Vážení zákazníci,

dovolujeme si Vás upozornit, že na tuto ukázku knihy se vztahují autorská práva, tzv. copyright.

To znamená, že ukázka má sloužit výhradně pro osobní potřebu potenciálního kupujícího (aby čtenář viděl, jakým způsobem je titul zpracován a mohl se také podle tohoto, jako jednoho z parametrů, rozhodnout, zda titul koupí či ne).

Z toho vyplývá, že není dovoleno tuto ukázku jakýmkoliv způsobem dále šířit, veřejně či neveřejně např. umisťováním na datová média, na jiné internetové stránky (ani prostřednictvím odkazů) apod.

redakce nakladatelství BEN – technická literatura
redakce@ben.cz

8 CYKLUS

8.1 FOR CYKLUS

V projektech, které budete zpracovávat, se často setkáte se situací, kdy je třeba nějakou akci provádět vícekrát za sebou. Je samozřejmě možné zapsat např. volání nějaké procedury nebo funkce opakovaně za sebe. Takový postup však můžete použít pouze v případě, když víte předem, kolikrát se má nějaký příkaz provést. Někdy ale tuto informaci nemáte k dispozici, počet opakování může záležet třeba na hodnotě náhodného čísla. Pascal nabízí řešení v podobě cyklu. Cyklus je příkaz, který umožňuje opakované provádění připravených akcí. I v případě, že znáte počet opakování, je výhodnější použít cyklus, než opakovaně zapisovat stejný kód. V takových případech se obvykle využívá nejjednodušší ze třech typů cyklu, které Pascal nabízí, cyklus for. Ještě dříve, než si ukážeme jeho použití, seznámíme se z komponentou Memo, kterou budeme často používat.

Pokud jsme potřebovali vypsat na formulář nějaký text, používali jsme komponentu label. Pokud bychom chtěli vypsat větší množství textu, komponenta label k tomu vhodná není. Chceme-li vypisovat více textu, je vhodné použít komponentu Memo (textové okno). Zástupce komponenty Memo naleznete na liště Standard pátého zleva. V této komponentě se typicky zobrazuje větší množství textu.

Obsah komponenty Memo určuje vlastnost Lines. Tato vlastnost je objekt typu TStrings a má svoje vlastnosti a metody. Pokud chceme při návrhu formuláře zadat do mema text, v Object Inspectoru kliknutím na tlačítko s třemi tečkami u vlastnosti Lines otevřeme String list editor, kde můžeme text zadat. Text do mema se obvykle přidává po řádcích. Využívá se k tomu metoda Memo.Lines.Add, která má jeden parametr typu textový řetězec. Hodnota parametru se přidává jako nový řádek na konec textu v memu. Pokud chceme pracovat s celým obsahem mema, můžeme využít vlastnost Text, jejíž hodnotou je veškerý text zapsaný v memu. Novou hodnotu do vlastnosti Text můžeme i přiřadit a přepsat tak text, který je zapsán v memu.

Zobrazení textu v komponentě memo je ovlivněno nastavením několika dalších vlastností. Vlastnost WordWrap, jejíž standardní hodnota je True, určuje, zda se budou zalamovat řádky podle šířky mema. Pokud je WordWrap True, delší řádky jsou rozloženy do více řádků. Pohybovat kurzorem v textu v memu můžeme pomocí kurzorových kláves. Pokud je v memu více textu, než lze zobrazit najednou, text se posunuje podle pozice kurzoru. K memu však můžeme přidat posuvné lišty, svislou a vodorovnou. Pokud text přesáhne velikost mema, pomocí lišt můžeme prohlížet i další nezobrazenou část textu. Použití posuvných lišt určuje hodnota vlastnosti ScrollBars. Standardní hodnota je ssNone, žádná lišta není použita.

Svislá lišta se zobrazí, pokud je hodnota ssVertical, vodorovná při nastavení ssHorizontal. Zobrazení obou lišt zajistí volba ssBoth. Často používané nastavení vlastností u mema je WordWrap na True a ScrollBars na ssVertical. Délka řádek se přizpůsobí šířce mema a celý text můžeme prohlížet pomocí svislé lišty. Pokud chceme zakázat uživateli měnit text v memu, nastavíme vlastnost ReadOnly na True. Typickou událostí mema je OnChange. Události mema se však příliš nevyužívají, memo nejčastěji slouží pro výpis většího množství textu.

Příklad 8.1

Umístěte na formulář edit, button a memo. Po stisku tlačítka přidejte do mema řádek s textem zadaným v editu.

Při návrhu formuláře prostřednictvím String list editoru vymažeme text v memu (standardně jméno komponenty), tedy změníme hodnotu vlastnosti Lines. Zapíšeme reakci na událost OnClick tlačítka. Zavoláme metodu Memo1.Lines.Add, jako parametr použijeme hodnotu vlastnosti Text editu.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
Memo1.Lines.Add(Edit1.Text);
end;
```

Zobrazíme vertikální posuvnou lištu, abychom mohli celý obsah mema jednoduše prohlížet. Opakovaným klikáním se přidávají stále další řádky. Zadání si ještě rozšíříme.

Počet řádek v memu udává hodnota vlastnosti Memo1.Lines.Count. Tuto informaci zobrazíme navíc v labelu. Hodnotu musíme aktualizovat po každé změně obsahu mema, tedy jako reakci na událost OnChange. Počet řádek v memu se může měnit i přepsáním zobrazeného textu v memu. Pokud zadáte v editu text delší, než je šířka mema, počet řádek v memu se zvyšuje více než o jednu.

Celý obsah mema můžeme vymazat. Využijeme k tomu metodu Clear komponenty memo. Provedení této metody však nezpůsobí vyvolání události OnChange a proto musíme současně nastavit hodnotu nula do labelu.

Příklad 8.2

Na formulář umístěte button a memo. Po stisku tlačítka přidejte do mema 10 řádků s textem „ahoj“.

V reakci na událost OnClick tlačítka můžeme volat desetkrát metodu Lines.Add. Pokud by však v zadání bylo vypsání 100 řádek, asi by vás nebavilo řádky ani kopírovat, natož psát. Pokud se má opakovat několikrát po sobě jeden příkaz, Pascal nabízí použití příkazu cyklu:

```
procedure TForm1.Button1Click(Sender: TObject);
var i:integer;
```

```
begin
for i:=1 to 10 do
  Memo1.Lines.Add('ahoj');
end;
```

Tento kód řeší velmi rychle a jednoduše zadanou úlohu.

Použili jsme příkaz cyklu. V Pascalu existuje více typů cyklu, cyklu, který jsme použili, se říká **for** cyklus. Jeho syntaxe je následující:

```
for proměnná:=dolní mez to horní mez do příkaz;
```

For cyklus můžeme použít vždy, když před jeho začátkem víme, kolikrát se má provést tělo cyklu. Počet opakování nemusíme znát přesně už při zápisu kódu, můžeme využít např. hodnotu nějaké proměnné při běhu aplikace. Za klíčové slovo for píšeme proměnnou, které se říká řídicí proměnná cyklu. Musí být ordinálního typu (podobně jako výraz v case příkazu), ze známých typů můžeme tedy použít integer (případně boolean, ale tělo cyklu s mezemi False a True by se provedlo pouze dvakrát a použití takového cyklu je dost neobvyklé). Následuje přiřazovací příkaz, výraz určující dolní mez, klíčové slovo **to** a výraz určující horní mez. Oba výrazy musí být stejného typu[1] jako řídicí proměnná. Za klíčovým slovem **do** je jeden příkaz (může být i složený), kterému říkáme tělo cyklu. Řídicí proměnná, která musí být lokální, mění postupně hodnoty v intervalu od dolní meze k horní mezi po jedničce[2]. Pro každou hodnotu řídicí proměnné se provádí příkaz za do.

V našem příkladu má proměnná i postupně hodnoty 1, 2..10, tělo se tedy provede desetkrát. Obecně se tělo provádí H–D+1 krát, kde D je dolní mez, H horní mez. Pokud je D>H, tělo cyklu se neprovede ani jednou. Hodnotu řídicí proměnné můžeme v těle cyklu využít, nesmíme ji však měnit.

Příklad 8.3

Vypište do mema 10 náhodných čísel z intervalu 1..100. U každého čísla přidejte ještě jeho pořadí.

Výpis čísel do mema provedeme v reakci na stisknutí tlačítka. Musíme sestavit textový řetězec, který přidáme do mema metodou Lines.Add. V řádce bude pořadové číslo a vygenerované náhodné číslo. Pořadové číslo bude určovat řídicí proměnná cyklu:

```
procedure TForm1.Button1Click(Sender: TObject);
var i,n:integer;
```

[1] Nebo typu kompatibilního. Existuje např. více celočíselných typů lišících se rozsahem přípustných hodnot. Pokud je hodnota z vhodného rozsahu, je možné použít kompatibilní typ.

[2] Toto platí pro řídicí proměnnou typu celé číslo. U dalších ordinálních typů, se kterými se seznámíme později, se mění řídicí proměnná na následující hodnotu. Ve verzi cyklu s **downto** (viz dále) na hodnotu předchozí.

```
    s:string;
begin
Memo1.Clear;
for i:=1 to 10 do
  begin
  n:=Random(100)+1;
  s:=IntToStr(i)+' ... '+IntToStr(n);
  Memo1.Lines.Add(s);
  end;
end;
```

Pro přehlednost jsou použity další 2 proměnné. Příkaz za **do** je složený, v těle cyklu využíváme hodnotu řídicí proměnné. V proceduře FormCreate je inicializován generátor náhodných čísel.

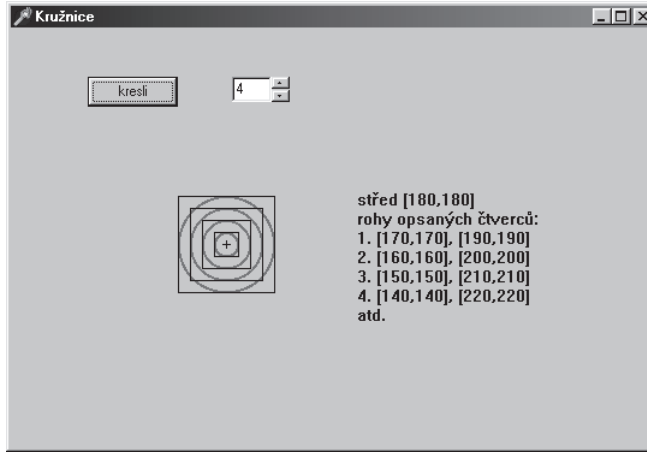
Zatím jsme použili cykly, které mají jako obě meze čísla a dolní mez je rovna jedné. Jako meze však můžu použít libovolné výrazy stejného typu jako je řídicí proměnná. Řídicí proměnná se vždy mění po jedničkách. Existuje ještě jedna verze for cyklu:

```
for proměnná:=horní mez downto dolní mez do příkaz;
```

V tomto případě prochází řídicí proměnná hodnoty od horní meze k dolní a mění se po –1. Jinak je použití této verze stejné.

Příklad 8.4

Zadejte v editu číslo z intervalu 0..10. Na formulář nakreslete tolik soustředných kružnic.



Obr. 38 Soustředné kružnice

Soustředné kružnice mají společný střed. V Delphi však při kreslení kružnice nepracujeme se souřadnicí středu, ale se souřadnicemi levého horního a pravého dolního vrcholu opsaného čtverce. Musíme tedy zjistit, jak nastavovat tyto hodnoty. Střed kružnic určíme na souřadnici [180, 180]. Tento bod musí být také středem opsaných čtverců *obr. 38*.

Podle pořadí čtverce jsou souřadnice vrcholů

[180-p*10,180-p*10], [180+p*10,180+p*10].

Hodnotu 10 zvolíme jako poloměr nejmenší kružnice. Každá další kružnice má poloměr o 10 větší. Pokud *p* bude postupně procházet hodnoty podle počtu kružnic, uvedené výrazy můžeme použit jako parametr metody Canvas.Ellipse. Nakreslení kružnic tedy zajistí tento cyklus:

```
for p:=1 to n do
   Canvas.Ellipse(180-p*10,180-p*10,180+p*10,180+p*10);
```

Proměnná *n* určuje, kolik se má kreslit kružnic. Pokud má hodnotu 0, tělo cyklu se neprovede ani jednou. Vykreslení kružnic zapíšeme jako reakci na stisk tlačítka. Tlačítko je možné stisknout opakovaně, plochu pro kreslení musíme tedy „vyčistit“. Nakreslíme obdélník barvou formuláře ve velikosti největší kružnice. Zatím jsme kreslili kruhy. Pokud chceme kreslit pouze okraj, nestačí nastavit barvu výplně na barvu formuláře. Větší kružnice by smazaly kružnice menší. Bylo by možné otočit for cyklus (`n downto 1`) a kreslit kružnice od největší k nejmenší. Můžeme však využít vlastnost Canvasu Brush.Style. Tato vlastnost určuje typ výplně geometrického obrazce. Standardně má hodnotu bsSolid, výplň se vykresluje. Pokud přiřadíme hodnotu bsClear, výplň se nenakreslí a nakreslíme pouze okraj obrazce. To přesně potřebujeme, tak nakreslíme kružnice. Styl výplně musíme měnit, když mažeme formulář, stejně tak musíme měnit barvu pera (mohli bychom měnit i hodnotu vlastnosti Pen.Style na psClear, obrys obdélníku by se nekreslil).

```
procedure TForm1.Button1Click(Sender: TObject);
var p,n:integer;
    i:integer;
begin
Canvas.Brush.Style:=bsSolid;
Canvas.Brush.Color:=Form1.Color;
Canvas.Pen.Color:=Form1.Color;
Canvas.Rectangle(80,80,280,280);
Canvas.Brush.Style:=bsClear;
Canvas.Pen.Color:=clGreen;
n:=UpDown1.Position;
for p:=n downto 1 do
  begin
   Canvas.Ellipse(180-p*10,180-p*10,180+p*10,180+p*10);
  for i:=1 to 10000000 do;
```

```
  end;
end;
```

Tělo procedury je trošku „zkomplikováno“. For cyklus je otočený, kružnice se tak vykreslují v pořadí od největší do nejmenší. Do těla cyklu s řídicí proměnnou *p* je přidán ještě jeden cyklus, který nic nedělá, za do je hned středník, to znamená prázdný příkaz. Ovšem řídicí proměnná *i* musí projít všechny hodnoty od dolní do (velké) horní meze. Aplikace se tím zdrží a můžeme sledovat postupné vykreslování kružnic. Můžete zkusit opět použít cyklus s to, pořadí vykreslování kružnic by se otočilo. Interval mezi vykreslením jednotlivých kružnic záleží na rychlosti počítače.

Toto ovšem není nejvhodnější řešení. Zkuste aplikaci přerušit při vykreslování. Nepodaří se to, aplikace se ukončí až po nakreslení všech kružnic. Provádí se totiž stále reakce na událost OnClick tlačítka a dokud provedení této reakce neskončí, aplikace nemůže reagovat na žádné další události, jako je třeba ukončení aplikace. Reakce na události, které v průběhu provádění „zdlouhavé“ reakce nastaly, se začnou postupně provádět až po ukončení probíhající reakce. Nikdy nemůžou probíhat současně reakce na více událostí[1]. Reakce na událost se vždy dokončí a pak se může provést další. Pokud např. stisknete Alt+F4 (uzavření okna) při vykreslování kružnic, aplikace skončí hned po tom, co se kružnice vykreslí.

Správné řešení „pozdrženého“ vykreslování kružnic by bylo pomocí časovače. Nešel by ale použít cyklus, muselo by se pracovat s globální proměnnou, která by určovala, jak velká kružnice se má kreslit a podle její hodnoty by se také časovač nakonec zastavil.

Příklad 8.5

Nakreslete na formulář šachovnici.

Šachovnici budeme kreslit na Canvas formuláře. Přizpůsobíme velikost plochy nastavením hodnot vlastností ClientWidth a ClientHeight na 320. Jednotlivá pole šachovnice budou mít rozměr 40 × 40. Šachovnici budeme kreslit po řadách. V každé řadě se pravidelně střídají bílá a černá pole. Proměnná *Bila* typu Boolean má hodnotu True, když kreslíme bílé pole, jinak False. Podle této proměnné nastavujeme vlastnosti Brush.Color a Pen.Color. V řadě je 8 polí, řad je také osm. Použijeme tzv. vnořené cykly. V těle jednoho příkazu cyklu se použije další cyklus. V tomto příkladu je cyklus s řídicí proměnnou *i* vnější, určuje, která řada se vykresluje. Cyklus s řídicí proměnnou *j* je vnitřní, prochází pole v jednotlivých řadách. Hodnoty proměnných *i*, *j* určují souřadnice nakresleného čtverce. V těle vnějšího cyklu je vnitřní cyklus, který se tedy provede osmkrát. Tělo vnitřního cyklu se provede osm krát osmkrát, dohromady tedy 64×. Nakreslí se tedy 64 polí

[1] V Delphi existují postupy, jak toto změnit. Je to však pokročilé téma, s kterým se zde nebudeme zabývat.

šachovnice. „Pomaleji“ se mění hodnota proměnné *i*, ta určuje pořadí řádku, tedy souřadnici „na výšku“ (y-ovou), proměnná *j* pro každou hodnotu proměnné *i* projde hodnoty 1..8, určuje souřadnici ve vodorovném směru (x-ovou). Vykreslení šachovnice provedeme jako reakci na kliknutí na formulář, tedy zapíšeme do procedury FormClick. Kdykoli uživatel klikne na plochu formuláře, kresba se obnoví.

```
procedure TForm1.FormClick(Sender: TObject);
var Bila:boolean;
    i,j:integer;
begin
Bila:=False;
for i:=1 to 8 do
  begin
  for j:=1 to 8 do
    begin
    if Bila then
      begin
       Canvas.Brush.Color:=clWhite;
       Canvas.Pen.Color:=clWhite;
       end
    else
      begin
       Canvas.Brush.Color:=clBlack;
       Canvas.Pen.Color:=clBlack;
       end;
     Canvas.Rectangle(j*40-40,i*40-40,j*40,i*40);
    Bila:=not Bila
    end;
  Bila:=not Bila; //řady končí a začínají stejnou barvou
  end;
end;
```

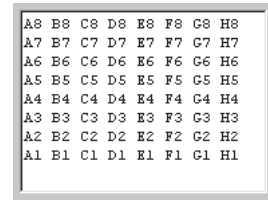
Pokud byste prohodili proměnné *i*, *j* ve volání metody Rectangle, šachovnice by se vykreslila po sloupcích. Zpomalit vykreslování můžete opět cyklem s prázdným tělem a velkým rozdílem mezí v těle vnitřního cyklu. Není to však vhodné, důvody jsou popsány v minulém příkladu.

Příklad 8.6

Vypište do mema souřadnice na šachovnici.

Souřadnice na šachovnici se určují pomocí čísla řádku a písmena sloupce. Jejich popis znázorňuje *obr. 39*, výsledek je vidět na *obr. 40*.

Obr. 39 Souřadnice na šachovnici



Obr. 40 Textový popis

V memu jsou tedy v jednotlivých řádcích zapsány souřadnice A8..H8, .., A1..H1. Do mema budeme přidávat pomocí metody Lines.Add, jejímž parametrem je textový řetězec, nový řádek. K sestavování souřadnic použijeme opět dva vnořené cykly. Vnější cyklus určuje číslo řádku a projde tedy hodnoty od 8 do 1. Vnitřní cyklus v každém řádku určí písmeno pro sloupec. Potřebovali bychom tedy cyklus, který projde písmena od A do H. Takový cyklus můžeme zapsat, použijeme-li jako řídicí proměnnou proměnnou typu **char** (znak, character). Typ char je ordinální a proměnné tohoto typu tedy můžou být řídicí proměnnou for cyklu. Hodnotou proměnných typu char je vždy jeden znak. Na proměnné typu char se dá tedy pohlížet jako na textové řetězce o délce jedna. Hodnoty proměnných typu char díky tomu můžeme přiřazovat do textových řetězců. Obráceně to ale nejde.

Pořadí znaků, které se v počítači používají, je dáno tzv. ASCII tabulkou. Řídicí proměnná typu char tak ve for cyklu prochází znaky od dolní k horní mezi (resp. obráceně při downto) podle pořadí znaků v této tabulce. Pokud tedy použijeme for cyklus s mezemi „A“ a „H“, řídicí proměnná projde písmena v tomto intervalu. V těle tohoto cyklu sestavíme textový řetězec, který pak vypíšeme do mema.

```
procedure TForm1.Button1Click(Sender: TObject);
var s:char;
    r:integer;
    t:string;
begin
for r:=8 downto 1 do
  begin
  t:='';
  for s:='A' to 'H' do
    t:=t+s+IntToStr(r)+' ';
  Memo1.Lines.Add(t);
  end;
end;
```