D automaticky ignoruje vymazané záznamy a zobrazí pouze zbylé záznamy sady.

7.33. Co když nejsou žádné záznamy k mazání?

Předpokládejme, že uživatel klepl na tlačítko vymazat, ale nevybral žádné záznamy. Vaše databáze by měla zobrazit následující zprávu: „Vymazat ze seznamu těchto 0 záznamů (Nelze zpět!)“. Uživatel očekává od vašeho software inteligentnější chování. Místo toho můžete zobrazit dialog, který uživateli oznámí, že nebudou mazány žádné záznamy, protože žádné nevybral.



7.33.1. Přiřazení metody k položce nabídky Záznamy ¬ Vymazat vybrané…

1. Vytvořte metodu projektu nazvanou M_GEN_DeleteUserSet následujícím způsobm:

```
If (False)
      ` Metoda: M_GEN_DeleteUserSet
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Vymaže záznamy vybrané uživatelem.

   <>fGeneric := False
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($LRecordsInSet)

$LRecordsInSet := Records in set ("UserSet")

If ($LRecordsInSet > 0)
   CREATE SET (pTable->; "SadaPuvodniVyber")
   USE SET ("UserSet") ` Změní platný výběr na vybrané.
   DELETE SELECTION (pTable->)     ` Omezí platný výběr na prázdný
      ` Navrátí původní výběr kromě vymazaných
   USE SET ("SadaPuvodniVyber ")
   CLEAR SET ("SadaPuvodniVyber ")`
   WIN_OutputWindowTitle
Else
   ALERT ("K vymazání musíte záznamy nejdříve vybrat!")
End if
   ` Konec metody
```

2. Jděte do prostředí Vlastní nabídky a otestujte tuto metodu.

7.34. Příkaz CONFIRM

Uživatel by měl dvakrát kontrolovat akce, které znamenají ztrátu dat. Takže před tím než budete mazat záznamy, by jste měli zobrazit dialog CONFIRM.

Příkaz CONFIRM pouze zobrazí dialog s vámi určeným textem a dotáže se uživatele. Uživatel klepne buď na tlačítko OK nebo Zrušit.

7.35. Dialogy by měly poskytovat užitečné informace

Dialog Confirm dovolí uživateli vybrat zda opravdu vymazat záznamy nebo ne. To však ještě není dostatečné. Např. uživatel může předpokládat, že pouze maže záznamy, které vidí v zobrazené části okna a přitom by mazal záznamy z celého výběru (můžou to být i všechny záznamy tabulky). Jestliže dialog Confirm bude obsahovat i informaci o tom kolik záznamů se bude mazat, dostane uživatel mnohem jednoznačnější informaci o tom co se provede. Tyto požadované informace můžete uživateli poskytnou jednoduše s pomocí funkce Records in set. Jak již název napovídá tato funkce navrací číslo, které znamená počet záznamů v sadě pro tabulku a sadu, kterou určíte. Hodnotu tohoto čísla můžete použít ve své zprávě Confirm.

7.36. Použití proměnných k zpřehlednění vašeho kódu

Některé řádky kódu, které napíšete mohou obsahovat několik příkazu 4D a funkcí v jednom řádku kódu. Příklad dialogu confirm je jedním z těchto příkladů. Aby jste sdělili uživateli kolik záznamů se bude mazat, potřebujete přidat do textu v příkazu CONFIRM funkci Records in set a to vše na jednom řádku. Výsledný kód by vypadal následovně:

```
CONFIRM ("Vymazat "+ String (Records in set ("UserSet"))+" záznamů?")
```

Kód napsaný tímto způsobem může být trochu matoucí, jestliže jste úplnní začátečníci v programování. Aby jste si daný kód zpřehlednili, můžete použít proměnné k uložení některých hodnot, které vám dovolí rozdělit kód do více řádek. V předchozím příkladu hodnota navracená funkcí Records in set, předávaná spolu s funkcí String a jí navracenou hodnotou textu, může být jako textová proměnná předána příkazu CONFIRM. K zpřehlednění kódu přesuneme funkci Records in set a vše co je do ní předáváno do řádky před příkaz CONFIRM. Hodnota navracená funkcí Records in set bude přiřazena proměnné, která pak bude předána funkci String na dalším řádku.

```
C_LONGINT ($LRecordsInSet)
$LRecordsInSet := Records in set ("UserSet")
CONFIRM ("Vymazat "+ String ($LRecordsInSet)+" vybraných záznamů?")
```

Hodnota uložená v proměnné $LrecordsInSet, kterou jsme použili v tomto příkladu, je potřeba pouze pro tuto metodu. Po odstranění a ukončení metody, bude proměnná $LRecordsInSet z paměti odstraněna. Proměnné začínající znakem dolar ($) jsou po ukončení metody odstraňovány z paměti automaticky. Při zavedení metody (jejím započetí) jsou těmto proměnným

přiřazovány nulové hodnoty. Tyto proměnné se nazývají lokální, protože jsou „lokální“ pouze danou metodu.

7.37. Optimalizace metody

Všimněte si, že bez přiřazení hodnoty navracené funkcí Records in set do proměnné, bude tato funkce v metodě z předchozí kapitoly prováděna dvakrát. Nahrazení tohoto zdvojeného místa pomocí přiřazení do proměnné, umožní 4D provést metodu o něco rychleji, protože nebude muset vyhledat v knihovně příkazu tento příkaz dvakrát.

7.37.1. Přidání CONFIRM do metody položky nabídky Záznamy ¬ Vymazat vybrané…

1. Upravte metodu projektu M_GEN_DeleteUserSet následovně:

```
If (False)
   ` Metoda: M_GEN_DeleteUserSet
   ` Kurs ACI University
   ` Generická procedura
   ` Vytvořeno: Jim Steinman
   ` Datum: 15/1/97

   ` Účel: Vymaže záznamy vybrané uživatelem.

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT ($LRecordsInSet)

$LRecordsInSet := Records in set ("UserSet")

If ($LRecordsInSet > 0)
•     CONFIRM ("Vymazat "+ String($LRecordsInSet)+" vybraných záznamů?")
•     If (OK = 1)                          ` Jestliže uživatel klepne OK v dialogu Confirm.
         CREATE SET (pTable->; "SadaPuvodniVyber")
      USE SET ("UserSet") ` Změní platný výběr na vybrané.
      DELETE SELECTION (pTable->)     ` Omezí platný výběr na prázdný
         ` Navrátí původní výběr kromě vymazaných
      USE SET ("SadaPuvodniVyber ")
      CLEAR SET ("SadaPuvodniVyber ")`
         WIN_OutputWindowTitle
•     End if
Else
   ALERT("K vymazání musíte záznamy nejdříve vybrat!")
End if
   ` Konec metody
```

2. Jděte do prostředí Vlastní nabídky a otestujte tuto metodu.

7.38. Přidání funkcí s použitím sad

Za pomoci sad můžete vylepšit některé již vytvořené metody. Vezměme např. smyčku Přidat záznam. Můžeme rozšířit platný výběr zobrazený na konci sekce přidávání záznamů o všechny přidané záznamy.

7.38.1. Rozšíření metody pro přidání záznamu.

1. Přepište metodu projektu M_GEN_AddRecords následujícím způsobm:

```
If (False)
   ` Metoda: M_GEN_AddRecords
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Přidat záznamy do tabulky.

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT ($LRecordsInSet)

CREATE SET(pTable->;"SadaPuvodniVyber")
CREATE EMPTY SET (pTable->;"SadaNoveZaznamy")
Repeat
   ADD RECORD(pTable->)
   If (OK=1)
      ADD TO SET (pTable->;" SadaNoveZaznamy ")
   End if
Until (OK=0)

$LRecordsInSet := Records in set ("SadaNoveZaznamy ")

If ($LRecordsInSet > 0)
   CONFIRM("Právě jste přidali  "+String($LRecordsInSet ) + " nových záznamů.";
                        "Ukázat pouze nové";"Ukázat předchozí plus nové")
   If (OK=1)
      USE SET("SadaNoveZaznamy ")
   Else
      UNION (" SadaPuvodniVyber ";" SadaNoveZaznamy ";"UnionSet")
      USE SET("UnionSet")
      CLEAR SET("UnionSet")
   End if
   CLEAR SET("SadaPuvodniVyber ")
   CLEAR SET("SadaNoveZaznamy ")
End if
WIN_OutputWindowTitle
      ` Konec metody
```

2. Přejděte do prostředí Vlastní nabídky a otestujte tuto metodu.

8. Přidání dalších tabulek do systému nabídek

8.1. Přidání faktur do nabídky tabulek

Nyní, když položka nabídky Soubor ¬ Zákazníci pracuje dobře, můžete přidat novou položku nabídky k přístupu do tabulky [Faktury].

8.2. Přidání nezbytných formulářů

Předtím než poprvé použijeme nabídku Soubor pro záznamy [Faktury], musíme vytvořit patřičné formuláře. Napsali jste určité generické metody, které pracují s libovolnou tabulkou za pomoci proměnné ukazatele pTable. Tyto metody však očekávají určité již existující formuláře. Především potřebujete formuláře nazvané „Vstupní“, „Výstupní“, „Dotazy“ a „ImportExport“. Potřebujete se přesvědčit, že tabulky [Faktury] a [Produkty] obsahují tyto čtyři formuláře.

8.2.1. Prověření, že tabulky [Faktury] a [Produkty] obsahují nezbytné formuláře

1. Ve struktuře, v okně Průzkumníka, na stránce Formuláře, klepněte na trojúhelník vedle názvu faktury.
2. Prověřte, že existují formuláře Vstupní, Výstupní, Dotazy a ImportExport.
3. Jestliže je to nezbytné, vytvořné chybějící formuláře.
4. Zopakujte tento postup pro tabulku [Produkty].

Nyní, když formuláře existují můžeme pokračovat v přidání položky nabídky pro [Faktury] v nabídce Soubor.

8.2.2. Přidání položky nabídky Soubor ¬ Faktury

1. Vytvořte metodu M_Invoices následujícím způsobem:

```
If (False)
      ` Metoda: M_Invoices
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Zobrazí výstupní formulář se seznamem záznamů [Faktury].

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

pTable := -> [Faktury]

MENU BAR (3) `                    Přepne záhlaví nabídek.
```

```
ALL RECORDS  (pTable->) `       Změní platný výběr.
WIN_OutputWindowTitle
MODIFY SELECTION (pTable->; *) ` Zobrazí záznamy na obrazovku.
MENU BAR (1) `                  Přepne zpět na Záhlaví #1.
   ` Konec metody
```

8.3. Vytvoření databáze metod MODIFY SELECTION

Každá z metod M_table je nyní nezbytně kopií části kódu. Nejúčinnější způsob použití této části kódu by byl ve vytvoření nové metody projektu, která bude obsahovat tento duplikovaný kód a bude volána z metody M_table.

8.3.1. Vytvoření metody projektu pro MODIFY SELECTION

1. Vytvořte metodu projektu nazvanou GEN_ModifySelection následujícím způsobem:

```
If (False)
      ` Metoda: M_Invoices
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Zobrazí výstupní formulář se seznamem záznamů tabulky pTable.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if


MENU BAR (3) `                  Přepne záhlaví nabídek.
ALL RECORDS  (pTable->) `       Změní platný výběr.
WIN_OutputWindowTitle
MODIFY SELECTION (pTable->; *) ` Zobrazí záznamy na obrazovku.
MENU BAR (1) `                  Přepne zpět na Záhlaví #1.
   ` Konec metody
```

8.3.2. Úprava metody M_Customers pro použití generického kódu GEN_ModifySelection

1. Upravte metodu projektu M_Customers následujícím způsobem:

```
If (False)
      ` Metoda: M_Customers
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Nastaví  pTable na [Zákazníci] pro užití ve všech metodách
```

```
      ` a volá výstupní formulář.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

pTable := ->[Zákazníci]
```

- ~~MENU BAR (3)~~
- ~~ALL RECORDS (pTable >)~~
- ~~WIN_OutputWindowTitle~~
- ~~MODIFY SELECTION (pTable >; *)~~
- ~~MENU BAR (1)~~
- GEN_ModifySelection

```
   ` Konec metody
```

8.3.3. Upravte metodu M_Invoices pro užití generického kódu GEN_ModifySelection

1. Upravte metodu projektu M_Invoices následujícím způsobm:

```
If (False)
    ` Metoda: M_Invoices
    ` Kurs ACI University
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Nastaví  pTable na [Faktury] pro užití ve všech metodách
    ` a volá výstupní formulář..

  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if

pTable := ->[Faktury]
```

- GEN_ModifySelection
  ` Konec metody

8.3.4. Přidání metody pro tabulku [Produkty]

1. Vytvořte metodu projektu nazvanou M_Products.

```
If (False)
    ` Metoda: M_Products
    ` Kurs ACI University
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Nastaví  pTable na [Produkty] pro užití ve všech metodách
    ` a volá výstupní formulář..

  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if

pTable := ->[Produkty]

GEN_ModifySelection
  ` Konec metody
```

2. Jděte do prostředí Vlastní nabídky a otestujte tuto metodu.

8.4. Zajištění použití správných formulářů Vstupní a Výstupní

Je možné, že z nějakých důvodů mohou být v jiných částech programu změněny výchozí vstupní a výstupní formuláře. Bude tedy nejlepší pokud při vstupu do agendy tabulky zajistíme, že budou vždy používány správné vstupní a výstupní formuláře. Z tohoto důvodu přidáme jejich definici do vstupní metody.

8.4.1. Úprava metody GEN_ModifySelection k zajištění správných formulářů

1. Upravte metodu projektu GEN_ModifySelection následujícím způsobem:

```
If (False)
    ` Metoda: M_Invoices
    ` Kurs ACI University
    ` Generická procedura
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Zobrazí výstupní formulář se seznamem záznamů tabulky pTable.

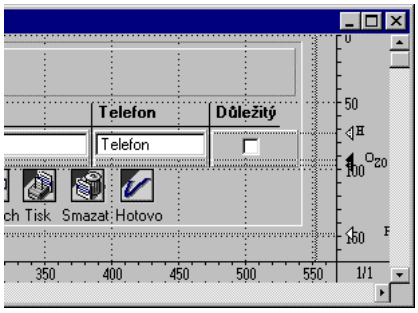
  <>fGeneric := True
  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if


MENU BAR (3) `                  Přepne záhlaví nabídek.
ALL RECORDS  (pTable->) `       Změní platný výběr.
WIN_OutputWindowTitle
INPUT FORM (pTable->; "Vstupní")        ` Zajistí správné formuláře
OUTPUT FORM (pTable->; "Výstupní")
MODIFY SELECTION (pTable->; *) ` Zobrazí záznamy na obrazovku.
MENU BAR (1) `                  Přepne zpět na Záhlaví #1.
  ` Konec metody
```

8.5. Řízení výšky zápatí

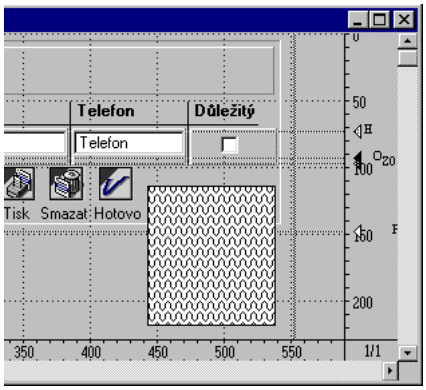
Řídící čára P ve výstupním formuláři určuje množství místa, které uvidíte na spodu okna při zobrazení formuláře v prostředí Vlastní nabídky. Jestliže potáhnete čáru P dolů, vytvoříte více místa pod posledním záznamem.

Tip: Potáhnout čáru lze uchopením trojúhelník vedle písmena P.



Formulář původně vypadal jako na obrázku výše. Zápatí formuláře má velikost, která odpovídá počtu bodů mezi čarami Z0 (zlom úrovně nula) a P. Jestliže je tato vzdálenost např. 4 body, má zápatí velikost 4 body.

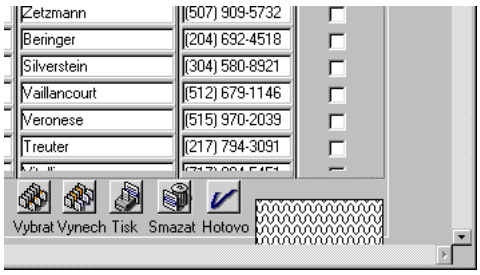
Vyzkoušejte si změny velikosti zápatí a zkontrolujte výsledek v prostředí Vlastní nabídky. Povšimněte si, že vše co je umístěno pod čárou P není v okně prostředí Vlastní nabídky vidět. Oblast pod čárou P, pod zápatím můžete tedy použít jako jistý druh schránky pro ukládání záložních obrázků, speciálních tlačítek atd.



8.5.1. Změny velikosti zápatí

1. Otevřete výstupní formulář [Zákazníci];“Výstupní“
2. Pod čárou zápatí nakreslete obdelník a vyplňte jej libovolným vzorem.

3. Potáhněte čáru P tak, aby přetínala vytvořený obdelník.
4. Přejděte do prostředí Vlastní nabídky a prohlédněte si nové zápatí se zobrazenou částí obdelníku.



8.6. Obrázková tlačítka ve výstupním formuláři

Mnoho lidí dává přednost tlačítkům ve spodu formuláře. Tato tlačítka obvykle vykonávají nejčastěji používané akce. Popisná tlačítka s textem však často zabírají příliš mnoho místa na obrazovce. Protože obrázek je obvykle přehlednější a lepší než popis dlouhým textem, můžeme použít obrázky umístěné nad zvýrazněná tlačítka, které nám zajistí tytéž úlohy jako textová tlačítka.

8.7. Obrázky zabírají paměť

Vícenásobné použití téhož obrázku v mnoha formulářích, může zabrat mnoho paměti. Kromě toho, že každý obrázek na formuláři je chápán 4D jako oddělený objekt, zabírá výpočet výsledného umístění obrázku na formuláři strojový čas. Jedna z metod, která šetří paměť a rychlost vybavování obrázků, je uložení obrázků ve zdrojích a přenesení těchto zdrojů do obrázkových proměnných. Jedna proměnná může být pak zobrazena v mnoha rozdílných formulářích a tento přístup ušetří paměť v každém formuláři, kde je použit. Kromě toho řady obrázků jako paleta tlačítek, mohou být sloučeny do jednoho obrázkového objektu a ušetříme tak i strojový čas na výpočet pozice v obrázku a vybavení formuláře.

8.8. Přiřazení klávesových zkratek tlačítkům

Časté přecházení z klávesnice k myši a naopak obvykle zpomaluje uživatele a užívání programu. Proto pro ty, kteří užívají raději klávesnici než myš, můžeme k tlačítkům přiřadit klávesové zkratky po jejichž stisknutí je provedena akce tlačítka. Prověřte si klávesové zkratky přiřazené tlačítkům ve výstupním formuláři.

8.9. Použití tlačítek uschovaných ve formuláři pod zápatím pro dodatečné klávesové zkratky

Uživatelé se často pokoušejí použít klávesové zkratky, které jste k tlačítkům nepřiřadili. Proto pro některé běžně používané klávesové zkratky (např. pro jiné platformy) můžete svá tlačítka

zduplikovat pod zápatí formuláře a přiřadit jim jiné (dodatečné) klávesové zkratky, aby jste tak pokryli všechny klávesové zkratky, které mohou uživatelé napadnout.

8.9.1. Přidání obrázkových tlačítek do výstupních formulářů

1. Otevřete formulář [Zákazníci];"Výstupní".
2. Odstraňte všechna tlačítka formuláře.
3. Otevřete formulář [zDialogy];"TlačítkaVýstupní": vyberte všechna tlačítka, překopírujte je do schránky a vložte je do formuláře [Zákazníci];"Výstupní".
4. Přemístěte čáru zápatí pod obrázková tlačítka.
5. Tento postup zopakujte pro formuláře [Produkty];"Výstupní" a [Faktury];"Výstupní".
6. Přejděte do prostředí Vlastní nabídky a vyzkoušejte je.

8.10. Přidání schopnosti tisku do výstupního formuláře

Tlačítka, která jsme právě vložili obsahují tlačítko pro tisk z výstupního formuláře. Jestliže však budeme jednoduše tisknout, budeme tisknout tento výstupní formulář včetně tlačítek naspodu formuláře. Proto musíme vytvořit formuláře, které budou při tisku platného výběru použity. Bude to jednoduchý formulář, který vypadá stejně jako výstupní formulář, ale bez tlačítek.

8.10.1. Vytvoření formuláře k tisku platného výběru ve výstupním formuláři

1. Vytvořte formulář [Zákazníci];"TiskVýstup" a použijte typ formuláře S obsahem pro tisk a vzor ACI Video Výstupní.
2. Vyberte všechny položky tohoto formuláře a vymažte je.
3. Otevřete formulář [Zákazníci];"Výstupní".
4. Vyberte všechny položky na formuláři a překopírujte je do schránky. Uzavřete formulář [Zákazníci];"Výstupní".
5. Vložte položky ze schránky do formuláře [Zákazníci];"TiskVýstup".
6. Se stisknutým Shift klepněte na všechna tlačítka a vymažte je.
7. Se stisknutým Shift klepněte na všechny obrázky tlačítek a vymažte je.
8. Se stisknutým tlačítkem myši nakreslete obdelník kolem zbývajících objektů pozadí v zápatí formuláře a vyberte je.
9. Se stisknutým tlačítkem Ctrl a šipka nahoru, zmenšete objekty pozadí tak, aby spodní okraj byl těsně pod čárou Z0.
10. Přemístěte čáru zápatí těsně pod čáru zlomu.
11. Zopakujte kroky 1 - 10 pro tabulku [Faktury].
12. Zopakujte kroky 1 - 10 pro tabulku [Produkty].
13. Přiřaďte formát zobrazení částka k polím ceny a dalším polím, které zobrazují finanční částku.

8.10.2. Vytvoření metody projektu pro položku nabídky Speciální zpráva

1. Vytvořte metodu projektu nazvanou M_SpecialReports následujícím způsobem:

```
If (False)
    ` Metoda: M_SpecialReports
    ` Kurs ACI University
    ` Generická procedura
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Tiskne platný výběr záznamů pro pTable
    ` na formuláři TiskVýstup.

  <>fGeneric := True
```

```
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

OUTPUT FORM (pTable->; "TiskVýstup")
PRINT SELECTION (pTable->)
OUTPUT FORM (pTable->; "Výstup")
   ` Konec metody
```

8.11. UserSet může způsobit problémy.

Předpokládejme, že uživatel vybere některé záznamy a pak aniž provede jakoukoliv změnu ukončí agendu a přepne se do jiné tabulky. V této tabulce jako první akci vybere Vymazat označené. Co se stane?

Jestliže jste odhadovali, že záznamy odpovídající číslům záznamů ze sady UserSet předchozí tabulky, budou vymazány, měli jste pravdu.

Sada UserSet nezáleží na vybrané tabulce. Proto, aby jsme se ujistili, že uživatel vstupuje do agendy vždy s prázdnou sadou UserSet, přidáme jednu řádku kódu s příkazem CREATE EMPTY SET, který zajistí vyprázdnění této sady.

8.11. Zneškodnění UserSet.

1. Upravte metodu projektu GEN_ModifySelection následujícím způsobem:

```
If (False)
      ` Metoda: M_Invoices
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Zobrazí výstupní formulář se seznamem záznamů tabulky pTable.

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if


MENU BAR (3) `                  Přepne záhlaví nabídek.
ALL RECORDS  (pTable->) `       Změní platný výběr.
CREATE EMPTY SET(pTable->; "UserSet")
WIN_OutputWindowTitle
INPUT FORM (pTable->; "Vstupní")          ` Zajistí správné formuláře
OUTPUT FORM (pTable->; "Výstupní")
MODIFY SELECTION (pTable->; *) `  Zobrazí záznamy na obrazovku.
MENU BAR (1) `                    Přepne zpět na Záhlaví #1.
```

```
` Konec metody
```

9. Metody objektu

Pole [Zákazníci]Stát je zobrazováno ve vstupním formuláři tabulky [Zákazníci]. Můžete napomoci uživateli tak, že 4D bude automaticky konvertovat vše co zde uživatel napíše do velkých písmen. Aby jsme to provedli, napíšeme pro toto pole metodu objektu.

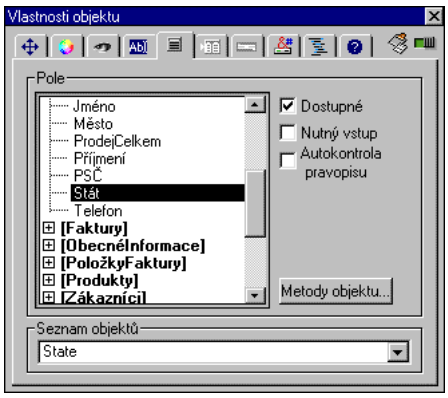
9.1. Kdy se spouštějí metody objektu?

Metoda objektu je sada instrukcí, která přesně říká 4D co má dělat krok po kroku. Metoda objektu může být spuštěna vícenásobně: když uživatel opouští pole po zadání něčeho, když uživatel stiskne libovolnou klávesu, když uživatel klepne na objekt, když objekt ztrácí fokus atd. Je ještě více příležitostí, kdy metoda objektu může být spuštěna. Některé z těchto příležitostí tzv. událostí formuláře v tomto kursu ještě využijeme.

Jestliže objekt jako např. pole obsahuje přiřazenou metodu objektu, je tento objekt zobrazen s černým troúhelníkem v levém horním rohu.

9.2. Přidání metody objektu

Otevřete formulář [Zákazníci];“Vstupní“ a poklepejte na pole Stát. Uvidíte následující dialog:



V dialogovém okně Vlastnosti objektu, přejděte na stránku Události (druhá záložka zprava) a klepněte na tlačítko Metoda objektu... Zobrazí se dialog pro typ metody.

V tomto kursu budeme používat pouze textové metody. Pokud je metoda objektu vytvořena Editorem vývojových diagramů, nemůžete ji již do textové formy konvertovat (a naopak), i když ji samozřejmě můžete vymazat a vytvořit znova v jiném typu.

Ujistěte se, že je vybrán typ metody Textová a klepněte na tlačítko OK.

9.3. Přiřazení nových hodnot polím

V následujícím cvičení se chystáme provést:

- Vytvořit kopii hodnoty pole [Zákazníci]Stát (ať již uživatel napíše do tohoto pole cokoliv).
- Zkonvertovat tuto zkopírovanou hodnotu do velkých písmen (“čr” se stane “ČR”).
- Vložit tuto hodnotu nyní velkými písmeny zpět do pole [Zákazníci]Stát a nahradit tak původní hodnotu napsanou uživatelem.

K tomuto využijeme funkci Uppercase, která provede konverzi do velkých písmen a operátor přiřazení, který vezme hodnotu čehokoliv co je na pravé straně a zkopíruje či přiřadí tuto hodnotu do čehokoliv co je na levé straně.

9.3.1. Vytvoření metody objektu k převodu do velkých písmen

1. Otevřete vstupní formulář pro [Zákazníci].
2. Poklepejte na pole [Zákazníci]Stát.
3. Přejděte na stránku události.
4. Odškrtněte všechny zaškrtnuté události.
5. Zaškrtněte událost Při změně
6. Klepněte na tlačítko Metoda objektu.
7. Ujistěte se, že je zvolena textová metoda a klepněte na tlačítko OK.
8. Do první řádky metody objektu napište:

```
[Zákazníci]Stát := Uppercase ([Zákazníci]Stát)
```

9.4.1. Testování metody objektu Uppercase

1. Vyberte Prostředí ¬ Uživatele (_ + U) (Ctrl + U).
2. Přepněte se do tabulky [Zákazníci]
3. Vyberte Vstup ¬ Nový záznam (_ + N) (Ctrl + N).
4. Klepněte na pole Stát.
5. Napište zkratku státu malými písmeny jako např čr.
6. Stiskněte Tab a sledujte pole Stát.

9.5. Psaní komentářů

O psaní komentářů metodám objektů platí totéž co bylo řečeno pro psaní komentářů metod projektu. Každá řádka komentáře musí začínat znakem (`)

9.5.1. Přidání komentáře do metody objektu

1. Přidejte do metody objektu následující komentář:

```
` METODA OBJEKTU: [Zákazníci]Stát.
` VYTVOŘENO: Jim Steinman. 15/1/97.
` ÚČEL: Tato metoda objektu konvertuje každý vstup dat do tohoto pole do velkých písmen.

[Zákazníci]Stát := Uppercase ([Zákazníci]Stát)

`Konec metody objektu.
```

9.6. Určení při které události je metoda objektu spuštěna

4D obsahuje mnoho událostí, při kterých může být váš kód spuštěn. Dále jsou uvedeny příklady některých událostí při kterých můžete chtít spouštět váš kód metody objektu:

Příležitost	Příklad
Při aktualizaci (formulář je používán)	Když uživatel napíše hodnotu do pole (a opouští pole), zde můžete chtít aktivovat váš kód jak jsme to učinili v poli Stát.
Při zavedení (záznam se objevuje na obrazovce)	Těsně předtím než uživatel uvidí nový záznam, zde můžete chtít vyplnit výchozí hodnotu pole, např. ČR pro Stát.

Při potvrzení vstupu (uživatel je hotov se záznamem)	Když uživatel klepne na tlačítko Přijmout k uložení záznamu, zde můžete chtít označit záznam časem, kdy byl naposled aktualizován.

Tyto příležitosti pro spuštění kódu se nazývají události formuláře.

Událost formuláře	Popis
Při aktualizaci	Uživatel právě skončil používání pole nebo tlačítka.
Při zavedení	Uživatel se chystá prohlédnout si záznam na obrazovce.
Při potvrzení vstupu	Uživatel právě uložil změněný záznam.

Nejdříve si probereme událost Při aktualizaci a dále v kursu se budeme spolu s dalšími událostmi zabývat událostmi Při zavedení a Při potvrzení vstupu.

9.7. O události Při aktualizaci

Doposud jsme viděli, že metoda objektu je spuštěna, když uživatel opouští pole po tom, kdy do něj něco napsal. Jednoduchý pohyb pomocí tabelátoru po jednotlivých polích nevyvolá spuštění této metody objektu. Spuštění metody se inicializuje až při události Při aktualizaci.

Tato událost nastane když:

- Uživatel opouští změněné pole nebo
- Uživatel klepne na tlačítko.

Jestliže chcete, aby byla metoda objektu spuštěna pouze, když uživatel změní pole tj. při události Při aktualizaci, zaškrtněte v dialogu Události, událost Při aktualizaci.

Všimněte si, že existuje ještě mnoho dalších událostí. Tyto události budou probírány v dalších kursech ACI Univerzity. V tomto kursu jich probereme pouze několik.

9.8. Chceme i další události?

Kdekoliv napíšete metodu objektu, musíte učinit důležité rozhodnutí: Kdy chci spouštět metodu objektu? Vždy když metodu napíšete musíte se zamyslet, jestli jsou ještě další příležitosti, kdy má být metoda spuštěna. V dalším cvičení uvidíte příkad na dvě další události, keré budou využity.

9.9. Příkaz ALERT

V tomto kurzu je několik příkazů, keré jsou využívány ke komunikaci s uživatelem. Jeden z prvních je upozornění – ALERT, který je používán k pouhému předání informace uživateli v dialogu. V tomto dialogu není tlačítko Storno, protože nemá smysl, uživatel nemůže zastavit běh programu, ani provést rozhodování. Je zde pouze tlačítko OK, po klepnutí na nějž program pokračuje v provádění dalších instrukcí.

9.9.1. Spuštění metody objektu během dalších událostí

1. Přejděte do prostředí návrháře
2. Otevřete vstupní formulář pro [Zákazníci].
3. Poklepejte na pole [Zákazníci]Stát.
4. Přejděte na stránku události.
5. Zatrhněte Při potvrzení vstupu a Při zavedení.
6. V editoru metod, změňte prováděcí část metody:

```
ALERT ("Metoda objektu Stát je spuštěna.")
[Zákazníci]Stát := Uppercase ([Zákazníci]Stát)
```

7. Uzavřete Editor metod.
8. Přejděte do Prostředí uživatele.
9. Vyzkoušejte metodu objektu poklepáním na záznam a úpravou pole Stát.

9.9.2. Úprava spouštění metody objektu pouze při události Při aktualizaci

1. Zopakujte kroky předchozího cvičení, ale odškrtněte události při Při potvrzení vstupu a Při zavedení.
2. Vyzkoušejte metodu objektu poklepáním na záznam a úpravou pole Stát.

9.10. Změna řádky kódu na komentář

Příležitostně můžete chtít dočasně odstranit řádku kódu z metody objektu. Pokud tuto řádku kódu převrátíte na komentář, 4D ji bude ignorovat. Pokud se rozhodnete opět tuto řádku použít, vymažete pouze znak komentáře. Tento postup se nazývá „odkomentování“ řádky. Je pouze potřeba mít na paměti, že řádky kódu mohou mít až tisíce znaků, komentář je však omezen na 80 znaků na řádek; takže buďte opatrní, když převádíte dlouhé řádky kódu do komentáře.

9.11. Vytvoření lokálního tlačítka

Dále budeme předpokládat, že jste si uvědomili, že většina vašich zákazníků je z jednoho místa. Takže než by jste požadovali po uživatelích, aby psali totéž město, stát a PSČ pro každý nový záznam, můžete přidat tlačítko, které tuto nudnou práci provede za ně.

Alernativně, ve skutečnosti, spíše než napsání metody objektu jak provedeme v dalším cvičení využijete nastavení výchozích hodnot v definici pole. To co napíšete v definici pole na stránku Řízení dat je používáno jako výchozí hodnota pro každý nový záznam. Výchozí hodnoty nemají žádný vliv na již existující záznamy. Zde volíme místo výchozích hodnot metodu objektu Tlačítka, protože vám uživatelé řekli, že jim více vyhovuje klepnutí na tlačítko než automatické výchozí hodnoty pro každý nový záznam.

9.12. Řetězce

Když píšete v Editoru metod, 4D předpokládá, že píšete názvy polí, proměnných nebo názvy příkazů programovacího jazyka. Když potřebujete komunikaci a zadat určité části textu, musíte text uzavřít do uvozovek. Např. přiřazení hodnoty „Johnson“ do pole Příjmení znamená, napsat následující kód:

```
[Zákazníci]Příjmení  := "Johnson"
```

Bez napsání uvozovek, 4D hledá pole s názvem Johnson. Pokud není takové pole nalezeno, 4D potom hledá příkaz jazyka s tímto názvem. Jestliže není splněna žádná z těchto podmínek, 4D předpokládá, že je to proměnná. Takže pokud se potřebujete odkázat na určitý konkrétní text, napište jej do uvozovek. Takovýto text se nazývá řetězec. Pro ukládání PSČ používáme pole alfa to znamená znakové pole, i když PSČ obsahuje pouze číslice. To znamená, že pokud budeme zadávat programovacím příkazem PSČ, musíme je rovněž uzavřít do uvozovek.

Část textu se nazývá řetězec, protože je to řetězec jednotlivých nezávislých písmen. Při programování by jste měli psát následovně:

Položka	Příklad
Řetězec/String	"Cupertino"
Pole	[Zákazníci]Město
Příkaz	Uppercase

9.12.1. Vytvoření tlačítka Místní

1. Jděte do Prostředí návrháře.
2. Otevřete formulář [Zákazníci];"Vstupní".
3. V paletě objektů klepněte na nástroj Tlačítka.

4. Táhněte ikonu do formuláře, k objektům polí Město a Stát.
5. Poklepejte na nové tlačítko.
6. V dialogu vlastností objektů, pojmenujte tlačítko bLocal.
7. Do textu tlačítka napište Místní.
8. Jako automatickou akci vyberte Žádná akce.
9. Klepněte na záložku Zobrazit.
10. Odškrtněte políčko Dostupné tabelátorem
11. Přejděte na stránku Události.
12. Přesvědčte se, že jediné vybrané události jsou Při klepnutí a Při poklepání.
13. Klepněte na tlačítko Metoda objektu...
14. Do metody objektu napište následující:

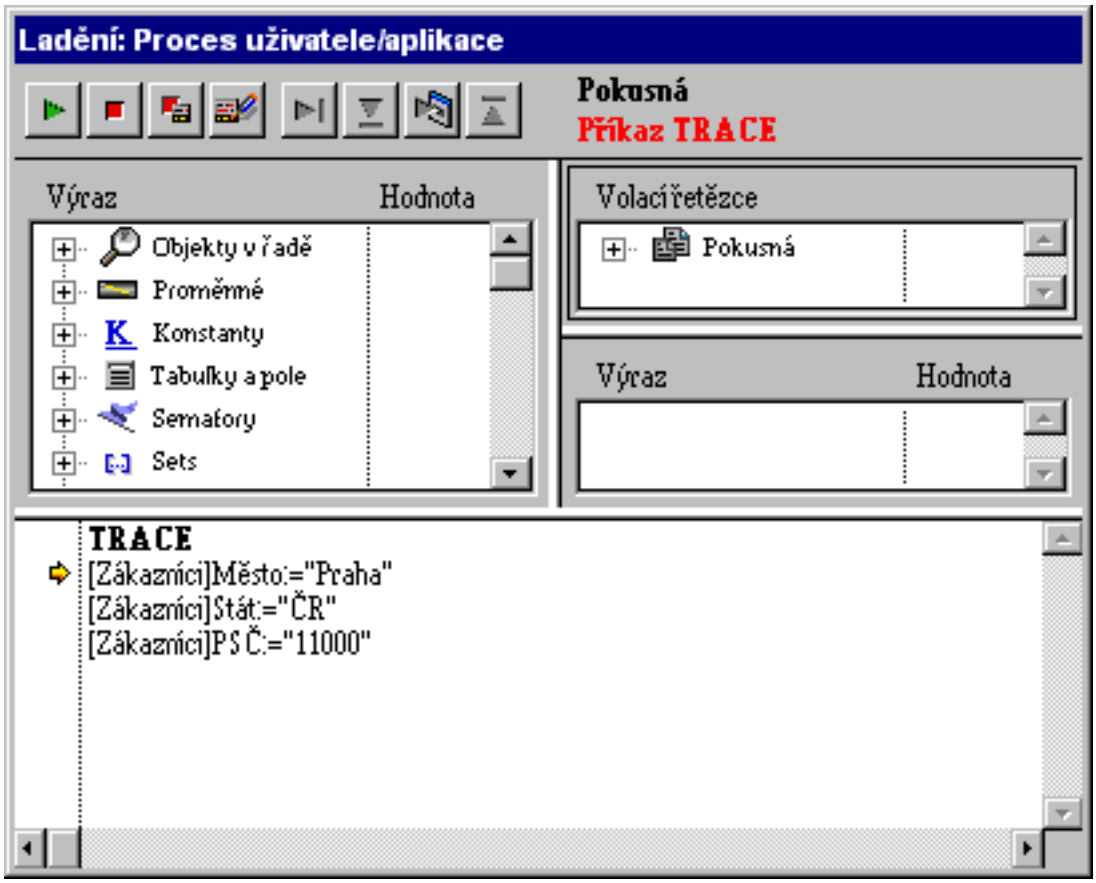
```
   ` Metoda objektu bLocal. [Zákazníci]; "Vstupní"
   ` Vytvořeno Jim Steinman. 15/1/97.

[Zákazníci]Město := "Praha 5"
[Zákazníci]Stát := "ČR"
[Zákazníci]PSČ := "15500"
   ` Konec metody objektu.
```

13. Přejděte do Prostředí uživatele a vyzkoušejte nové tlačítko.

9.13. Zkoumání běhu vašich metod objektu

Můžete napsat metodu objektu nebo jiný kód, který sice proběhne, ale neprovede přesně to co by jste očekávali. V těchto situacích bude užitečné vědět, které řádky metody objektu se spouštějí a jaké hodnoty klíčových polí a proměnných v nich jsou obsaženy. Jedině tak lépe porozumíte chování metody. Okno ladění je nástroj, který vám umožní sledovat běh vašeho programu řádek po řádku.



V další tabulce jsou popsána čtyři první tlačítka v okně Ladění.

Ikona	Akce
Nekrokovat	Pokračuje v běhu metody bez krokování, uzavře okno Ladění.
Odstranit	Ukončí provádění metody.
Odstranit a Upravit	Ukončí běh metody a přepne do Prostředí uživatele, běžící metoda je zobrazena v Editoru metod a poslední provedený řádek je vysvícen.

Upravit	Dočasně pozastaví provádění metody a přepne do Prostředí návrháře, poslední prováděný řádek metody je vysvícen. Po opuštění Prostředí návrháře se můžete vrátit do okna Ladění a pokračovat v provádění metody.

9.14. Použití okna Ladění

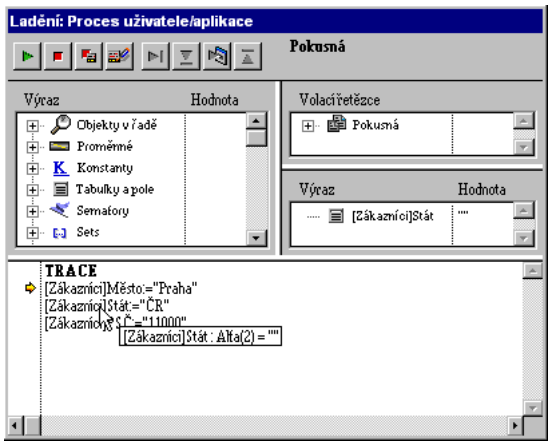
V okně Ladění můžete zkontrolovat hodnoty polí a proměnných za běhu procedury. Okno je rozděleno do několika částí, každou z nich můžete zvětšit či zmenšit a zobrazit si tak tolik informací kolik z této části potřebujete.

Žlutá šipka ukazující na řádek kódu znamená, že tento řádek bude prováděn. Klepnutí na tlačítko Enter nebo Return způsobí, že 4D tento řádek provede. Je rovněž možné klepnout na šipku myší a potáhnout ji na nové místo a provést tak buď část kódu ještě jednou nebo přeskočit určitou část kódu.

Část okna v levé horní části se nazývá Výrazy a je složena z několika hierarchických seznamů objektů. První seznam se nazývá Objekty v řadě a zobrazuje úplnný seznam všech objektů v platném řádku. Tak jak krokujete proceduru a přecházíte z řádku na řádek, budou se tyto objekty měnit. Pokud si chcete zkontrolovat počet záznamů v platném výběru a hodnotu platného záznamu, můžete tak učinit v seznamu Tabulky a pole. Je ještě mnoho dalších speciálních rysů okna Ladění, ale těmi se budeme zabývat později.

9.15. Zkratky – rychlé vkládání výrazu do okna Ladění

V pravé části pod oblastí Volací řetězce je oblast, kde můžete sledovat vámi vybrané proměnné, pole a výrazy. Spíše než psaní názvů tabulek, polí, proměnných budete využívat metodu klepnutí na tyto objekty v řádku kódu ve spodní části nebo přetažením tohoto objektu do části Výraz.



V dalším cvičení použijete příkaz TRACE k sledování práce tlačítka bLocal a sledování zda doopravdy mění pole Město, Stát a PSČ.

9.15.1. Užití příkazu TRACE k zobrazení okna Ladění

1. Otevřete metodu objektu pro tlačítko bLocal.

2. Do první řádky metody, napište příkaz TRACE.

```
TRACE

   ` Metoda objektu bLocal. [Zákazníci]; "Vstupní"
   ` Vytvořeno Jim Steinman. 15/1/97.

[Zákazníci]Město := "Praha 5"
[Zákazníci]Stát := "ČR"
[Zákazníci]PSČ := "15500"
   ` Konec metody objektu.
```

3. Přidejte nový záznam a vyzkoušejte tlačítko.
4. V okně Ladění klepněte na každé pole vašeho kódu a přidejte je tak do oblasti Výrazy k sledování změny hodnoty.

Všimněte si, že tato pole jsou na počátku prázdná

5. V oblasti výrazy poklepejte a vytvořte zde novou položku.
6. Napište následující: Length ([Zákazníci]Město)
7. Klepněte na tlačítko Enter nebo Return a proveďte tak první řádek kódu, sledujte změny v oblasti výrazy.
8. Pro provedení celé procedury přejděte zpět do metody objektu v Prostředí návrháře a „odkomentujte“ příkaz TRACE.

Vyvolání okna Ladění lze provést následujícími způsoby:

- Přidat příkaz TRACE do vašeho kódu.
- Při provádění metody provést Option + klepnout (Alt + klepnout).
- V Prostředí návrháře zvolit Nástroje ¬ Seznam procesů..., klepněte na proces (momentálně proces Uživatele/Aplikace) a vyberte z nabídky Proces ¬ Krokovat.
- V Prostředí návrháře vybrat Nástroje ¬ Seznam přerušení…a vytvořit tak přerušení (provádění kódu bude přerušeno vždy, když 4D narazí na určitý příkaz uvedený v seznamu přerušení).
- Napsat záměrnou chybu ve vašem kódu!

9.16. Problémy způsobované příkazy TRACE a ALERT

Důrazně doporučujeme, aby jste se vyvarovali příkazům TRACE a ALERT v metodách formulářů a to proto, že použití těchto příkazů na zmíněném místě může vyvolat nekonečnou smyčku.

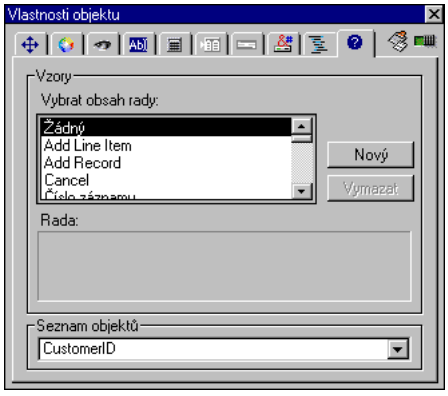
Později v pokročilejších stádiích vašeho programování můžete používat příkaz ON EVENT CALL. Tento příkaz je nemožné krokovat pomocí příkazu TRACE, protože každé klepnutí myší či na klávesnici bude inicializovat tento příkaz a okno ladění, takže se dostanete do nekonečné smyčky.

Občas jediný způsob jak přerušit takovouto smyčku je vynutit konec programu a nejlépe pak restartovat počítač.

Extra Kredit Diskuse

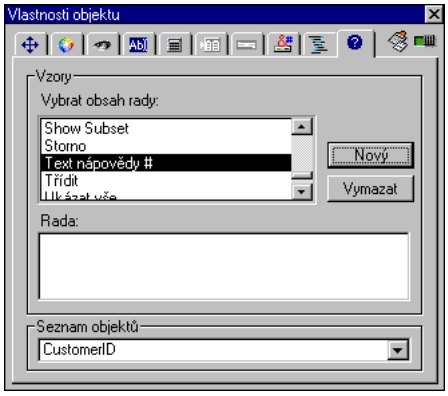
4D verze 6 vám nabízí příležitost přidat tipy/nápovědu do vašich polí a tlačítek. Můžete takovýto tip přidat k tlačítku bLocal tak, aby vysvětloval uživateli co toto tlačítko provádí.

Tipy jsou organizovány jako soubor pojmenovaných zpráv. Vytvoříte zprávu a nazvete ji jak je ukázáno na obrázku níže, když je tato zpráva vytvořena můžete ji přiřadit poli nebo tlačítku. Každou zprávu můžete použít vícekrát s různými objekty. Jestliže změníte zprávu, tato změněná zpráva se okamžitě objeví ve všech přiřazených objektech.



Prvním krokem je v dialogu Definic objektů přejít na stránku Nápověda.

K vytvoření zprávy, klepněte na tlačítko Nový, vytvoříte tak nový prázdný tip.

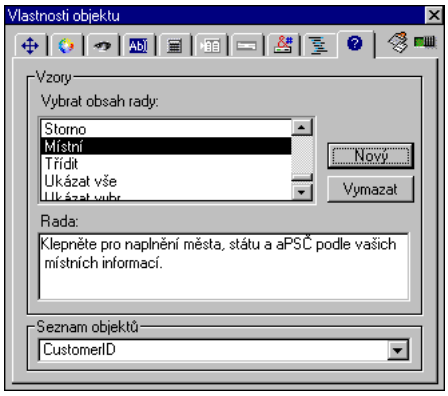


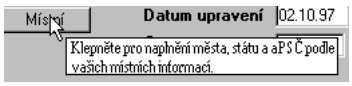
Do oblasti Zpráva napište text vašeho tipu.

9.17. Vložení znaku Nový řádek do zprávy Rada

V předchozích verzích 4D nešlo v hlášeních Rady oddělovat jednotlivé řádky, znakem Nový řádek. Ve verzi 6 je to umožněno.

Pokud jste skončili se zadáváním typu, klepněte uzavírací políčko a uzavřete okno definic objektů.





Extra kredit Cvičení

9.17.1. Přidání tipu k tlačítku bLocal

1. V Prostředí návrháře otevřete formulář [Zákazníci];"Vstupní".
2. Poklepejte na tlačítko bLocal.
3. Přejděte na stránku Nápověda (záložka úplnně vpravo)
4. Klepněte na tlačítko Nový a vytvořte text nápovědy #21.
5. Klepněte do oblasti Rada a napište: Klepněte a vyplňtě tak pole Město, Stát a PSČ vašimi místními informacemi.
6. Klepněte se stisknutým tlačítkem Comma (Ctrl) na text nápovědy #21 a změňte jeho název na bLocal.
7. V pravém horním rohu klepněte na uzavírací okénko.
8. Přejděte do Prostředí uživatele a vyzkoušejte.

10. Provádění metod formulářů

10.1. Zkratka – uzavření všech oken v Prostředí návrháře

Jestliže se pro vás již Prostředí návrháře stává nepřehledným z důvodu počtu otevřených oken, můžete je všechny najednou uzavřít. Zkratka pro uzavření všech oken v prostředí návrháře je tatáž jako v systému:

- Se stisknutou klávesou Alt poklepejte na uzavírací okénko v libovolném okně

Již jsme podstatně vylepšili formulář [Zákazníci];"Vstupní", ale ještě nejsme hotovi. Nyní do tabulky [Customers] přidáme další pole, která nám umožní sledovat prováděné změny. Jsou to pole:

- DatumVytvoření
- ČasVytvoření
- DatumÚpravy
- ČasÚpravy

Tato čtyři pole by měla být naplňována automaticky. Když se tato pole chystají k zobrazení ve vstupním formuláři, můžete vytvořit metodu objektu, která jim přiřadí hodnoty. Ale co když nechcete, aby se tato pole objevila ve vstupním formuláři. Kam by jste měli umístit svůj kód? Kromě libovolné metody objektu, můžete kód rovněž umístit do metody formuláře.

10.1.1. Přidání polí ke sledování změn do struktury a formuláře

1. Přejděte do Prostředí návrháře.
2. Vyberte Návrh ¬ Struktura.
3. V okně struktury klepněte na tabulku [Zákazníci].
4. Vyberte Struktura ¬ Nové pole.
5. Přidejte následující pole:

Název pole	Typ	Vlastnosti
DatumVytvoření	Datum	Pouze zobrazit, Neviditelné
ČasVytvoření	Čas	Pouze zobrazit, Neviditelné
DatumÚpravy	Datum	Pouze zobrazit, Neviditelné
ČasÚpravy	Čas	Pouze zobrazit, Neviditelné

6. Otevřete formulář [Zákazníci];"Vstupní".
7. Přidejte čtyři nová pole a jejich textové popisky.

10.2. Metody formuláře

Zatímco metody objektu patří určitému poli nebo určitému tlačítku ve formuláři, metody formuláře patří celému formuláři. Metody formuláře se spouštějí pouze po určitých událostech formuláře, jako je klepnutí, poklepání atd., které jsou určeny programátorem. Tyto události mohou opakovaně spouštět více částí kódu (tj. jestliže klepnete např. na tlačítko, může být spuštěna metoda objektu a současně může být spuštěna i metoda formuláře).

Velice často můžete kód umístit buď do metody objektu nebo metody formuláře. Programátoři 4d jsou známí svými debatami o výhodách metod objektů proti metodám formulářů a naopak. Ve verzi 6 by tyto debaty měly skončit, protože již není důvod proto neužívat metody formulářů z důvodu, že nejste přesně schopni řídit ve které události se která část kódu spustí.

Mohli jste si všimnout, že 4D umísťuje v levém horním rohu objektu s metodou objektu černý trojúhelník. Tento černý trojúhelník je viditelný pouze v prostředí Návrháře, uživatel jej nevidí. 4D však neumísťuje černý trojúhelník v levém horním rohu formuláře, který obsahuje metodu formuláře. To samozřejmě může způsobovat určité těžkosti, někomu kdo se snaží převzít po někom jiném kód databáze.

10.3. Kdy se spouští metoda formuláře

Protože metoda formuláře může být potenciálně spuštěna během všech událostí formuláře, budete určitě chtít zajistit, aby se váš kód spouštěl pouze jako odpověď na zcela určité události.

10.4. Vnořování rozhodování If...End if

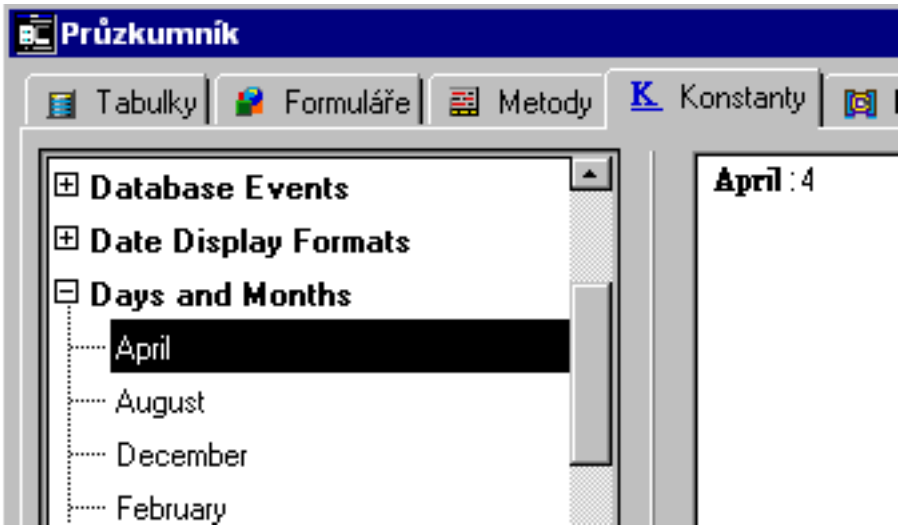
V tomto cvičení, napíšete metodu formuláře obsahující vnořená rozhodování If...End if. Jestliže je první podmínka splněna pak 4D pokračuje v testování druhé podmínky. Jestliže je druhá podmínka splněna, je proveden kód. Druhá zde použitá podmínka, testuje zda je záznam nový nebo zda je zobrazován záznam již v databázi uložený. Jestliže první podmínka není splněna, druhá podmínka se netestuje. Všimněte si jak pro vás 4D zarovnává bloky If...End if. Když dojdete na konec řady vnořených bloků If...End if, můžete obvykle říci, jestli vám chybí jedno či více End if, protože není provedeno správné zarovnání.

10.5. Konstanty

4D sleduje mnoho položek jako čísla, takže např. den není Neděle ale číslo 1. Aby jste si nemuseli všechna tato čísla pamatovat, 4D zavádí a používá konstanty. Různé typy konstant mohou být nalezeny v Prostředí návrhá v okně Průzkumníka, na stránce Konstanty. Zde jsou konstanty organizovány do logických skupin a mohou být myší přetaženy z okna Průzkumníka do Editoru metod.

Konstanty jsou velmi užitečné, protože významně snižují čas, který potřebuje k programování tím, že nemusíte hledat v knihovnách hodnoty určitých proměnných a rovněž zpřehledňují výsledný kód (tj. místo aby jste si museli pamatovat co znamená iDay=1, iDay=Sunday je

jednodušší). Aby jste v kódu rozeznali, že použitý název je konstanta, jsou konstanty v textu metody podtrženy.



10.6. Testování nového záznamu s použitím příkazu Record number

Když uživatel přijme a uloží nový záznam do databáze, 4D přiřadí tomuto záznamu číslo. Ve 4D je toto číslo nazýváno číslo záznamu. Aby jste toto číslo zjistili, můžete použít příkaz Record number. Poznamenejme pouze, že toto číslo není trvalé. Některá čísla mohou být využívána znovu a znovu. Pokud je záznam nový a nebyl ještě uložen má záznam číslo –3. Uložený záznam obdrží kladné číslo.

10.7. Události formuláře

Jestliže potřebujete testovat jestli program právě zavádí záznam, můžete testovat jestli událost formuláře je On Load. Tato podmínka testování bude vracet hodnotu true nebo false (pravda či nepravda). V závislosti na tom jestli je daná událost formuláře aktivní. Existuje samozřejmě rovněž mnoho dalších událostí, které mohou být obdobným způsobem testovány.

10.7.1. Příkaz Form event navracející číslo

Příkaz Form event vrací číslo, které reprezentuje událost formuláře, která je právě aktivní v daném metodě. Protože lidé rozumějí lépe slovům než číslům, budeme používat pro události formuláře konstanty, takže budeme spíše testovat, že událost formuláře je rovna On Load spíše než číslu 1 (toto je právě hodnota konstanty On Load).

10.7.2. Naplňování polí datumu a času při vytváření záznamu

1. Přejděte do Prostředí návrháře.
2. Otevřete vstupní formulář pro [Zákazníci].
3. Vyberte Formulář ¬ Vlastnosti formuláře.
4. Prověřte, že je zaškrtnuta událost Při zavedení.
5. Vyberte Návrh ¬ Upravit metodu .
6. Klepněte na tlačítko Upravit pro metodu formuláře [Zákazníci];“Vstupní“ .
7. Vyberte typ Textová a klepněte na tlačítko OK.
8. Vložte následující metodu (konstanty můžete přetáhnout z okna Průzkumníka)

```
If (False)
      ` Metoda formuláře: [Zákazníci] "Vstupní"
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Nastaví výchozí hodnoty při užití vstupního formuláře.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

If (Form event=On Load)
   If (Record Number ([Zákazníci]) = -3) `          Jestliže je to nový záznam
      [Zákazníci]DatumVytvoření := Current Date
      [Zákazníci]ČasVytvoření := Current Time
   End if
End if
   ` Konec metody
```

9. Přejděte do Prostředí uživatele a vyzkoušejte tuto novou metodu vytvořením nového záznamu.

10.8. Naplnění pole datumu a času při úpravě záznamu

Již jsme použili dvě ze čtyř polí datumu a času pro sledování změn záznamu. Nyní potřebujeme naplnit pole a čas pro úpravy záznamů. Abychom to mohli provést musíme nejdříve určit místa, kde může být upravený záznam uložen. Nemusíte dělat nic jestliže sice uživatel ukládá záznam, ale neprovedl žádné změny a rovněž nemusíte dělat nic jestliže uživatel sice provedl změny, ale záznam stornuje. Zajímají nás tedy pouze situace, kdy uživatel upravil některé pole (ať již vstupem z klávesnice nebo pomocí příkazů jazyka) a pak záznam přijal a uložil. Pro tento účel můžeme využít událost Při potvrzení vstupu (konstanta On Validate).

10.8.1. Přidání řádku kódu pro sledování změn.

1. Upravte metodu formuláře [Zákazníci]; „Vstupní“ následujícím způsobem:

```
If (False)
      ` Metoda formuláře: [Zákazníci] "Vstupní"
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Nastaví výchozí hodnoty při užití vstupního formuláře.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

If (Form event=On Load)
   If (Record Number ([Zákazníci]) = -3) `          Jestliže je to nový záznam
      [Zákazníci]DatumVytvoření := Current Date
      [Zákazníci]ČasVytvoření := Current Time
   End if
End if
If (Form event=On Validate)
      [Zákazníci]DatumÚpravy := Current Date
      [Zákazníci]ČasÚpravy := Current Time
End if

   ` Konec metody
```

10.9 Blok Case pro vylučující se podmínky

Jestliže v bloku If...End if máte řadu podmínek, keré se vzájemně vylučují, můžete je nahradit příkazy Case of...End case (Případ). Příkazy Case of...End case jsou účinnější, protože automaticky přeskočí všechny ostatní podmínky, pokud se zjistí, že je jedna z podmínek splněna.

Jako analogie, předpokládejme, že jste se někoho zeptali jesti je ráno a on odpověděl „Ano“. Potřebujete se jej ještě zeptat jestli je odpoledne? Samozřejmě ne, protože byl-li případ splněn automaticky to zanmená, že druhý případ nemůže být splněn.

Při programování můžete tyto vzájemně se vylučující otázky považovat za následnou řadu bloků If...End if, ale je mnohem účinnější a přehlednější použít Case of...End case. Zde je způsob dotazu pomocí bloku Case of...End case:

```
Case of
   : (Ráno)
      ALERT("Dobré ráno!")
   : (Odpoledne)
      ALERT("Dobré odpoledne!")
   : (Večer)
      ALERT("Dobrý večer!")
End case
```

Ve vstupním formuláři jsou některé události (většina) příkladem na tyto vylučující se podmínky. Je-li platná (děje se) jedna událost není současně platná jiná. Proto zde lze místo If...End if bloků využít Case of...End case.

Kromě toho si tím zajistíme i optimalizaci kódu a budeme mít jistotu, že příkaz Form event voláme pouze jednou.

Tip: Umístěte podmínky v tom pořadí jak jsou nejčastěji pravda, tak že nejčastěji splněná bude první. Urychlí to i provádění kódu (Ve většině případů se testuje jen jednou).

Poznámka: Kromě toho, že jsou zde zachytávané události, je tento typ použití s Case připraven i na případné rozšíření zachytávání dalších událostí, aniž by se ztratila přehlednost, proto vždy v metodách formulářů a triggerech používáme bloky Case of.

10.9.1. Nahrazení vícenásobných bloků If…End if blokem Case of…End case

1. Otevřete metodu vstupního formuláře [Zákazníci].
2. Nahraďte bloky If…End if blokem Case of…End následovně:

```
If (False)
      ` Metoda formuláře: [Zákazníci] "Vstupní"
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Nastaví výchozí hodnoty při užití vstupního formuláře.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if
```

- C_LONGINT($LFormEvent)
- $LFormEvent := Form event
- Case of
- : ($LFormEvent = On Load)
  If (Record Number ([Zákazníci]) = -3) ` Jestliže je to nový záznam
  [Zákazníci]DatumVytvoření := Current Date
  [Zákazníci]ČasVytvoření := Current Time
  End if
- : ($LFormEvent = On Validate)
  [Zákazníci]DatumÚpravy := Current Time
  [Zákazníci]ČasÚpravy := Current Time

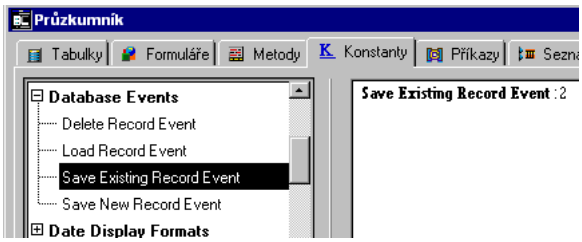
End case
` Konec metody

10.10. Triggery

Tak jako formulář má události má události také tabulka. Tyto události jsou známy jako triggery (spouštěče) a provádějí se, kdekoliv má je prováděna určitá událost. Nazývají se triggery, protože určitá akce/událost spouští kód.

Jsou čtyři zákaldní události, které spustí trigger ve 4D: uložení nového záznamu, Uložení existujícího záznamu, Mazání záznamu, Zavádění záznamu. V tomoto kurzu budeme probírat pouze jednu z nich Uložení existujícího záznamu.

Náš kód v předchozím cvičení ukládá do pole datum a čas změny dokonce i u nových záznamů. Můžeme toto chování vylepšit tak, že tato pole budou naplněna pouze v případě změny původně uloženého záznamu, přesuneme tuto část kódu do triggeru.



Událost Uložení existujícího záznamu se děje když:

- Uživatel přijme upravený záznam

Událost Uložení existujícího záznamu se neděje když:

- Uživatel zruší záznam, nebo
- Uživatel klepne na tlačítko Přijmout, ale nebylo upraveno žádné pole.

Poslední dvě podmínky znamenají, že záznam je jako by vůbec nebyl užit.

Pole může být upraveno, když:

- Uživatel napíše data do pole, nebo
- Program změní hodnoty v polích.

Nezáleží na tom jaký typ změny byl proveden. Například: Změna pole jméno z “Bob” na “Bob” znamená provedenou změnu.

Je důležité si pamatovat, že každá tabulka má pouze jeden trigger. Tento kód může být volán čtyřmi odlišnými způsoby (jestliže jsou všechny čtyři události databáze použity), takže musíte kontrolovat, kerá událost způsobila vyvolání triggeru.

Je možné zapnout, nebo vypnout události databáze, keré způsobují vyvolání a spuštění triggeru. Může tak být provedeno v prostředí návrháře: položka Vlastnosti tabulky v nabídce Struktura. Jako výchozí nastavení jsou všechny neaktivní a musí být zapnuty návrhářem.

10.10.1. Úpravy sledování změn dle datumu a času

1. Otevřete okno průzkumníka a přejděte na stránku Metody.
2. Klepněte na trojúhelník Metody formulářů a triggery a pak na tabulku [Zákazníci].
3. Klepněte na tlačítko Upravit.
4. Přidejte do triggeru [Zákazníci] následující kód:

```
If (False)
      ` Trigger: Zákazníci
      ` Kurz ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 1/15/97

      ` Účel: Sleduje změny záznamů zákazníků

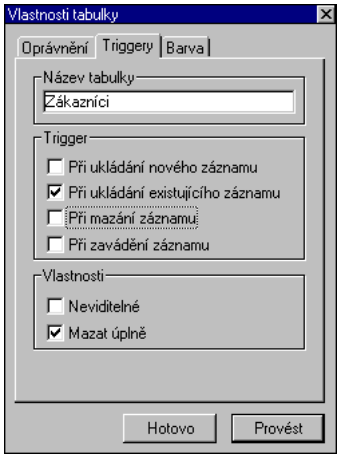
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if
```

- ```
  C_LONGINT($LDatabaseEvent)
  ```

- ```
  $LDatabaseEvent := Database event
  ```

- ```
  Case of
     : ($LDatabaseEvent =  Save Existing Record Event)
        [Zákazníci]DatumÚpravy := Current Date
        [Zákazníci]ČasÚpravy := Current Time
  End case
     `Konec triggeru
  ```

5. V okně struktury klepněte na tabulku [Zákazníci] a vyberte ji tak.
  ```



6. Vyberte položku nabídky Struktura Vlastnosti tabulky.
7. Klepněte na stránku Triggery a zaškrtněte políčko Při ukládání existujícího záznamu.
8. Upravte metodu formuláře[Zákazníci] " následujícím způsobem:

```
.....
        Case of
          : ($LFormEvent =  On Load)
            If (Record Number ([Zákazníci]) = -3) `        Jestliže je to nový záznam
            [Zákazníci]DatumVytvoření := Current Date
            [Zákazníci]ČasVytvoření := Current Time
          End if
        End case
          ` Konec metody
```

9. Přejděte do prostředí návrháře a vyzkoušejte si úpravy existujících záznamů.
10. Rovněž zkuste přidat nový záznam.

11. Nabídky pro vstupní formuláře

11.1. Nabídky vstupních formulářů musí vyhovovat doporučením TWI nebo HIG

Náš program je již téměř přijatelný, kromě toho, že ve vstupních formulářích nabídky buď nepracují, nebo jsou nesprávně zobrazeny. Jestliže jsme ve vstupním formuláři a nechceme, aby uživatel pracoval s nabídkami, musíme tyto nabídky vypsat v šedém písmu (znepřístupnit). Můžeme si zjednodušit úlohu, že k vstupním formulářům přiřadíme záhlaví nabídky –2, vše pak bude pracovat, ale ve skutečnosti rozhodně nechceme, aby uživatel ve vstupním formuláři se mohl dotazovat na záznamy a třídit je, když má zobrazen pouze jeden. Chceme tedy některé položky znepřístupnit. Nejjednodušší cesta je vyměnit záhlaví nabídek, tak aby bylo zobrazováno správně.

Protože se rovněž chystáme změnit tituly okna ve vstupních formulářích, aby odráželi zobrazované záznamy, provedeme to vše v jednom místě, nezapomeneme při tom použít proměnné, keré se sami vymaží z paměti, když je již nebudeme potřebovat.

11.1.1. Vytvoření Záhlaví #4 použité ve vstupních formulářích

1. Vytvořte Záhlaví #4.
2. Připojte nabídku Záznamy ze Záhlaví #1.
3. V Záhlaví #4 s vybranou nabídkou Záznamy stiskněte tlačítko Přidat.
3. Pojmenujte novou nabídku `Zprávy`.
4. Klepněte na tlačítko Přidat položku
5. Vložte položku nabídky `Rychlé…/R`
6. Přidejte z Knihovny obrázků patřičný obrázek do položky nabídky.
7. Opakujte kroky 4 až 6 dokud nevložíte:

```
Rychlé…/R
Štítky…/J
Diagamy…/K
-
Specielní zprávy…
```

8. V nabídce Zprávy odškrtněte pro všechny položky políčko Viditelná.

11.1.2. Vytvoření metody projektu pro ovládání nových nabídek a titulu okna

1. Vytvořte metodu projektu WIN_InputWindowTitle.
2. Vložte následující metodu:

```
If (False)
      ` Metoda: WIN_InputWindowTitle
      ` Kurz ACI
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 1/15/97

      ` Účel: Tato metoda mění záhlaví nabídek
      ` a titul okna pro vstupní formulář
      ` CB: Všechny metody vstupních formulářů

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_STRING (81; $sWindowTitle)
C_STRING (31; $sSelectedRecord ;$sRecordsInSelection)

$sWindowTitle := Table name(pTable)

If (Record number (pTable->) = -3) `je nový záznam
   $sWindowTitle := $sWindowTitle + " nový záznam"
Else `               je existující záznam
   $sSelectedRecord := String (Selected record number (pTable->))
   $sRecordsInSelection := String (Records in selection (pTable->))
   $sWindowTitle := $sWindowTitle + ": #" + $sSelectedRecord + " z " + $sRecordsInSelection
End if

SET WINDOW TITLE ($sWindowTitle)
MENU BAR (4)
   ` Konec metody
```

11.2. Tlačítka vztahů způsobují problém

Tlačítka ve fakturách, kerá nám dovolují upravit patřičný záznam zákazníka a tlačítko ve vstupním formuláři Zákazníků, které nám umožňuje vidět patřičnou fakturu jsou pěkný rys naší databáze. Použití vztahů nám zde však nyní způsobuje problém. Například: Přejděme do konkrétního záznamu zákazníka, přepněme se na stránku 2, veberme nějakou fakturu a otvřme ji a v tomto formuláři kleněme na tlačítko Zákazník, keré nám otevře tentýž záznam zákazníka, z nějž jsme vyšli, ale jme o úroveň hlouběji. Odtud opět přes tlačítko faktury zobrazme fakturu ..... dokud se paměť nezhltí. Takovýmto situacím se musíte vyhnout.

Počáteční tlačítko je dobré, ale to musí být vše. Z druhé úrovně již nesmíte být schopni jít dále. Takže když budeme v druhé úrovni musíme některá tlačítka udělat nedostupná.

11.3. SET VISIBLE ({*;}objekt; logické)

Příkaz SET VISIBLE způsobí, že daný objekt zmizí z obrazovky. Volitelný parametr hvězdička způsobí, že bude použit název objektu a ne název zdroje dat. Zdroj dat je něco jako název proměnné nebo název pole. Název objektu je oddělený název, který je přiřaditelný každému objektu formuláře včetně čar, obdelníků, statického textu atd. Když používáte název objektu můžete rovněž použít znak „@“ k výběru a zachycení skupiny objektů, které začínají stejnými písmeny a provést tak akci na všechny objekty takovéto skupiny.

Kvíz

- Jak můžeme zjistit, že jsme se ocitli ve formuláři [Faktury];"Vstupní" přes formulář [Zákazníci];"Vstupní"?

11.3.1. Volání kódu výměny nabídky ve vstupním formuláři a změny titulu

1. Otevřete formulář [Zákazníci];"Vstupní".
2. Na stránce 2 přidělte tlačítku pOpen název objektu `INVOICEButtonOpen` (názvy objektů jsou na stránce Souřadnice)
3. Tlačítku bCalculate dejte název `INVOICEButtonCalculate`.
4. Upravte metodu objektu [Zákazníci];"Vstupní" následujícím způsobem:

```
…

Case of
   : ($LFormEvent =  On Load)
      If (pTable = (->[Faktury])) ` Přicházíme z tabulky [Faktury] pomocí Upravit zákazníka
         SET VISIBLE (*;"INVOICEButton@";False) `      Nelze provádět funkce faktur
         SET WINDOW TITLE ("Upravuje se "+[Zákazníci]Firma)
      Else
         WIN_InputWindowTitle
      End if

      If (Record Number ([Zákazníci]) = -3) `Záznam je nový.
         [Zákazníci]DatumVytvoření := Current Date
         [Zákazníci]ČasVytvoření := Current Time
      End if
End case
…
```

5. Vytvořte novou metodu pro formulář [Produkty];"Vstupní" následujícím způsobem.

```
If (False)
```

```
      ` Metoda formuláře: Vstupní
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Metoda formuláře [Produkty]; "Vstupní"

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($LFormEvent)

$LFormEvent := Form event

Case of
   : ($LFormEvent =  On Load)
      WIN_InputWindowTitle
End case
   ` Konec metody
```

6. Vytvořte novou metodu formuláře pro [Faktury];"Vstupní".
7. Do metody napište následující:

```
If (False)
      ` Metoda formuláře: [Faktury]; "Vstupní"
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Metoda formuláře  [Faktury]; "Vstupní"

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($LFormEvent)

$LFormEvent := Form event

Case of
   : ($LFormEvent =  On Load)
      If (pTable = (-> [Zákazníci])) ` Přicházíme ze [Zákazníci] tlačítkem bOpen
         SET VISIBLE (bModify; False) `Nelze zpět upravovat zákazníka
         SET WINDOW TITLE ("Upravuje se faktura  " + String([Faktury]InvoiceNo))
      Else
         WIN_InputWindowTitle
      End if
End case
   ` Konec metody
```

8. V Prostředí návrháře otevřete vstupní formuláře pro [Zákazníci], [Faktury] a [Produkty] a v pravém horním rohu vymažte proměnnou zobrazující informace o počtu záznamů. (Ve formuláři [Zákazníci] je proměnná na stránce 0.)

11.3.2. Zajištění, že okno otevírané voláním MODIFY RECORD bude bez posuvníku.

1. Otevřete formulář [Faktury];"Vstupní".
2. Upravte metodu objektu bModify následujícím způsobem:

```
...
MODIFY RECORD([Zákazníci];*)
WIN_InputWindowTitle
   ` Konec metody objektu
```

3. Otevřete formulář [Zákazníci];"Vstupní".
4. Přejděte na stránku 2 formuláře.
5. Upravte metodu objektu tlačítka bOpen následujícím způsobem:

```
...

MODIFY RECORD([Faktury];*)
WIN_InputWindowTitle
   ` Konec metody objektu
```

11.2. Návrat k záhlaví nabídek a titulu výstupního formuláře

Nyní když opustíme vstupní formulář je nabídka ve výstupním formuláři i titul okna nesprávně. Můžeme tento nedostatek napravit buď vložením kódu do metody výstupního formuláře nebo vytvořením metody objektu pro každé tlačítko, které způsobí, že se vracíme zpět do výstupního formuláře. Protože zatím neřešíme požadavky na chování, které znamenají pokud možno vyloučení kódu z metod formuláře nebo vůbec zákaz jejich používání, a protože opakování vkládání metody objektu do několika tlačítek je nudná práce, použijeme metodu výstupního formuláře a novou událost formuláře On Close Detail. Tato událost se děje, když uživatel opouští vstupní formulář a navrací se z něj do výstupního formuláře. Tato událost s děje pouze ve výstupním formuláři a jeho patřičných metodách.

11.2.1. Volání kódu k obnově záhlaví nabídek a titulu okna

1. Vytvořte metodu formuláře pro [Zákazníci];"Výstupní".

```
If (False)
      ` Metoda formuláře: [Zákazníci];"Výstupní"
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Metoda formuláře pro [Zákazníci]; "Výstupní"

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($LFormEvent)

$LFormEvent := Form event

Case of
   : ($LFormEvent =  On Close Detail)
      WIN_OutputWindowTitle
      MENU BAR(3)
End case
   ` Konec metody
```

2. Vytvořte novou metodu formuláře pro [Produkty];"Výstupní".

```
If (False)
      ` Metoda formuláře: [Produkty];"Výstupní"
      ` Kurs ACI Universita
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Metoda formuláře pro [Produkty]; "Výstupní"
```

```
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($LFormEvent)

$LFormEvent := Form event

Case of
   : ($LFormEvent =  On Close Detail)
      WIN_OutputWindowTitle
      MENU BAR(3)
End case
   ` Konec metody
```

3. Vytvořte novou metodu formuláře pro [Faktury];"Výstupní".

```
If (False)
      ` Metoda formuláře: [Faktury];"Výstupní"
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Metoda formuláře pro [Faktury]; "Výstupní"

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($LFormEvent)

$LFormEvent := Form event

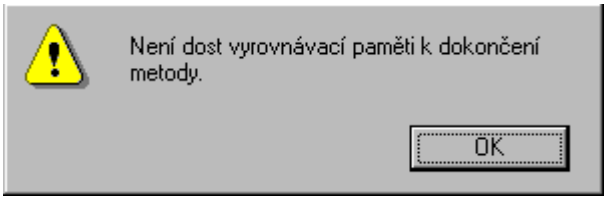
Case of
   : ($LFormEvent =  On Close Detail)
      WIN_OutputWindowTitle
      MENU BAR(3)
End case
   ` Konec metody
```

4. Přejděte do prostředí Vlastní nabídky a otestujte je.

12. Procesy

12.1. Ukládání ve vyrovnávací paměti/Stack

Nyní máme v nabídce Soubor zpřístupněny všechny položky tak, že můžeme po sobě vybrat Zákazníci a výstupním formuláři zákazníků potom Faktury. Nyní zjistíme, že 4D do paměti (stack) ukládá výstupní formuláře. Vyzkoušejte si: vyberte Soubor ¬ Zákazníci a pak vyberte Soubor ¬ Faktury. Když nyní klepnete na tlačítko Hotovo ve výstupním formuláři [Faktury] uvidíte, že se opět objeví výstupní formulář tabulky [Zákazníci]. 4D si pamatuje předchozí formulář na obrazovce tak, že jej umístí do speciální oblasti paměti nazývané Stack. Samozřejmě tato část paměti je omezená takže jestliže budete opakovaně vybírat Soubor ¬ Zákazníci a Soubor ¬ Faktury, na přibližně pátý či šestý pokus nebude již 4D schopna vyplnit úlohu a na obrazovce se objeví zpráva „Není dost vyrovnávací paměti pro provedení metody.“ Jestliže uzavřete některé formuláře, uvolní se paměť stack a vy opět budete schopni vybrat další položku nabídky.



Ukládání více výstupních formulářů do paměti stack (způsobené opakovaným voláním MODIFY SELECTION) není dobrým způsobem a to z následujících tří důvodů:

- Paměť stack bude přeplněna přibližně po uložení šesti formulářů.
- Uživatel může být zmaten, když klepne na tlačítko Hotovo a uvidí opět nějaký výstupní formulář.
- Když klepnete na tlačítko Hotovo a uzavřete jeden výstupní formulář, proměnná pTable bude stále ukazovat na poslední zobrazenou tabulku, která již nyní není aktivní. Takže uživatel může být fyzicky na obrazovce v tabulce [Zákazníci], ale protože předchozí otevřené okno bylo z tabulky [Faktury] vybráním Záznamy ¬ Editor dotazů nebo Záznamy ¬ Třídit uvidí seznam polí tabulky [Faktury].

Ve verzi 2 4D programátoři zabraňovali uživatelům v několikanásobném překrývání výstupních formulářů na obrazovce tím, že znepřístupnili nabídku tabulek. To samozřejmě řeší tento problém, protože uživatel nemůže podruhé vybrat Soubor ¬ Zákazníci atd.

Ve verzi 6 (a 3) 4D však existuje elegantnější řešení, když uživatel vybere Soubor ¬ Zákazníci, zobrazíte nový výstupní formulář v jeho vlastním novém okně a první výstupní formulář ponecháte v původním okně. 4D umožňuje otevřít více nezávislých oken tak, že každé toto okno pracuje odděleně a nezávisle. Počet takovýchto oken je omezen pouze pamětí přidělenou aplikaci. Každé takovéto okno je spuštěno uvnitř nezávislého procesu.

12.2. Nastartování nového procesu

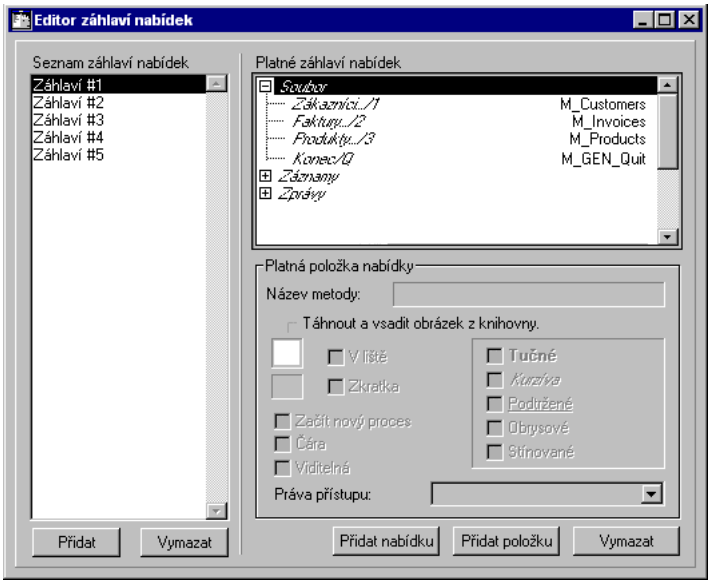
Proces je nejlépe si představit jako novou kopii celé 4D s jejím vlastním:

- Oknem
- Záhlavím nabídek
- Souborem procedur
- Platnými výběry pro každou tabulku databáze
- Proměnnými
- Pamětí stack

Nastartování nového procesu je překvapivě jednoduché: zaškrtnutí políčka

12.2.1. Úprava nabídky Soubor k nastartování nových procesů

1. Otevřete libovolné záhlaví s nabídkou Soubor.
2. Klepněte na položku nabídky Soubor ¬ Zákazníci.
3. Zaškrtněte políčko Nový proces.
4. Zopakujte pro položku Soubor ¬ Faktury.
5. Zopakujte pro položku Soubor ¬ Produkty.
6. Ukončete a restartujte 4D.
7. Jděte do prostředí Vlastní nabídky a vyberte Soubor ¬ Zákazníci.



Kvíz

- Mohli by jste přidat do systému nabídek tabulku [PoložkyFaktury]?

12.3. Změna titulu okna úvodní obrazovky

Jediný titul okna, který jsme ještě neměnili je titul okna úvodní obrazovky, který stále říká „Aplikace“. Určitě takovýto podivný název budete chtít změnit. Nejlepší místo pro změnu titulu okna úvodní obrazovky je těsně před jeho zobrazením. 4D obsahuje jednu metodu, která je spouštěna při otevírání aplikace tato metoda se nazývá Při spuštění. Tato metoda je spuštěna pouze jednou, a to když je databáze poprvé otevírána. Tuto metodu můžeme naplnit vlastním kódem s vlastními nastaveními databáze, kterými změníme výchozí nastavení.

Metoda Při spuštění je jednou ze sedmi speciálních metod nazývaných metody databáze. Tyto metody databáze jsou spuštěny kdykoliv se stane určitá specifická událost, která je charakterizuje jako např. spuštění či ukončení databáze. Další metody databáze jsou probrány v kurzech programování pro pokročilé.

12.3.1. Vytvoření metody databáze Při spuštění

1. Vytvořte metodu databáze Při spuštění následujícím způsobem:

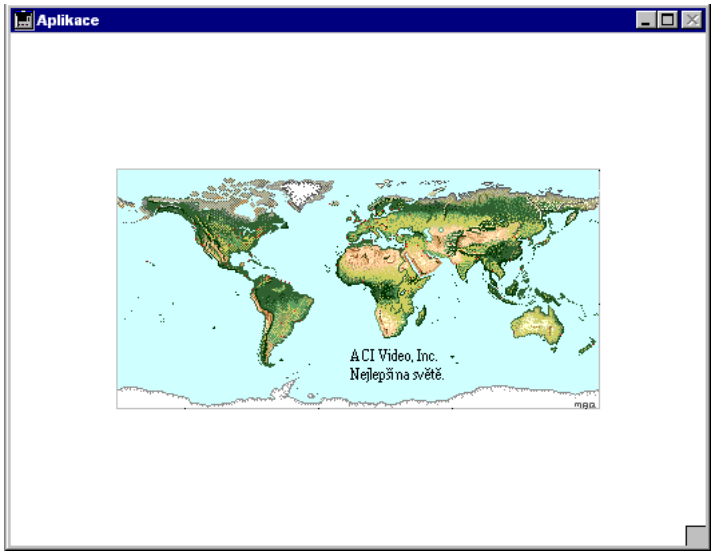
```
` Metoda databáze: Při spuštění
` Kurs ACI University
` Vytvořeno: Jim Steinman
` Datum: 15/1/97

` Účel: Nastaví prostředí podle přání programátora.

` Řídící proměnné verze a vývojáře
<>f_Version6x10 := True `Kurs programování
<>fJ_Steinman := True

SET WINDOW TITLE ("ACI Video")
```

2. Ukončete a restartujte 4D.



12.4. Nastavení výchozího prostředí po spuštění databáze

Jestliže jste se pozorně dívali během otevírání databáze viděli jste, že se titulek okna změnil během spouštění. Jakmile však přejdete do Prostředí návrháře a pak zpět do prostředí Vlastní nabídky, úvodné okno opět říká Aplikace. Aby jste udrželi titul okna podle vašeho přání, musíte zůstat v prostředí Vlastní nabídky.

12.4.1. Prostředí Vlastní nabídky jako výchozí prostředí po otevření

1. Vraťte se do Prostředí návrháře.
2. Vyberte Soubor ¬ Předvolby databáze.
3. Zaškrtněte v oblasti Začít v volič Vlastních nabídkách.
4. Ukončete a restartujte 4D.

12.5. Nastavení velikosti okna procesu

Výchozí okno procesu je příliš malé než, aby správně zobrazovalo vaše výstupní formuláře. Převezmeme řízení velikosti okna pomocí programovacích instrukcí. 4D rovněž předpokládá, že tyto rysy ošetříte spíše programovacími příkazy a vlastní prací. Jedna z úloh, kterou provedeme je určení velikosti otevíraného okna. Pokud neotevřete okno samy otevře ho 4D za vás, ve velikosti jakou stanoví sama.

12.5.1. Úprava metody GEN_ModifySelection pro otevření většího okna

1. Upravte metodu GEN_ModifySelection následujícím způsobem:

```
If (False)
    ` Metoda: GEN_ModifySelection
    ` Kurs ACI University
    ` Generická procedura
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Zobrazí výstupní formulář se seznamem záznamů tabulky určené pomocí pTable

  <>fGeneric := True
  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if

MENU BAR (3)                    ` Přepne záhlaví nabídky
INPUT FORM (pTable->; "Vstupní")    ` Nastaví výchozí formuláře
OUTPUT FORM (pTable->; "Výstupní")
• $LWidth := 605                    ` Požadovaná šířka.
• $LHeight := 420                   ` Požadovaná výška.
• $LLeft := 10
• $LTop := 75
• $LRight := $LLeft + $LWidth
• $LBottom := $LTop + $LHeight
• $LWindowType := 4
• $LWindowID := Open window ($LLeft; $LTop; $LRight; $LBottom; $LWindowType)
ALL RECORDS  (pTable->)         ` Změní platný výběr.
CREATE EMPTY SET(pTable->; "UserSet")
WIN_OutputWindowTitle
MODIFY SELECTION (pTable->; *)    ` Zobrazí záznamy na obrazovku.
CLOSE WINDOW
   ` Konec metody
```

2. Přejděte do prostředí Vlastní nabídky a otestujte ji.

14. Porozumění předávaným parametrům

14.1. Předávání parametrů

Od prvního příkazu v tomto kurzu, t.j Uppercase, jste předívali parametry do 4D příkazů. Parametry jsou hodnoty předávané příkazům nebo funkcím, keré jazyk 4D potřebuje, aby mohl správně vykonat určitý příkaz založený na těchto parametrech. Parametry mohou být ve formě tabulek, polí nebo proměnných. Některé příkazy očekávají nebo dokonce vyžadují více parametrů, tak jako například funkce Open window, prováděná v předchozím cvičení.

Tak jako příkazy a funkce 4D mohou používat parametry, dovoluje vám jazyk 4D napsat vaše vlastní metody projektu, které budou používat nebo dokonce vyžadovat parametry. Tento silný rys jazyka můžeme využit k napsání vlastních metod, které budou vícenásobně používány uvnitř našeho programu. Tyto metody mohou vykonávat i velmi složité úlohy poměrně jednoduše. Vše co musíme provést je předat parametry našim vlastním příkazům.

Když předáváte parametry do vestavěných příkazů jazyka 4D, 4D si vytvoří kopii těchto paramterů a pracuje pouze s těmito kopiemi. Originální hodnoty parametrů zůstávají nezměněny nezávisle na tom co příkaz s předávanými hodnotami uvnitř sám sebe provede. Tentýž princip je použit pro vaše vlastní metody, které napíšete. Jazyk umístí všechny vaše paramtery do očíslovaných lokálních proměnných. První parametr je vždy $1, druhý $2 atd. Jestliže chcete parametr použít uvnitř vaší metody, musíte se na něj odkazovat pomocí očíslovaných proměnných a ne pomocí původních proměnných polí nebo tabulek. Existuje jeden pokročilý způsob, ve kterém toto nemusíme, ale ten se naučíme později.

Začneme psaním metody otevírající okna. Tento typ metody je běžný téměř ve všech databázích 4D a velice zjednodušuje proces otevírání oken.

14.2. Meziprocesní proměnné

Je to v pořadí třetí typ proměnné, který budeme probírat. Meziprocesní proměnné jsou dostupné v celé databázi a jsou sdíleny všemi procesy. Meziprocesní proměnné jsou primárně používány ke sdílení informací mezi procesy. Meziprocesní proměnné začínají znaky: Windows „<>“ (menší než následované větším než), Macintosh název začíná “<>”.

Dokonce i na počítači Macintosh můžete použít znaky používané na Windows. Tyto znaky se však překonvertují při vyhodnocení řádky na znaky používané na platformě Macintosh.

Meziprocesní proměnné budeme používat ke kaskádovitému otevírání oken tak, že se otevíraná okna nebudou zcela překrývat a vždy bude vidět alespoň kus z již otevřeného předchozího okna.

14.3. Nové přiřazení parametrů lokálním proměnným

Očíslované lokální proměnné ($1, $2, atd.) jsou používány uvnitř celé 4D jako vstupní parametry. Takže tyto proměnné mohou mít různý význam a různý typ v rozdílných metodách. Aby jste zpřehlednili váš kód a lépe jej zdokumentovali, důrazně vám doporučujeme, aby jste vždy na počátku procedury vložili několik řádek, které přiřadí parametry do jiných lokálních proměnných. 4D provede těchto několik přiřazení za velmi krátký okamžik a váš kód bude rozhodně přehlednější a lépe se v něm vyznáte.

14.4. Přiřazení nejčastěji volaných funkcí 4D do lokálních proměnných

V následující metodě budeme často používat počet parametrů, které budeme předávat metodám v bloku Case of...End case. Můžeme samozřejmě volat funkci Count parameters pokaždé, když tuto informaci potřebujeme. Znamená to však, že 4D volá tuto funkci vždy znovu a znovu, je mnohem úspornější přiřadit hodnotu navracenou touto funkcí do lokální proměnné a pak místo volání používat tuto lokální proměnnou. Tento způsob je rovněž použitelný pro opakované odkazování na tatáž data polí.

14.5. Vytvoření vlastních konstant s použitím proměnných IP

Můžete emulovat 4D konstanty vytvořením vlastních meziprocesních proměnných a přiřadit jim hodnoty při spuštění databáze. Tuto techniku použijeme k vytvoření našich vlastních definovaných konstant pro každý typ okna a další dostupné parametry oken. Pak když píšeme vlastní kód potřebujeme se pouze na tyto proměnné odkázat při použití okna určitého typu a na určité pozici.

14.6. Určení platformy uživatele

Protože je 4D na platformě nezávislá (běží na Macintosh nebo Windows), mohou existovat položky, které se týkají velikostí oken a případně dalšího chování, které se rozdílné pro jednotlivé platformy. V další metodě příkaz PLATFORM PROPERTIES bude použit k tomu, aby určitl na kterém typu stroje je kód spuštěn. Tato metoda může být volána kdykoliv v kódu potřebujete vědět na kterém typu stroje je metoda spuštěna.

14.6.1. Vytvoření generického příkazu WIN_LNewWindow

1. Vytvořte metodu projektu WIN_IsWindows.
2. Překopírujte metodu z textového souoru WIN_IsWindows , nebo napište:

```
If (False)
    ` Metoda: WIN_IsWindows
    ` Modul: Window Management
    ` Kurz ACI University
    ` Generická procedura
    ` Vytvořeno: Jim Steinman
    ` Datum:  15/1/97
```

```
    ` Účel: Určí zda běží pod Windows

  <>fGeneric :=True
  <>f_Version6x10 :=True
  <>fJ_Steinman :=True
End if

    ` Declarace localních proměnných
C_LONGINT($LPlatform)          ` Typ platformy
C_LONGINT($LSystemVersion)     ` System verze
C_LONGINT($LProcessor)         ` Procesor typ

PLATFORM PROPERTIES ($LPlatform;$LSystemVersion;$LProcessor)

If($LPlatform=Windows )
  <>WIN_fIsWindows := True
Else
  <>WIN_fIsWindows := False
End if
    `Konec metody
```

3. Vytvořte metodu projektu WIN_InitializeWindowVariables.
4. Zkopírujte z WIN_InitializeWindowVariables nebo napište:

```
If (False)
    ` Metoda: WIN_InitializeWindowVariables
    ` Modul: Window Management
    ` Kurz ACI University
    ` Generická procedura
    ` Vytvořeno: Jim Steinman
    ` Datum:  15/1/97

    ` Účel: Initializuje Meziprocesní proměnné pro ovládání oken

  <>fGeneric:=True
  <>f_Version6x10:=True
  <>fJ_Steinman:=True

End if

C_LONGINT(<>WIN_LStaardResizable;<>WIN_LModal;<>WIN_L1Pixel;<>WIN_LShadow)
C_LONGINT(<>WIN_LStaardNoResize;<>WIN_LMovableModal;<>WIN_LStaardZoom)
C_LONGINT(<>WIN_LDeskAccessory;<>WIN_LFloatingPalette;<>WIN_LFloatingScrollCoefficient)
C_LONGINT(<>WIN_LFloatingTitleCoefficient;<>WIN_LFloatingToggelableCoef;
                <>WIN_LFloatingZoomCoefficient)

C_LONGINT(<>WIN_DesignerSize;<>WIN_DesignerSizeCascade;<>WIN_LCentered;<>WIN_LCascade)
C_LONGINT(<>WIN_LUpper;<>WIN_LLower;<>WIN_LUpperLeft;<>WIN_LUpperRight)
C_LONGINT(<>WIN_LLowerLeft;<>WIN_LLowerRight;<>WIN_LFillScreen;<>WIN_LOffScreen)

C_LONGINT(<>WIN_LMenuBarHeight;<>WIN_LToolBarHeight;<>WIN_LWindowHorizonOpen)
C_LONGINT(<>WIN_LWindowVerticalOpen)
```

```
C_REAL(<>WIN_rResize)

    ` Initializuje konstanty typu okna
<>WIN_LStaardResizable:=0      ` Staardní nemodalní okno bez uzavírátka
<>WIN_LModal:=1                    ` Staardní modalní okno
<>WIN_L1Pixel:=2                   ` 1 bodové modalní okno
<>WIN_LShadow:=3                ` Stínované modalní okno
<>WIN_LStaardNoResize:=4       ` Staardní nemodalní okno bez uzavírátka a zvětšování
<>WIN_LMovableModal:=5          ` Pohyblivé modální okno
<>WIN_LStaardZoom:=8            ` Staardní nemodalní okno se zvětšováním
<>WIN_LDeskAccessory:=16       ` DA styl okna
<>WIN_LFloatingPalette:=-720    ` Plovoucí okno bez posuvníku
<>WIN_LFloatingScrollCoefficient:=-4`Plovoucí okno bez posuvníku
<>WIN_LFloatingTitleCoefficient:=-2  ` Plovoucí okno s titulem
<>WIN_LFloatingToggelableCoef:=-1  ` Plovoucí okno s hlavičkou
<>WIN_LFloatingZoomCoefficient:=-8` Plovoucí okno se zvětšením

    ` Initializace konstant pozice okna
<>WIN_DesignerSize:=-2             ` Použít návrhářem zvolenou velikost okna a XY pozici
<>WIN_DesignerSizeCascade:=-1 ` Použít návrhářem zvolenou velikost okna a kaskádově
<>WIN_LCentered:=0                 ` Centrované okno
<>WIN_LCascade:=1               ` Kaskádová okna
<>WIN_LUpper:=2                    ` Horní 1/3 a středěný
<>WIN_LLower:=3                    ` Dolní 1/3 a středěný
<>WIN_LUpperLeft:=4             ` Vrchní levý roh
<>WIN_LUpperRight:=5               ` Vrchní pravý roh
<>WIN_LLowerLeft:=6             ` Levý dolní roh
<>WIN_LLowerRight:=7               ` Pravý dolní roh
<>WIN_LFillScreen:=8            ` výplň obrazovky
<>WIN_LOffScreen:=10               ` okno mimo obrazovku

    ` Initializace vrchu okna
<>WIN_LMenuBarHeight :=40     `Výška nabídky
<>WIN_LToolBarHeight:=31       ` Výška palety nástrojů

<>WIN_LWindowHorizonOpen:=3     ` Začátek pozice horizontálního kaskádování
<>WIN_LWindowVerticalOpen:=3    ` Začátek pozice vertikálního kaskádování

<>WIN_rResize:=1.35  ` hodnota pro funkci změny velikosti na Windows™

    `konec metody
```

5. Nahraďte metodu Při spuštění následujícín kódem z On Startup:

```
` Metoda databáze: On Startup
` Kurz ACI University
` Vytvořeno: Jim Steinman
` Datum:15/1/97

` Účel: Nastaví prostředí na uživatelem zvolené hodnoty

` <>f_Version6x00 :=True ` verze z kurzu úvod do 4d
```

```
<>f_Version6x10 :=True          ` Verze z kurzu Programování ve 4D
<>fJ_Steinman :=True            ` Autor ACI University
      ` používáno pro označení generické metody
      ` <>fGeneric := True
```

- WIN_IsWindows ` Zkontroluje zda běží na Windows™
- WIN_InitalizeWindowVariables ` Initializuje proměnné pro otevíraní oken

```
SET WINDOW TITLE ("ACI Video")

    `Konec metody
```

6. Vytvořte metodu projektu WIN_WindowTypeOffsets.
7. Kopírujte ze souboru WIN_WindowTypeOffsets, nebo napište následující:

```
If (False)
    ` Metoda: WIN_WindowTypeOffsets (longint;pointer;pointer)
    ` Modul: Window Management
    ` Kurz ACI University
    ` Generická procedura
    ` Vytvořeno: Jim Steinman
    ` Datum:15/1/97

    ` Účel: Navrátí offsety pro každý typ okna

  <>fGeneric:=True
  <>f_Version6x10:=True
  <>fJ_Steinman:=True

End if

    ` Deklarace parameterů
C_LONGINT($1;$WIN_LWindowType)    ` Požadovaný typ okna
C_POINTER($2)                     ` ukazatel na proměnnou k navracení horizontálního ofsetu
C_POINTER($3)                     ` ukazatel na proměnnou k navracení vertikálního ofsetu

    ` Deklarace lokálních proměnných
C_LONGINT($WIN_LHorizontal)  ` Horizontalni offset
C_LONGINT($WIN_LVertical)  ` Verticalni offset


$WIN_LWindowType := Abs($1) ` Získání absolutní hodnoty typu okna, plovoucí budou stejné velikosti

If (<>WIN_fIsWindows ) `windows offsety
  Case of
    : ($WIN_LWindowType=<>WIN_LStaardResizable) | ($WIN_LWindowType=<>WIN_LStaardZoom)
      $WIN_LHorizontal := 6
      $WIN_LVertical := 28
    :($WIN_LWindowType=<>WIN_LStaardNoResize) | ($WIN_LWindowType=<>WIN_LDeskAccessory)
      $WIN_LHorizontal := 3
      $WIN_LVertical := 25
    : ($WIN_LWindowType=<>WIN_LModal)
      $WIN_LHorizontal := 7
      $WIN_LVertical := 10
```

```
      : ($WIN_LWindowType=<>WIN_L1Pixel) | ($WIN_LWindowType=<>WIN_LShadow)
         $WIN_LHorizontal := 3
         $WIN_LVertical := 6
      : ($WIN_LWindowType=<>WIN_LMovableModal)
         $WIN_LHorizontal := 7
         $WIN_LVertical := 29
      Else   ` floating window palette
         $WIN_LHorizontal := 3
         $WIN_LVertical := 20
   End case
Else   ` Macintosh Offsety


   Case of
      :($WIN_LWindowType=<>WIN_LStaardResizable) | ($WIN_LWindowType=
                  <>WIN_LStaardNoResize) | ($WIN_LWindowType=
                  <>WIN_LStaardZoom) | ($WIN_LWindowType=<>WIN_LDeskAccessory)
         $WIN_LHorizontal := 3
         $WIN_LVertical := 21
      : ($WIN_LWindowType=<>WIN_LModal)
         $WIN_LHorizontal := 10
         $WIN_LVertical := 10
      : ($WIN_LWindowType=<>WIN_L1Pixel) | ($WIN_LWindowType=<>WIN_LShadow)
         $WIN_LHorizontal := 3
         $WIN_LVertical := 4
      : ($WIN_LWindowType=<>WIN_LMovableModal)
         $WIN_LHorizontal := 8
         $WIN_LVertical := 25
      Else   ` floating window palette
         $WIN_LHorizontal := 3
         $WIN_LVertical := 13
   End case

End if

$2-> := $WIN_LHorizontal  ` navrací the offset
$3-> := $WIN_LVertical
      `Konec metody
```

6. Vytvořte metodu projektu WIN_LNewWindow.
7. Kopírujte ze douboru WIN_LnewWindow, nebo napište:

```
If (False)
      ` Metoda: WIN_LNewWindow(width;height;position;type;title;closebox;resize)
      ` Modul: Window Management
      ` Kurz ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum:15/1/97

      ` Účel:   Tato proceduraotevře okno, kdekoliv je potřeba

   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
```

```
End if

    ` Deklarace parametrů
C_LONGINT($0)          ` ID okna k návratu do metody volání
C_LONGINT($1;$LWindowWidth )     ` Šířka vnitřního obdélníku (volitelné; výchozí celá)
C_LONGINT($2;$LWindowHeight )    ` Výška vnitřního obdélníku (volitelné; výchozí celá)
    ` -1 v těchto parametrech určuje vývojářem stanovenou velikost
    ` INPUT FORM musí být nastaven před voláním této metody
C_LONGINT($3;$LScreenPosition )  ` Pozice okna (volitelné; výchozí = centrované)
    ` V události bude užita velikost stanovená vývojářem
    ` Parametery $1 & $2 reprezentují vrchní horní rok jestliže pozice je -2
C_LONGINT($4;$LWindowType )      ` Typ okna (volitelné; výchozí = modal dialog)
C_STRING(255;$5;$sWindowTitle )  ` Titul okna (volitelné)
C_STRING(31;$6;$sCloseBoxMethod )      ` Procedura uzavíracího boxu (volitelné)
C_BOOLEAN($7;$fResizeForWindows )   ` Potlačit změnu velikosti (volitelné; True = potlačit)

    ` Deklarace lokálních proměnných
C_BOOLEAN($fProcessVisible)        ` Uschování flag
C_LONGINT($LNumberParmeters)           ` Počet parametrů
C_LONGINT($LScreenHeight)          ` Výška obrazovky nebo aplikačního okna
C_LONGINT($LScreenWidth)           ` Šířka obrazovky nebo aplikačního okna
C_LONGINT($LScreenTotalHeight)     ` Celková výška obrazovky, nebo Mac všech obrazovek
C_LONGINT($LScreenTotalWidth)      ` Celková šířka obrazovky, nebo Mac všech obrazovek
C_LONGINT($LWindowBottom)          ` Spodní okraj vnitřního obdélníku
C_LONGINT($LWindowID)              ` ID okna velikosti okna návrháře
C_LONGINT($LWindowLeft)            ` Levý okraj vnitřního obdélníku
C_LONGINT($LWindowRight)           ` Pravý okraj vnitřního obdélníku
C_LONGINT($LWindowTop)             ` Vrchní okraj vnitřního obdélníku

$LScreenWidth := Screen width
$LScreenHeight := Screen height

$LNumberParmeters := Count parameters

    `Define default values
$LWindowWidth := $LScreenWidth-6       ` Výchozí celá obrazovka
$LWindowHeight := $LScreenHeight-6     ` Výchozí celá obrazovka
$LScreenPosition := <>WIN_LCentered          ` Výchozí Centered
$LWindowType := <>WIN_LStaardResizable   ` Výchozí staardní nemodální s změnou  vel
$sWindowTitle := ""                    ` Výchozí bez titulu
$sCloseBoxMethod := ""                 ` Výchozí uzavírací box ne
$fResizeForWindows := False            ` Výchozí ne změna velikosti

    ` Předefinice na základě paramentrů
If ($LNumberParmeters>0)
   $LWindowWidth := $1
   If ($LNumberParmeters>1)
      $LWindowHeight := $2
      If ($LNumberParmeters>2)
         $LScreenPosition := $3
         If ($LNumberParmeters>3)
            $LWindowType := $4
            If ($LNumberParmeters>4)
```

```
            $sWindowTitle := $5
            If ($LNumberParmeters>5)
               $sCloseBoxMethod := $6
               If ($LNumberParmeters>6)
                  $fResizeForWindows := $7
               End if
            End if
         End if
      End if
   End if
End if

   ` Získat offset okna
WIN_WindowTypeOffsets ($LWindowType;->WIN_LHorizonOffset;->WIN_LVerticalOffset)

If (($LWindowWidth=-1) & ($LWindowHeight=-1)) ` Potřeba najít velikost návrháře
   ` K zajištění otevření mimo obrazovku použít volitelnou hvězdičku
   $LScreenTotalWidth := Screen width(*) - 2
   $LScreenTotalHeight := Screen height(*) - 2

   $LWindowID := Open window($LScreenTotalWidth;$LScreenTotalHeight;-1;-
1;<>WIN_L1Pixel)
                  ` Otevřít okno velikosti návrháře

   GET WINDOW RECT($LWindowLeft;$LWindowTop;$LWindowRight;$LWindowBottom;
$LWindowID)
   CLOSE WINDOW
   $LWindowWidth := $LWindowRight-$LWindowLeft
   $LWindowHeight := $LWindowBottom-$LWindowTop
End if


   ` Adjustovat šířku a výšku při změně
If ($fResizeForWindows)
      ` See WIN_InitializeWindowVariables for <>WIN_rResize
   $LWindowWidth := $LWindowWidth*<>WIN_rResize
      ` See WIN_InitializeWindowVariables for <>WIN_rResize
   $LWindowHeight := $LWindowHeight*<>WIN_rResize
End if

Case of
   : (Abs($LScreenPosition)=<>WIN_LCascade)      ` 1 kaskáda

   If(($LWindowWidth+<>WIN_LWindowHorizonOpen+WIN_LHorizonOffset)>$LScreenWidth
)

    | (($LWindowHeight+<>WIN_LWindowVerticalOpen+<>WIN_LMenuBarHeight+
                  <>WIN_LToolBarHeight+WIN_LVerticalOffset)>$LScreenHeight)
         ` okno by se otevřelo mimo, levý vrchní roh znovu
      <>WIN_LWindowHorizonOpen := 3
      <>WIN_LWindowVerticalOpen := 3
```

```
      End if
      $LWindowLeft := <>WIN_LWindowHorizonOpen+WIN_LHorizonOffset
      $LWindowTop := <>WIN_LWindowVerticalOpen+<>WIN_LMenuBarHeight+
                  <>WIN_LToolBarHeight+ WIN_LVerticalOffset

         ` zvýšit odpovídající parametry
      <>WIN_LWindowHorizonOpen := <>WIN_LWindowHorizonOpen+20
      <>WIN_LWindowVerticalOpen := <>WIN_LWindowVerticalOpen+20

   : ($LScreenPosition=<>WIN_LUpper)          ` 2 1/3 od vrchu a centrované
      $LScreenHeight := $LScreenHeight\3
      $LScreenWidth := $LScreenWidth\2
      $LWindowLeft := $LScreenWidth-($LWindowWidth\2)
      $LWindowTop := $LScreenHeight-($LWindowHeight\2)

   : ($LScreenPosition=<>WIN_LCentered)       ` 0 centrované
      $LScreenHeight := $LScreenHeight\2
      $LScreenWidth := $LScreenWidth\2
      $LWindowLeft := $LScreenWidth-($LWindowWidth/2)
      $LWindowTop := $LScreenHeight-($LWindowHeight/2)

   : ($LScreenPosition=<>WIN_DesignerSize)    ` -2 užít návrhářovu velikost a XY position
      If ($LNumberParmeters>0)
         $LWindowLeft := $1
         $LWindowTop := $2
      Else
         $LWindowLeft := 20
         $LWindowTop := <>WIN_LMenuBarHeight+<>WIN_LToolBarHeight+20
      End if

   : ($LScreenPosition=<>WIN_LUpperLeft)          ` 4 vrchní levý roh
      $LWindowLeft := WIN_LHorizonOffset
      $LWindowTop :=
<>WIN_LMenuBarHeight+<>WIN_LToolBarHeight+WIN_LVerticalOffset

   : ($LScreenPosition=<>WIN_LUpperRight)         ` 5 vrchní pravý roh
      $LWindowLeft := $LScreenWidth-$LWindowWidth-WIN_LHorizonOffset
      $LWindowTop :=
<>WIN_LMenuBarHeight+<>WIN_LToolBarHeight+WIN_LVerticalOffset

   : ($LScreenPosition=<>WIN_LLower)          ` 3 dolní centrované
      $LWindowLeft := ($LScreenWidth-$LWindowWidth)/2
      $LWindowTop := ($LScreenHeight-$LWindowHeight)/1.5

   : ($LScreenPosition=<>WIN_LLowerLeft)          ` 6 spolní levý roh
      $LWindowLeft := WIN_LHorizonOffset
      $LWindowTop := $LScreenHeight-$LWindowHeight-WIN_LHorizonOffset-2

   : ($LScreenPosition=<>WIN_LLowerRight)         ` 7 spodní pravý roh
      $LWindowLeft := $LScreenWidth-$LWindowWidth-WIN_LHorizonOffset
      $LWindowTop := $LScreenHeight-$LWindowHeight-WIN_LHorizonOffset-2
```

```
   : ($LScreenPosition=<>WIN_LFillScreen)          ` 8 vyplnit obrazovku nebo okno aplikce
      $LWindowLeft := WIN_LHorizonOffset
      $LWindowTop :=
<>WIN_LMenuBarHeight+<>WIN_LToolBarHeight+WIN_LVerticalOffset
      $LWindowWidth := $LScreenWidth-(WIN_LHorizonOffset*2)
      $LWindowHeight := $LScreenHeight-WIN_LVerticalOffset-$LWindowTop

   : ($LScreenPosition=<>WIN_LOffScreen)          ` 10 okno mimo obrazovku
      $LWindowLeft := 0
      $LWindowTop := 0
      $LScreenWidth := 1
      $LScreenHeight := 1

End case

      ` zajištění, že se okno neotevře vyšší než očekáváno
If
(($LWindowTop<(<>WIN_LMenuBarHeight+<>WIN_LToolBarHeight+WIN_LVerticalOffset))
&
                  ($LScreenPosition # <>WIN_LOffScreen))
   $LWindowTop := <>WIN_LMenuBarHeight+<>WIN_LToolBarHeight+WIN_LVerticalOffset
End if

      ` zajištění, že se okno neotevře příliš vlevo než očekáváno
If (($LWindowLeft<WIN_LHorizonOffset) & ($LScreenPosition # <>WIN_LOffScreen))
   $LWindowLeft := WIN_LHorizonOffset
End if

If ($LScreenPosition>-2)
   $LWindowRight := $LWindowLeft+$LWindowWidth
   $LWindowBottom := $LWindowTop+$LWindowHeight
Else
   $LWindowRight := -1
   $LWindowBottom := -1
End if

$0 := Open window($LWindowLeft;$LWindowTop;$LWindowRight;$LWindowBottom;
   $LWindowType;$sWindowTitle;$sCloseBoxMethod)
      `Konec metody
```

8. Upravte metodu projektu GEN_ModifySelection, aby používala metodu WIN_LNewWindow následujícím způsobem:

```
If (False)
      ` Metoda: GEN_ModifySelection
      ` Kurz ACI University
      ` Generická
      ` Vytvořeno: Jim Steinman
      ` Datum:15/1/97

      ` Účel: Zobrazí výstupní formulář se seznamem záznamů pro tabulku určenou
      ` pTable
```

```
   <>fGeneric :=True
   <>f_Version6x10 :=True
   <>fJ_Steinman :=True

End if
```

- C_LONGINT($LWindowID)

  MENU BAR (3) ` Změní záhlaví

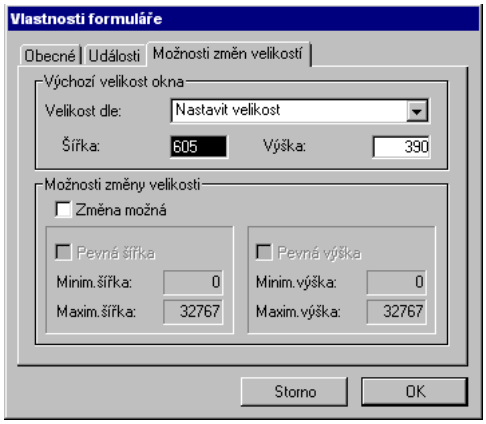
- INPUT FORM(pTable->;"Vstupní";*) ` Příprava k použití navrženého okna
  OUTPUT FORM(pTable->;"Výstupní")

- $LWindowID := WIN_LNewWindow (-1;-1;<>WIN_LCascade;<>WIN_LStaardNoResize)

```
ALL RECORDS(pTable->)  ` Změní platný výběr
CREATE EMPTY SET(pTable->; "UserSet")
WIN_OutputWindowTitle
MODIFY SELECTION(pTable->;*)  ` Ukáže záznamy na obrazovku
CLOSE WINDOW
   `Konec metody
```

9. Otevřete každý vstupní formulář a nastavte v Formulář ¬ Vlastnosti formuláře velikosti následujícím způsobem:

   Pro všechny formuláře vyberte Velikost dle – Nastavit velikost:



Vstupní formulář	Šířka	Výška
Zákazníci	605	390
Faktury	605	400
Produkty	515	350

10. Restartujte databáze a vyzkoušejte nové metody pro otevírání kaskádových oken.

15. Co musí být dodrženo při přidávání tabulek

Když přidáváte novou tabulku do vaší databáze je několik věcí, které musíte udělat, aby jste vyhověli požadavkům, které jste vložili do generických procedur.

1. Vytvořte tabulky a pole ve struktuře.
2. Natáhněte libovolné potřebné vztahy mezi tabulkami.
3. Vytvořte vstupní a výstupní formuláře.
   (Buď přejděte do Prostředí uživatele a pak zpět do Prostředí návrháře nebo vytvořte každý formulář ručně.)
4. Pojmenujte vstupní a výstupní formuláře „Vstupní“ a „Výstupní“.
5. Vytvořte metodu formuláře pro všechny vstupní formuláře, která bude volat metodu WIN_InputWindowTitle při události On Load.
6. Vytvořte metodu formuláře pro všechny výstupní formuláře, která bude volat metodu WIN_OutputWindowTitle a MENU BAR(3) při události On Close Detail.
7. Přiřaďte Záhlaví # -2 to ke všem formulářům „Vstupní“ a „Výstupní“.
8. Vytvořte další formuláře pro používání s položkami nabídek Záznamy ¬ Import a Záznamy ¬ Export.
   (Pojmenujte tento formulář importu/exportu jako ImportExport.)
9. Vytvořte vstupní formulář pro užití v položce nabídky Záznamy ¬ Dotaz dle příkladu.
   (Pojmenujte formulář Dotazy.)
10. Vytvořte výstupní formulář pro užití s položkou nabídky Zprávy ¬ Speciální zprávy.
    (Pojmenujte formulář Tisk výstupní.)
11. Udělejte vše co je potřebné pro přiřazení výchozích hodnot pro nové záznamy.
12. Udělejte vše co je potřebné pro přiřazení hodnot do polí s vlastností jedinečné.
13. Udělejte vše co je potřebné pro přiřazení hodnot polím vytvářejícím vztah nových záznamů k rodičovským záznamům.
14. Otevřete libovolné záhlaví nabídek s nabídkou Soubor a vložte do ní položku nabídky pro novou tabulku.
15. Pro novou položku nabídky přiřaďte metodu M_NázevTabulky.

16. Vytvořte novou metodu M_NázevTabulky následujícím způsobem:

```
If (False)
      ` Metoda: M_TheTable
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Napište své jméno
      ` Datum: Napište datum vytvoření

      ` Účel: Nastaví pTable pro [NázevTabulky] pro užití ve všech metodách a voláních


   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

pTable := -> [NázevTabulky]

GEN_ModifySelection
   ` Konec metody
```

16. Vytvoření uživatelských dialogů

S pomocí 4D vytvoříte uživatelský dialog. Vytvoření dialogů ve 4D je překvapivě jednoduché, protože můžete použít editor formulářů, tentýž editor, který jste použili pro vytváření vtupních a výstupních formulářů.

Někteří vyši uživatelé považují editor dotazů za příliš složitý, takže pro tyto lidi vytvoříte zjednodušený dialog pro dotazy. Nejdříve zde použijete vestavěný příkaz Request k žádosti o zadání informace. Tento příkaz umožní uživaeli napsat textovou ingormaci do níž vloží kategorii videa, jako Komedie nebo Drama. Později pro dialog přidáme více rysů.

Doposud jsme hlavně používali tabulky [Zákazníci] a [Faktury]. Nyní použijeme záznamy [Produkty].

16.1. Použití příkazu Request k získání informace od uživatele.

1. Vytvořte tlačítko ve formuláři [Produkty];"Výstupní" s následujícími vlastnostmi:

Název:	bQueryCategory
Typ:	Tlačítko
Automatická akce:	Žádná akce
Popiska:	Kategorie?
Metody událostí objektu:	Klepnutí a poklepání

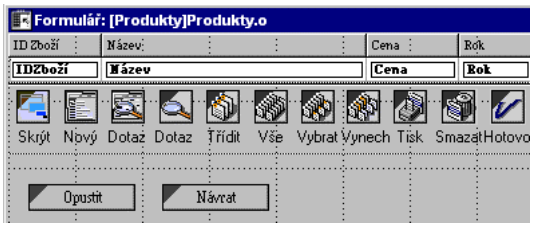
2. Vytvořte metodu objektu pro tlačítko následovně:

```
If (False)
      ` Metoda objektu: bQueryCategory
      ` Kurz ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Vytvoří rys dotaz na kategorii

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

$sCategory := Request ("Hledat produkty kategorie:")
QUERY ([Produkty]; [Produkty]Kategorie = $sCategory)
WIN_OutputWindowTitle
   ` Konec metody objektu
```



16.2. Váš vlastní dialog ve třech krocích

K zobrazení dialogu musíte provést tři věci:

- Otevřít okno.
- Zobrazit formulář, který jste pro tento účel vytvořili.
- Uzavřít okno.

16.3. Formulář pro záznamy vs. příkaz DIALOG

Záznamy se normáně zobrazují pomocí vstupních a výstupních formulářů. Tyto vstupní a vyýstupní formuláře se zobrazují „sami“ jako výsledek příkazu MODIFY SELECTION. Ve výstupním formuláři se seznamem záznamů může uživatel poklepat na záznam a zobrazí se mu tento záznam ve vstupním formuláři.

Dialog je používán, když potřebuje uřivateli předat informace, nebo potřebujete, aby je zadal pro něco jiného, než jsou pole databáze. Například zde pro zadání kategorií do podmínky dotazu.

16.4. Otevření okna pro váš dialog

V kódu použitém níže uvidíle použití příkazů OPEN WINDOW a CLOSE WINDOW s příkazem Dialog mezi nimi. Příkaz dialog zobrazuje formulář dle vašeho výběru. Jakékoliv pole na formuláři je zobrazeno, ale není dostupné, takže uživatel nemůže provádět změny v záznamech. Podobně vložené formuláře jsou zobrazeny, ale uživatel nemůže zadávat data do záznamů. Formulář dialogu zůstane na obrazovce dokud uživatel neklepne na tlačítko s automatickou akcí buď Přijmout nebo Storno. Po klepnutí na tlačítko je formulář uzavřen, ale zbývá otevřené okno, takže za příkazem DIALOG musí následovat příkaz k uzavření okna CLOSE WINDOW.

16.5. 4D vs. 4D First

Poznamenejme, že toto je způsob jakým pracuje plná 4D. Jestliže otevřete databázi vytvořenou 4D First uvidíte, že je příkaz DIALOG používán samostatně. 4D First má některé rysy zjednodušené, takže nepotřebuje pro oevření okna s dialogem OPEN WINDOW a CLOSE WINDOW. Je to však za cenu, že tato okna (velikost a umístění) nemůžete řídit. 4D First otevře okno tak, aby se do něj vešly všechny objekty formuláře, třeba i ty co nechcete, aby uživatel viděl. 4D vám dává tedy více možností a kontroly než 4D First.

16.6. Vytvoření formuláře dialogu

1. Vytvořte nový formulář pro tabulku [zDialogs] pojmenovaný DotazKategorie.
2. Vytvořte následující objekty:

Název:	bOK
Typ:	Tlačítko
Automatická akce:	Přijmout
Popiska:	OK
Metoda objektu:	Ne
Styl:	Tlačítka

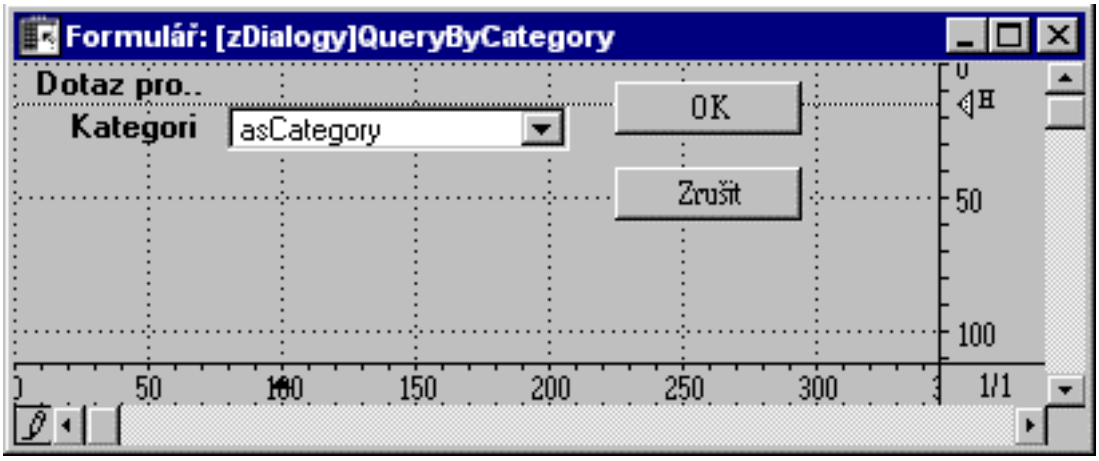
Název:	bCancel
Typ:	Tlačítko
Automatická akce:	Zrušit
Popiska:	Storno
Metoda objektu:	Ne
Styl:	Tlačítka

Název:	sCategory
Typ:	Enterable
Automatická akce:	Ne
Popiska:	Žádna
Metoda objektu:	Ne
Styl:	Pole vstupní

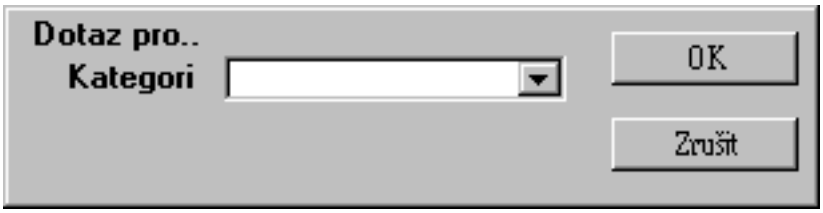
Název:	Dotaz pro …
Typ:	Statický text
Styl:	Pole vstupní

Název:	Kategorie:
Typ:	Statický text
Styl:	Pole vstupní

3. Klepněte na objekt sCategory.
4. Dejte mu vzhled ponořený.
5. Ujistěte se, že pořadí vstupu začíná sCategory. (Je to nezbytné pro uživatele Windows)



V prostředí uživatele by měl váš dialog vypadat následovně:



16.7. Použití proměnných ve formulářích

V dialogu můžete použít proměnné, ale musí to být minimálně proměné procesu a nikoliv lokální proměnné s počátečním znakem dolar. V předchozí metodě objektu jsme použili lokální proměnou $sCategory k zachycení požadavku napsaného uživatelem do dialogu Request. V dalším tento znak dolaru odstraníme a přeměníme tak proměnnou na proměnnou procesu.

V dialogu pro zadání informace musíte použít proměnno a ne pole. Pole můžete pouze zobrazit, ale uživatel do nich nemůže zadat data. Dialog není určen pro žádnou tabulku, takže pole v pořadí vstupu nepracují.

16.8. Zobrazení dialogu

K zobrazení dialogu msíte napsat kód pro otevření okna, zobrazujícího dialogový formulář a posléze okno uzavřít. Formulář zůstane na obrazovce dokud uživatel neklepne na tlačítko automatické akce Přijmout nebo Zrušit, nebo na tlačítka kde je použit příkaz ACCEPT nebo CANCEL. Po klapnutí na takovéto tlačítko 4D automaticky nastaví systémovou proměnnou OK na 1 nebo 0 podle akce.

16.9. Změna lokální proměnné na proměnnou procesu

Všimněte si v kódu níže, že provádíme změnu proměnné $sCategory na sCategory. Lokální proměnné nemohou být v dialogu použity (protože jejich doba života je pouze trvání metody), takže musíme změnit typ proměnné na proměnnou procesu.

16.10. Proměnná procesu trvá

Proměnná procesu zůstává v paměti (RAM) uvnitř jednoho procesu, dokud proces neskončí. Jestliže spustíte tentýž dialog ještě jednou v tomtéž procesu uvidíte, že se objeví s výchozí hodnotou z předchozího zadávání uživatele.

16.10.1. Úprava metody objektu pro použití s formulářem dialogu

1. Otevřete výstupní formulář pro [Produkty].
2. Otevřete metodu objektu bQueryCategory.
3. Upravte metodu následujícím způsobem:

```
If (False)
      ` Metoda objektu: bQueryCategory
      ` Kurz ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Vytvoří rys dotaz na kategorii

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if


•     C_LONGINT ($LWidth; $LHeight;$LWindowID )
•     C_STRING (15;sCategory)

•     $LWidth := 300                          `Požadovaná šířka okna.
•     $LHeight := 70                          `Požadovaná výška
•     $LWindowID := WIN_LNewWindow ($LWidth; $LHeight;<>WIN_LUpper;<>WIN_LModal)

•     DIALOG ([zDialogy]; "DotazKategorie")
•     CLOSE WINDOW

•        ` $sCategory := Request ("Hledat produkty kategorie:")
•        ` QUERY ([Produkty]; [Produkty]Kategorie = $sCategory)
•     QUERY ([Produkty]; [Produkty]Kategorie = sCategory)
      WIN_OutputWindowTitle
        ` Konec metody
```

4. Vyzkoušejte tlačítko v prostředí Vlastní nabídky.

16.11. Vylepšení metody objektu bQueryCategory

Nyní vylepšeme tuto metodu. Jestliže jste studovali tuto metodu, objevili jste, že zcela ignoruje akci provedenou uživatelem. Vaše metody objektu provede dotaz jestli uživatel klepne na tlačítko OK i Storno. Dále vložíte blok if...end if pro kontrolu akce uživatele.

16.11.1. Úprava metody objektu bQueryCategory Object k určení akce Zrušit

1. Vložte blok If…End if do metody objektu následovně:

```
If (False)
    ` Metoda objektu: bQueryCategory
    ` Kurz ACI University
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Vytvoří rys dotaz na kategorii

  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if

C_LONGINT ($LWidth; $LHeight;$LWindowID )
C_STRING (15;sCategory)

$LWidth := 300                          `Požadovaná šířka okna.
$LHeight := 70                          `Požadovaná výška
$LWindowID := WIN_LNewWindow ($LWidth; $LHeight;<>WIN_LUpper;<>WIN_LModal)

DIALOG ([zDialogy]; "DotazKategorie")
CLOSE WINDOW

  ` $sCategory := Request ("Hledat produkty kategorie:")
  ` QUERY ([Produkty]; [Produkty]Kategorie = $sCategory)
If ((OK = 1) & (Length(sCategory)>0))
  QUERY ([Produkty]; [Produkty]Kategorie = sCategory)
  WIN_OutputWindowTitle
End if
  ` Konec metody
```

2. Přejděte do prostředí Vlastní nabídky a opět vyzkoušejte.

16.12. Použití rozevírací nabídky k výběru kategorie

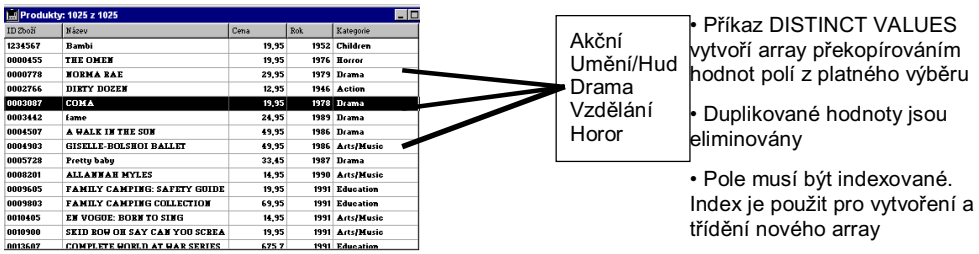
Váš dialog nyní pracuje správně. Můžete jej však chtít poměkud přívětivější k uživateli, tím že uživateli poskytnete možnost vybrat si z dostupných hodnot pomocí rozevírací nabídky/seznamu. V tomto případě si uživatel nemusí pamatovat a místo zadání zcela přesné hodnoty vybere ze seznamu pomocí myši.

Rozevírací nabídky/seznamy, sloupcové nabídky, texty se seznamy zobrazují skupiny hodnot z array. Array podobně jako proměnné „žijí“ pouze v paměti počítače a když 4D ukončíte zmizí, jsou vymazány. Proměnné mají pouze jednu hodnotu např. „Drama“, zatímco Array může mít libovolný počet hodnot např. „Drama“, „Horor“, „Komedie“...současně. Array pro tyto objekty musí již existovat v paměti 4D, když je objekt zobrazen na obrazovku. Tyto objekty nemohou zobrazovat přímo seznamy záznamů můžete však tyto záznamy překopírovat do array. Způsob jak kopírovat jedno pole platného výběru do array a eliminovat přitom zdvojení hodnot v array je příkaz DISTINCT VALUES.

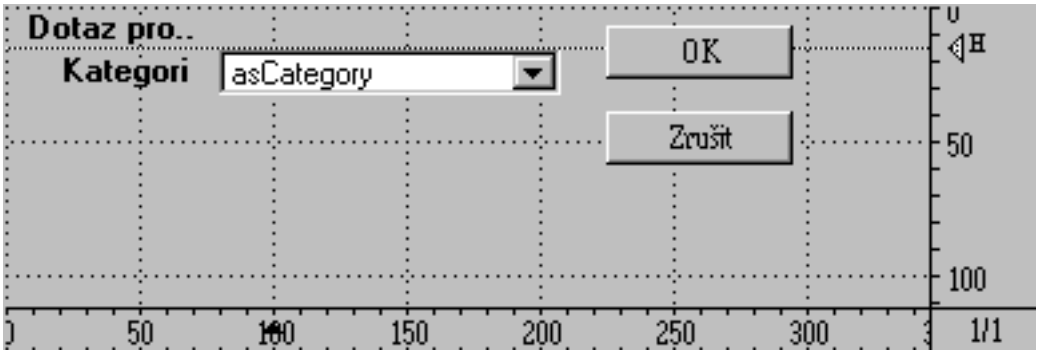
16.13. Příkaz DISTINCT VALUES

Příkaz DISTINCT VALUES vytvoří array na základě platného výběru. Příkaz kopíruje určené pole nejdříve z prvního záznamu a umístí jej do prvního prvku array, pak příkaz pokračuje na další záznamy. Jestliže hodnota v polí dalšího záznamu není již v arrary, příkaz zkopíruje pole do dalšího prvku array. Pokud se hodnota v poli záznamu v array již vyskytuje je záznam přeskočen. Tímto způsobem se pokračuje do konce platného výběru. Nakonec je array setříděn abecedně. Příkaz DISTINCT VALUES vyžaduje pole:

- Typ Alfa
- Indexované



Váš dialog bude vypadat následovně:



16.13.1. Vytvoření Nabídky/seznam k dotazům na kategorii

1. Otevřete formulář [zDialogy];"DotazKategorie".
2. Poklepejte na proměnnou sCategory.
3. Změňte nastavení následovně:

Název	asCategory
Typ	Nabídka/Seznam

4. Otevřete metodu objektu bQueryCategory a upravte ji následujícím způsobem:

```
If (False)
      ` Metoda objektu: bQueryCategory
      ` Kurz ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Vytvoří rys dotaz na kategorii

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT ($LWidth; $LHeight;$LWindowID )
•        ` C_STRING (15;sCategory)

•        DISTINCT VALUES ([Produkty]Kategorie; asCategory)

$LWidth := 300                    `Požadovaná šířka okna.
$LHeight := 70                    `Požadovaná výška
```

```
$LWindowID := WIN_LNewWindow ($LWidth; $LHeight;<>WIN_LUpper;<>WIN_LModal)

DIALOG ([zDialogy]; "DotazKategorie")
CLOSE WINDOW

   ` $sCategory := Request ("Hledat produkty kategorie:")
   ` QUERY ([Produkty]; [Produkty]Kategorie = $sCategory)
   If ((OK = 1) & (asCategory>0))
      `QUERY ([Produkty]; [Produkty]Kategorie = sCategory)
      QUERY ([Produkty]; [Produkty]Kategorie = asCategory{asCategory})
   WIN_OutputWindowTitle
End if
   ` Konec metody
```

5. Přejděte zpět do okna struktury a přesvědčte se, že je pole kategorie indexováno.

Kvíz

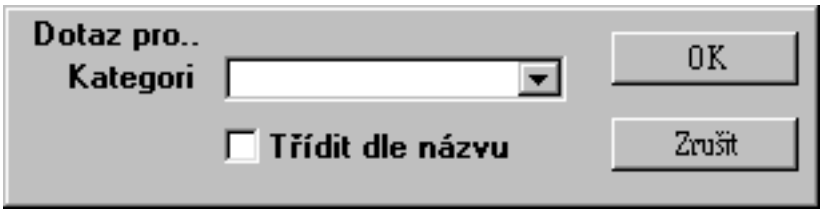
- Proč se vždy array vytvořený pomocí DISTINCT VALUES objeví v abecedním pořadí?
- Na jakou chybu 4D nejpravděpodobněji narazíte, když se poprvé pokusíte použít příkaz DISTINCT VALUES?

16.14. Použití zaškrtávacího políčka v dialogu

Dejme uživateli možnost třídit vyhledané záznamy podle názvu videa. Vytvořte zaškrtávací políčko, které bude nastavovat příznak Třídit dle názvu a pak upravte svou metodu, aby tento příznak testovala (je zapnut či vypnut)

Poznamenejme, že tak jako pole je zaškrtávací políčko nastaveno buď na true nebo false (pravda či nepravda). Na druhou stranu jako proměnná je zaškrtávací políčko buď 1 (zapnuto) nebo 0 (vypnuto).

Váš dialog bude vypadat následovně:



16.14.1. Vytvoření zaškrtávacího políčka pro třídění dle názvu

1. Otevřete metodu objektu a upravte ji následujícím způsobem:

```
If (False)
      ` Metoda objektu: bQueryCategory
      ` Kurz ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Vytvoří rys dotaz na kategorii

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT ($LWidth; $LHeight;$LWindowID )
      ` C_STRING (15;sCategory)

DISTINCT VALUES ([Produkty]Kategorie; asCategory)

$LWidth := 300                          `Požadovaná šířka okna.
$LHeight := 70                          `Požadovaná výška
$LWindowID := WIN_LNewWindow ($LWidth; $LHeight;<>WIN_LUpper;<>WIN_LModal)

DIALOG ([zDialogy]; "DotazKategorie")
CLOSE WINDOW

   ` $sCategory := Request ("Hledat produkty kategorie:")
   ` QUERY ([Produkty]; [Produkty]Kategorie = $sCategory)
```

```
      If ((OK = 1) & (asCategory>0))
•        `QUERY ([Produkty]; [Produkty]Kategorie = sCategory)
•        QUERY ([Produkty]; [Produkty]Kategorie = asCategory{asCategory})
•        If (ckOrderSelection= 1)                ` Políčko Třídit dle názvu je zapnuto.
•           ORDER BY ([Produkty]; [Produkty]Název)
•        End if

         WIN_OutputWindowTitle
      End if
         ` Konec metody
```

2. Otevřete formulář dialogu.
3. Přidejte do formuláře aktivní objekt s těmito nastaveními:

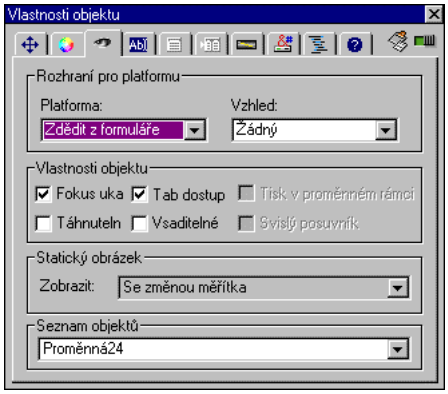
Název:	ckOrderSelection
Typ:	Zaškrtávací políčko
Automatická akce:	Ne
Popiska:	Třídit dle názvu
Styl:	Logické

4. Přejděte do prostředí Vlastní nabídky a otestujte jej.

16.14.2. Odstranění tečkovaného rámečku ze zaškrtávacího políčka

Tečkovaný rámeček kolem zaškrtávacího políčka nemusí být požadovaným rysem, na druhou stranu nám umožňuje pouhým klepnutím na mezerník zapnout či vypnout zaškrtávací políčko. Jestliže chcete tento rys vynechat a přinutit uživatele klepnout myší na zaškrtávací políčko, proveďte následující:

1. Poklepáním na zaškrtávací políčko, otevřete definici objektů ckORDERSelection.
2. Přejděte na stránku Zobrazit.



3. Odškrtněte políčko Tab dostup.

16.15. Vylepšení akce tlačítka Dotaz dle kategorie

V případě, že jste si ještě nevšimli - když klepnete na tlačítko Dotaz dle kategorie více než jednou, příkaz DISTINCT VALUES vytvoří seznam s pouze jednou hodnotou a to proto, že tento příkaz vytváří seznam na základě platného výběru. Po prvním klepnutí na tlačítko dotazů budete mít ve výběru záznamy pouze jedné kategorie. Tento nedostatek může být vyřešen tím, že těsně před příkaz DISTINCT VALUES umístíme příkaz ALL RECORDS a zajistíme tak vytváření seznamu z celé databáze.

Druhý drobnější detail je, že při vstupu do dialogu se nám v seznamu nezobrazuje žádná hodnota. Tento nedostatek může bý vyřešen pomocí inicializační hodnoty array asCategory těsně před zobrazením.

Posledním problémem je, že staartní indikátor průběhu může být občas poněkud matoucí. 4D zná způsob jak tento indikátor průběhu vypnout v oblastech, kde by mohl působit rušivým dojmem. Vypnutí indikátoru průběhu dosáhneme příkazem MESSAGES OFF. Tento příkaz vypne všechny automatické zprávy, které 4D odesílá uživateli. Nezapomeňte je však opět zapnout pomocí příkazu MESSAGES ON.

16.15.1. Vylepšení metody objektu bQueryCategory

1. Otevřete metodu objektu a upravte ji následujícím způsobem:

```
If (False)
      ` Metoda objektu: bQueryCategory
      ` Kurz ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Vytvoří rys dotaz na kategorii

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT ($LWidth; $LHeight;$LWindowID )
      ` C_STRING (15;sCategory)
```

- ALL RECORDS ([Produkty])
- MESSAGES OFF
  DISTINCT VALUES ([Produkty]Kategorie; asCategory)
- MESSAGES ON

```
$LWidth := 300                          `Požadovaná šířka okna.
$LHeight := 70                          `Požadovaná výška
$LWindowID := WIN_LNewWindow ($LWidth; $LHeight;<>WIN_LUpper;<>WIN_LModal)
```

```
DIALOG ([zDialogy]; "DotazKategorie")
CLOSE WINDOW

   ` $sCategory := Request ("Hledat produkty kategorie:")
   ` QUERY ([Produkty]; [Produkty]Kategorie = $sCategory)
   If ((OK = 1) & (asCategory>0))
      `QUERY ([Produkty]; [Produkty]Kategorie = sCategory)
      QUERY ([Produkty]; [Produkty]Kategorie = asCategory{asCategory})
   If (ckOrderSelection= 1)                  ` Políčko Třídit dle názvu je zapnuto.
      ORDER BY ([Produkty]; [Produkty]Název)
   End if

   WIN_OutputWindowTitle
End if
   ` Konec metody
```

16.16. Obnova platného výběru při zrušení uživatelem

Jedna věc, kterou by program neměl dělat je úprava čehokoli jestliže uživatel akci zruší.
V metodě objektu Dotaz dle kategorie je použit příkaz ALL RECORDS, který přepíše původní platný výběr. Pomocí příkazu CUT NAMED SELECTION si můžeme původní platný výběr odložit a v případě potřeby jej opět použít.

16.16.1. Úprava bQueryCategory k obnově výběru.

1. Upravte metodu objektu bQueryCategory následujícím způsobem:

```
If (False)
      ` Metoda objektu: bQueryCategory
      ` Kurz ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Vytvoří rys dotaz na kategorii

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT ($LWidth; $LHeight;$LWindowID )
      ` C_STRING (15;sCategory)

•  CUT NAMED SELECTION ([Produkty]; "PuvodniVyber") `  Vytváří pro případ potřeby

   ALL RECORDS ([Produkty])

   MESSAGES OFF
   DISTINCT VALUES ([Produkty]Kategorie; asCategory)
   MESSAGES ON
```

```
$LWidth := 300                          `Požadovaná šířka okna.
$LHeight := 70                          `Požadovaná výška
$LWindowID := WIN_LNewWindow ($LWidth; $LHeight;<>WIN_LUpper;<>WIN_LModal)

DIALOG ([zDialogy]; "DotazKategorie")
CLOSE WINDOW

   ` $sCategory := Request ("Hledat produkty kategorie:")
   ` QUERY ([Produkty]; [Produkty]Kategorie = $sCategory)
   If ((OK = 1) & (asCategory>0))
      `QUERY ([Produkty]; [Produkty]Kategorie = sCategory)
      QUERY ([Produkty]; [Produkty]Kategorie = asCategory{asCategory})
   If (ckOrderSelection= 1)             ` Políčko Třídit dle názvu je zapnuto.
      ORDER BY ([Produkty]; [Produkty]Název)
   End if
WIN_OutputWindowTitle
•  CLEAR NAMED SELECTION ("PuvodniVyber") ` Odstraní výběr není-li potřeba
•  Else
•     USE NAMED SELECTION ("PuvodniVyber") ` Obnoví výběr při klepnutí na storno

End if
   ` Konec metody
```

2. Přejděte do prostředí Vlastní nabídky a vyzkoušejte tuto metodu.

16.17. Vytvoření přijatelnějšího rozhraní

Podle TWI a HIG by měly mít Nabídky/seznamy určité vlastnosti. Měly by uživateli okamžitě nabízet nějakou výchozí hodnotu aniž by ji musel vybírat.

16.17.1. Úprava bQueryCategory k vylepšení rozhraní

1. Otevřete bQueryCategory a upravte ji následujícím způsobem:

```
If (False)
      ` Metoda objektu: bQueryCategory
      ` Kurz ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Vytvoří rys dotaz na kategorii

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT ($LWidth; $LHeight;$LWindowID )
      ` C_STRING (15;sCategory)
```

```
CUT NAMED SELECTION ([Produkty]; "PuvodniVyber") `  Vytváří pro případ potřeby

ALL RECORDS ([Produkty])

MESSAGES OFF
DISTINCT VALUES ([Produkty]Kategorie; asCategory)
• asCategory := 1                                   `     Naplní nabídku seznam hodnotou
• USE NAMED SELECTION ("PuvodniVyber") ` Obnoví výběr před vykreslením po close

MESSAGES ON

$LWidth := 300                          `Požadovaná šířka okna.
$LHeight := 70                          `Požadovaná výška
$LWindowID := WIN_LNewWindow ($LWidth; $LHeight;<>WIN_LUpper;<>WIN_LModal)

DIALOG ([zDialogy]; "DotazKategorie")
CLOSE WINDOW

  ` $sCategory := Request ("Hledat produkty kategorie:")
  ` QUERY ([Produkty]; [Produkty]Kategorie = $sCategory)
  If ((OK = 1) & (asCategory>0))
    `QUERY ([Produkty]; [Produkty]Kategorie = sCategory)
    QUERY ([Produkty]; [Produkty]Kategorie = asCategory{asCategory})
  If (ckOrderSelection= 1)              ` Políčko Třídit dle názvu je zapnuto.
    ORDER BY ([Produkty]; [Produkty]Název)
  End if
WIN_OutputWindowTitle

End if
  ` Konec metody
```

2. Přejděte do Prostředí vlastní nabídky a vyzkoušejte.

16.18. Array vyžadují mnoho paměti

Array jsou pravděpodobně největší „žrout“ paměti. Když nejsou potřeba měli by být z paměti vymazány. Vymazání z paměti provedete předeklarováním array na nulovou délku.

16.20.1. Upravení metody objektu bQueryCategory k šetření paměti

1. Přidejte do metody objektu bQueryCategory pro array typu String asCategory následující kód k vymazání zabrané paměti.
   Vložte těsně před DISTINCT VALUES řádek.

   ```
   ARRAY STRING (15; asCategory; 0)
   ```

   Těsně před poslední řádek metody „`Konec metody“ vložte tento řádek znovu

   ```
   ARRAY STRING (15; asCategory; 0)
   ```

Extra Credit Diskuze

16.19. Jiný způsob automatického generování array

Jsou ještě dva další způsoby jak 4D pro vás vygeneruje array.

Jeden způsob je kopírovat pole ze všech záznamů platného výběrum tento příkaz je:

```
SELECTION TO ARRAY  ([NějakáTabulka]NějakéPole; asMůjArray)
```

Jiný způsob je kopírovat výběrový seznam ze struktury, příkaz je:

```
LIST TO ARRAY ("MůjSeznam"; asMůjArray)
```

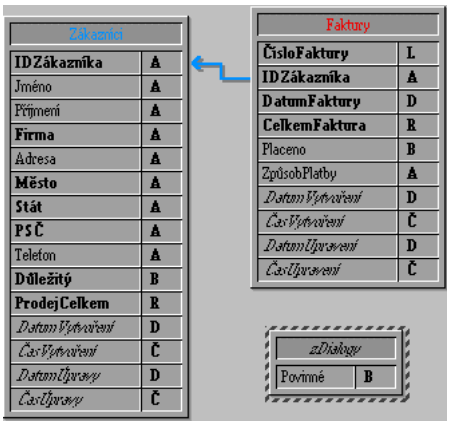
Oba tyto příkazy mají své protějšky ve fungujícím druhém směru. ARRAY TO LIST změní existující seznam uložený ve struktuře. ARRAY TO SELECTION změní určité (á) pole záznamů v platném výběru tak, že překopíruje první prvek array do prvního záznamu, druhý prvek array do druhého záznamu atd. Jestliže platný výběr neobsahuje dost záznamů, aby pokryl celý array, jsou vytvořený nové záznamy.

17. Vytváření specializovaných zpráv

17.1. Vytváření vlastních zpráv s použitím formulářů

Editor rychlých zpráv je vhodný pro většinu zpráv obsahujících sloupce. Má však svá omezení. Např. nemůžete přidat grafiku (např. logo společnosti). Editor rychlých zpráv vytváří pouze zprávy sloupcového typu. Jestliže potřebujete zprávu, která jde za rámec těchto omezení, musíte použít Editor formulářů. Pro tisk takovéhoto formuláře použijete příkaz PRINT SELECTION. Tento příkaz tiskne platný výběr na platném výstupním formuláři. V Prostředí uživatele existuje ekvivalent tohoto příkazu položka nabídky Soubor Tisk.

Doposud jsme pracovali především se záznamy [Zákazníci]. Nyní při sestavování této zprávy budeme pracovat se záznamy z [Faktury]. Při práci v další kapitole si pamatujme, že i když nevytvoříme možnost tisknout seznam záznamů, můžeme vytvořit kód, který tiskne jednotlivé faktury.



První krok bude vylepšení vaší položky nabídky Speciální zprávy tak, aby umožnila provádět několik speciálních zpráv. Abychom eleminovali potřebu vytvářet stále nové položky nabídky pokaždé, když chceme vytvořit speciální zprávy, upravíme metodu projektu této položky nabídky. Pak vytvoříme nový formulář a použijeme tento formulář s novou metodou.

17.2. Vytvoření generického nástroje pro speciální zprávy

Vytváření tiskových zpráv je obvykle primární funkce všech databází. Existuje jen velmi málo databází, kde lze vystačit pouze s Editorem rychlých zpráv. Samozřejmě přizpůsobovat databázi tak, že budeme neustále přidávat položky nabídky je poněkud nevhodné. Udržet všechny takové položky nabídky správně dostupné a nedostupné je také velice náročné.

17.2.1. Úprava metody projektu M_SpecialReports

1. Vymažte veškerý text z metody projektu M_SpecialReports a pak vložte následující text buď ze souboru M_SpecialReports nebo napište:

```
If (False)
      ` Metoda: M_SpecialReports
      ` Kurz ACI University
      ` Generická metoda
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Ovládá specielní zprávy
      ` přidané do bloku case při události zavedení On Load formuláře
      ` [zDialogy]; "VybratZprávu" vytvořeného k tomuto účelu


   <>fGeneric := True
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_STRING(30; $sReport)
C_LONGINT ($LWidth; $LHeight)

$LWidth := 340 `                        stanovená šířka okana.
$LHeight := 290 `                           stanovená výška okna.
$LWindowID := WIN_LNewWindow ($LWidth; $LHeight; <>WIN_LUpper; <>WIN_LModal)
DIALOG ([zDialogy]; "VybratZprávu")
CLOSE WINDOW

If (OK = 1)
   $sReport := Replace string(asReports{asReports}; " "; "") ` Vynechat mezery vložené pro
přehlednost

   Case of
      : (bReport = 1)
         M_GEN_QuickReport

      : (bLabel = 1)
         M_GEN_Labels

      : (bChart = 1)
         M_GEN_Chart

      : (Position ("…"; $sReport) > 0) ` ASCII elipsy znamenají provést metodu
         $sReport := Replace string ($sReport; "…"; "") `Vynechat elipsy
         EXECUTE ($sReport)

      : (Substring($sReport;Length($sReport)-2;3)="...")                `Tři tečky jsou totéž
         $sReport:=Substring($sReport;1;Length($sReport)-3)             `Vynechat tři tečky
         EXECUTE($sReport)
```

```
    : (Substring ($sReport; 1; 3) = "QR-") `        Použití existující uložené rychlé zprávy
      $sReport := Substring ($sReport; 4)
      If(<>WIN_fIsWindows)
        $sReport :=  $sReport + ".4qr"
      End if
      REPORT (pTable->; $sReport)

    : (Substring ($sReport; 1; 7) = "Labels-")`     Použití existujícího uloženého návrhu štítků
      $sReport := Substring ($sReport; 8)
      If(<>WIN_fIsWindows)
        $sReport :=  $sReport + ".4lb"
      End if
      PRINT LABEL (pTable->; $sReport)

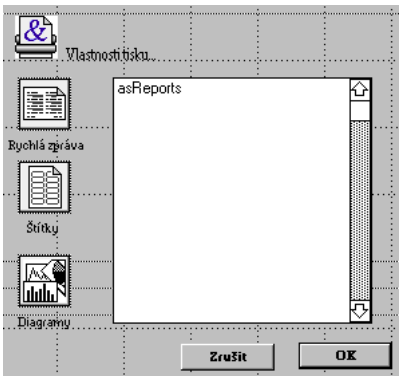
    Else

      OUTPUT FORM (pTable->; $sReport) `    Přepnout na tiskový formulář
      PAGE SETUP (pTable->; $sReport) `Nastavit výchozí vzhled stánky
      PRINT SELECTION (pTable->)
      OUTPUT FORM (pTable->; "Výstupní") `      Zpět k výstupnímu formuláři
  End case

End if
  ` Konec metody
```

17.2.2. Vyzkoušení metody formuláře [zDialogy];"VybratZprávu"

Formulář [zDialogy];"VybratZprávu" námi připravený dopředu vypadá následovně



1. Otevřete metodu formuláře [zDialogy];"VybratZprávu.

```
If (False)
   ` Metoda formuláře: VybratZprávu
   ` Kurz ACI Universita
   ` Vytovřeno: Jim Steinman
   ` Datum: 15/1/97

   `Účel: Tato metoda naplňuje array posuvné oblasti

   `V popise níže mohou být použity mezery
   `Mezery budou vynechány před provedením

   `Jestliže vaše zpráva je touto cestou spouštěna
   `ukončete váš popis (název metody) zankem elipsy
   `nebo tři tečky
   `Příklad: asReports{1} := "Prodej po státech…"

   `Jetliže vaše zpráva spouští existující návrh Rychlé zprávy
   `Předřaďte názvu zprávy znaky "QR-"
   `Příklad: asReports{2} := "QR-Zakaznici zpráva"

   ` Jetliže vaše zpráva spouští existující návrh Štítků
   ` Předřaďte názvu zprávy znaky "Labels-"
   `Příklad: asReports{3} := "Labels-Zakaznici Stitky"

   `Pozn.: Názvy metod nemohou obsahovat mezery
   `Příklad: "Prodejpostátech" = Skutečný název metody

   `Pozn.: Existující názvy návrhu rychlých zpráv a štítků
   `nesmí obsahovat mezery v názvech
   `Příklad: "ZakazniciZpráva" = Skutečý název návrhu RZ
   ` Příklad: "ZakazniciStitky" = Skutečný název návrhu štítků
```

```
<>f_Version6x10 := True
<>fJ_Steinman := True
End if

If (Form event =  On Load)

   Case of
      : (Current process = 1) `                Proces uživatele/aplikace
         DISABLE BUTTON (bReport) ` Tyto zprávy nejsou dostupné
         DISABLE BUTTON (bLabel) `  z úvodní obrazovky
         DISABLE BUTTON (bChart)
         ARRAY STRING (40; asReports; 1)
         asReports{1} := "Prodej po státech…"

      : (pTable = (->Products]))
         ARRAY STRING (40; asReports; 1)
         asReports{1} := "Tisk Výstup"

      : (pTable = (->[Invoices]))
         ARRAY STRING (40; asReports; 5)
         asReports{1} := "Tisk Výstup "
         asReports{2} := "Tisk Faktura"
         asReports{3} := "QR-Zakaznici prodeje zprava"
         asReports{4} := "QR-Zakaznici prodeje celkem"
         asReports{5} := "QR-Mesicni rocni prodeje celkem"

      : (pTable = (->[Customers]))
         ARRAY STRING (40; asReports; 3)
         asReports{1} := "Tisk Výstup"
         asReports{2} := "QR-Zakaznici zprava"
         asReports{3} := "Labels-Zakaznici stitky"
   End case
   asReports := 1
End if
   ` Konec metody
```

17.2.3. Vytvoření metody projektu ProdejPoStátech

1. Vytvořte metodu projektu nazvanou ProdejPoStátech následovně:

```
If (False)
      ` Metoda: ProdejPoStátech
      ` Kurz ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Spustí zprávu se seznamem faktur po státech

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if
```

```
ALL RECORDS ([Invoices]) `       Změní platný výběr.
PRINT SELECTION ([Faktury])
   ` Konec metody
```

2. Přejděte do Prostředí uživatele a vyzkoušejte.

17.3. Přepnutí formuláře při tisku

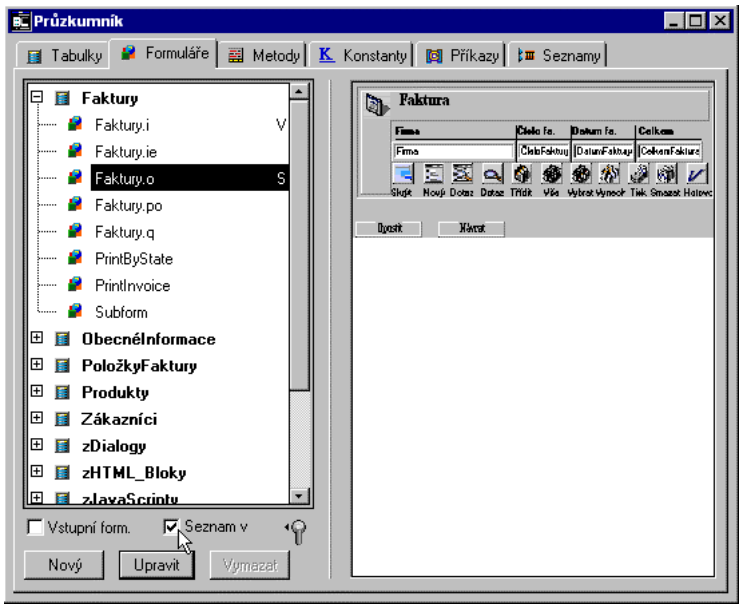
Současné platné výstupní formuláře mají příliš mnoho polí v jednom řádku než, aby se vešli na jednu tiskovou stránku. Naštěstí můžete vytvořit rozdílné formuláře jen pro účely tisku. Jako výchozí 4D používá jakýkoliv formulář, který je právě v okamžiku tisku označen jako výstupní formulář. Přepnout na jiný formulář lze příkazem OUTPUT FORM. Takže upravme metodu tak, aby nás přepnula na formulář s méně poli.

Tak jako příkaz INPUT FORM, říká příkaz OUTPUT FORM 4D, který formulář má použít, příště až bude potřebovat výstupní formulář pro určitou tabulku. Příkaz OUTPUT FORM neprovádí nic co by se stalo okamžitě, pouze říká 4D: „Kdykoliv od této doby budeš potřebovat výstupní formulář použij bez jakýchkoliv dalších instrukcí tento formulář.“.

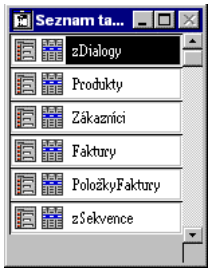
17.4. Určování vstupního a výstupního formuláře

Poznamenejme, že názvy fromulářů nemají žádný vliv na určování vstupního a výstupního formuláře. Určit vstupní či výstupní formulář lze třemi způsoby:

- Zaškrtnutím políček Vstupní form nebo Seznam v okně Průzkumníka na stránce Formuláře



- V Prostředí uživatele klepnutí na klávesy Comma + mezerník (Ctrl + mezerník) přenese na popředí tabulku Seznam tabulek, kde z patřičných nabídek formulářů si vyberete vhodný formulář.



- V metodě objektu nebo projektu použijete příkazy INPUT FORM nebo OUTPUT FORM.

```
OUTPUT FORM ([Zákazníci]; "Výstupní")
```

Protože váš původní výstupní formulář má příliš mnoho polí, vytvoříte nový formulář pouze pro tisk. Před tiskem můžete příkazem vyměnit výstupní formulář a po tisku se opět tímtéž příkazem s jinými parametry navrátit k původnímu výstupnímu formuláři pro zobrazování na obrazovku.

17.4.1. Vytvoření nového tiskového formuláře

1. Pro tabulku [Faktury] vytvořte nový formulář TiskPoStátech. Vyberte do něj následující pole:

- Firma (z tabulky [Zákazníci])
- Stát (z tabulky [Zákazníci])
- ČísloFaktury
- DatumFaktuy
- CelkemFaktura

2. Zvolte typ formuláře: Seznam pro tisk
3. Jako užitý vzor zvolte: 3D Look, Banner nebo Simple.
4. Pro pole [Faktury]CelkemFaktura přiřaďte formát zobrazení částka
5. Vymažte všechny objekty ze záhlaví.
6. Vytvořte svou vlastní hlavičku zprávy v oblasti záhlaví formuláře.

17.4.2. Úprava metody ProdejPoStátech pro přepínání formulářů

1. Otevřete metodu ProdejPoStátech a proveďte následující úpravy:

```
If (False)
      ` Metoda: ProdejPoStátech
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Spouští zprávu se seznamem faktur po státech.

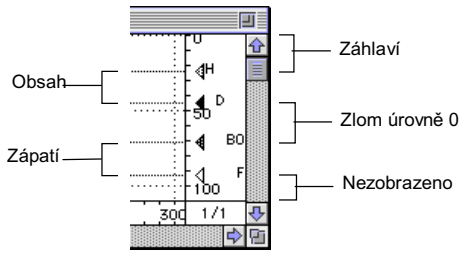
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

ALL RECORDS ([Faktury]) `        Změní platný výběr.
   ` Přepne do formuláře zprávy.
• OUTPUT FORM ([Faktury]; "TiskPoStátech")
PRINT SELECTION ([Faktury])
   ` Přepne zpět na správný výstupní formulář.
• OUTPUT FORM ([Faktury]; "Výstupní")
   ` Konec metody
```

17.5. Vytvoření zápatí pro PRINT SELECTION

Protože příkaz PRINT SELECTION používá k tisku formulář, můžete jej jednoduše upravovat a dát mu libovolný vzhled a formát. Např. chcete přidat oblast zápatí, která tiskne cokoliv chcete naspodu každé stránky zprávy.

Nejdříve určíte velikost zápatí, potažením řídící čáry P ve formuláři. Řídící čáry jsou označeny trojúhelníky na pravé straně měřítka, v okně formuláře. Vše co umístíte mezi čáry P a Z0 se bude tisknout v zápatí každé stránky.



Dvě položky jsou běžné téměř ve všech zprávách, jsou to datum tisku a číslo tištěné stránky. Dobré umístění pro tyto položky je v zápatí. Abychom přidali datum tisku a číslo stránky, musíte přidat do formuláře proměnné. Metoda objektu těchto proměnných bude pak použita k přiřazení hodnot (datumu a čísla stránky do těchto proměnných).

17.5.1. Přidání datumu tisku do zápatí zprávy

1. Otevřete formulář [Faktury];"TiskPoStátech".
2. Potáhněte řídící čáru P asi 2 cm dolů..
3. Klepněte na nástroj Aktivní objekt a potáhněte jej na levou stranu zápatí.
4. Až se objeví dialog definic objektu napište do názvu sDate.
5. Zaškrtněte na stránce Události, Při tisku zápatí.
6. Klepněte na tlačítko Metoda objektu.
7. Napište následující metodu objektu:

```
If (False)
      ` Metoda objektu: sDate
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Nastaví hodnotu proměnné na dnešní datum.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

sDate := "Tisk dne: "+ String (Current date)
   ` Konec metody objektu
```

8. Přejděte do prostředí Vlastní nabídky a otestujte tuto metodu.

17.6. Přidání čísla stránky do zprávy

Abychom přidali číslo stránky na každou stránku zprávy, vytvoříme další proměnnou.

17.6.1. Přidání čísla stránky do zápatí zprávy

1. Otevřete formulář [Faktury];"TiskPoStátech".
2. Potáhněte řídící čáru P asi 2 cm dolů..
3. Klepněte na nástroj Aktivní objekt a potáhněte jej na levou stranu zápatí.
4. Až se objeví dialog definic objektu napište do názvu sPage.
5. Zaškrtněte na stránce Události, Při tisku zápatí.
6. Klepněte na tlačítko Metoda objektu.
7. Napište následující metodu objektu:

```
If (False)
      ` Metoda objektu: sPage
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Nastaví hodnotu proměnné na číslo stránky.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if


sPage := "Stránka: "+ String (Printing page)
   ` Konec metody objektu
```

17.7. Přidání označení konce zprávy

Pro ty, kteří budou číst tuto zprávu bude užitečné jestliže nějak označíme konec této zprávy. Nejlepším místem pro toto označení je zlom úrovně 0. Tento zlom je tištěn jednou na konci celé zprávy po tisku posledního záznamu. Zlom úrovně 0 je oblast mezi řídící čárou Z0 a čárou těsně nad ní.

17.7.1. Přidání textu do oblasti zlomu Z0

1. Otevřete formulář [Faktury];"TiskPoStátech".
2. Potáhněte řídící čáru Z0 asi o 6mm níže.
3. Pokud je potřeba potáhněte proměnné datumu tisku a čísla stránky níže.
4. Do oblasti zlomu Z0 přidejte statický text s textem ***Konec zprávy***
5. Přejděte do prostředí Vlastní nabídky a vyzkoušejte.

17.8. Přidání mezisoučtů do zprávy

Jistě jste si všimli, že oblast zlomu Z0 je tištěna na konci každé stránky místo jak jsme zamýšleli na konci zprávy. Je ještě několik dodatečných kroků, které musíme udělat, abychom dosáhli kýženého výsledku.

17.9. Co jsou oblasti zlomu?

Oblasti zlomu vám dovolí zobrazovat sumární informace mezi skupinami setříděných záznamů. Např. jestliže zpráva bude tříděna podle státu může oblast zlomu zahrnovat součet pro všechny faktury pro jeden stát. V tomto případě by oblast zlomu byla tištěna, když 4D dokončí tisk všech faktur pro jeden určitý stát. Oblasti zlomu nemohou být tištěny pokud nejsou záznamy tříděny, to znamená pokud nejsou rozděleny do určitých skupin. K setřídění záznamů použijeme příkaz ORDER BY.

Project	Expense	Description
House	$2280	Siding
Garage	$1010	Roofing
Yard	$100	Seed
Yard	$280	Fertilizer
Garage	$600	Concrete
House	$1200	Carpet
Yard	$450	Shrubs

Project	Expense	Description
Garage	$1010	Roofing
Garage	$600	Concrete
House	$2280	Siding
House	$1200	Carpet
Yard	$100	Seed
Yard	$280	Fertilizer
Yard	$450	Shrubs

Project	Expense	Description
Garage	$1010	Roofing
Garage	$600	Concrete

Subtotal	**$1610**	
House	$2280	Siding
House	$1200	Carpet

Subtotal	**$3480**	
Yard	$100	Seed
Yard	$280	Fertilizer
Yard	$450	Shrubs

Subtotal	**$830**	

Když jsou již záznamy tříděny můžete vypočítat mezisoučty pro každou skupinu záznamů. Zde jsou údaje o ceně sčítány pro jednotlivé projekty.

17.10. Třídění záznamů procedurálně

Již víte, že příkaz ORDER BY umí zobrazit Editor třídění, když nejsou určena žádná pole. Např.:

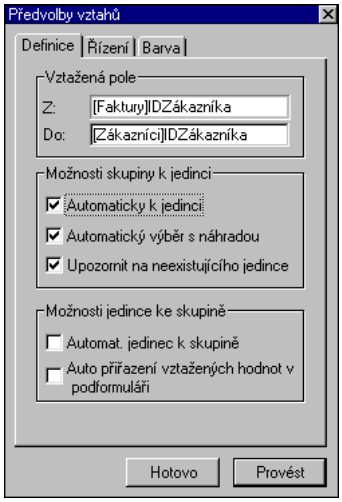
```
ORDER BY ([Produkty])
```

Pomocí programovacích instrukcí si můžete naprogramovat vlastní třídění aniž uživatel uvidí okno Editoru třídění. K programování třídění přidáte do příkazu ORDER BY více parametrů oddělených středníky. Můžete určit jestli bude třídění vzestupné (A-Z), pomocí znaku „>“, či sestupné (Z-A) pomocí znaku „<“. Např. můžete třídit záznamy produktů vzestupně podle jejich kategorie.

```
ORDER BY ([Produkty]; [Produkty]Kategorie; > )
```

17.11. Třídění přes relace

Jestliže se pozorně podíváte na následující proceduru všimnete si, že třídíme [Faktury], ale používáme k tomu cizí pole z jiné tabulky ([Zákazníci]Stát). V tomto případě to můžeme provést, protože jsme nakreslili vztah mezi tyto dvě tabulky a zapnuli rys vztahu Automaticky k jedinci.




```
ORDER BY ([Faktury];[Zákazníci]Stát;>)
```

Tabulka, kde chcete výsledek.

Tabulka použitá pro třídění.

17.11.1. Úprava metody ProdejPoStátech pro automatické třídění záznamů

1. Upravte metodu ProdejPoStátech následujícím způsobem:

```
If (False)
    ` Metoda: ProdejPoStátech
    ` Kurs ACI University
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Spouští zprávu se seznamem faktur po státech.

  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if

ALL RECORDS ([Faktury]) `       Změní platný výběr.
ORDER BY ([Faktury]; [Zákazníci]Stát; >)
` Přepne do formuláře zprávy.
OUTPUT FORM ([Faktury]; "TiskPoStátech")
PRINT SELECTION ([Faktury])
  ` Přepne zpět na správný výstupní formulář.
OUTPUT FORM ([Faktury]; "Výstupní")
  ` Konec metody
```

17.12. Výpočet součtu polí pro úroveň zlomu

Jestliže chcete provést součet polí skupiny a zobrazit jej v oblasti zlomu, musíte 4D říct, aby postupně akumulovala hodnoty v těchto polích. Tato instrukce musí být sdělena 4D dostatečně dopředu, aby mohla akumulovat hodnoty při provádění tisku. Tato instrukce se zadává příkazem ACCUMULATE. Tento příkaz musí obdržet název akumulovaného pole jako parametr. Pro správné provedení tohoto příkazu musíme rovněž 4D sdělit, pro kterou úroveň zlomu má akumulace provádět. Toto provedeme pomocí příkazu BREAK LEVEL (samozřejmě žádný z těchto příkazů přímo nevytvoří patřičné skupiny záznamů, úrovně zlomů, to musíte provést již zmíněným příkazem ORDER BY).

17.13. Provádění zlomů s a bez kompilátoru

V interpretované databázi (tj. nezkompilované) můžete funkční zlomy provádět bez použití příkazů BREAK LEVEL a ACCUMULATE. Váš programátorský život však bude jednodušší, jestliže budete tyto příkazy používat pokaždé. Vyhnete se tak chybám a nekonzistentnostem programu po kompilaci. Rozhodně vám to asi neublíží, ale pomůže.

17.14. Provádění celkových součtů

Jestliže chcete 4D bude sledovat data během tisku a postupně nasčítávat hodnoty číselných polí tak, aby jste na konci zprávy mohli zobrazit celkový součet. Celkový součet je jeden z případů již zmíněných mezisoučtů. To znamená platí zde tatáž pravidla. Musíte instruovat 4D dostatečně dopředu, že budete chtít akumulovat určité pole. Rovněž musíte použít příkaz BREAK LEVEL do které úrovně se má tato akumulace provádět. Oba příkazy ACCUMULATE i BREAK LEVEL musí být použity před příkazme PRINT SELECTION a nejlépe po příkazu ORDER BY.

17.15. Přidání součtu do zprávy

Když jste řekli 4D, aby akumulovala hodnoty polí, musíte přidat ještě další proměnnou do oblasti Z0. Po přidání proměnné ji musíte přidat metodu objektu s příkazem Subtotal, který přiřadí této proměnné naakumulované hodnoty pro tuto úroveň zlomu. V úrovni zlomu (PRINT SELECTION) nelze použít příkaz Sum, protože tento příkaz k tomu, aby vrátil správnou hodnotu potřebuje mít fyzicky k dispozici všechny sčítané záznamy, což není případ úrovně zlomu.

17.15.1. Přidání celkových součtů do zprávy

1. Upravte metodu ProdejPoStátech následujícím způsobem:

```
If (False)
      ` Metoda: ProdejPoStátech
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Spouští zprávu se seznamem faktur po státech.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

ALL RECORDS ([Faktury]) `        Změní platný výběr.
ORDER BY ([Faktury]; [Zákazníci]Stát; >)
•  BREAK LEVEL (0) `Umožní celkové součty  v úrovni 0.
•  ACCUMULATE ([Faktury]CelkemFaktura)
` Přepne do formuláře zprávy.
OUTPUT FORM ([Faktury]; "TiskPoStátech")
PRINT SELECTION ([Faktury])
   ` Přepne zpět na správný výstupní  formulář.
OUTPUT FORM ([Faktury]; "Výstupní")
   ` Konec metody
```

2. Nad řídící čáru Z0 tj. do úrovně zlomu 0 přidejte proměnnou nazvanou rTotal.
3. Vytvořte metodu objektu pro rTotal spouštěnou při události Při tisku zlomu:

```
If (False)
      ` Metoda objektu: rTotal
```

```
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Vypočte celkový součet pro záznamy ve zprávě.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

rTotal := Subtotal ([Faktury]CelkemFaktura)
   ` Konec metody objektu
```

4. Pro proměnnou rTotal vyberte formát zobrazení Částka
5. Přejděte do prostředí Vlastní nabídky a vyzkoušejte ji.

17.16. Přidání úrovní zlomu

Ve vaší zprávě můžete mít mezisoučty jak již bylo zmíněno podle skupin faktur dle států, které obsahují všechny prodeje do jednoho státu. K tomuto potřebujeme vytvořit další úroveň zlomu, která by tyto informace zobrazovala. Tato úroveň zlomu bude úroven 1 s označením Z1. Čísla v úrovních zlomu odpovídají pořadí polí podle kterých se třídí v příkaze ORDER BY. Z toho výplývá, že pokud chceme zobrazovat údaje např. v úrovni zlomu 3, musíme provést třídění minimálně podle třech parametrů třech polí.

17.17. Přidání řídící čáry

Aby jste vymazali nebo setřídili řídící zprávu, musíte klepnout myší způsobem popsaným v následující tabulce.

Cíl	Akce
Přidání řídící čáry	Option + klepnout (Alt + klepnout) na řídící čáru Z0
Vymazání řídící čáry	Comma + klepnout (Ctrl + klepnout) na patřičnou řídící čáru

17.17.1. Přidání zlomu a mezisoučtu do Zprávy

1. Otevřete formulář [Faktury];"TiskPoStátech".
2. Prove'dte Option + klepnout (Alt + klepnout) the na řídící čáru Z0 a vytvořte tak řídící čáru Z1.
3. Potáhněte řídící čáry Z0 a F dolů spolu s jejich objekty.
4. Zduplikujte (kopírova, vložit) proměnnou rTotal a potáhněte ji nad řídící čáru Z1..
5. Přejděte do prostředí Vlastní nabídky a otestujte.

17.17.2. Změna v příkaze BREAK LEVEL

K tomu, aby byly mezisoučty ve zprávě správně sčítány je potřeba instruovat 4D, aby akumulovala i pro vyšší úroveň zlomu.

1. V metodě projektu ProdejPoStátech změňte řádek BREAK LEVEL (0) na BREAK LEVEL (1)

```
`BREAK LEVEL (0) `Umožní celkové součty v úrovni 0.
BREAK LEVEL (1) `Umožní mezisoučty v úrovni 1 i celkové součty.
```

2. Přejděte do prostředí Vlastní nabídky a otestujte tuto změnu

18. Provádění dotazů procedurálně

18.1. Dotazy pro Editor dotazů se mohou stát složitými

Doposud byly námi zadávané dotazy v kódu velice jednoduché, pouze jedna řádka kódu. Postupným skládáním lze však vytvářet velice složité dotazy. Složitý dotaz vytvoříte několikanásobným použitím příkazu QUERY. To, že dotaz ještě není kompletní sdělíme 4D pomocí volitelného paramteru "*". Na konci každého pokračovacího řádku příkazu QUERY. Příkazy QUERY jsou pak spojeny dohromady pomocí logických spojek. Logické spojky, které máme k dispozici v kódu jsou tytéž jako v Editoru dotazů.

Logická spojka	Symbol
A	&
Nebo	|
Kromě	#

Poslední řádek složeného dotazu nesmí na konci řádku obsahovat volitelný parametr "*" . Tento poslední řádek dotaz fyzicky provede.

18.1.1. Úprava metody ProdejPoStátech pro provádění složených dotazů

1. Vymažte současnou metodu ProdejPoStátech a z textového souboru Prodej po státech vložte následující:

```
If (False)
      ` Metoda: ProdejPoStátech
      ` Kurs ACI Universita
      ` Vytvořeno: Jim Steinman
      ` Datum: 1/15/97

      ` Účel: Spustí zprávu se seznamem faktur po státech.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT ($LWidth; $LHeight;$LRecordsInSelection;$LWindowID)

$LWidth := 200 `                    Zadaná šířka otevíraného okna.
$LHeight := 185 `                   Zadaná výška.
$LWindowID := WIN_LNewWindow ($LWidth; $LHeight; <>WIN_LUpper; <>WIN_LModal)
DIALOG ([zDialogy]; "DotazDatumy")
CLOSE WINDOW
```

```
If ((OK = 1) & (dFrom#!00/00/00!))
   Case of
      : (rb4 = 1)                          ` Rozsah datumu pro dotaz.
            ` Počátek skládání dotazu
         QUERY ([Faktury]; [Faktury]DatumFaktury >= dFrom; *)
            ` Konec skládání dotazu a provedení dotazu
         QUERY ([Faktury]; & ; [Faktury]DatumFaktury <= dTo) `  Konec

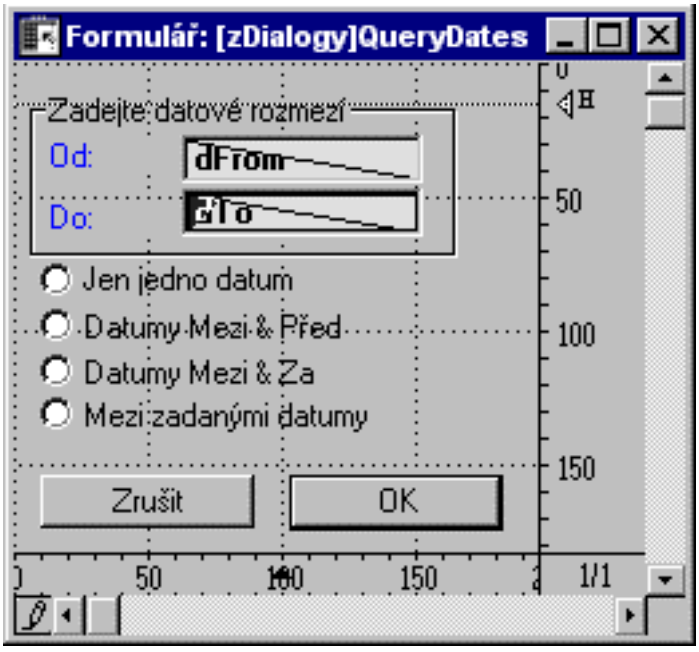
      : (rb1 = 1)                          ` Dotaz na jeden datum
         QUERY ([Faktury]; [Faktury]DatumFaktury = dFrom)

      : (rb2 = 1)                           ` Dotaz na datum a předchozí.
         QUERY([Faktury]; [Faktury]DatumFaktury<= dFrom)

      Else                                 ` Dotaz na datum a následující
         QUERY ([Faktury]; [Faktury]DatumFaktury >= dFrom)
   End case

   $LRecordsInSelection := Records in selection([Faktury])
   If ($LRecordsInSelection > 0)
      ORDER BY ([Faktury]; [Zákazníci]Stát; >; [Faktury]DatumFaktury)
      BREAK LEVEL (1) `                                Umožňuje mezisoučty a celkové součty.
      ACCUMULATE ([Faktury]CelkemFaktura)
      OUTPUT FORM ([Faktury]; "TiskPoStátech")
         ` Přepne do formuláře Zprávy.
      PRINT SELECTION ([Faktury])
      OUTPUT FORM ([Faktury]; "Výstupní")
         ` Přepne zpět na správný výstupní formulář.
   Else
      ALERT ("Nenalezen žádný záznam v tomto období!")
   End if

End if
   ` Konec metody
```


Formulář: [zDialogy]QueryDates
Zadejte datové rozmezí
Od:
dFrom
Do:
dTo
Jen jedno datum
Datumy Mezi & Před
Datumy Mezi & Za
Mezi zadanými datumy
Zrušit
OK
0
50
100
150
1/1

18.2. Použití proměnných z dialogu k sestavení popisného textu

Vytvořili jsme dialog, ze kterého nyní získáváme informace pro sestavení dotazu. Tatáž informace může být užitečná, jestliže ji umístíme do samotné zprávy jako popis, který pomůže vyjasnit jaké časové období zpráva reprezentuje.

18.3. Použití parametrů formátu při sestavení řetězce z proměnných či polí typu Datum a Čas.

Formát zobrazení, který bude použit pro konverzi datumu a času do řetězce není uvnitř 4D předpověditelný. Tento formát, pokud nic dalšího neřekneme, je 4D přebírán z nastavení systému. Proto jestliže chceme zcela určitý formát zobrazení je nejlepší určit tento formát přímo v příkaze konvertujícím datum a čas na textový řetězec. V níže uvedené tabulce vidíte jaké formáty máte k dispozici a jaký bude výsledek po konverzi do řetězce.

String (datum; format) -> String

Č.form.	Formát	Příklad
1	Krátký	2.10.95
2	Zkratky	Po 2 říj 1995
3	Dlouhý	Pondělí 2. říjen 1995
4	MM DD YYYY	02.10.95 nebo 02.10.1895
5	Měsíc Datum Rok	2 říjen 1995
6	Zkr.: Měsíc Datum	2 říj 1995
7	MM DD YYYY Forced	02.10.1995

String(čas; format) -> String

Č.form	Formát	Příklad
1	HH MN SS	13:22:12
2	HH MN	13:22
3	Hod Min Sek	13 hodin 22 minut 12 sekund
4	Hod Min	13 hodin 22 minut
5	HH MN DOP/ODP	1:22 ODP

18.3.1. Přidání popisu do zprávy

1. Otevřete formulář [Faktury]"TiskPoStátech".
2. V paletě nástrojů klepněte na nástroj Aktivní objekt a potáhněte jej do oblasti záhlaví formuláře.
3. V dialogovém okně definic objektu napište název objektu: sMessage.
4. Uzavřete okno definic a zvolte Styl ¬ Středěný.

5. Upravte metodu ProdejPoStátech následujícím způsobem:

```
…
If ((OK = 1) & (dFrom#!00/00/00!))
   Case of
      : (rb4 = 1)                          ` Rozsah datumu pro dotaz.
           ` Počátek skládání dotazu
         QUERY ([Faktury]; [Faktury]DatumFaktury >= dFrom; *)
           ` Konec skládání dotazu a provedení dotazu
         QUERY ([Faktury]; & ; [Faktury]DatumFaktury <= dTo) `   Konec
•        sMessage := “Tato zptáva je za období od "+ String(dFrom; Short) + " do " +
              String(dTo; Short)

      : (rb1 = 1)                          ` Dotaz na jeden datum
         QUERY ([Faktury]; [Faktury]DatumFaktury = dFrom)
•        sMessage := “Tato zpráva je pro den "+ String(dFrom; Short)
      : (rb2 = 1)                           ` Dotaz na datum a předchozí.
         QUERY([Faktury]; [Faktury]DatumFaktury<= dFrom)
•        sMessage := “Tato zpráva je pro den a dny předchozí včetně "+ String(dFrom; Short)
      Else                                 ` Dotaz na datum a následující
         QUERY ([Faktury]; [Faktury]DatumFaktury >= dFrom)
•        sMessage := “ Tato zpráva je pro den a dny následující včetně "+ String(dFrom; Short)

   End case

.
…
```

6. Přejděte do prostředí Vlastní nabídky a vyzkoušejte tuto metodu.

18.4. Vytvoření záhlaví skupiny

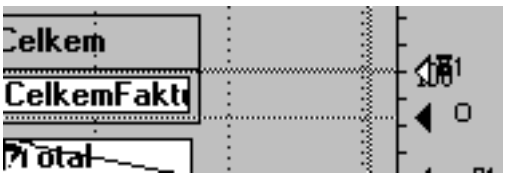
Současné záhlaví pro každou skupinu není asi to co bychom chtěli.

	Rock Video	AK	10226	14.11.92	493,8
	Pleasure Video	AK	10466	29.05.91	415,1
*** Konec státu ***					1 836,9
	North Beach Video	HI	10017	16.07.92	264,8
	Stoney Video	HI	10129	29.05.93	219,4
	Angle Video	HI	10258	25.03.92	439,3
	National Video	HI	10351	16.03.92	413,4
*** Konec státu ***					1 336,9
	Vern's Video	OR	10053	18.09.91	1 117,3
	Riddle Video	OR	10121	03.07.93	578,9

Tak jako jsme vytvořili řídící čáru a oblast zlomu 1, můžeme vytvořit řídící čáru a oblast záhlaví (hlavičky) H1.

18.4.1. Vytvoření jednoduché hlavičky

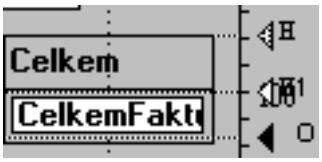
1. Otevřete formulář [Faktury];"TiskPoStátech".
2. Proveďte Option + klepnout (Alt + klepnout) na řídící čáru H a vytvořte tak řídící čáru H1.
3. Oddělte řídící čáry H a H1.
4. Grafickým nástrojem Čára vytvořte vodorovnou čáru a umístěte ji jeden bod nad řídící čáru H1.
5. Umístěte řídící čáru H přesně na nakreslenou čáru.



6. Přejděte do prostředí Vlastní nabídky a proveďte testování.

18.4.2. Opakování záhlaví pro každou úroveň zlomu

1. Přemístěte řídící čáru H o 1 bod nad nakreslenou čáru ve formuláři.



Poznámka: Pokud se vám nepodaří trojúhelník řídící čáry H uchopit, budete muset nejdříve řídící čáru H1 posunout o něco níže a pak teprve posunout řídící čáru H výše. Nezapomeňte řídící čáru H1 vrátit na původní pozici.

2. Přejděte do Prostředí Vlastní nabídky a proveďte testování.

	Pleasure Video	AK	10466	29.05.91

*** Konec státu ***

Firma	Stát	Číslo fa.	Datum fa.
North Beach Video	HI	10017	16.07.92
Stoney Video	HI	10129	29.05.93
Angle Video	HI	10258	25.03.92
National Video	HI	10351	16.03.92

*** Konec státu ***

Firma	Stát	Číslo fa.	Datum fa.
Vern's Video	OR	10053	18.09.91
Riddle Video	OR	10121	03.07.93

19. Zajištění konzistence dat

19.1. Využití počítače k zajištění konzistence formátu dat

Během životnosti databáze se obvykle při zadávání dat vystřídá několik lidí v různých dobách. Tato změna obsluhy může vést k nekonzistenci ve formátech vkládaných dat. Např. jeden člověk je zvyklý psát PŘÍJMENÍ a jiný pouze Příjmení. Již z prvního kursu víme, že můžeme využít 4D k tomu, aby nám pomohla formátovat data v poli [Zákazníci]Příjmení.

19.2. Odkazy na znaky

V jednom textovém řetězci můžeme adresovat i jednotlivé znaky pomocí symbolů odkazů. Tyto znaky jsou "[[" a "]]" syntaxe je následující:

```
sString[[1]] `    Jestliže sString = "abc" výsledek je "a"
```

Na počítačích Macintosh jsou tyto znaky “•” a “•”.

Tento odkaz je funkce a vždy vrací jeden znak jehož pořadí v původním řetězci uvádíme v závorkách.

19.2.1. Napsání metody objektu zajišťující, že pouze první písmeno bude velké

1. Přejděte do Prostředí návrháře.
2. Otevřete formulář [Zákazníci];"Vstupní".
3. Poklepejte na pole Příjmení a otevřete tak dialog definic objektu.
4. Na stránce Události, klepněte se stisknutou klávesou _ (Ctrl) na událost Při aktualizaci a odškrtněte tak všechny události.
5. Zaškrtněte událost Při aktualizaci
6. Klepněte na tlačítko Metoda objektu a otevřete tak metodu objektu pro pole Příjmení.
7. Napište následující metodu objektu:

```
If (False)
      ` Metoda objektu: LastName
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Překonvertue všechna písmena na malá a pak
      ` převrátí první písmeno na velké

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

[Zákazníci]Příjmení := Lowercase ([Zákazníci]Příjmení)
```

```
[Zákazníci]Příjmení [[1]] := Uppercase ([Zákazníci]Příjmení [[1]])
   ` Konec metody objektu
```

8. Přepněte se do prostředí Vlastní nabídky.
9. Přidejte nový záznam a do pole Příjmení napište velkými písmeny své příjmení.

19.3. Přepsání předchozího makra do generické metody

Nyní je zřejmé, že pokud budeme potřebovat rys z předchozího cvičení v mnoha polích formulářů, čeká nás mnoho programování. Cílem veškerého našeho programátorského úsilí by mělo být psaní generických metod tj. kódu, který může být používán mnohonásobně v celé databázi. Bylo by tedy užitečné, kdybychom mohli znovu použít předchozí makro jako metodu projektu pro změnu stylu psaní ve více polích a formulářích. Vyhnuli bychom se tím neustálému překopírovávání téže části kódu. Kromě toho jestliže je generická procedura již napsána a otestována, můžeme ji směle použít kolikrát chceme a máme zajištěnu bezchybnost kódu. Samozřejmě si rovněž ušetříme čas programování, protože se můžeme odkazovat na již hotové části programu. Aby mohla být naše část kódu použita vícekrát, musíme najít způsob jak napsat kód bez použití zcela určitých tabulek a polí.

19.3.1. Vytvoření metody GEN_Capitalize

1. Přepněte se do Prostředí návrháře.
2. Vytvořte metodu projektu nazvanou GEN_Capitalize.

```
If (False)
      ` Metoda: GEN_Capitalize
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Konvertuje poskytnutý text na malá písmena a pak převrátí první písmeno na velké.

   <>fGeneric := False
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

sText := Lowercase (sText)
sText•1• := Uppercase (sText[[1]])
   ` Konec metody
```

3. Upravte metodu objektu pro pole Příjmení následujícím způsobem:

```
If (False)
      ` Metoda objektu: Příjmení
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Konvertuje příjmení do malých písmen
      ` a pak převrátí první písmeno na velké.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

sText := [Zákazníci]Příjmení
GEN_Capitalize
[Zákazníci]Příjmení := sText
   ` Konec metody objektu
```

4. Přepněte se do prostředí Vlastní nabídky a proveďte testování.

19.4. Některé generické kódy jsou lepší než jiné

Náš generický kód pracuje, ale není to o mnoho lepší než to bylo. Nyní máme místo původních dvou řádek, tři řádky kódu, které musíme pokaždé vložit do metody objektu. Navíc používáme proměnné procesu, pro které si musíme pamatovat jejich názvy pokaždé, když je chceme použít v kódu. A nakonec proměnné procesu velice zatěžují paměť a proto ji nyní zbytečně plýtváme.

19.5. Vytvoření vlastní funkce

Tak jako 4D má funkce, které navracejí hodnotu, můžeme si napsat vlastní metody projektu, které budou navracet hodnotu. Musíme jen trochu rozšířit naše znalosti o předávání parametrů, které jsme získali v předchozím. 4D používá speciální lokální proměnnou $0 pro navracení jedné hodnoty z volané metody, tak nám dovoluje vytvořit naši vlastní funkci.

19.5.1. Vytvoření funkce GEN_Capitalize

1. Přepněte se do Prostředí návrháře
2. Upravte metodu GEN_Capitalize následujícím způsobem:

```
If (False)
      ` Metoda: GEN_Capitalize
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Konvertuje poskytnutý text na malá písmena a pak převrátí první písmeno na velké.

   <>fGeneric := False
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_STRING (255; $0; $1)
$1 := Lowercase ($1)
$1•1• := Uppercase ($1[[1]])
$0 := $1
   ` Konec metody
```

3. Upravte metodu objektu pole Příjmení následujícím způsobem:

```
If (False)
      ` Metoda objektu: Příjmení
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Konvertuje příjmení do malých písmen
      ` a pak převrátí první písmeno na velké.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

[Zákazníci]Příjmení := GEN_Capitalize ([Zákazníci]Příjmení)
   ` Konec metody objektu
```

4. Přepněte se do Prostředí uživatele.
5. Vložte nový záznam do tabulky zákazníků a do pole příjmení napište své jméno.

6. Upravte metodu GEN_Capitalize následujícím způsobem a zvyšte tak její účinnost:

```
If (False)
      ` Metoda: GEN_Capitalize
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Konvertuje poskytnutý text na malá písmena a pak převrátí první písmeno na velké.

   <>fGeneric := False
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_STRING (255; $0; $1)
$0 := Lowercase ($1)
$0[[1]] := Uppercase ($0[[1]])
   ` Konec metody
```

7. Vytvořte metodu objektu pro pole Jméno následujícím způsobem:

```
If (False)
      ` Metoda objektu: Jméno
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Konvertuje jméno do malých písmen
      ` a pak převrátí první písmeno na velké.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

[Zákazníci]Jméno := GEN_Capitalize ([Zákazníci]Jméno)
   ` Konec metody objektu
```

8. Přepněte se do prostředí Vlastní nabídky a proveďte testování.

19.6. Použití smyčky k ošetření více než jednoho písmena, které musí být velké

Naše metoda pracuje dobře, ale jen pokud budeme převádět na velká písmena pouze první písmeno. To se však nestává ve všech případech. Vezměme jako příklad název ulice na pražačce, zde potřebujeme převést na velká písmena začátek každého slova, pro tento účel můžeme využít jednu ze tří možností 4D pro vytváření programových smyček a procházet řetězec znak po znaku a sledovat přitom jestli některý znak potřebujeme převrátit na velké písmeno.

- Repeat…Until
- While…End while
- For…End for

19.6.1. Smyčka ve funkci GEN_Capitalize

1. Přepněte se do Prostředí návrháře.
2. Upravte metodu GEN_Capitalize následujícím způsobem.

```
If (False)
      ` Metoda: GEN_Capitalize
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Převede řetězec na malá písmena a pak převede velké první písmeno
      ` a písmena po mezeře, pomlčce a apostrofu.

   <>fGeneric := False
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_STRING (255; $0; $1)
C_LONGINT ($i)

$0 := Lowercase ($1)
$0[[1]] := Uppercase ($0[[1]])
For ($i; 1; Length ($0) - 1)
   If (($0[[$i]] = " ") | ($0[[$i]] = "-") | ($0[[$i]] = "'"))
      $0[[$i + 1]] := Uppercase ($0[[$i + 1]])
   End if
End for
   ` Konec metody
```

4. Přepněte se do prostředí Vlastní nabídky a vyzkoušejte pole jméno napsáním alois jiří

19.7. Ukazatele dovolují pracovat přímo s originálními daty

Vaše funkce vás omezují na jednu navracenou hodnotu. S použitím ukazatelů, můžete pracovat na více než jedné položce současně a rovněž získáte výhody přímého přístupu k samotným datům a ne pouze ke kopii. Pamatujete si, že ukazatel je něco jako zástupce na položku dat. Protože je to pouze zástupce musíme ukazatel dereferencovat, abychom se dostali ke skutečným datům. A nakonec, protože pracujeme s ukazateli tj. přímo s originálními daty nemusíme již navracet žádnou hodnotu, protož i po ukončení metody zůstanou originální data upravena.

19.7.1. Použití ukazatele v metodě GEN_Capitalize

1. Přepněte se do Prostředí návrháře
2. Upravte metodu GEN_Capitalize následujícím způsobem.

```
If (False)
      ` Metoda: GEN_Capitalize
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Převede řetězec na malá písmena a pak převede velké první písmeno
      ` a písmena po mezeře, pomlčce a apostrofu.

   <>fGeneric := False
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_POINTER ($1)
C_LONGINT ($i)

$1-> := Lowercase ($1->)
$1->[[1]] := Uppercase ($1->[[1]])
For ($i; 1; Length($1->) - 1)
   If (($1->[[$i]] = " ") | ($1->[[$i]] = "-") | ($1->[[$i]] = "'"))
      $1->[[$i + 1]] := Uppercase ($1->[[$i + 1]])
   End if
End for
   ` Konec metody
```

3. Otevřete formulář [Zákazníci];"Vstupní".
4. Klepněte s pomocí Option (Alt) na pole objektu a otevřete metodu objektu.
5. Upravte metodu objektu následujícím způsobem:

```
If (False)
      ` Metoda objektu: Příjmení
      ` Kurs ACI University
```

```
      ` Vytvořeno: Jim Steinman
      ` Datum: 1/15/97

      ` Účel: Konvertuje příjmení do malých písmen
      ` a pak převrátí první písmeno na velké.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

GEN_Capitalize (-> [Zákazníci]Příjmení)
   ` Konec metody objektu
```

19.8. Časté dereferencování ukazatele zpomaluje provádění

Ukazatele jsou ideálním způsobem k ovládání dat ve 4D. Musíte-li je však často dereferencovat spotřebováváte mnoho času procesoru. K tomu, aby jste urychlili provádění metody je vhodnější překopírovat si data do lokální proměnné, pracovat s těmito daty v proměnných, které by opakovaně dereferencovaly ukazatele a do originální proměnné převést až konečný výsledek. Další výhodou tohoto postupu je, že váš kód bude čitelnější a přehlednější až se k němu budete v budoucnu navracet.

19.8.1. Přiřazení lokálních proměnných pro často dereferencované ukazatele

1. Přepněte se do Prostředí návrháře.
2. Upravte metodu GEN_Capitalize následujícím způsobem:

```
If (False)
      ` Metoda: GEN_Capitalize (ukazatel)
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Legenda
      ` $1 - ukazatel na řetězec ke konverzi

      ` Účel: Převede řetězec na malá písmena a pak převede velké první písmeno
      ` a písmena po mezeře, pomlčce a apostrofu.


      `$sStringToCap - používán pro čitelnost a vyloučení dereferencování
      `POZNÁMKA: Metoda pracuje pouze pro řetězce do 255 znaků.

   <>fGeneric := False
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_POINTER ($1)
C_STRING (255; $sStringToCap )
C_LONGINT ($i)

$sStringToCap := Lowercase ($1->)
$sStringToCap [[1]] := Uppercase ($sStringToCap [[1]])
For ($i; 1; Length ($sStringToCap ) - 1)
   If (($sStringToCap [[$i]] = " ") | ($sStringToCap [[$i]] = "-") |
      ($sStringToCap [[$i]] = """))
      $sStringToCap [[$i + 1]] := Uppercase ($sStringToCap [[$i + 1]])
   End if
End for
$1-> := $sStringToCap `                    Zpětné přiřazení hodnoty originálu
```

```
` Konec metody
```

19.9. Generické volání metody GEN_Capitalize

Když vytváříte metodu objektu nebo ji kopírujete z jiného objektu, musíte si stále pamatovat, že je potřeba změnit pole, které se používá jako parametr metody GEN_Capitalize. Jestliže je skutečně naším cílem vytvořit kód tak generický jak je jen možné, máme k v jazyce 4D k dispozici příkaz, který vrátí ukazatel na objekt, ve kterém je proveden. S použitím tohoto příkazu se metoda objektu stane natolik generickou, že může být beze změn zkopírována do kteréhokoliv objektu řetězce.

19.9.1. Úprava metod objektů volajících GEN_Capitalize tak, aby byly generické

1. Přepněte se do Prostředí návrháře.
2. Otevřete formulář [Zákazníci];"Vstupní".
3. Se stisknutým Option (Alt) klepněte na pole Příjmení a otevřete tak metodu objektu.
4. Upravte metodu následujícím způsobem:

```
If (False)
      ` Metoda objektu: alfa pole
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 1/15/97

      ` Účel: Konvertuje do malých písmen
      ` a pak převrátí první písmeno na velké.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

GEN_Capitalize (Self)
   ` Konec metody objektu
```

5. Upravte metody objektu všech polí užívajících metodu GEN_Capitalize (Firma, Jméno, Adresa, Město atd.). Ujistěte se, že v každém z těchto objektů je zaškrtnuta pouze událost Při aktualizaci.

Extra kredit Diskuse

19.10. Přemostění metody GEN_Capitalize.

Příjmení jako McDonald, MacPherason, DeAngelo, de Mornay, ACI US, IBM, spol. s r. o. nevyhovují myšlence této metody. Neexistuje způsob jak napsat metodu která je stoprocentně úspěšná (příjmení MacIntosh vs. počítač Macintosh). Nejlepší způsob jak tento problém vyřešit je umožnit přemostění volání metody GEN_Capitalize. Ve formuláři můžeme vytvořit zaškrtávací políčko, které pokud je vybráno způsobí, že metoda nebude volána nebo neprovede nic.

Extra kredit Cvičení

1. Přepněte se do Prostředí návrháře.
2. Upravte metodu GEN_Capitalize následujícím způsobem:

```
If (False)
      ` Metoda: GEN_Capitalize (ukazatel)
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Legenda
      ` $1 - ukazatel na řetězec ke konverzi

      ` Účel: Převede řetězec na malá písmena a pak převede velké první písmeno
      ` a písmena po mezeře, pomlčce a apostrofu.


      `$sStringToCap - používán pro čitelnost a vyloučení dereferencování
      `POZNÁMKA: Metoda pracuje pouze pro řetězce do 255 znaků.

   <>fGeneric := False
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_POINTER ($1)
C_STRING (255; $sStringToCap )
C_LONGINT ($i)
```

-
```
If (ckCapitalizeOff = 0)
$sStringToCap := Lowercase ($1->)
$sStringToCap [[1]] := Uppercase ($sStringToCap [[1]])
For ($i; 1; Length ($sStringToCap ) - 1)
   If (($sStringToCap [[$i]] = " ") | ($sStringToCap [[$i]] = "-") |
      ($sStringToCap [[$i]] = """))
      $sStringToCap [[$i + 1]] := Uppercase ($sStringToCap [[$i + 1]])
   End if
```

```
            End for
            $1-> := $sStringToCap `                 Zpětné přiřazení hodnoty originálu
•           End if
•           ckCapitalizeOff   := 0

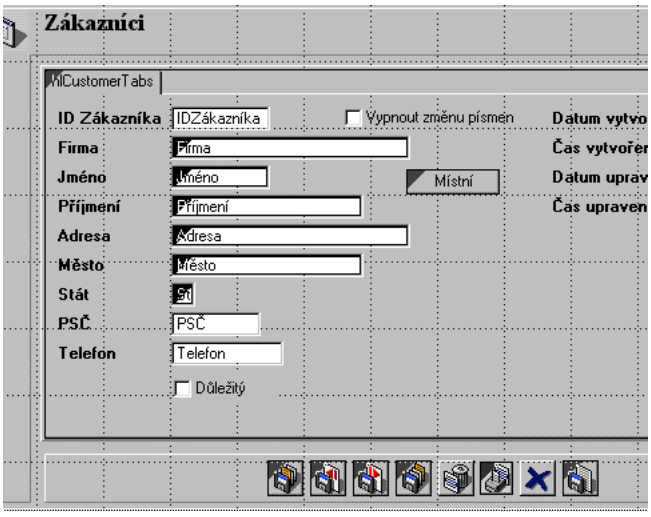
              ` Konec metody
```

3. Otevřete formulář [Zákazníci];"Vstupní".
4. Přidejte zaškrtávací políčko nazvané ckCapitalizeOff s textem Vypnout změnu písmen.
5. Ujistěte se, že políčko není dostupné tabelátorem.
6. Otevřete metodu formuláře [Zákazníci];"Vstupní".
7. Nastavte hodnotu políčka ckCapitalizeOff v události formuláře On Load jak je ukázáno níže:

```
…

Case of
   : ($LFormEvent =   On Load)
      …….

      ckCapitalizeOff   := 0
End case
…
```

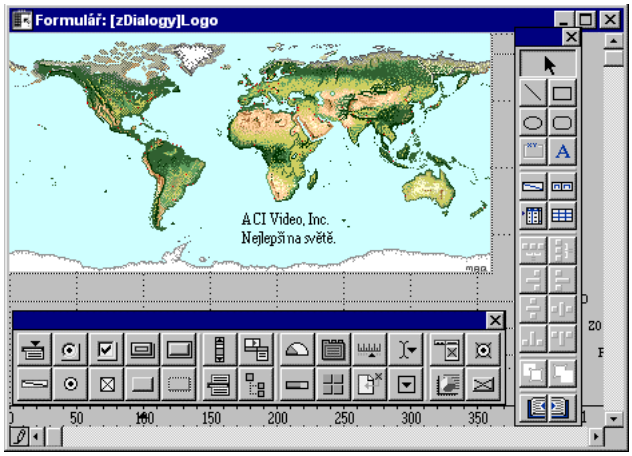


20. Něco navíc

20.1. Přidání vašeho podpisu

Většina softwarových programů, které používáte mají v položce nabídky O aplikaci v nabídce Nápověda něco jako O 4th Dimension…O aplikaci Microsoft Word…(na počítačích Macintosh pod nabídkou Apple).

4D má položku nabídky Nápověda ¬ Co je 4th Dimension…, kterou můžete změnit. Nejenom, že můžete změnit text této položky nabídky, můžete také instruovat 4th Dimension, aby zobrazila vaše vlastní dialogové okno, které jste navrhli v Editoru formulářů.



K tomuto všemu slouží příkaz SET ABOUT, který řekne, aby spustila určitou metodu. V této metodě zobrazíte dialog tak jak jsme se již naučili v dříve.

Váš dialog se objeví na obrazovce s přidanou oblastí nad vrchním okrajem formuláře, kde zobrazí ikonu a číslo verze 4th Dimension a pokud je databáze kompilována rovněž verzi 4D Compiler a případně další informace o ACI. Tato přidaná oblast je veliká přibližně 280 x 72 bodů. To znamená, že při otevírání okna dialogu, musíte pamatovat na toto místo navíc a otevřít o 72 bodů vyšší okno.

Extra kredit Cvičení

20.1.1. Zobrazení dialogu About (Co je…)

1. Otevřete metodu databáze Při spuštění
2. Upravte metodu databáze Při spuštění tak, že vložíte následující kód:

```
…

SET WINDOW TITLE ("ACI Video")

SET ABOUT ("Co je  ACI Video…"; "About")

…
```

3. Vytvořte metodu projektu nazvanou About, obsahující následující kód:

```
If (False)
    ` Metoda: About
    ` Kurs ACI University
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Zobrazí dialog About při volání z nabídky Nápověda .

  <>f_Version6x10 := True
```

```
   <>fJ_Steinman := True
End if

C_LONGINT ($LWidth; $LHeight; $L4DCredits)

$LWidth := 325   `                              Požadovaná šířka okna.
$LHeight := 200 `                               Požadovaná výška okna
$L4DCredits := 72
$LHeight := $LHeight + $L4DCredits
   `            <--- POZNÁMKA použití 50 bodů navíc.
$LWindowID := WIN_LNewWindow ($LWidth; $LHeight; 0) `           Středěné
DIALOG ([zDialogy]; "Logo")
CLOSE WINDOW
   ` Konec metody
```

4. Ukončete a restartujte 4D a v prostředí Vlastní nabídky vyberte Nápověda ¬ Co je ACI Video… a proveďte testování.

Kvíz

- Do nabídky Soubor přidáváte volání tabulky [Kontakty]. Otevřete Záhlaví #1 nebo Záhlaví #2?

20.2. Úprava více záznamů najednou

Tabulka [Zákazníci] obsahuje pole [Zákazníci]ProdejeCelkem. V tomto poli by měly být uloženy celkové prodeje na každého jednotlivého zákazníka, tj. součet všech faktur. Potřebujete způsob jak toto pole naplnit správným číslem. To znamená, alespoň podle našich současných znalostí, že musíte jít do každého záznamu zákazníka nalézt všechny vztažené zánamy ve [Faktury], vypočítat celkový součet pro tyto faktury a výsledek vložit do pole [Zákazníci]ProdejeCelkem. Samozřejmě můžete napsat metodu, která to za vás provede pro jeden záznam zákazníka ve formuláři [Zákazníci];"Vstupní". Později se naučíme jak použít tuto metodu najednou na více než jeden záznam souboru zákazníci.

20.3. Zjištění vztažených záznamů

Abychom nalezli vztažené záznamy potřebujeme se dotázat v tablce [Faktury] na ty záznamy, kde [Faktury]Idzákazníka je rovno poli [Zákazníci]Firma pro záznam zákazníka, se kterým právě zacházíme. Naštěstí 4D poskytuje příkaz, který to provede za vás: RELATE MANY. Jestliže použijete jako parametr pole v tabulce jedinců (v tomto případě [Zákazníci]), použije 4D vztah podle šipky mezi [Zákazníci] a [Faktury] k dotazu na záznamy s toutéž hodnotou v [Faktury]Idzákazníka. Výsledné záznamy můžete sečíst pomocí příkazu SUM

20.4. Zaokrouhlování čísel

Potom, kdy provedete matematické operace (zde je to součet) měli by jste výsledné číslo zaokrouhlit. Způsob jakým počítač provádí matematické operace může vést k malým odlišnostem na zadních desetinných místech. Takže 1+1 nemusí být 2, ale může být rovno 2,00000187 nebo něco podobného. K zaokrouhlení výsledku můžete použít příkaz Round a zakrouhlit výsledek na tolik desetinných míst kolik určíte.

20.4.1. Vytvoření metody k součtu vztažených záznamů

1. Vytvořte metodu nazvanou SumFaktury
2. Napište následující metodu:

```
If (False)
      ` Metoda: SumFaktury
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Sečte faktury zákazníka a uloží výsledek

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_REAL ($rTotal )
```

```
RELATE MANY ([Zákazníci]IDzákazníka)
$rTotal := Sum ([Invoices]CelkemFaktura)
[Zákazníci]ProdejeCelkem := Round ($rTotal; 2)
   ` Konec metody
```

3. Změňte nápis tlačítka bCalculate na Sečíst faktury .
4. Pro toto tlačítko vytvořte následující metodu objektu:

```
If (False)
      ` Metoda objektu: bCalculate
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Vypočte celkové prodeje pro zákazníka

   <>f_Version6x00 := True
   <>fJ_Steinman := True

   ` Modified  1/16/97
   <>f_Version6x10 := True
End if

SumFaktury
   ` Konec metody objektu
```

5. Pole [Zákazníci]ProdejeCelkem umístěte do formuláře [Zákazníci];"Vstupní".
6. Ujistěte se, že toto pole není dostupné a přidělte mu formát zobrazení Částka.
7. Vyzkoušejte je v prostředí Vlastní nabídky v některém ze záznamů klepnutím na nové tlačítko.

Kvíze

- Jestliže chcete zaokrouhlit na koruny, co napíšete v druhém parametru následující řádky?

```
[Zákazníci]ProdejeCelkem := Round($SumTotal; ???)
```

- Povolí vám Finanční úřad zaokrouhlovat daň z přidané hodnoty na koruny?

20.5. Použití metody projektu na více záznamů najednou

Nyní, když můžete sčítat faktury pro jednoho zákazníka, potřebujete napsat metodu, která použije metodu SumFaktury na více záznamů zákazníka.

20.5.1. Použití metody na více záznamů

1. Vytvořte metodu nazvanou PostFaktury následujícím způsobem:

```
If (False)
      ` Metoda: PostFaktury
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Použije metodu SumFaktury opakovaně na
      ` všechny záznamy platného výběru tabulky [Zákazníci].

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

APPLY TO SELECTION ([Zákazníci]; SumFaktury )
   ` Konec metody
```

2. Otestujte tuto metodu v Prostředí uživatele pod položkou nabídky Zvláštní ¬ Provést metodu.

Kvíz

- Pro ty z vás, kteří již víte více o jazyce 4D či o programování obecně, jaké příkazy mohou být použity místo APPLY TO SELECTION?

20.6. Získání informace od uživatele

Provedení metody PostFaktury může zabrat nějaký čas. Uživatel nemusí chtít čekat na dokončení zvláště jestli tuto metodu spustil náhodně. Měli by jste od uživatele chtít potvrzení před tím než se tato metoda spustí.

20.6.1. Úprava metody PostFaktury pro získání potvrzení

1. Upravte metodu PostFaktury následujícím způsobem:

```
If (False)
      ` Metoda: PostFaktury
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Použije metodu SumFaktury opakovaně na
      ` všechny záznamy platného výběru tabulky [Zákazníci].

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

CONFIRM ("Sečíst faktury pro každého zákazníka? (Nelze zpět.) Zabere určitý čas!")
If (OK = 1)
   APPLY TO SELECTION ([Zákazníci]; SumFaktury )
End if
   ` Konec metody
```

2. Proveďte testování.

21. Generování zprávy za pomoci dialogu

Předpokládejme, že manažer společnosti chce vidět neustále aktuální informaci o dnešních prodejích. A to kdykoliv si vzpomene. Tento rys můžete přidat do zápatí formuláře [Faktury];"Výstupní" pomocí tlačítka.

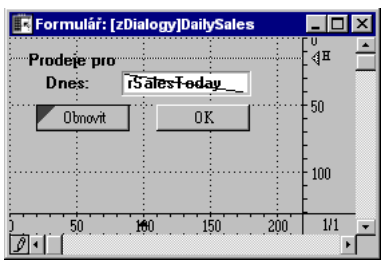
21.1. Použití hlášení příkazem MESSAGE pro informování uživatele

Jestliže provádíte operaci, která zabere více než jen chvíli, měli by jste na to uživatele upozornit. Jinak si uživatel může myslet, že se vyskytl problém s počítačem. Příkaz MESSAGE byl navržen právě pro tento účel. Tento příkaz dočasně zobrazí zprávu v okně do té doby než bude okno překresleno. Můžete jednoduše zavolat příkaz MESSAGE a nemusíte nic uzavírat a mazat, protože se o to 4D postará za vás.

21.2. Použití pojmenovaných výběru k zapamatování si platného výběru

Uživatel si bude prohlížet určité faktuy na obrazovce ve výstupním formuláři a pak klepne na tlačítko Zprávy. Pro splnění akce tohoto tlačítka musíte provést vyhledání dnešních faktur, ale to změní a zruší platný výběr prohlížený uživatelem. Pokud uživatelem k tomuto výběru dospět nějakým složitým způsobem asi nebude příliš spokojen. Odpověď na tento problém je zapamatovat si platný výběr předtím, než provedete dotaz a po dokončení akce změnit platný výběr zpět na původní. Jsou dva způsoby jak to lze ve 4D provést:

Zapamatování platného výběru	Pro	Proti	Paměť/RAM
Sady/Sety	Může být porovnávána s jinými sadami pomocí INTERSECTION, DIFFERENCE a UNION.	"Zapomíná" pořadí třídění. (Pořadí záznamů se vrátí k přirozenému pořadí na disku.)	Používá 1/8 byte (1 bit) na záznam použité tabulky.
Výběry/Named Selection	Pamatuje si pořadí třídění.	Nemůže být porovnáván s jinými pojmenovanými výběry.	Používá 4 bytes na záznam ve výběru.



21.2.1. Přidání tlačítka Dnešní prodeje

1. Vytvořte formulář nazvaný "DnešníProdeje" v tabulce [zDialogy] s následujícími aktivními objekty:

Název:	bOK
Typ:	Tlačítko
Automatická akce:	Přijmout
Popiska:	OK
Styl:	Tlačítka

Název:	bCancel
Typ:	Tlačítko
Automatická akce:	Zrušit
Popiska:	Storno
Styl:	Tlačítka

Název:	rSalesToday
Typ:	Nedostupné
Formát:	|Částka
Vzhled:	Ponořený
Popiska:	Ne
Styl:	Popisky vstupní

2. Vytvořte statické texty "Prodeje pro" a "Dnes:"
3. Otevřete formulář [Faktury];"Výstupní".
4. Do zápatí formuláře [Faktury];"Výstupní" přidejte aktivní objekt s následujícími vlastnostmi

Název:	bDailySales
Typ:	Tlačítko
Automatická akce:	Žádná
Popiska:	Dnešní prodeje
Událost formuláře:	Při klepnutí a Při poklepnutí
Styl:	Tlačítka

5. Přidejte metodu objektu pro tlačítko bDailySales.

```
If (False)
      ` Metoda objektu: bDailySales
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum:  15/1/97

      ` Účel: Dotáže se na dnešní faktury, sečte je
      ` a zobrazí výsledek v dialogu.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT ($LWidth; $LHeight;$LWindowID)

MESSAGE ("Čekejte prosím." + Char(Carriage return) + "Výpočet dnešních prodejů v běhu.")
CUT NAMED SELECTION ([Faktury]; "BeforeQueryInvoicesNS")
QUERY ([Invoices]; [Faktury]DatumFaktury = Current Date)
rSalesToday := Sum ([Faktury]FakturaCelkem)
rSalesToday := Round (rSalesToday; 2)
USE NAMED SELECTION ("BeforeQueryInvoicesNS")

$LWidth := 200 `                    Požadovaná šířka okna.
$LHeight := 100 `                       Požadovaná výška.
$LWindowID := WIN_LNewWindow ($LWidth; $LHeight; <>WIN_LUpper;<>WIN_LModal)

DIALOG ([zDialogy]; "DnešníProdeje")
CLOSE WINDOW
   ` Konec metody objektu
```

6. Přepněte se do Prostředí uživatele.
7. Přepněte se do tabulky [Faktury].
8. Vyberte několik faktur a zvolte Dotazy ¬ Vybrat označené.
9. Použijte následující výraz s pomocí položky nabídky Vstup ¬ Užít výraz a změňte tak datum faktur na dnešní datum, takže dotaz pro dnešní faktury nalezne nějaké záznamy:

```
[Faktury]InvoiceDate := Current date
```

10. Otevřete v prostředí Vlastní nabídky.

Kvíz

- Co vyžaduje více paměti sady nebo pojmenované výběry?
- Jaký je základní účel sad a pojmenovaných výběrů?

- Jaký je rozdíl mezi sadami a pojmenovanými výběry?
- Proč je předchozí otázka nepovedenou slovní hříčkou (nápověda:rozdíl)?

21.3. Používejte tlačítka a nabídky správně, aby jste vytvořili intuitivní rozhraní.

Umístěním akcí do nabídek uživatel bude vědět, kde má hledat, jestliže chce určitou akci provést. Tlačítka na formuláři uživatel nevidí, pokud není formulář zobrazen. V tomto případě nemá uživatel vizuální přehled, kerý by mu pomohl si připomenout jak určitou akci provést. Jestliže jsou akce v nabídce jsou vždy viditelné.

21.4. Uchování dialogu o denních prodejích na obrazovce

Je často příjemné, mít otevřeno více oken s různými funkcemi současně. Dialog DenníProdeje může být jeden z těch, keré je příležitostně dobré držet na obrazovce stále otevřené. S naším současným modálním oknem však musíme být opatrní. Chceme proto změnit současné okno na nemodální typ, tak že i s tímto otevřeným oknem bude umožněna práce i s jinými okny na obrazovce.

21.4.1. Přemístění metody objektu tlačítka bDailySales do metody peojektu a odděleného procesu

1. Přepněte se do Prostředí návrháře.
2. Otevřete formulář [Faktury];"Výstupní".
3. Zkopírujte metodu objektu tlačítka bDailySales do schránky.
4. Vytvořte novou metodu projektu nazvanou M_DailySales.
5. Vložte metodu objektu do této nové metody.
6. Otevřete Záhlaví #1.
7. Do nabídky Zprávy přidejte novou položku nazvanou Denní Prodeje.
8. Přiřaďte metodu M_DailySales této nové položce nabídky.
9. Otevřete Záhlaví #3.
10. Do nabídky Zprávy, přidejte novou položku nazvanou Denní prodeje.
11. Přiřaďte metodu M_DailySales této nové položce.
12. Otevřete Záhlaví #4.
13. Do nabídky Zprávy přidejte novou položku nazvanou Denní prodeje.
14. Přiřaďte metodu M_DailySales této nové položce nabídky.
15. Upravte řádek kódu metody M_DailySales, který se odkazuje na metodu projektu WIN_LNewWindow následovně:

```
$LWindowID := WIN_LNewWindow ($LWidth; $LHeight; <>WIN_LUpper;
<>WIN_LStaardNoResize)
```

16. Přepněte se do prostředí Vlastní nabídky.

17. Klepněte na úvodní obrazovku a přeneste ji tak na popředí.
18. Klepněte na tlačítka Option (Alt) + f.
19. Přepněte se zpět do prostředí Vlastní nabídky a vyzkoušejte položku Denní prodeje.

21.5. Použití více úloh k přidělení stejné váhy všem otevřeným oknům

Když nyní klepnete na jiné okno, stane se zvláštní věc. Klepnutím do jiného okna způsobíte, že nynější okno na popředí (dialog Denní prodeje) bude překryto. Když se aktivní okno, ve kterém chceme provést následující akci dostane na popředí změní se ukazatel myši tak, že tento ukazatel znázorňuje, že používáme neaktivní okno. Musíme proto spustit metodu M_DailySales tak, aby byla provedena ve svém vlastním procesu a tím ji dáme stejnou důležitost jakou mají ostatní okna na obrazovce. Pak bude přecházení mezi okny přecházení mezi procesy a ne mezi aktivním a neaktivním oknem.

21.6. Procesy vyžadující okno si otevřou své vlastní, pokud jej neotevřete vy!

Když nastartujeme nový proces, příkaz MESSAGE potřebuje okno, ve kterém by zobrazoval informace. Jestliže je neotevřeme pro zobrazování hlášení, 4D za nás automaticky okno otevře. Toto okno může nebo nemusí být to co chceme. Jestliže však přemístíme řádek kódu WIN_LNewWindow dopředu před příkaz MESSAGE, budeme mít tuto zprávu zobrazenu v našem vlastním okně.

21.7. Text měnící pozici na obrazovce je nepříjemný

Lidé mají rádi, když mohou svou pozornost zaměřit na omezenou oblast. Hlášení MESSAGE se objevují na jednom místě obrazovky a důležitá data se zobrazují na jiném. Můžeme stanovit, kde přesně se naše hlášení bude zobrazovat a to přemístěním neviditelného kurzoru na pozici obrazovky, kde chceme zprávu zobrazit. Tento příkaz se jmenuje XY.

21.8. Procesy mají své vlastní platné výběry

Protože každý proces má svůj vlastní platný výběr a my již pro zobrazení dialogu nepoužíváme proces faktur, je kód obnovující pojmenovaný výběr zbytečný, jednoduše jej z kódu vymažeme.

21.9. Každý proces má vlastní záhlaví nabídek

Když se pokusíme použít tento proces, beze změny bude záhlaví nabídek prázdné, každý proces musí mít definováno své vlastní záhlaví nabídek. Takže musíme definovat, které záhlaví nabídek proces používá.

21.9.1. Úprava dialogu Denní prodeje tak, aby byl skutečně nemodální

1. Otevřete Záhlaví #1.
2. Z nabídky Zprávy vyberte položku Denní prodeje.
3. Zaškrtněte políčko Začít nový proces.
4. Opakujte tento postup pro Záhlaví #3 a Záhlaví #4.
5. Otevřete formulář [zDialogy];"DenníProdeje".
6. Přidělte záhlaví nabídek pro tento formulář pomocí položky nabídky Formulář->Záhlaví nabídek… (pamatujeme si, že musíme použít zápornou hodnotu (“-2”)).
7. Upravte metodu M_DailySales method následovně:

```
If (False)
      ` Metoda: M_DailySales
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum:  15/1/97

      ` Účel: Vyhledá dnešní prodeje, sečte je a zobrazí výsledek
      ` v dialogu.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT ($LWidth; $LHeight; $LWindowID)

MENU BAR (4)
$LWidth := 200 `                    Požadovaná šířka okna.
$LHeight := 100 `                   Požadovaná výška.
$LWindowID := WIN_LNewWindow ($LWidth; $LHeight;
<>WIN_LUpper;<>WIN_LStaardNoResize)

GOTO XY (0; 2)
MESSAGE ("Čekejte prosím." + Char(Carriage return) + "Výpočet dnešních prodejů v běhu.")

QUERY ([Faktury]; [Faktury]DatumFaktuy = Current Date)
rSalesToday := Sum ([Faktury]CelkemFaktura)
rSalesToday := Round (rSalesToday; 2)

DIALOG ([zDialogy]; "DenníProdeje")
CLOSE WINDOW
   ` End of method
```

8. Odstraňte tlačítko bDailySales z formuláře [Faktury];"Výstupní".
9. Přepněte se do prostředí Vlastní nabídky a vyzkoušejte položku Denní prodeje.

21.10. Obnova hodnot v jiném čase než při prvním spuštění

Dialogové okno denní prodeje se obnoví pouze při prvním otevření. Abychom jej obnovili musíme je uzavřít a znovu otevřít. Jednoduchým řešením je přidat tlačítko pro obnovu částky denních prodejů. Kód, který to provede již existuje v metodě M_DailySales. Můžeme pouze kód zkopírovat a uložit do metody objektu tlačítka, ale tak bychom zduplikovali část kódu což není nezbytné. Navíc pokud budeme potřebovat upravit kód později, budeme jej muset upravit na dvou místech. Místo toho vytvoříme metodu projektu, umístíme do ní kód a zavoláme jej kdykoliv bude potřeba.

21.10.1. Vytvoření tlačítka obnovujícího dialog Denní prodeje

1. Vytvořte metodu prodeje nazvanou CalculateDailySales.
2. Otevřete metodu M_DailySales.
3. Kopírujte řádky s QUERY, Sum a Round.
4. Vložte je do metody CalculateDailySales.

```
If (False)
      ` Metoda: CalculateDailySales
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Vypočítává částky pro denní prodeje.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

MESSAGES OFF
QUERY ([Faktury]; [Faktury]DatumFaktury = Current Date)
MESSAGES ON
rSalesToday := Sum ([Faktury]FakturaCelkem)
rSalesToday := Round (rSalesToday; 2)
   ` Konec metody
```

5. V metodě M_DailySales nahraďte řádky s QUERY, Sum a Round řádkou CalculateDailySales.
6. Otevřete formulář [zDialogy];"DenníProdeje".
7. Změňte tlačítko bCancel na tlačítko s žádnou akcí nazvané bCalculate a s popiskou Obnovit.
8. Otevřete metodu objektu nového tlačítka
9. Přidejte následující metodu objektu.

```
If (False)
      ` Metoda objektu: bCalculate
      ` Kurs ACI University
```

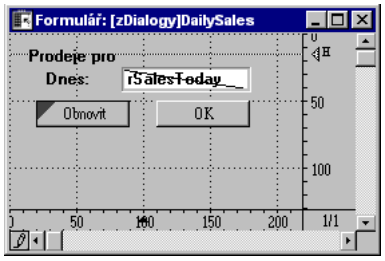
```
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Umožní uživateli obnovit součet pro denní prodeje.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

CalculateDailySales
   ` Konec metody objektu
```

10. Přepněte so do prostředí Vlastní nabídky a vyzkoušejte nové tlačítko.



22. Vytváření jedinečných čísel a další úkoly

Vestavěné pořadové číslo 4D (sequence number má určitá omezení). Nemůžete začít na hodnotě, kterou si vyberete. Jestliže budete muset exportovat záznamy do jiné databáze a záznamy budou mezitím vymazány je možné, že dostanete zdvojená čísla. Proto je potřeba ukládat důležitá čísla někam, kde uživatel může nastavit počáteční hodnotu. My splníme tuto úlohu vytvořením tabulky, která bude pouze generovat pořadová čísla.

22.1. Vytvoření tabulky zSekvence a metody LNextSequence

1. Vytvořte novou tabulku a pojmenujte ji [zSekvencences].
2. Přidejte pole Alfa 20 nazvané NázevSekvence s následujícími vlastnostmi:

- Povinné
- Neměnné
- Indexované
- Jedinečné

3. Přidejte pole Longint nazvané Hodnota.
4. Vytvořte metodu projektu nazvanou LNextSequence následovně:

```
If (False)
      ` Metoda: LNextSequence (string)
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Legenda
      ` $1 - String - Název čísla, které bude navraceno

      ` Účel: Udržuje tabulku čísel pro generování jedinečných čísel.

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_STRING (15; $1)
C_LONGINT ($0)

QUERY([zSekvence]; [zSekvence]NázevSekvence = $1)
```

```
If (Records in selection([zSekvence]) = 0)
   $0 := -1
Else
   [zSekvence]Hodnota := [zSekvence]Hodnota + 1
   $0 := [zSekvence]Hodnota
   SAVE RECORD ([zSekvence])
End if
   ` Konec metody
```

5. Přepněte se do Prostředí uživatele.
6. Vytvořte nový záznam pro tabulku [zSekvence]:

NázevSekvence:	ČísloFaktury
Hodnota:	10469
NázevSekvence:	IDzákazníka
Hodnota:	234

7. Přepněte se zpět do Prostředí návrháře.
8. Otevřete formulář [Faktury];"Vstupní".
9. Poklepejte na objekt pole ČísloFaktury.
10. Na stránce Řízení dat odstraňte z políčka Výchozí #N.
11. Otevřete metodu formuláře [Faktury];"Vstupní".
12. Upravte metodu následovně:

```
If (False)
      ` Metoda formuláře: Vstupní
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Metoda vstupní formuláře pro  [Faktury]; "Vstupní"

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($LFormEvent)

$LFormEvent := Form event

Case of
   : ($LFormEvent =   On Load)
      If (pTable =(-> [Zákazníci]))          ` Přicházíme ze zákazníků.
         SET VISIBLE(bModify;False)          ` Nelze jít dále na úpravu zákazníka.
         SET WINDOW TITLE("Upravuje se faktura  " + String ([Faktury]ČísloFaktury))
      Else
•        If (Record number([Faktury]) = -3) `Nový záznam
•           [Faktury]ČísloFaktury := LNextSequence ("ČísloFaktury")
```

```
                End if
                WIN_InputWindowTitle
            End if
        End case
        ` Konec metody
```

13. Otevřete metodu formuláře [Zákazníci];"Vstupní".

14. Upravte metodu následujícím způsobem:

```
If (False)
      ` Metoda formuláře: [Zákazníci];"Vstupní"
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Metoda formuláře [Zákazníci];"Vstupní"

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($LFormEvent)

$LFormEvent := Form event

Case of
   : ($LFormEvent =   On Load)
      If (pTable=(->[Faktury])
         SET VISIBLE (*;"INVOICEButton@";False) `      Nelze provádět funkce pro faktury
         SET WINDOW TITLE("Upravuje se "+[Zákazníci]Firma)
      Else
         WIN_InputWindowTitle
      End if

      If (Record number([Zákazníci])= -3)
•        [Zákazníci]IDzákazníka := String(LNextSequence ("IDzákazníka"))
         [Zákazníci]DatumVytvoření := Current date
         [Zákazníci]ČasVytvoření := Current time
      End if

      cKCapitalizeOff:= 0
End case
      ` Konec metody
```

15. Nastavte pole IDzákazníka na Pouze zobrazit
16. Přepněte se zpět do prostředí Vlastní nabídky a přidejte novou fakturu k otestování nové funkce.

Extra kredit Diskuse

22.2. Vytváření následných pořadových čísel bez chybějících čísel

Pro některé položky jako čísla faktur a další finanční dokumenty je nezbytné udržovat perfektní následnost čísel. S tímto máme zatím problém. Jestliže použijete výše popsanou metodu a po získání dalšího čísla, uživatel zruší záznam při přidání dalšího nového záznamu, budeme o číslo dále. Jestliže při zrušení záznamu je ještě číslo v pořadových číslech stejné, můžeme toto číslo jednoduše snížit. V prostředí více uživatelů a současně běžících procesů přidávání faktur se může stát, že od vytvoření nového záznamu do jeho zrušení již někdo přidal další záznam. V tomto případě si čísla neodpovídají a samozřejmě nelze jednoduše číslo vrátit. Jediné řešení je potom uložit někam toto navracené číslo a při vytváření nových záznamů se podívat jestli existují nějaká navrácená nepoužitá čísla.

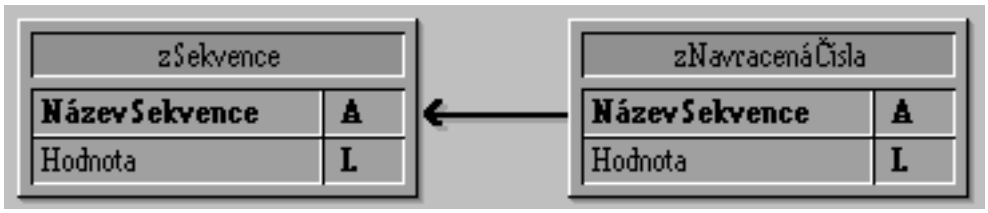
Extra kredit Cvičení

22.2.1. Udržování pořadových čísel bez chybějících čísel

1. Vytvořte novou tabulku a nazvěte ji [zNavracenáČísla].
2. Přidejte pole Alfa 20 pojmenované NázevSekvence s následujícími vlastnostmi:

- Povinné
- Neměnné
- Indexované

3. Přidejte pole Longint field a nazvěte jej Hodnota.
4. Vytvořte vztah z [zNavracenáČísla]NázevSekvence do [zSekvence]NázevSekvence.



4. Nahraďte metodu projektu nazvanou LNextSequence následující metodou:

```
If (False)
    ` Metoda: LNextSequence (string; Longint)
    ` Kurs ACI University
    ` Generická procedura
    ` Vytvořeno: Jim Steinman
    ` Datum: 1/15/97

    ` Legenda
    ` $1 - String - Název čísla k vrácení.
    ` $2 - Longint - (Volitelné) číslo, které se navrací.

    ` Účel: Udržuje tabulky čísel pro generování jedinečný čísel.

  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if

C_STRING (15; $1)
C_LONGINT ($0; $2)

Case of
  : (Count parameters = 1) `                              Potřebujeme nové číslo
    QUERY ([zSekvence]; [zSekvence]NázevSekvence = $1)
    If (Records in selection ([zSekvence]) = 1)
      RELATE MANY ([zSekvence] NázevSekvence)
      If (Records in selection ([zNavrácenáČísla]) > 0) `  Nejdříve zkontrolujte navrácená čísla
        ORDER BY ([zNavrácenáČísla]Hodnota)
        $0:= [zNavrácenáČísla]Hodnota
        DELETE RECORD ([zNavrácenáČísla])
      Else
        $0 := [zSekvence] Hodnota
        [zSekvence] Hodnota:= [zSekvence] Hodnota + 1
        SAVE RECORD ([zSekvence])
      End if

    Else
      CREATE RECORD ([zSekvence])
      [zSekvence] NázevSekvence:= $1
      [zSekvence] Hodnota:= 2
      $0 := 1
      SAVE RECORD ([zSekvence])
    End if

  Else `                                                  Navracíme číslo do databáze
    CREATE RECORD ([zNavrácenáČísla])
    [zNavrácenáČísla] NázevSekvence:= $1
    [zNavrácenáČísla]Hodnota := $2
    SAVE RECORD ([zNavrácenáČísla])
    $0 := 0
End case
  ` Konec metody
```

5. Upravte metodu formuláře [Faktury];"Vstupní" následujícím způsobem:

```
If (False)
      ` Metoda formuláře: Vstupní
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Metoda vstupní formuláře pro  [Faktury]; "Vstupní"

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($LFormEvent)

$LFormEvent := Form event

Case of
   : ($LFormEvent =   On Load)
•     fNewInvoice := False
      If (pTable =(-> [Zákazníci]))          ` Přicházíme ze zákazníků.
         SET VISIBLE(bModify;False)          ` Nelze jít dále na úpravu zákazníka.
         SET WINDOW TITLE("Upravuje se faktura  " + String ([Faktury]ČísloFaktury))
      Else
         If (Record number([Faktury]) = -3) `Nový záznam
•           fNewInvoice := True
            [Faktury]ČísloFaktury := LNextSequence ("ČísloFaktury")
         End if
         WIN_InputWindowTitle
      End if
End case
   ` Konec metody
```

6. Nahraďte všechna tlačítka naspodu formuláře [Faktury];"Vstupní" tlačítky z formuláře [zDialogy];"Input buttons".
7. Prohlédněte si následující kód pod tlačítkem bCancel ve formuláři [Faktury];"Vstupní". (Poznámka: Toto tlačítko je pod zápatím)

```
If (False)
      ` Metoda objektu: bCancel
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Provede metodu zrušení faktury

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if
```

```
EXECUTE("E_"+Table name(Current form table)+"Storno")
   ` Konec metody objektu
```

7. Tentýž kód existuje pod tlačítky bEscape a bDelete.
8. Vytvořte novou metodu projektu nazvanou E_FakturyStorno s použitím následujícího kódu:

```
If (False)
      ` Metoda: E_FakturyStorno
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Spustí se, když uživatel zruší formulář Faktury vstupní
      ` Je-li potřeba navrátí číslo do souboru

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($x)                        `nevyužitá proměnná pro funkci navracení hodnoty

If (fNewInvoice )
   $x := LNextSequence ("ČísloFaktury";[Faktury]ČísloFaktury)
End if
      `Konec metody
```

Extra kredit Diskuse

22.3. Optimalizace dotazů pro hodnoty vztažených dat

Dotazy používající pole ze vztažených tabulek, nemohou být nikdy dost rychlé. Když jsou pole ze vztažených tabulek použita v dotazu, dotak je vždy sekvenční. Dotazy podle vztažených polí mohou být optimalizovány pomocí příkazů RELATE ONE SELECTION a RELATE MANY SELECTION.

Extra kredit Cvičení

22.3.1. Úpravy tlačítek dotazů pro optimalizaci dotazů

1. Ve formuláře [Faktury];"Výstupní" vytvořte nové tlačítko.
2. Změňte metodu objektu následovně:

```
$hStart := Current time
QUERY ([Faktury]; [Zákazníci]Stát = "CA")
$hStop := Current time
$hTotal := $hStop - $hStart
ALERT ("Dotaz trval:" + String($hTotal))
WIN_OutputWindowTitle
```

3. Duplikujte tlačítko bQuery ve formuláři [Faktury];"Výstupní".
4. Změňte název tlačítka na bQueryPS a změňte jeho text na Promítnutí, metodu objektu změňte následovně:

```
$hStart := Current time
QUERY ([Zákazníci]; [Zákazníci]Stát = "CA")
RELATE MANY SELECTION ([Faktury]Firma)
$hStop := Current time
$hTotal := $hStop - $hStart
ALERT ("Dotaz trval:" + String($hTotal))
WIN_OutputWindowTitle
```

5. Pomocí klepnutí s klávesou Shift, vyberte obě tlačítka, které jste právě vytvořili a použijte nástroj duplikace k duplikování obou tlačítek. Obě tlačítka vyjměte a vložte je do formuláře [Zákaznící];"Výstupní".
6. Změňte metodu objektu tlačítka bQuery ve formuláři [Zákazníci];"Výstupní" na následující:

```
$hStart := Current time
QUERY ([Zákazníci]; [Faktury]DatumFaktury = !09/06/91!)
$hStop := Current time
$hTotal := $hStop - $hStart
ALERT ("Dotaz trval:" + String($hTotal))
WIN_OutputWindowTitle
```

7. Otevřete dialog definic objektu tlačítka bQueryPS, ve formuláři [Zákazníci];"Výstupní". Změňte název tlačítka na bJoin, změňte text na Propojení a upravte metodu objektu následujícím způsobem:

```
$hStart := Current time
QUERY ([Faktury]; [Faktury]DatumFaktury = !09/06/91!)
RELATE ONE SELECTION ([Faktury]; [Zákazníci])
$hStop := Current time
$hTotal := $hStop-$hStart
ALERT ("Dotaz trval:" + String($hTotal))
WIN_OutputWindowTitle
```

8. Přepněte se do prostředí Vlastní nabídky a při testování porovnejte výsledky obou tlačítek ve formulářích [Zákazníci] i [Faktury]. Všimněte si, zvýšení rychlosti při použití RELATE ONE SELECTION nebo RELATE MANY SELECTION oproti prostému hledání přes relace.

22.4. Exportování záznamů v pevné délce

Mnoho z programů a uživatelů počítačového světa nepoužívá pole a záznamy oddělované oddělovači. Místo toho užívají pevnou délku polí a záznamů. K tomu, aby jste mohli exportovat data v pevné délce, musíte vytvořit vlastní exportní metodu psanou přesně podle tabulek. Použitím příkazu SET CHANNEL, můžete otevřít diskový soubor a použitím příkazu SEND PACKET do něj můžete zapisovat data v tabulkové formě.

22.4.1. Vytvoření metody k exportu záznamů v pevné délce.

1. Vytvořte metodu projektu nazvanou IMPEXP_ExportFixed a pak napište následující:

```
If (False)
      ` Metoda: IMPEXP_ExportFixed
      ` Modul: Import a Export
      ` Kurs ACI University
      ` Generická procedura
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Exportuje záznamy v pevné délce.

   <>fGeneric := False
   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT ($i; $LWidth; $LHeight;$LWindowID)

SET CHANNEL (12; "")
If (OK = 1)
   $LWidth := 115 `                    Nastaví šířku vhodnou pro hlášení a dialog.
   $LHeight := 35
   $LWindowID := WIN_LNewWindow ($LWidth; $LHeight; 0;
$LWindowType;<>WIN_LCentered)
   GOTO XY (1; 1) `                    Řídí umístění hlášení v dialogu, jeden znak zleva
   MESSAGE ("Probíhá export záznamů")
   FIRST RECORD ([Zákazníci])
   For ($i; 1; Records in selection ([Zákazníci]))
      SEND PACKET ([Zákazníci]Jméno + (20 - Length ([Zákazníci] Jméno) * " "))
      SEND PACKET ([Zákazníci]Příjmení + (30 - Length ([Zákazníci]Příjmení) * " ") +
            Char (13))
      NEXT RECORD ([Zákazníci])
   End for
   CLOSE WINDOW
End if
SET CHANNEL (11)
```

```
    ` Konec metody
```

2. Přepněte se do Prostředí uživatele.
3. Zobrazte tabulku [Zákazníci].
4. Vyberte Zvláštní ¬ Provést metodu a proveďte metodu IMPEXP_ExportFixed.
5. Otevřete soubor na disku textovým procesorem.

22.5. Vylepšení chování metody IMPEXP_ExportFixed

Opakované SEND PACKET opakovaně přistupuje k pevnému disku. Diskový přístup je nejpomalejší vlastností počítače a my jsme schopni vylepšit chování jestliže omezíme množství diskových přístupů naší metody na minimum. Protože textová proměnná může obsahovat až 32000 znaků, jsme schopni skládat náš export do textové proměnné a zapisovat na disk pouze čas od času, když jsme již blízko limitu proměnné.

22.5.1. Vylepšení chování IMPEXP_ExportFixed

1. Otevřete metodu projektu IMPEXP_ExportFixed a upravte ji následujícím způsobem:

```
If (False)
    ` Metoda: IMPEXP_ExportFixed
    ` Modul: Import a Export
    ` Kurs ACI University
    ` Generická procedura
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Exportuje záznamy v pevné délce.

  <>fGeneric := False
  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if

C_LONGINT ($i; $j; $LWidth; $LHeight; $LWindowID; $LLastPosition)
C_TEXT ($tText)

SET CHANNEL (12; "")
If (OK = 1)
  $LWidth := 115 `                Nastaví šířku vhodnou pro hlášení a dialog
  $LHeight := 35
  $LWindowType := 1
  $LWindowID := WIN_LNewWindow ($LWidth; $LHeight; 0;
$LWindowType;<>WIN_LCentered)
  GOTO XY (1; 1) `                Řídí umístění hlášení v dialogu, jeden znak zleva
  MESSAGE ("Probíhá export záznamů")
  FIRST RECORD ([Zákazníci])
  $tText := " " * 32000
  $LLastPosition := 0
  For ($i; 1; Records in selection ([Zákazníci]))
```

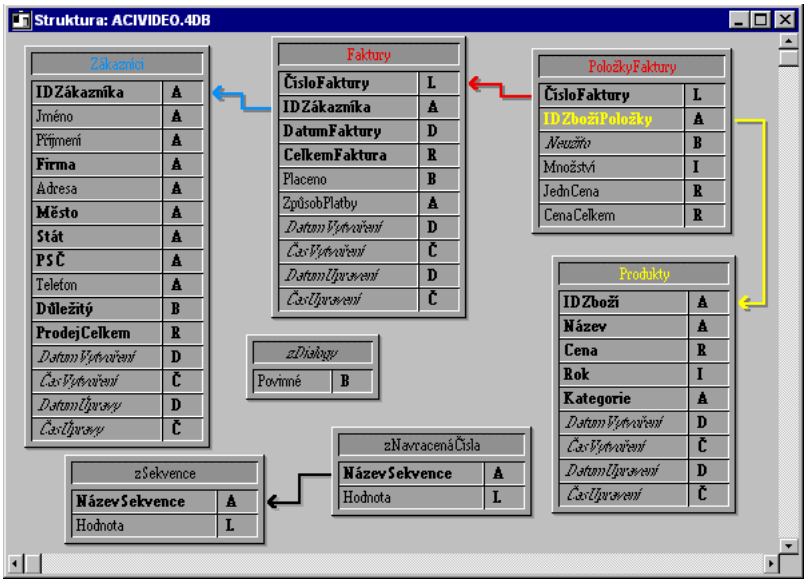
```
      For ($j; 1; Length ([Zákazníci]Jméno))
         $tText•$LLastPosition + $j• := [Zákazníci]Jméno •$j•
      End for
      $LLastPosition := $LLastPosition + 20
      For ($j; 1; Length ([Zákazníci]Příjmení))
         $tText•$LLastPosition + $j• := [Zákazníci]Příjmení•$j•
      End for
      $LLastPosition := $LLastPosition + 30
      $tText•$LLastPosition + 1• := Char (Carriage return)
      $LLastPosition := $LLastPosition + 1
      If ($LLastPosition > 30000) `   Zapisuje do souboru blízko limitu
         SEND PACKET (Substring ($tText; 1; $LLastPosition))
         $tText := " " * 32000
         $LLastPosition := 0
      End if
      NEXT RECORD ([Zákazníci])
   End for
   SEND PACKET (Substring ($tText; 1; $LLastPosition)) `   Poslat poslední kousek
   CLOSE WINDOW
End if
SET CHANNEL (11)
   ` Konec metody
```

22.6. Generické označování záznamů datumem a časem.

Budeme určitě chtít označovat si všechny záznamy hlavních tabulek datumem a časem vytvoření a změny. Náš současný kód pracuje pouze pro tabulku [Zákazníci]. Kód můžeme upravit pro každou ze zamýšlených tabulek, bylo by však podstatně účinnější vytvořit generickou metodu, která bude požadovanou provádět. To nám sníží množství duplikovaného kódu pro obnovu každé tabulky.

Nyní přidejme čtyři pole do tabulek [Faktury] a [Produkty]. Jsou to:

- DatumVytvoření
- ČasVytvoření
- DatumÚpravy
- ČasÚpravy



22.6.1. Přidání polí ukládání datumu a času do struktury a formuláře

1. Jděte do Prostředí návrháře.
2. Zvolte Návrh¬ Struktura.
3. V okně struktury, klepněte na tabulku [Faktury].
4. Vyberte Struktura ¬ Nové pole.
5. Přidejte následující pole:

Název pole	Typ	Vlastnosti
DatumVytvoření	Datum	Pouze zobrazit, Neviditelné
ČasVytvoření	Čas	Pouze zobrazit, Neviditelné
DatumÚpravy	Datum	Pouze zobrazit, Neviditelné
ČasÚpravy	Čas	Pouze zobrazit, Neviditelné

22.6.2. Přidání polí ukládání datumu a času do struktury a formuláře

1. Jděte do Prostředí návrháře.
2. Zvolte Návrh¬ Struktura.
3. V okně struktury, klepněte na tabulku [Produkty].
4. Vyberte Struktura ¬ Nové pole.
5. Přidejte následující pole:

Název pole	Typ	Vlastnosti
DatumVytvoření	Datum	Pouze zobrazit, Neviditelné
ČasVytvoření	Čas	Pouze zobrazit, Neviditelné
DatumÚpravy	Datum	Pouze zobrazit, Neviditelné
ČasÚpravy	Čas	Pouze zobrazit, Neviditelné

22.6.3. Přidání polí do vstupních formulářů

1. Otevřete formulář [Produkty];"Vstupní".
2. Umístěte čtyři nová pole do formuláře.
3. Na formuláři [Faktury];"Vstupní" není místo kam bychom tato pole umístili, přesto použijeme kód k naplňování těchto informací a pokud je budeme potřebovat v budoucnu, můžeme si je vytisknout rychlou zprávou.

22.6.4. Přidání generické metody k ovládání polí datum a čas

1. Vytvořte novou metodu nazvanou GEN_TimeStamp následujícím způsobem:

```
If (False)
      Metoda: GEN_TimeStamp (datum;čas)
      `Kurs ACI University
      `Generická procedura
      `Vytvořeno: Jim Steinman
      `Datum: 15/1/97

      `Účel: Procedurálně naplňuje pole Datum a Čas při vytvoření a úpravách

   <>fGeneric:=True
   <>f_Version6x10:=True
   <>fJ_Steinman:=True

      `$1 - Ukazatel na datum
      `$2 - Ukazatel na čas

End if

C_POINTER($1;$2)

$1-> := Current date(*)
$2-> := Current time(*)
   ` Konec metody
```

2. Otevřete metodu formuláře [Zákazníci];"Vstupní".
3. Upravte tuto metodu následovně:

```
If (False)
      ` Metoda formuláře: [Zákazníci];"Vstupní"
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Metoda formuláře [Zákazníci];"Vstupní"

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

C_LONGINT($LFormEvent)

$LFormEvent := Form event

Case of
   : ($LFormEvent =   On Load)
      If (pTable=(->[Faktury])
         SET VISIBLE (*;"INVOICEButton@";False) `      Nelze provádět funkce pro faktury
```

```
      SET WINDOW TITLE("Upravuje se "+[Zákazníci]Firma)
    Else
      WIN_InputWindowTitle
    End if

    If (Record number([Zákazníci])= -3)
      [Zákazníci]IDzákazníka := String(LNextSequence ("IDzákazníka"))
      GEN_TimeStamp (>[Zákazníci]DatumVytvoření;->[Zákazníci]ČasVytvoření)
    End if

    cKCapitalizeOff:= 0
End case
    ` Konec metody
```

4. Otevřete metodu formuláře [Faktury];"Vstupní".
5. Upravte metodu následovně:

```
If (False)
    ` Metoda formuláře: Vstupní
    ` Kurs ACI University
    ` Vytvořeno: Jim Steinman
    ` Datum: 15/1/97

    ` Účel: Metoda vstupní formuláře pro  [Faktury]; "Vstupní"

  <>f_Version6x10 := True
  <>fJ_Steinman := True
End if

C_LONGINT($LFormEvent)

$LFormEvent := Form event

Case of
  : ($LFormEvent =  On Load)
    fNewInvoice := False
    If (pTable =(-> [Zákazníci]))          ` Přicházíme ze zákazníků.
      SET VISIBLE(bModify;False)           ` Nelze jít dále na úpravu zákazníka.
      SET WINDOW TITLE("Upravuje se faktura  " + String ([Faktury]ČísloFaktury))
    Else
      If (Record number([Faktury]) = -3) `Nový záznam
        fNewInvoice := True
        [Faktury]ČísloFaktury := LNextSequence ("ČísloFaktury")
      GEN_TimeStamp (->[Faktury]DatumVytvoření;->[Faktury]ČasVytvoření)
      End if
      WIN_InputWindowTitle
    End if
End case
  ` Konec metody
```

6. Otevřete metodu formuláře [Produkty];"Vstupní".
7. Upravte metodu následovně:

```
If (False)
      ` Metoda formuláře: Vstupní
      ` Kurs ACI University
      ` Vytvořeno: Jim Steinman
      ` Datum: 15/1/97

      ` Účel: Metoda vstupního formuláře pro [Produkty]; "Vstupní"

   <>f_Version6x10 := True
   <>fJ_Steinman := True
End if

Case of
   :  (Form event = On Load)
      WIN_InputWindowTitle
         If (Record number([Produkty])=-3)
            GEN_TimeStamp (->[Produkty]DatumVytvoření;->[Produkty]ČasVytvoření)
         End if
End case
   ` Konec metody
```

8. Přepněte se do Prostředí uživatele a přidejte nový záznam na otestování nových polí.

9. Vytvořte triggery pro Faktury a Produkty a upravte trigger pro Zákazníci tak, aby používali tuto novou metodu pro sledování datumu a času.

Kvíz

- Co je pro vás nejdůležitější věcí, kterou si chcete odnést z tohoto kurzu?

Konec (tohoto kurzu)

Příloha

A. Klávesové zkratky na Macintosh a Windows

V této tabulce jsou uvedeny klávesové zkratky mnoha akcí na platformách Macintosh a Windows.

Prostředí Uživatel

Funkce	Macintosh	Windows
Zobrazit/uschovat seznam tabulek	Stisknout Cmnd - mezerník	Stisknout Ctrl-mezerník
Uvolnit a uložit buffery	Stisknout Cmnd-W	Stisknout Ctrl-W
Přeskočit na název metody v okně Provést metodu, napsáním několika prvních písmen z názvu	Stisknout Cmnd -Shift a napsat písmeno(a)	Stisknout Ctrl- Shift a napsat písmeno(a)
Přeskočit na název metody/příkazu v okně Užít výraz, napsáním několika prvních písmen z názvu	Stisknout Cmnd -Shift a napsat písmeno(a)	Stisknout Ctrl-Shift a napsat písmeno(a)
Ukázat nápovědu/tipy	Stisknout Control-esc	Ne / automatické

Prostředí Vlastní nabídky a Uživatel

Funkce	Macintosh	Windows
Přejít do prostředí Uživatel	Stisknout Option-f v úvodní obrazovce	Vybrat Zavřít v MDI okně úvodní obrazovky, nebo poklepat na ikonu, nebo stisknout ALT+F4 (v úvodní obrazovce).
Ukázat okno ladění (debugger)	Stisknout Option a klepnout myší při provádění metody	Stisknout a klepnout myší při provádění metody
Přerušit provádění při Receive Record nebo Receive Packet	Stisknout Čárku-Shift-Option	Stisknout Ctrl-Shift-ALT
Vybrat řadu záznamů za sebou ve výstupním formuláři	Stisknout Shift a klepnout na záznamy	Stisknout Shift a klepnout na záznamy
Vybrat řadu záznamů ne za sebou ve výstupním formuláři	Stisknout Cmnd a klepnout na záznamy	Stisknout Ctrl a klepnout na záznamy
Jít na předchozí záznam (ekvivalent tlačítka Předchozí záznam)	Stisknout Cmnd a klávesu šipka vlevo	Stisknout Ctrl a klávesu šipka vlevo
Jít na další záznam (ekvivalent tlačítka Další záznam)	Stisknout Cmnd a klávesu šipka vpravo	Stisknout Ctrl a klávesu šipka vpravo
Storno transakce (ekvivalent tlačítka Storno)	Stisknout Cmnd-. (tečka)	Stisknout Ctrl-. (tečka)
Konvertovat Event proces do normálního procesu, když On Event Call zpracovává event	Stisknout Cmnd-Shift-Option-Control-Backspace	Stisknout Ctrl-Shift-ALT-s
Přesunout libovolné okno	Stisknout Cmnd-Control a potáhnout	Ne
Přidat záznam ve vložené oblasti	Stisknout Cmnd-Tab	Stisknout Ctrl + / (Tuto zkratku lze nastavit pomocí Customizer Plus)

Zobrazit sloupcovou nabídku tabulek v editorech zpráv, štítků, dotazů, třídění, užít výraz.	Podržet tlačítko myši na názvu tabulky (dolistovat na název tabulky)	Podržet tlačítko myši na názvu tabulky (dolistovat na název tabulky)
Přeskok zpět proti pořadí vstupu ve formuláři	Stisknout Shift-Tab	Stisknout Shift-Tab
Přepnout logický přepínač	Stisknout mezerník nebo (A nebo N) nebo první písmeno názvu	Stisknout mezerník nebo (A nebo N) nebo první písmeno názvu
Přepnout logické okénko zaškrtávání	Stisknout mezerník nebo (A nebo N)	Stisknout mezerník nebo (A nebo N)
Rolovat textové pole	Stisknout šipku Nahoru nebo Dolu	Stisknout šipku Nahoru nebo Dolu
Přejít na počátek nebo konec pole	Stisknout šipku Nahoru nebo Dolu	Stisknout šipku Nahoru nebo Dolu
Rolovat v seznamu	Stisknout šipku Nahoru nebo Dolu	Stisknout šipku Nahoru nebo Dolu
Zrušit vstup ze seznamu	Stisknout Tab	Stisknout Tab
Přenést aktivní okno na popředí (s více okny v jednom procesu)	Klepnout na libovolnou klávesu	Klepnout na libovolnou klávesu

Průzkumník metody a formuláře

Funkce	Macintosh	Windows
Přejít na metodu napsáním prvních písmen	Stisknout Cmnd-Shift a napsat písmeno(a)	Stisknout Ctrl-Shift a napsat písmeno(a)
Rozsířit nebo zúžit seznam ve skupině	Stisknout šipku Vpravo nebo Vlevo	Stisknout šipku Vpravo nebo Vlevo
Pohyb nahoru, dolů seznamem	Stisknout šipku Nahoru nebo Dolu	Stisknout šipku Nahoru nebo Dolu

Vstoupit do názvu k přepsání názvu	Stisknout Option a klepnou na název	Stisknout ALT a klepnou na název

Editor formulářů

Funkce	Macintosh	Windows
Vytvořit další úroveň zlomu	Stisknout Option a klepnout na označení zlomu (trujúhelník vedle Z)	Stisknout ALT a klepnout na označení zlomu (trujúhelník vedle Z)
Vytvořit dodatečnou úroveň záhlaví	Stisknout Option a klepnout na označení záhlaví (trujúhelník vedle H)	Stisknout ALT a klepnout na označení záhlaví (trujúhelník vedle H)
Zobrazit nabídku polí k vložení pole do statického textu	Stisknout Option a držet stisknuté tlačítko myši v místě vložení	Stisknout ALT a držet stisknuté tlačítko myši v místě vložení
Zobrazit hierarchický seznam tabulek a polí k vložení pole do statického textu	Stisknout Shift-Option a držet stisknuté tlačítko myši v místě vložení	Stisknout Shift-ALT a držet stisknuté tlačítko myši v místě vložení
Vytvořit pevnou (nelámatelnou) mezeru pro formáty a filtry	Stisknout Option mezerník	Ne
Otevřít metodu objektu pro objekt ve formuláři	Stisknout Option a klepnout na objekt	Stisknout ALT a klepnout na objekt
Změnit vzhled objektu ve formuláři	Stisknout Cmnd- *číslo* (0 do 5)	Stisknout Ctrl- *číslo* (0 do 5)
Automaticky vybrat a použít poslední užitý nástroj	Stisknout Cmnd a klepnout do formuláře	Stisknout Ctrl a klepnout do formuláře
Vymazat úroveň zlomů	Stisknout Cmnd a klepnout na označení úrovně	Stisknout Ctrl klepnout na označení úrovně
Vymazat úroveň záhlaví	Stisknout Cmnd a klepnout na označení úrovně	Stisknout Cmnd a klepnout na označení úrovně
Zmenšit nebo zvětšit objekt o jeden bod	Stisknout Cmnd a šipky nahoru, dolu, vlevo nebo vpravo	Stisknout Ctrl a šipky nahoru, dolu, vlevo nebo vpravo

Zmenšit nebo zvětšit objekt o jeden krok sítě	Stisknout Cmnd-Control a šipky nahoru, dolu, vlevo nebo vpravo	Klepnout na pravé tlačítko myši a změnit velikost
Zobrazit dialog pozic	Stisknout Control poklepat na objekt	Poklepat na objekt pravým tlačítkem
Vybrat objekty potažením myši	Stisknout Control a nakreslit obdélník	Stisknout pravé tlačítko a klepnout
Přesun objektů o nastavení sítě	Stisknout Control a šipky nahoru, dolu, vlevo nebo vpravo	Klepnout na pravé tlačítko myši a změnit pozici
Vybrat či odvybrat více objektů	Stisknout Shift klepat postupně na objekty	Stisknout Shift klepat postupně na objekty
Zajistit ovál jako kruh a čáru přesně horizontální, nebo vertikální	Stisknout Shift a vytvořit objekt	Stisknout Shift a vytvořit objekt
Přesunout řídící čáry pod vybranou najednou	Stisknout Shift táhnout	Stisknout Shift táhnout
Posunout vybrané objekty o bod	Stisknout šipky nahoru, dolu, vlevo nebo vpravo	Stisknout šipky nahoru, dolu, vlevo nebo vpravo

Editor nabídek

Funkce	Macintosh	Windows
Otevřít přiřazenou metodu vybrané položde nabídky	Stisknout Cmnd-P	Stisknout Ctrl-P

Editor seznamů

Funkce	Macintosh	Windows
Zobrazit a přiřadit seznamy k položce	Stisknout Option a táhnout vybranou položku na název seznamu	Stisknout ALT a táhnout vybranou položku na název seznamu
Třídit v sestupném pořadí	Stisknout Shift při výběru položky	Stisknout Shift při výběru položky

Editor metod

Funkce	Macintosh	Windows
Přeformátovat celou metodu	Stisknout Cmnd-Enter	Stisknout Ctrl-Enter z numerické klávesnice
Napsat -> znak ukazatele	Napsat – a >	Napsat – a >
Napsat <> označení meziprocesní proměnné	Stisknout Option- a 5	Napsat < a >
Napsat • a • pro adresování znaku řetězce $String•1•, atd.	Stisknout Option-, a Option-. (period)	Napsat [[pak]]
Napsat † pro čas: †12:00:00†	Stisknout Option-T	Napsat ?
Napsat datum !00/00/00!	Napsat !	Napsat !
Uzavřít všechna otevřená okna vyjma struktury	Stisknout Option a klepnout uzavírací box	Stisknout ALT a klepnout uzavírací box
Zobrazit nabídku tabulek	Podržet myš na názvu tabulek	Podržet myš na názvu tabulek
Rozšířit zúžit seznam příkazů a metod	Stisknout šipku vpravo nebo vlevo	Stisknout šipku vpravo nebo vlevo
Pohyb v seznamu příkazů a metod	Stisknout šipku nahoru nebo dolu	Stisknout šipku nahoru nebo dolu
Zkrácený způsob zadání příkazu	Napsat první písmena a @. t.j.: “Open w@” --> “OPEN WINDOW”	Napsat první písm. a @. t.j.: “Open w@” --> “OPEN WINDOW”
Otevřít použitou metodu z textu editoru metod	Vysvítit název a stisknout Cmnd-P	Vysvítit název a stisknout Ctrl-P
Otevřít použitý formulář z textu editoru metod	Vysvítit název a stisknout Cmnd-L	Vysvítit název a stisknout Ctrl-L
Zkontrolovat řádek kódu; (přemístí kurzor na konec řádku)	Stisknout Enter	Stisknout Enter na numerické klávesnici

Debugger/ Okno ladění

Funkce	Macintosh	Windows

Změnit velikost	Stisknout Option klepnout nad čáru dělení, nebo	Stisknout ALT klepnout nad čáru dělení, nebo
	Stisknout Option potáhnout čáru dělení	Stisknout ALT potáhnout čáru dělení
Zobrazit seznam polí	Stisknout Option a podržet myš	Stisknout ALT a podržet myš
Zobrazit seznam příkazů	Stisknout Cmnd a podržet myš	Stisknout Ctrl a podržet myš
Provést metody bez krokování	Stisknout Shift a klepnout na tlačítko krok	Stisknout Shift a klepnout na tlačítko krok
Oevřít další okno debuggeru pro nový proces	Stisknout Control a klepnout na tlačítko krok v řádku oevírajícím nový proces	Stisknout Ctrl+ALT a klepnout na tlačítko krok v řádku oevírajícím nový proces

Programování ve 4D

Příloha