



Ďalšie vzdelávanie učiteľov
základných škôl a stredných škôl
v predmete *informatika*



ŠTÁTNY PEDAGOGICKÝ ÚSTAV
NATIONAL INSTITUTE FOR EDUCATION

Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Programovanie 1

Predmet: Programovanie

Línia: Vlastný odborový kontext informatiky a informatickej výchovy



EURÓPSKA ÚNIA



Európsky sociálny fond



Európska únia
Európsky sociálny fond

Moderné vzdelávanie pre vedomostnú spoločnosť/Projekt je spolufinancovaný zo zdrojov ES

Programovanie 1

Identifikácia modulu

Aktivita projektu: 1.2 Vzdelávanie nekvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ

Línia aktivity: Vlastný odborový kontext informatiky a informatickej výchovy

Predmet: Programovanie 1-9

Garant predmetu:

RNDr. Andrej Blaho
KAI FMFI UK, Bratislava
andrej.blaho@gmail.com

Autori:

RNDr. Andrej Blaho
KAI FMFI UK, Bratislava

RNDr. Ľubomír Salanci, PhD.
KZVI FMFI UK, Bratislava

Rukopis odovzdaný:

1. jún 2009

Zaradenie modulu

Moduly Programovanie 1 až Programovanie 9 nadväzujú na modul Programovanie 0. Všetky spolu vytvárajú ucelený kurz programovania, ktorý pokrýva stredoškolský obsah programovania aj s množstvom metodicky vhodných úloh, problémov a projektov. Všetkých 9 modulov je v prvých dvoch semestroch štúdia, nakoľko na nich nadväzujú ďalšie predmety z informatickej línie ale aj z didaktiky programovania.

Prvý semester vzdelávania:



Abstrakt modulu

Vytváranie prvých jednoduchých aplikácií v programovacom prostredí TurboDelphi, resp. Lazarus s použitím elementárnych komponentov a príkazov grafiky.

Programovanie 1.....	1
Identifikácia modulu	1
Zaradenie modulu	1
Abstrakt modulu	1
Obsah	2
Úvod	3
Softvérové a hardvérové požiadavky a odporúčania	3
Cieľ modulu.....	4
Vstupné vedomosti	4
Požadované prerekvizity	4
Predpokladané vstupné vedomosti, skúsenosti a zručnosti	4
1. Prvý program	5
1. Programátorské vývojové prostredie.....	5
2. Prvý program	6
3. Základné komponenty a ich vlastnosti	8
4. Udalosť kliknutie na tlačidlo	11
5. Nastavovanie vlastností komponentov počas behu programu	16
2. Úvod do grafiky.....	20
1. Grafická plocha a súradnicová sústava	20
2. Príkaz na nakreslenie obdĺžnika, vymazanie plochy	21
3. Pero, štetec a ich atribúty	24
4. Kreslenie čiar	28
5. Kreslenie elipsy a kruhu.....	31
6. Písanie textov do grafickej plochy	33
Čo sme sa naučili v tomto module	36
Preverenie výstupných vedomostí	36
Literatúra a použité zdroje	37
Základné materiály.....	37
Internetové zdroje pre prostredie Delphi	37
Internetové zdroje pre prostredie Lazarus.....	37

Úvod

TurboDelphi je pomenovanie programátorského prostredia, v ktorom sa programuje v jazyku **Objektový Pascal**. Existuje aj veľmi podobné prostredie s názvom **Lazarus**, ktoré poskytuje skoro identický **Objektový Pascal**. Toto prostredie je vyvíjané ako otvorený kód. Obe tieto prostredia sú voľne stiahnuteľné a každé má svoje výhody aj nevýhody.

Samotný jazyk pascal je komunitou informatikov na celom svete považovaný za jazyk, ktorý je určený nielen na učenie programovania, ale aj na rozvoj algoritmického myslenia.

Pascal, podobne ako jazyk C, vznikol začiatkom 70-tych rokov. Oba tieto jazyky boli postavené na základe štruktúrovaných konštrukcií jazyka Algol a obidva sa inšpirovali vtedajšími jazykmi ako Fortran, Cobol a PL/1.

Pascal sa začal šíriť vďaka univerzitám na celom svete. Na rozdiel od neho jazyk C vznikol pre systémových programátorov, teda pre už veľmi skúsených programátorov. Postupom času (v polovici 80-tych rokov) sa z Pascalu vyvinul moderný Objektový Pascal a C sa zmodernizovalo na objektovo orientované C++.

Programátorské prostredie (IDE - integrated development environment) je softvérový balík umožňujúci programátorovi:

- programy navrhovať, písať, upravovať, vyvíjať
- kompilovať
- testovať, ladiť

Súčasnú modernú programátorskú prostredia, ktoré umožňujú vyvíjať nielen aplikácie pre grafické rozhranie (napr. Windows), ale aj aplikácie pre web - sú založené na vizuálnom princípe: všetko, čo bude mať v bežiackej aplikácii vizuálne znázornenie, sa už počas návrhu bude dať vizuálne poskladať z nejakých predpripravených častí. Programátor potom veľmi často "iba" doprogramováva správanie týchto komponentov v rôznych situáciách a grafická nadstavba mu pritom zabezpečí celkové správne fungovanie aplikácie..

Tento predmet je úvodom do Objektového pascalu a do prostredia TurboDelphi, resp. Lazarus. V 9. moduloch sa postupne zoznámite so základnými pojmami programovania, jednoduchej tvorby algoritmov a tiež so spôsobmi ladenia programov. Keďže TurboDelphi aj Lazarus sú určené hlavne na vývoj aplikácií pod Windows, aj materiál v tomto učebnom texte je zameraný na tento operačný systém (hoci Lazarus je prostredie, v ktorom sa dá pracovať aj Linuxových operačných systémoch). Všetky nové pojmy vysvetľujeme podľa možnosti na grafických prvkoch, aby boli čo najnázornejšie a čo najjednoduchšie na pochopenie.

Softvérové a hardvérové požiadavky a odporúčania

Nainštalované jedno z Freeware programovacích prostredí

- Turbo Delphi - verzia min. z 2006
 - <http://downloads.embarcadero.com/free/turbo>
- Lazarus - verzia min. z 2009
 - <http://www.lazarus.freepascal.org/>

Odporúčame, aby sa v prostredí nastavilo:

- zapnutá kontrola pretečenia indexov (Range Checking)
- automatické ukladanie projektu pri spúšťaní projektu (Run)

Cieľ modulu

Modul obsahuje úvodné pojmy z programovania v Objektovom Pascale. Predstaví programové prostredie TurboDelphi, resp. Lazarus. Ďalej elementárne komponenty a udalosti. Cieľom bude vytvoriť a odladiť prvé programy.

Druhý blok modulu sa venuje úvodu do grafiky. Vysvetlí grafický komponent, mechanizmus súradnej sústavy a elementárne grafické príkazy.

Vstupné vedomosti

Požadované prerekvizity

Modul Programovanie 0 - úvod do korytnačej grafiky.

Predpokladané vstupné vedomosti, skúsenosti a zručnosti

Účastník vzdelávania

- ovláda základné príkazy korytnačej grafiky,
- dokáže rozoznať v obrázku opakujúce sa časti, vie používať cyklus s pevným počtom opakovaní na vykresľovanie rôznych obrázkov pomocou korytnačky,
- vie vytvárať vlastné príkazy bez parametrov aj s parametrami,
- vie čítať jednoduché postupnosti príkazov v jazyku Logo a analyzovať ich,
- dokáže vytvoriť jednoduchý projekt využívajúc tlačidlá, udalosti a procesy,
- rozumie pojmom príkaz, postupnosť príkazov, procedúra, parameter, udalosť.

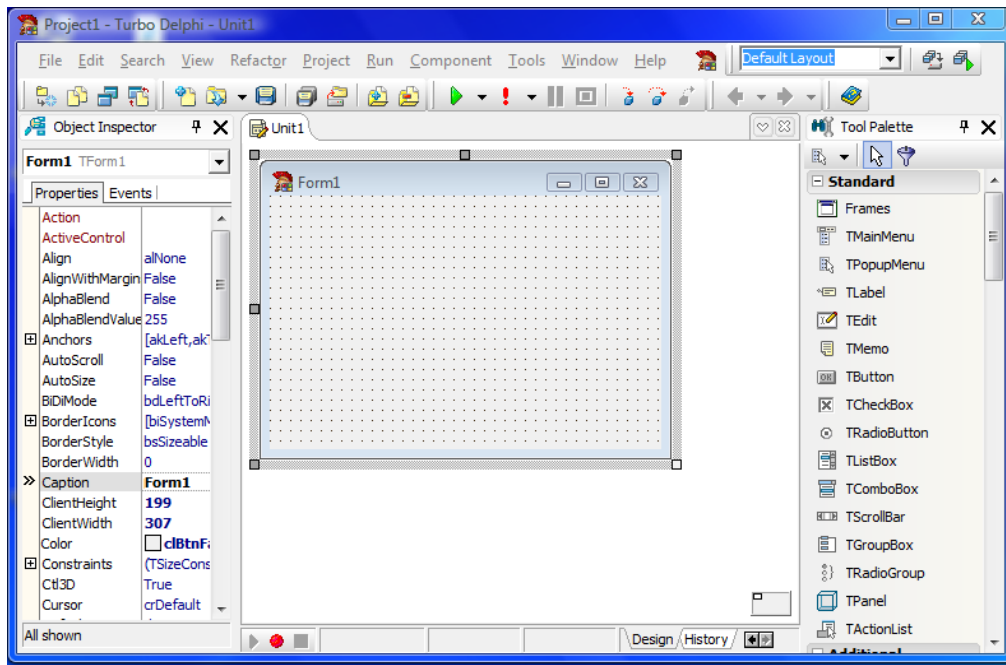
1. Prvý program

Na vývoj programov (aplikácií) väčšinou pre niektorý konkrétny operačný systém slúžia rôzne programátorské prostredia (napr. TurboDelphi a Lazarus). My sa budeme zaoberať vývojom programov len pre operačný systém Windows. Obe uvažované prostredia sú voľne stiahnuteľné z internetu a bez obmedzení sa môžu nainštalovať v škole ale aj na domácich počítačoch. TurboDelphi je freeware verzia profesionálneho prostredia od firmy Embarcadero (pôvodne Borland). Lazarus je open source projekt, ktorý je tiež voľne stiahnuteľný.

Materiál je logicky rozdelený na dve štvorhodinové tematické jednotky

1. Programátorské vývojové prostredie

Po načatovaní TurboDelphi sa na obrazovke objaví:



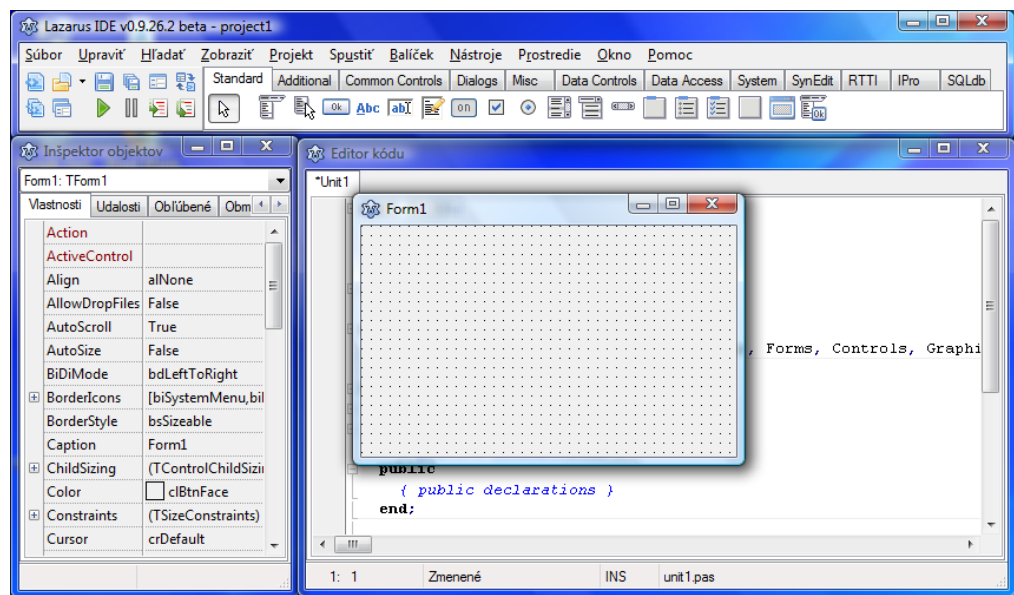
Obrázok 1: Prostredie TurboDelphi

Okrem hornej ponuky a pásu s nástrojovými tlačidlami sa skladá z týchto troch častí:

- vľavo je inšpektor objektov (Object Inspector)
- v strede je editovacie okno spolu s formulárom
- vpravo je paleta komponentov (Tool Palette)

Prostredie Lazarus po načatovaní vyzerá veľmi podobne ako TurboDelphi. Najväčší viditeľný rozdiel je v rozdelení častí prostredia:

- vľavo je inšpektor objektov (Object Inspector)
- vpravo je editovacie okno spolu s formulárom
- nad editovacím oknom je paleta komponentov (Tool Palette)



Obrázok 2: Prostredie Lazarus

Editovacie okno slúži na zápis algoritmov - programovať budeme v programovacom jazyku objektový pascal. Programovanie vlastne bude znamenať to, že budeme popisovať, ako sa bude program správať v rôznych situáciách.

Formulár (ak nie je momentálne vidieť, stlačte kláves F12) slúži na vizualizáciu nášho budúceho programu. Programy vo Windows majú väčšinou tvar obdĺžnikového okna a v ňom obsahujú rôzne väčšinou štandardné prvky ako texty, tlačidlá, posúvače, obrázky a pod. My budeme počas prípravy nášho budúceho projektu práve do tohto formuláru (budúceho okna aplikácie) rozmiestňovať a aj upravovať najrôznejšie súčiastky (komponenty).

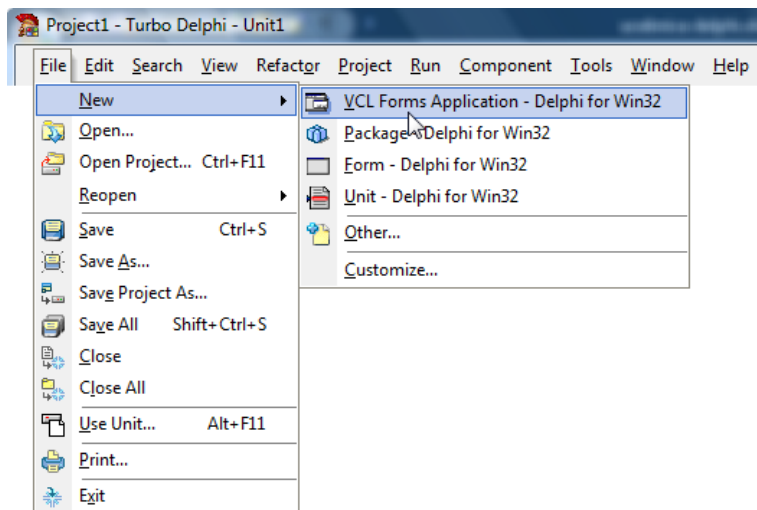
Paleta komponentov obsahuje predpripravené "súčiastky", z ktorých sa bude skladať okno našej vytváranej aplikácie. Táto paleta sa skladá z viacerých záložiek, ale my sa postupne zoznámime len s ich veľmi malou množinou.

Inšpektor objektov nám pomôže nastavovať rôzne parametre pre súčiastky (komponenty), ktoré sme už umiestnili v našom formulári.

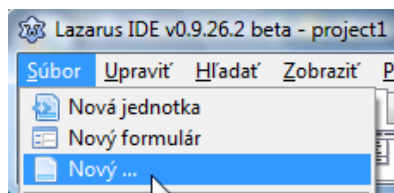
2. Prvý program

Skôr ako sa naučíme tvoriť nové programy v Delphi (hovoríme im aj projekty alebo aplikácie), najprv si zautomatizujeme jednoduchý postup, ktorý nám na začiatku môže pomôcť pri vytváraní nových projektov a zrejme aj vyvarovať sa niektorých začiatočných chýb.

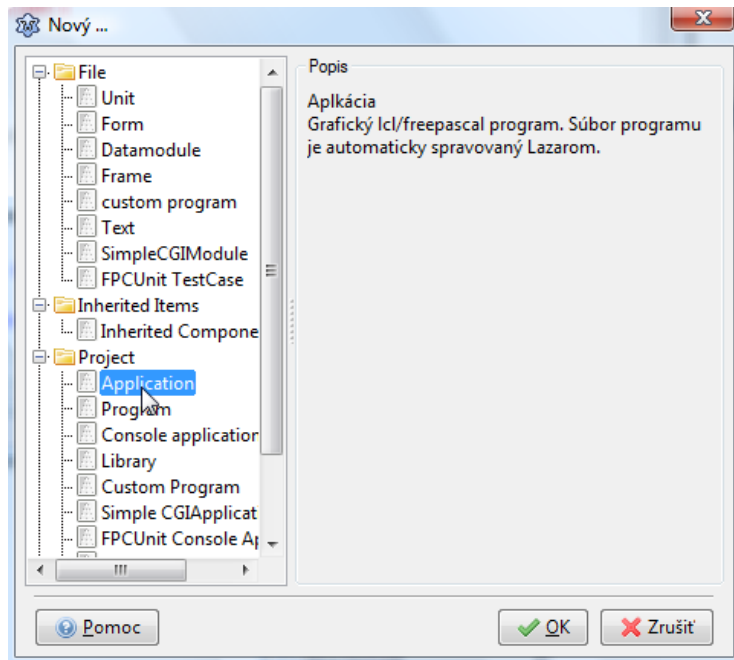
1. V TurboDelphi v hlavnej ponuke (menu) **File** zvolíme **New** a potom **VCL Application**



V Lazarus to bude podobne: v ponuke **Súbor** vyberieme **Nový**



a potom zvolíme **Application**:

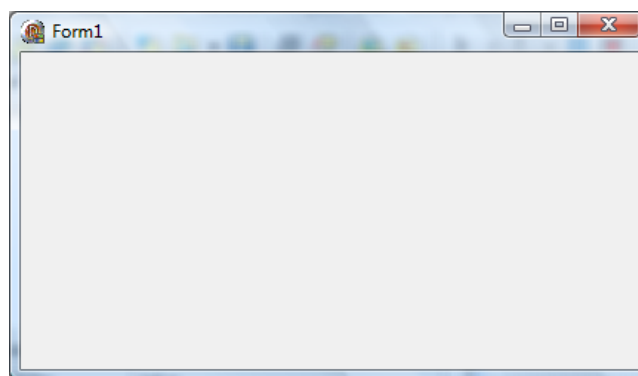


2. Programátorské prostredie je teraz pripravené na to, aby sme mohli začať tvoriť náš nový program - editovacie okno obsahuje len zopár základných predpísaných konštrukcií a formulár je prázdny.
3. V tomto momente (práve pred samotným začiatkom tvorby programu) zvykneme takýto zatiaľ "prázdny" program uložiť - samozrejme, že to môžeme aj neskôr, ale začiatok pri tom môže urobiť množstvo chýb. Takže, z hlavnej ponuky **File** vyberieme **Save Project As...** , resp. v ponuke **Súbor** zvolíme **Uložiť všetko**

...

- Zvolíme priečinok, do ktorého budeme projekt ukladať - odporúčame ukladať každý projekt do nového priečinku. Každý projekt, kým ho ešte programujeme, sa na disku skladá z viacerých súborov a tieto by mali byť spolu v jednom priečinku a nemali by sa radšej prekryvať s inými projektmi.
 - Pri ukladaní projektu sa TurboDelphi aj Lazarus opýtajú na uloženie dvoch súborov: Unit1.pas - to je samotný program aj s formulárom a Project1.bdsproj (project1.lpi) - to je projektový súbor, ktorý sa vytvára automaticky a obsahuje dôležité informácie o celkovej štruktúre nášho projektu. Tieto dva súbory môžete premenovávať (zatiaľ to neodporúčame), ale dôležité je, aby boli spolu v rovnakom priečinku.
4. Teraz už predpokladáme, že máme projekt úspešne uložený a môžeme ho spustiť:
- Hoci sme ešte nič neprogramovali, prostredie nám s novou aplikáciou všetko pripraví tak, aby sme už mali funkčné jedno Windows okno.
 - Stlačíme kláves **F9** (alebo tlačidlo na ovládacom paneli so zeleným trojuholníkom) a ak je program bez chýb, tak sa spustí: objavilo sa prázdne šedé okno s titulovým modrým pásom, s textom Form1 a s malými systémovými tlačidlami na minimalizovanie, maximalizovanie a ukončenie aplikácie:

Prekladač vytvoril v našom priečinku súbor **Project1.exe**, ktorý je už spustiteľnou aplikáciou aj bez samotného programátorského prostredia.

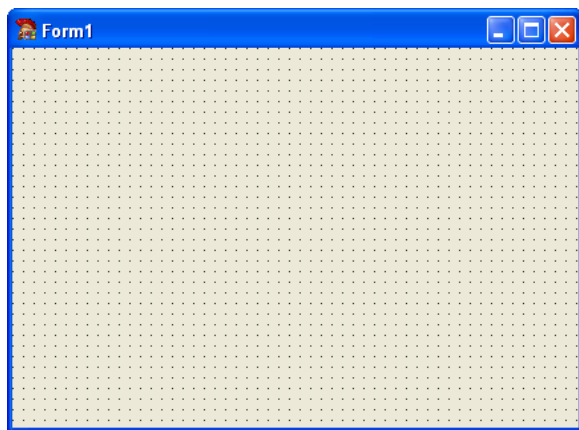


V Linuxe Lazarus vytvorí spustiteľný binárny súbor **project1** (bez prípony).

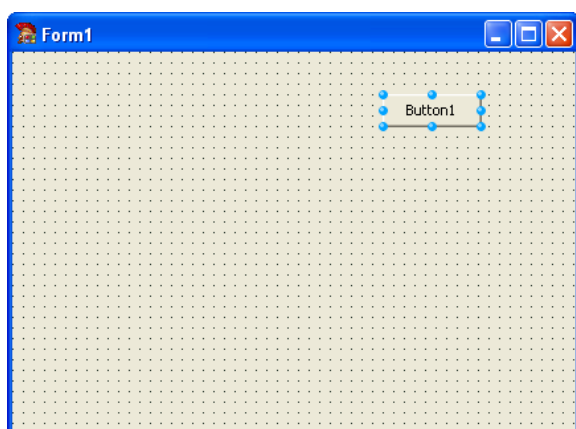
- Spustený program môžeme na pracovnej ploche Windows posúvať, meniť mu veľkosť, prípadne sa prepnúť do inej bežiackej aplikácie, napr. prostredie TurboDelphi (to že naša aplikácia práve beží, vidíme v TurboDelphi aj v Lazarus, napr. tak, že tlačidlo so zeleným trojuholníkom je zablokované - zašedené).
 - Bežiaci program ukončíme buď tlačidlom **Close** (v pravom hornom rohu) alebo klávesmi **Alt+F4**.
5. Po zastavení nášho bežiaceho prvého programu, ho môžeme teraz začať naozaj programovať.

3. Základné komponenty a ich vlastnosti

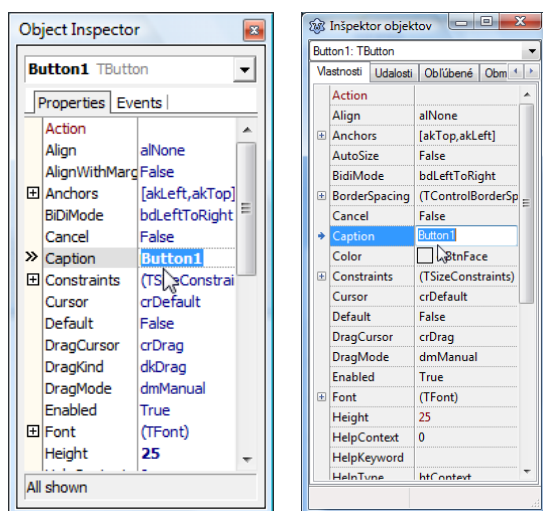
Väčšina Windows aplikácií má v svojom hlavnom okne najrôznejšie súčiastky. My sa teraz naučíme vytvárať toto okno aj s týmito Windows prvkami. Aby sme mohli ukladať komponenty do formulára, tento musí byť vpredu - ešte pred editovacím oknom. Ak ešte nie je, stlačíme kláves **F12** a objaví sa formulárové okienko (spoznáme ho hneď, je celé posiate bodkami):



Teraz môžeme upraviť jeho polohu a rozmer a pripraviť ho na ukladanie súčastok z palety komponentov. Zo štandardnej záložky zvolíme súčastku tlačidlo (TButton). Teraz klikneme niekde do formuláru a objaví sa tu tlačidlo s popisom Button1:



Toto tlačidlo môžete presunúť na ľubovoľnú pozíciu, prípadne mu zmeniť veľkosť ťahaním za modré guľôčky (resp. čierne štvorčeky v Lazarus). Popis na tlačidle zmeníme veľmi jednoducho: všimnite si, že keď je naše nové tlačidlo označené (má okolo seba modré alebo čierne značky), v **Inšpektore objektov** (Object Inspector) sa zobrazujú nejaké informácie, ktoré sa týkajú práve tohto tlačidla:



Inšpektor sa skladá z dvoch stĺpcov: ľavý obsahuje meno nastavenia (napr. **Caption** označuje popis na tlačidle) a v príslušnom pravom je momentálna hodnota (teraz je tam **Button1**). Informácie v pravom stĺpci môžeme meniť a tým sa bude meniť vzhľad tlačidla a niekedy aj jeho správanie. Zmeňte v Inšpektore nastavenie **Caption** na text, napr. **Tlačidlo**. Všimnite si, že zároveň sa zmenil popis na tlačidle.

Ak teraz spustíme projekt (kláves **F9**) na formulári sa objavilo tlačidlo s popisom **Tlačidlo** - na tlačidlo môžeme klikat', ale zatiaľ toto nemá žiadnu funkčnosť.

V Inšpektore môžete experimentovať ešte s týmito nastaveniami:

- **Caption** text na tlačidle
- **Font** nastavenie písma textu na tlačidle
- **Height** výška tlačidla
- **Width** šírka tlačidla
- **Left** x-ová súradnica polohy tlačidla
- **Top** y-ová súradnica polohy tlačidla

Pozor: zatiaľ neodporúčame meniť položku **Name**.

Teraz môžete vo formulári vytvoriť viac tlačidiel rôznych rozmerov, zmeňte im popisy a vyskúšajte, ako sa správajú, keď takúto aplikáciu spustíme (**F9**).

Ak potrebujete nejaké tlačidlo z formuláru odstrániť, stačí aby bolo označené (kliknite naň) a potom stlačte kláves **Del**.

Pozrime ešte paletu komponentov: skladá sa z viacerých záložiek a my sa postupne zoznámime len s niektorými z nich:

- zo štandardnej palety **Standard**
 - súčiastka **jednoduchý text** - vnútorné meno TLabel
 - súčiastka **editovací riadok** - vnútorné meno TEdit
 - súčiastka **textová plocha** - vnútorné meno TMemo
 - súčiastka **tlačidlo** - vnútorné meno TButton
 - súčiastka **posúvač** - vnútorné meno TScrollBar
- z prídavnej palety **Additional**
 - súčiastka **grafická plocha** - vnútorné meno TImage
- z prídavnej palety **System**
 - súčiastka **časovač** - vnútorné meno TTimer

Úlohy na precvičenie

Zadanie 1	Poukladajte do formulára 4 tlačidlá tak, aby sa navzájom neprekrývali, aby pokrývali celú plochu formulára a pritom každé malo iné rozmery.
Zadanie 2	Sledujte niektoré položené tlačidlo vo formulári a meňte mu veľkosť a polohu len zmenou nastavení v inšpektore objektov: <ul style="list-style-type: none">• Height výška tlačidla• Width šírka tlačidla• Left x-ová súradnica polohy tlačidla• Top y-ová súradnica polohy tlačidla
Zadanie 3	Postupne do formulára ukladajte rôzne komponenty (napr. jednoduchý text TLabel , editovací riadok TEdit , textová plocha TMemo a pod.). Experimentujte s ich rôznymi veľkosťami. V inšpektore objektov môžete meniť nastavenie písma (Font) pričom meníte nielen meno a veľkosť fontu, ale pri niektorých komponentoch sa dá meniť aj farba písma. Otestujte to a prípadne to otestujte aj po spustení aplikácie (F9).

4. Udalosť kliknutie na tlačidlo

Program s tlačidlami je už zaujímavejší, ako čisté okno, ale tlačidlá zatiaľ nerobia žiadne akcie. Každé tlačidlo môže mať priradenú nejakú akciu, ktorá sa spustí, keď naň počas behu programu klikneme - spustí sa zakaždým, keď naň klikneme. Priradenie akcie sa robí nasledovne:

- Na dané tlačidlo vo formulári dvojklikneme (predpokladajme, že to bolo tlačidlo **Button1** - prostredie dáva automaticky mená komponentom ako **Button1**, **Button2**, ... atď. Ak zmeníme text na tlačidle, meno tohto komponentu stále ostáva **Button1**).
- Objaví sa editovacie okno, v ktorom môžeme vidieť tieto riadky:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  |
end;
```

Všimnite si, že kurzor je nastavený medzi riadkami **begin** a **end**; - editovacie okno je pripravené, aby sme doň na správne miesto pripisovali nové príkazy programu. Pre nás je teraz dôležité si všimnúť slovo **Button1Click** - toto označuje, že to čo teraz naprogramujeme medzi slovami **begin** a **end** sa spustí vždy, keď sa (počas behu programu) klikne na tlačidlo **Button1**.

Na otestovanie tlačidiel sa naučíme jednoduchý príkaz na zmenu textu na tlačidlo:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    Button1.Caption := 'zmena';
end;
```

Program spustíme (F9) a keď teraz klikneme myšou na tlačidlo **Button1** (pravdepodobne s textom **Tlačidlo**), jeho popis sa zmení na slovo **zmena**. Ak máme v programe viac tlačidiel, každému môžeme naprogramovať inú akciu - dvojkliknutím sa predpripraví časť programu, do ktorej dopíšeme nové akcie.

Pozrime sa, čo sa v skutočnosti stalo s celým našim programom - v editovacom okne sme totiž videli len koniec programu, takže po vyrolovaní okna vidíme:

```
unit Unit1;

interface

uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics,
    Controls, Forms, Dialogs;

type
    TForm1 = class(TForm)
        Button1: TButton;
        procedure Button1Click(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
    Button1.Caption := 'zmena';
end;

end.
```

V prostredí Lazarus je to veľmi podobné:

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, LResources, Forms, Controls,
  Graphics, Dialogs, StdCtrls;

type

  { TForm1 }

  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  Form1: TForm1;

implementation

{ TForm1 }

procedure TForm1.Button1Click(Sender: TObject);
begin
  Button1.Caption := 'zmena';
end;

initialization
  {$I unit1.lrs}

end.
```

Pre začiatočníka je to asi priveľa textu, v ktorom zatiaľ poznáme len jeden nami pripísaný riadok. Všetky zvyšné riadky nám dopísalo samotné programátorské prostredie. Vďaka tomuto máme istotu, že sú to správne zapísané príkazy a vôbec im zatiaľ nemusíme rozumieť. Zapamätajte si, že prostredie sa nám bude snažiť pomáhať v písaní programu, ale my by sme mali na začiatok dodržiavať tieto jednoduché pravidlá:

- Pre nás sú určené len riadky programu medzi slovami **implementation** a záverečným **end**. Riadky pred **implementation** udržiava prostredie a my sem zatiaľ zasahovať nebudeme.
- Riadok `{$R *.dfm}` (resp. `{$I unit1.lrs}`) označuje, že náš program pracuje s formulárom, a preto ani tento riadok nebudeme rušiť.
- Akcie pre tlačidlá sú v špeciálnych blokoch začínajúcich slovom **procedure** a končiacich slovom **end**. Táto konštrukcia sa vytvára automaticky dvojklikom na tlačidlo a nesmieme ju do textu zapisovať sami. Tejto konštrukcii budeme neskôr hovoriť procedúra alebo podprogram.
- Ak chceme takúto akciu zrušiť, nesmieme ručne vymazať kompletne riadky tejto konštrukcie. Urobí to za nás Delphi, ak medzi **begin** a **end** budú len prázdne riadky a my tento program uložíme pomocou povelu **Save** z hlavnej ponuky **File** (alebo klávesom Ctrl+S). Ako sa to robí v prostredí Lazarus uvidíme neskôr.

Z týchto dôvodov budeme v ďalšom texte uvádzať len riadky programu v našej povolenej časti, t.j. medzi **implementation** a záverečným **end** s bodkou.

Pozrime sa teraz na príkaz, ktorý zmení text na tlačidle:

```
Button1.Caption := 'zmena';
```

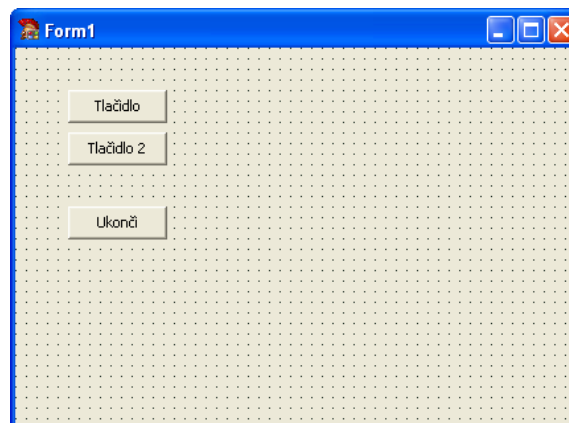
Každý znak v tomto riadku je veľmi dôležitý a ak sa pri písaní nejako preklepneme, prostredie nám to pri pokuse o spustenie (**F9**) oznámi takzvanou chybovou správou. Pozor, tieto správy sú v angličtine.

Tento riadok začína menom komponentu (súčiastka **Button1**), za bodkou nasleduje názov nastavenia (sú to názvy z Inšpektora Objektov) v našom prípade slovo **Caption**. Týmto sme zapísali, že budeme meniť text na tlačidle **Button1**. Ďalej nasleduje špeciálny dvoj-znak := (dvojbodka a znak rovná sa) - voláme ho znak priradenia. Za týmto dvoj-znakom nasleduje nová hodnota, ktorou chceme nahradiť pôvodné nastavenie - v našom príkaze je to takzvaný znakový reťazec '**zmena**', t.j. postupnosť znakov uzavretá v apostrofoch. Celý príkaz je ukončený znakom ; bodkočiarka.

Naučíme sa ešte jeden užitočný príkaz: ukončenie programu. Hoci používateľ môže náš program ukončiť napr. pomocou Alt+F4, my mu ponúkame na ukončenie programu aj tlačidlo **Ukonči**.

```
Close;
```

Na otestovanie teraz do formulára postupne vložíme 3 tlačidlá, zmeníme im texty - takto to vyzerá počas návrhu formulára:



a priradíme tieto akcie (nezabudnite dvojkliknúť na každé tlačidlo, aby sa vám zobrazila prázdna konštrukcia **procedure** pre každé tlačidlo):

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
    Button1.Caption := 'zmena';  
end;  
  
procedure TForm1.Button2Click(Sender: TObject);  
begin  
    Button1.Caption := 'Tlačidlo';  
    Button2.Caption := 'zmena 2';  
end;
```

```

end;

procedure TForm1.Button3Click(Sender: TObject);
begin
    Close;
end;

```

Na tomto príklade môžete vidieť, že v rámci jednej akcie (pre druhé tlačidlo) môžeme uviesť aj viac príkazov. Tieto sa potom budú spracovávať postupne: najprv sa zmení text na prvom tlačidle a potom na druhom. Tretie tlačidlo ukončí program. Poexperimentujte s tým, ako to funguje.

Na záver tejto časti sa naučíme ešte jeden veľmi jednoduchý komponent, ktorý sa nám môže hodiť vo formulári: je to súčiastka **jednoduchý text** (s vnútorným menom **TLabel**). Slúži na ukladanie textov do formulára, prípadne ho môžeme použiť aj na výpis jednoduchých oznamov vo formulári. Použitie tejto súčiastky ukážeme na tomto jednoduchom príklade: Vo formulári sa objaví nejaká jednoduchá otázka, napr. **Ako sa máš?** a pod ňou budú dve tlačidlá s textami: **Dobre** a **Zle**. Pod týmito dvoma tlačidlami je ešte pripravené miesto na odpoveď (zatiaľ sú tam tri bodky ...). Keď používateľ odpovie stlačením prvého tlačidla, počítač mu odpovie: **Mám z teba radosť**. Stlačenie druhého tlačidla vypíše správu: **Chúďa moje**.

Pustime sa do nového projektu:

- Do prázdneho formulára vložíme komponent **jednoduchý text** (zo záložky **Standard** zvolíme komponent **TLabel** a klikneme niekam do plochy formulára) - vo formulári sa objavil text **Label1** (to je jeho momentálny text podobne ako **Button1** na novom tlačidle).
- Zmeňme v Inšpektore Objektov tento text na **Ako sa máš?** - rovnako ako pri tlačidlách meníme nastavenie **Caption**.
- V Inšpektore Objektov zmeníme ešte aj písmo, aby bol tento text čitateľnejší - meníme nastavenie **Font**.
- Pod tento text vložíme dve tlačidlá a zmeníme im texty na slová **Dobre** a **Zle**.
- Na koniec pod tlačidlá pridáme ešte jeden text (jeho meno bude **Label2**): tiež mu zmeníme písmo aj text na ...

Formulár teraz vyzerá nejak takto:



Priradenie textov pre komponent **Label2** (druhý text) je veľmi podobné, ako sme to robili pri tlačidlách:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label2.Caption := 'Mám z teba radosť';
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Label2.Caption := 'Chúďa moje';
end;
```

Úlohy na precvičenie

Zadanie 1

Upravte posledný program tak, aby v ňom bolo viac možností odpovedí, t.j. pridajte ďalšie tlačidlá aj s príslušnými akciami. Napr. na otázku **Ako sa dnes cítiš?** by mohli byť odpovede na tlačidlách **Výborne**, **Som unavený**, **Cítim sa chorý**, **Normálne**, ... Ku každej odpovedi pripravte reakciu počítača, napr. jedna z odpovedí **To ma teší**, **Neuč sa už toľko**, **Chod' k doktorke**, **Ja tiež**, ...

Zadanie 2

Aplikácia zadá kvízovú otázku (alebo hádanku) a pritom na niektorých tlačidlách sú umiestnené možné odpovede. Tlačidlo so správnou odpoveďou vypíše gratuláciu. Ostatné tlačidlá vypíšu upozornenie alebo návod na správnú odpoveď.

Zadanie 3

Podobným postupom vytvorte program, ktorý bude návodom na uvarenie vajíčka na mätko: vo formulári bude napr. 5 tlačidiel s textami **1 krok**, **2 krok**, ... Po zatlačení každého tlačidla sa vypíše (do komponentu **jednoduchý text**) jeden riadok návodu, napr.

1. do hrnca daj vodu a vajíčko
2. daj vodu v hrnci variť na šporák
3. čakaj, kým nezačne voda vriieť
4. nechaj vriieť 3 minúty
5. vyber vajíčko z hrnca

5. Nastavovanie vlastností komponentov počas behu programu

Pripomeňme si riadok programu, pomocou ktorého sme menili text na tlačidle, resp. jednoduchom riadku:

```
Button1.Caption := 'zmena';
```

alebo

```
Label2.Caption := 'Chúďa moje';
```

Vidíme že takýto príkaz sa skladá z

- meno komponentu, napr. **Button1**, **Label2**, ...
- názov nastavenia, napr. **Caption**
- nová hodnota nastavenia, napr. znakový reťazec '**zmena**', '**Chúďa moje**', ...

Pritom vieme, že v tomto riadku má každý znak dôležitý význam a na každý preklep alebo nepresnosť nás prekladač upozorní chybovou správou.

Pomocou takýchto príkazov, môžeme meniť (priradovať) nielen textové hodnoty (menili sme text na tlačidle na nejakú novú hodnotu znakového reťazca) ale aj skoro všetky ďalšie nastavenia tak, ako ich vidíme v inšpektore objektov. Príkazu hovoríme priradovací príkaz a jeho úlohou je zmeniť momentálnu hodnotu nejakého nastavenia na novú hodnotu. Dôležitou podmienkou pri tom je aby nastavenie aj menená hodnota boli rovnakého typu, t.j. do textového nastavenia môžeme **priradiť** len textovú hodnotu a do číselného nastavenia len číslo.

Priradovací príkaz má teda takýto tvar:

```
komponent.nastavenie := nová hodnota;
```

Medzi komponentom a názvom nastavenia píšeme znak bodka, medzi ľavou časťou (menom nastavenia) a pravou časťou (priradovanou hodnotou) píšeme dvojnak := , tzv. znak priradenia. Celý príkaz je ukončený znakom bodkočiarka ;

Už sme videli použitie znakového reťazca: medzi dva apostrofy sme zapísali nejakú postupnosť znakov. Mali by sme vedieť, že celý znakový reťazec by sa mal zmestiť na jeden riadok programu a tiež to, že ak by sme chceli, aby obsahoval znak apostrof, tak tento v reťazci musíme zdvojiť, napr. takto

```
Label2.Caption := 'I'm busy';
```

Pozrime sa teraz ako môžeme meniť hodnoty netextových nastavení komponentov. Napr. nastavenia **Left** a **Top** (X-ová a Y-ová súradnica komponentu vo formulári) môžu obsahovať len celé čísla, napr.

```
Button1.Left := 100;
```

alebo

```
Label1.Top := 0;
```

Do jednej procedúry, napr. **Button1Click**, môžeme zadať aj viac príkazov, ktoré sa budú vykonávať postupne jeden za druhým. Budeme pritom dodržiavať takúto dohodu

- každý príkaz zapisujeme do nového riadka,
- na konci každého príkazu zapíšeme bodkočiarku,
- pred každý príkaz zapisujeme 2 medzery, teda každý riadok začína v tretom stĺpci.

Príkazy sa po spustení (F9) budú vykonávať postupne jeden za druhým.

Pozrime, ako môžeme zmeniť polohu nejakým komponentom, napr. tlačidlám. Ak predpokladáme, že ich počiatočná poloha je na súradniciach (50, 50) a (150, 50), t.j. sú vedľa seba najprv prvé a potom druhé, tak príkaz

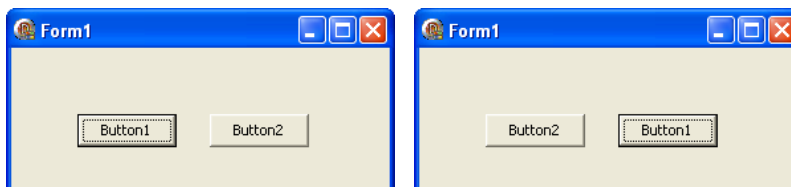
```

procedure TForm1.Button1Click(Sender: TObject);
begin
  Button1.Left := 150;
  Button2.Left := 50;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Button1.Left := 50;
  Button2.Left := 150;
end;

```

Ich navzájom presťahuje, t.j. **Button1** bude už napravo a **Button2** naľavo.



Ďalší príklad ukazuje spôsob zmeny veľkosti písma pre jednoduchý text (**Label**). Všimnite si, že názov nastavenia je zložitejší, lebo pre nastavenie **Font** meníme jeho atribút veľkosť, t.j. **Height**:

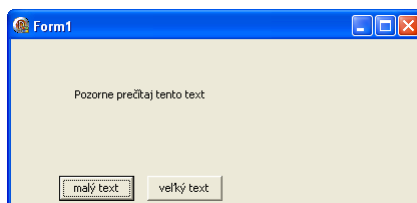
```

procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.Font.Height := 8;
end;

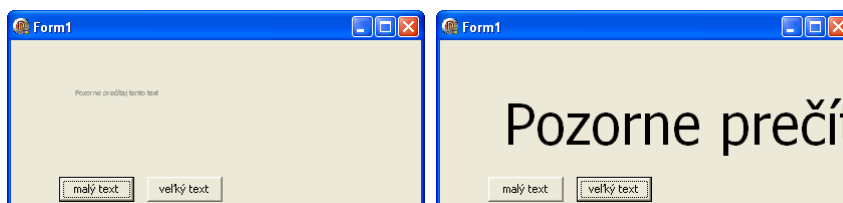
procedure TForm1.Button2Click(Sender: TObject);
begin
  Label1.Font.Height := 68;
end;

```

Keď takúto aplikáciu spustíme, dostaneme:



Prvé tlačidlo tento text ešte zmenší a druhé naopak zväčší:



Hoci takto zväčšený text nevidíme celý, celé okno môžeme natiahnuť tak, aby sme ho celý prečítali.

Úlohy na precvičenie

Zadanie 1	Vo formulári sa nachádza jeden komponent jednoduchý text s nejakou krátkou vetou. Rôzne tlačidlá menia rôzne atribúty tohto textu: polohu, veľkosť, písmo.
Zadanie 2	Tlačidlá menia veľkosť okna aplikácie: nastavenia bez uvedenia komponentu menia hodnoty pre formulár, teda Width zmení šírku a Height mení výšku okna.
Zadanie 3	Tlačidlá menia rôzne atribúty aplikácie, napr. nadpis okna (Caption := 'Nadpis'), farbu formuláru (Color := clRed), polohu (Top := 0; Left := 0;)

Čo sme sa naučili

Vieme naštartovať programovacie prostredie a pripraviť v ňom veľmi jednoduchú aplikáciu. Mali by sme rozumieť, že náš pripravený program sa skompiluje (vytvorí sa EXE) a potom sa spustí.

2. Úvod do grafiky

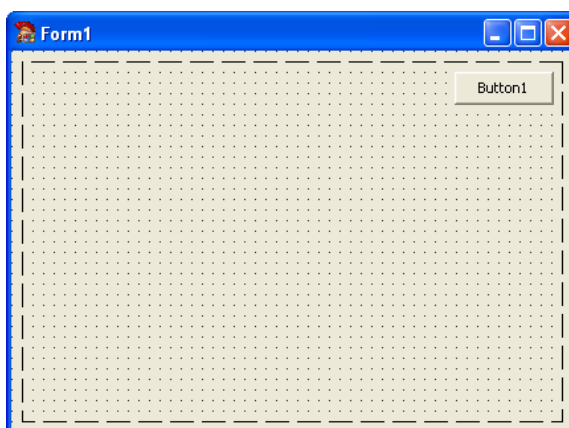
1. Grafická plocha a súradnicová sústava

Jedným z najdôležitejších komponentov bude pre nás grafická plocha (s vnútorným menom **TImage**). Pod týmto budeme rozumieť obdĺžnikovú oblasť, do ktorej môžeme rôznymi nástrojmi kresliť, písať, môžeme ju jednoducho uložiť na disk, resp. zo súboru do nej prečítať nejaký obrázok.

Neskôr sa naučíme, ako môže grafická plocha reagovať aj na klikanie, resp. ťahanie myšou.

Začneme s novým projektom:

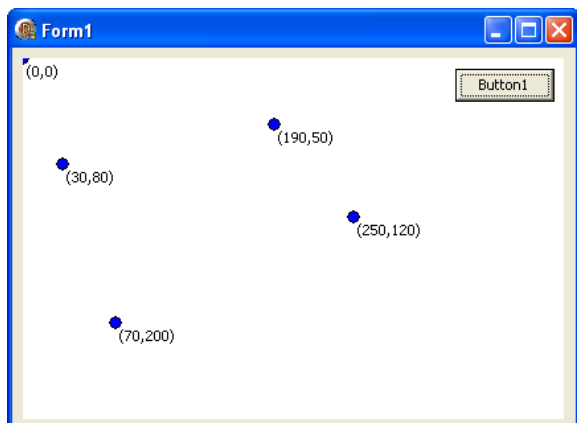
- Z palety komponentov **Additional** vyberieme komponent **TImage** a položíme ho do formulára: objavil sa tu rámik veľkosti asi 100x100 pričom ho hneď zväčšíme ťahaním za modré, resp. čierne značky podobne ako sme to robili s tlačidlami.
- Do formulára vložíme ešte jedno tlačidlo, pomocou ktorého budeme neskôr kresliť do tejto novej grafickej plochy.



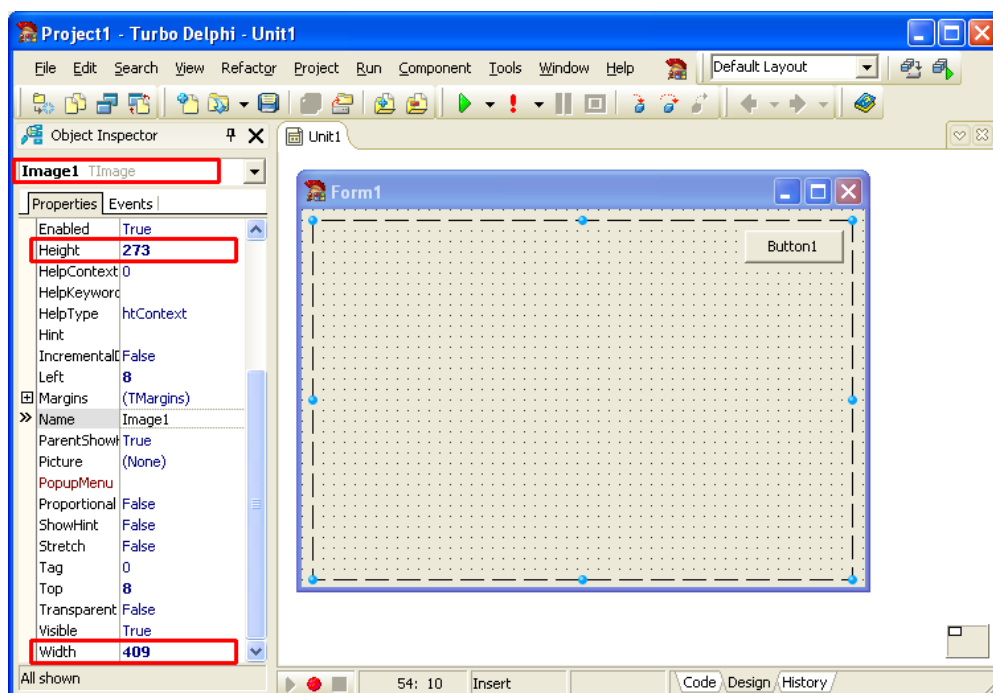
Grafickú plochu teda už máme. Kresliť do nej budeme pomocou príkazov, ktoré sa vyvolajú pri zatlačení tlačidla - teda, napr. do procedúry **Button1Click** budeme zapisovať grafické príkazy medzi **begin** a **end**. Skôr ako začneme kresliť, vysvetlíme si základné pojmy:

- Naša grafická plocha má meno **Image1** (podobne ako tlačidlo **Button1** a aj jednoduchý text **Label1**)
- Každá grafická plocha má svoje plátno a kreslíme práve do tohto plátna (je to niečo podobné, ako je plátno v zarámovaných obrazoch) - plátno sa volá **Canvas**.
- Na plátno kreslíme pomocou pera - **Pen** alebo vyfarbujeme nejaké plochy pomocou štetca **Brush**.
- Pero aj štetec majú svoje farby - **Color**, pero má aj svoju hrúbku - **Width**.
- Príkazy, ktoré pohybujú a kreslia perom väčšinou vyžadujú zadávanie súradníc: súradná sústava je ale v Objektovom Pascale trochu inak natočená, ako ju poznáme z matematiky - ľavý horný roh má súradnice (0,0) - teda toto je počiatok. X-ová súradnica ide zľava doprava na hornom okraji plochy a y-ová súradnica ide zhora nadol pri ľavom okraji plochy. Niekedy má zmysel používať aj záporné súradnice, hoci tieto popisujú body mimo plochy.

Pozrite, ako vyzerajú súradnice niekoľkých bodov v rovine:



Maximálna x-ová a y-ová súradnica viditeľnej časti grafickej plochy závisí od toho, ako sme natiahli komponent **Image1** do formulára. Ak potrebujete zistiť, aké je momentálne maximum, tak sa môžeme pozrieť do Inšpektora objektov:



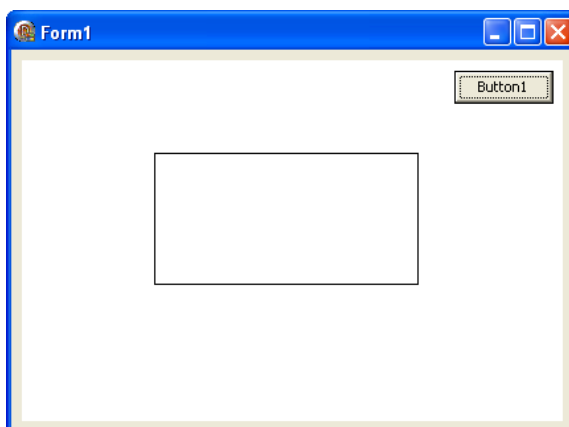
Nastavenia **Height** a **Width** pre komponent **Image1** vyjadrujú výšku a šírku grafickej plochy. Preto, ak chceme aby bol nejaký bod v ploche viditeľný, x-ová súradnica v našom konkrétnom príklade musí byť od 0 do 408 a y-ová od 0 do 272.

2. Príkaz na nakreslenie obdĺžnika, vymazanie plochy

Ako prvý grafický príkaz sa naučíme kreslenie obdĺžnika: príkaz **Rectangle** má 4 parametre, ktoré vyjadrujú súradnice dvoch protilahlých vrcholov obdĺžnika. Takto nakreslený obdĺžnik má strany rovnobežné s osami. Zadáme:

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
    Image1.Canvas.Rectangle(100, 70, 300, 170);  
end;
```

Po spustení programu (**F9**) a zatlačení tlačidla sa nakreslí požadovaný obdĺžnik:



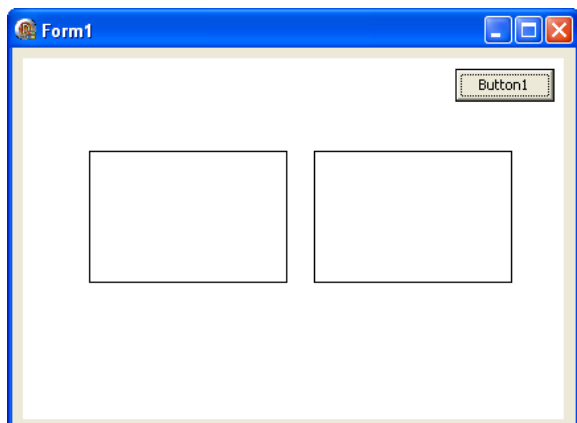
Pozorne si pozrime grafický príkaz na kreslenie obdĺžnika. Vidíme, že príkaz sa skladá z viacerých slov oddelených bodkami (**Image1.Canvas.Rectangle**) - poradie týchto slov a samozrejme aj presný zápis je veľmi dôležitý a vyjadruje toto: ideme pracovať s grafickou plochou **Image1** - budeme kresliť na jej plátno **Canvas** - konkrétne použijeme nástroj obdĺžnik - **Rectangle**. Štyri čísla v okrúhlych zátvorkách vyjadrujú súradnice nejakých dvoch protiláhlých vrcholov - prvé dve sú x-ová a y-ová súradnica prvého vrcholu a druhé dve opäť x-ová a y-ová súradnica druhého vrcholu. Treba si zapamätať, že tieto súradnice musia byť vždy celé čísla. Ak je niektorá z nich mimo veľkosť grafickej plochy, tak zrejme aj nejaká časť obdĺžnika bude mimo plochy.

Nakoľko budeme teraz postupne experimentovať s rôznymi nástrojmi na kreslenie do grafickej plochy, nezabudnite zakaždým program skončiť (napr. Alt+F4) a až potom robiť zmeny do programu.

V ďalšom príklade nakreslíme dva obdĺžniky vedľa seba, napr.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Rectangle(50, 70, 200, 170);
  Image1.Canvas.Rectangle(220, 70, 370, 170);
end;
```

A po spustení dostávame



Všimnite si, že oba obdĺžniky sú rovnako veľké a ich rozmery sú 150x100.

Niekedy budeme potrebovať zmazať celú grafickú plochu. Môžeme na to použiť príkaz **Rectangle**, nakoľko nakreslený obdĺžnik je vyplnený bielou farbou. Vytvoríme ďalšie tlačidlo, ktorého úlohou bude zmazať obsah plochy. Procedúra pre kliknutie na toto nové tlačidlo môže vyzeráť aj takto:

```
procedure TForm1.Button2Click(Sender: TObject);  
begin  
    Image1.Canvas.Rectangle(-1, -1, 1000, 1000);  
end;
```

Súradnice protilehlých vrcholov obdĺžnika sme zvolili tak, aby oba tieto body boli mimo grafickej plochy.

Úlohy na precvičenie

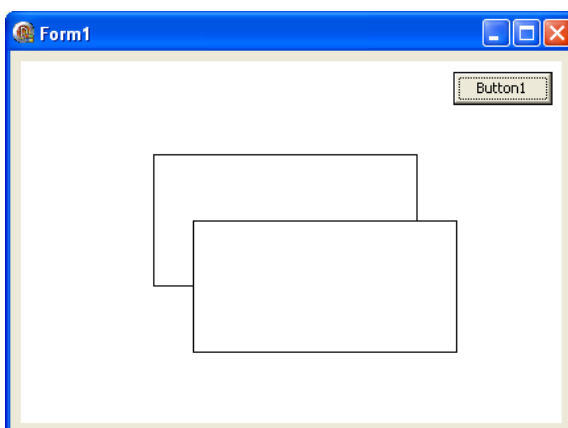
Zadanie 1	Nakreslite štvorec so stranou 85.
Zadanie 2	Nakreslite vedľa seba 3 dotýkajúce sa štvorce veľkostí 50, 70 a 90.
Zadanie 3	Do formulára dajte 3 tlačidlá, pre ktoré 1. tlačidlo nakreslí veľký štvorec (hlava robota) 2. tlačidlo nakreslí dva menšie štvorce vedľa seba (oči robota) 3. tlačidlo nakreslí podlhovastý obdĺžnik (ústa robota) tak, aby po postupnom zatlačení nakreslili hlavu robota s očami a ústami.

3. Pero, štetec a ich atribúty

V ďalšom príklade nakreslíme dva obdĺžniky tak, aby sa navzájom prekryvali, napr.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Rectangle(100, 70, 300, 170);
  Image1.Canvas.Rectangle(130, 120, 330, 220);
end;
```

A po spustení vidíme



Zapamätajte si, že oba obdĺžniky sú vyplnené bielou farbou a preto druhý zakryl časť prvého. Príkaz obdĺžnik (teda grafický nástroj **Rectangle**) pri kreslení použije na jeho obvod svoje pero (**Pen**) a konkrétne jeho farbu a hrúbku a vždy jeho vnútro vyfarbí farbou svojho štetca (**Brush**). Ak sme doteraz nič nenastavovali, tak obdĺžnik sa nakreslí s čiernym obrysom s hrúbkou pera 1 a vyplní sa bielou farbou štetca. Podobne, ako sme menili text na tlačidle pomocou znaku priradenia (napr. **Button1.Caption := 'zmena'**), môžeme nastavovať aj parametre pera a štetca. Len zápis bude trochu rozsiahlejší:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Pen.Color := clRed;
  Image1.Canvas.Brush.Color := clBlue;
  Image1.Canvas.Rectangle(100, 70, 300, 170);

  Image1.Canvas.Pen.Width := 5;
  Image1.Canvas.Brush.Color := clYellow;
  Image1.Canvas.Rectangle(130, 120, 330, 220);
end;
```

Skôr ako tento program spustíme, vysvetlíme jednotlivé časti nastavovania pera a štetca:

```
Image1.Canvas.Pen.Color := clRed;
```

Pred dvoj-znakom priradenia **:=** musí byť veľmi presne špecifikované, čo chceme zmeniť: v našom prípade budeme grafickej ploche **Image1** jej plátnu **Canvas** jeho peru **Pen** nastavovať farbu **Color**. Za znakom priradenia musíme nejako určiť požadovanú farbu. Zatiaľ sa naučíme najjednoduchší spôsob: t.j. výber jednej z už preddefinovaných farieb. Neskôr uvidíme aj iné spôsoby. Pascal nám ponúka niekoľko najčastejších farieb a my si z nich môžeme ľubovoľnú vybrať, napr.

- `clBlack` čierna
- `clRed` červená
- `clBlue` modrá
- `clGreen` zelená
- `clYellow` žltá
- `clWhite` biela
- `clGray` šedá

Nastavenie farby pera funguje veľmi podobne ako namočenie skutočného štetca do nejakej farby: od tohto momentu sa budú všetky čiary kresliť touto farbou až pokým ju nezmeníme namočením do inej farby. Samozrejme, že zmena farby pera nemá žiaden vplyv na už nakreslené obrázky.

Teraz by nám nemuselo robiť veľký problém pochopiť nastavenie farby štetca, t. j. farby, ktorou sa obdĺžnik vyfarbí:

```
Image1.Canvas.Brush.Color := clBlue;
```

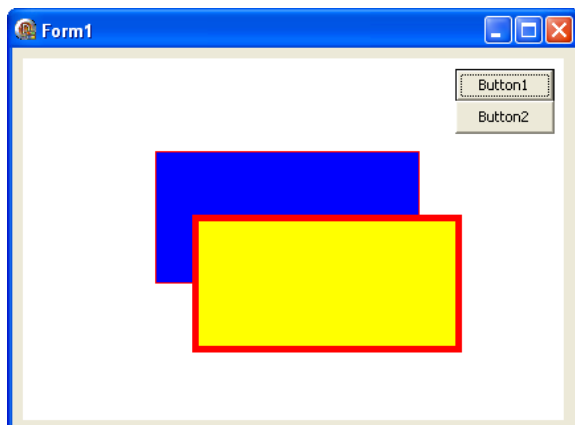
Rozdiel od nastavenia pera je len v slove **Brush**.

Hrúbku pera nastavujeme úplne analogickým spôsobom:

```
Image1.Canvas.Pen.Width := 5;
```

Pero môže mať hrúbku určenú ľubovoľným celým číslom väčším ako 0.

A teraz po spustení programu (F9) sa v grafickej ploche sa objaví

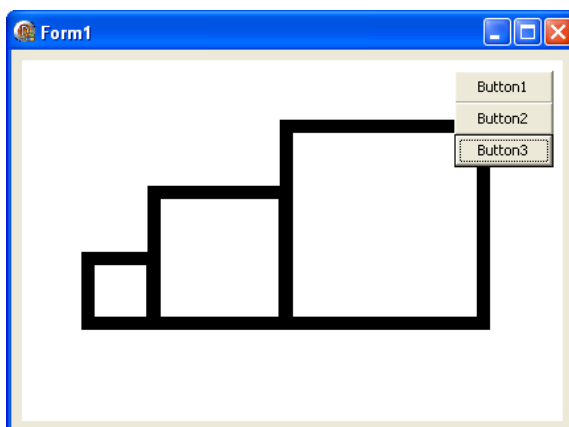


Vidíme, že prvý obdĺžnik má tenký červený obvod a je vyplnený modrou farbou, druhý obdĺžnik má hrubý červený obvod a je vyfarbený žltou farbou.

Pridajme do nášho projektu tretie tlačidlo (druhé mazalo plochu), pomocou ktorého sa vedľa seba nakreslia tri rôzne veľké štvorce:

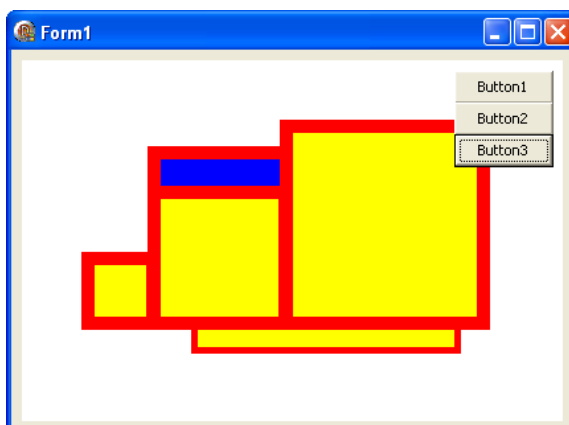
```
procedure TForm1.Button3Click(Sender: TObject);
begin
  Image1.Canvas.Pen.Width := 10;
  Image1.Canvas.Rectangle(50, 200, 100, 150);
  Image1.Canvas.Rectangle(100, 200, 200, 100);
  Image1.Canvas.Rectangle(200, 200, 350, 50);
end;
```

a po spustení

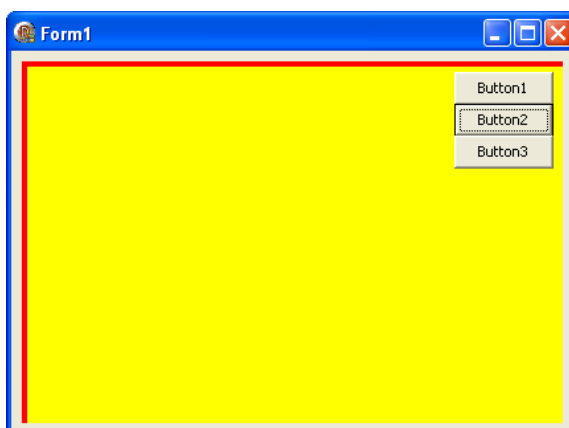


Všimnite si, že hrúbku sme nastavili len raz a odvtedy platí pre všetky ďalšie kresby. Niekedy sa musíme so súradnicami "trochu pohrať", aby sme dostali útvary podľa našich predstáv - tu sú štvorce nakreslené tesne vedľa seba a ich spodná strana leží na jednej priamke.

Uvedomte si, že grafická plocha si pamätá nastavenie svojho pera a štetca (pri štarte programu si pamätá čierne tenké pero a biely štetec). Takže, ak v tomto programe stlačíme po tlačidlo **Button3** tlačidlo **Button1** a opäť **Button3**, vidíme, že program si pamätal svoje nastavenia a druhé zatlačenie tlačidla **Button3** už nakreslí žltou vyfarbené štvorce.



Prekvapením pre nás bude aj tlačidlo **Button2**, ktoré slúžilo na zmazanie grafickej plochy. Ak ho zatlačíme po zatlačení **Button1** aj **Button3**, dostaneme:



Dôvodom, prečo plocha nie je po zmazaní biela ale žltá a ešte u nej vidíme aj kúsok červeného rámiku, je ten, že z predchádzajúceho kreslenia si grafická plocha pamätá žltý štetec a hrubé červené pero. Preto pred samotným mazaním plochy musíme tieto atribúty zmeniť.

Naučíme sa ale nový príkaz, ktorý prefarbí celú grafickú plochu jednou farbou a nepotrebuje na to pero (nebudeme musieť meniť hrúbku ani farbu pera). Teda druhé tlačidlo **Button2** zmeníme takto:

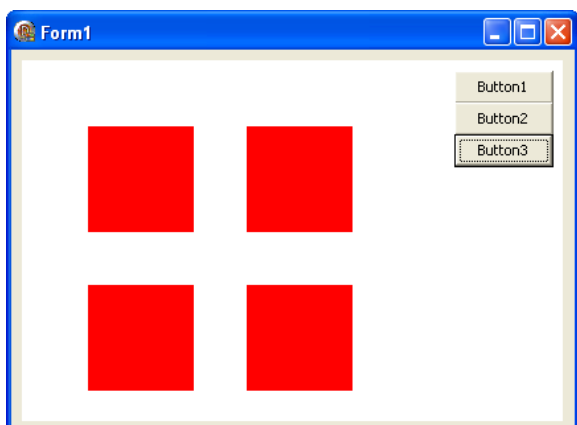
```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Image1.Canvas.Brush.Color := clWhite;
  Image1.Canvas.FillRect(Image1.ClientRect);
end;
```

Skôr ako zmažeme grafickú plochu príkazom **FillRect**, nastavíme farbu štetca na bielu, resp. na ľubovoľnú inú, ktorou chceme vymazať (prípadne vyfarbiť) grafickú plochu.

Niekedy môžeme využiť aj biely obdĺžnik na zmazanie nejakej časti kresby. Nasledujúci program nakreslí jeden veľký červený štvorec a potom z neho časť zmaže pomocou dvoch bielych obdĺžnikov:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  Image1.Canvas.Pen.Color := clRed;
  Image1.Canvas.Brush.Color := clRed;
  Image1.Canvas.Rectangle(50, 50, 250, 250);
  Image1.Canvas.Pen.Color := clWhite;
  Image1.Canvas.Brush.Color := clWhite;
  Image1.Canvas.Rectangle(50, 130, 250, 170);
  Image1.Canvas.Rectangle(130, 50, 170, 250);
end;
```

Po spustení dostaneme:



Úlohy na precvičenie

Zadanie 1	Pomocou farebných obdĺžnikov nakreslite štátne zástavy niektorých štátov, napr. pre Belgicko, Bulharsko, Francúzsko, Estónsko, Nemecko, Rumunsko, ...
Zadanie 2	Nakreslite šachovnicu 4x4 na najmenší počet volania príkazu Rectangle . Na šachovnici by sa mali striedať dve farby, napr. žltá a červená.
Zadanie 3	Nakreslite pyramídu pomocou rôzne dlhých obdĺžnikov pričom všetky majú rovnakú výšku, napr. 20. Šírky obdĺžnikov by mohli byť napr. 60, 100, 140, 180.
Zadanie 4	Nakreslite 5 štvorcov, ktoré majú spoločný stred, ich strany sú postupne 300, 240, 180, 120, 60 a každý je vyplnený inou farbou.
Zadanie 5	Pomocou rôzne veľkých vyplnených obdĺžnikov nakreslite niektoré písmená, napr. H, T, E, F, L.
Zadanie 6	V programe vytvorte 7 malých tlačidiel: <ul style="list-style-type: none">• tri tlačidlá, ktoré len menia farbu pera na červenú, modrú a zelenú• ďalšie tri tlačidlá, ktoré menia farbu štetca (výplne) na žltú, bielu a šedú• jedno tlačidlo, ktoré nakreslí obdĺžnik s hrúbkou pera 10, pričom sa použijú nastavenia farieb rámiku a výplne pomocou 6 tlačidiel.

4. Kreslenie čiar

Grafická plocha má na kreslenie rôznych obrázkov väčšie množstvo nástrojov a my sme sa zatiaľ naučili pracovať len s príkazom **Rectangle**.

Dvojicou príkazov na kreslenie v grafickej ploche sú **MoveTo** a **LineTo**. Oba presúvajú grafické pero na nové pozície: **MoveTo** pritom nekreslí čiary a **LineTo** kreslí čiary nastavenou farbou a hrúbkou.

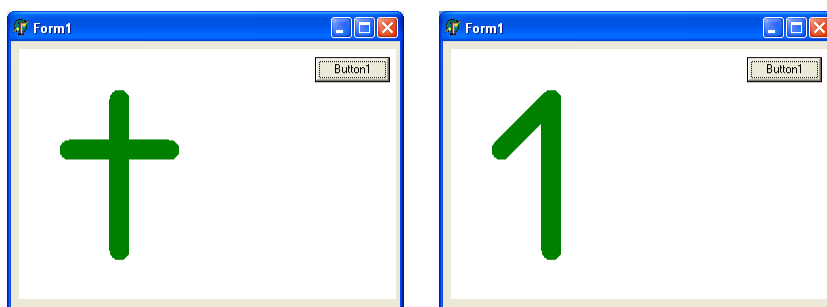
Teda **MoveTo** nastaví začiatok úsečky a **LineTo** jej koniec. Napr. pomocou dvoch úsečiek môžeme nakresliť krížik:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Pen.Width := 20;
  Image1.Canvas.Pen.Color := clGreen;
  Image1.Canvas.MoveTo(100, 200);
  Image1.Canvas.LineTo(100, 50);
  Image1.Canvas.MoveTo(150, 100);
  Image1.Canvas.LineTo(50, 100);
end;
```

Ak je viac **LineTo** za sebou, tak sa kreslí lomená čiara. Ak vyradíme druhé **MoveTo**, nakreslí sa znak jednotka. Tento riadok programu môžeme buď vyhodit', alebo urobiť z neho komentár: na začiatok riadka pridáme dve lomky (napíšeme ich tesne za sebou) a text za týmito dvoma znakmi sa ďalej už ignoruje. Komentáre sa najčastejšie používajú na písanie poznámok do textu programu.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Pen.Width := 10;
  Image1.Canvas.Pen.Color := clGreen;
  Image1.Canvas.MoveTo(100, 200);
  Image1.Canvas.LineTo(100, 50);
  // Image1.Canvas.MoveTo(150, 100);
  Image1.Canvas.LineTo(50, 100);
end;
```

Nakreslené obrázky sú pred úpravou a po úprave:



Veľmi často sa obrázky skladajú z viacerých čiar a potom aj náš program vyzerá ako postupnosť príkazov **LineTo**, ktorá začína **MoveTo**. Zistite, čo nakreslí

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Image1.Canvas.Pen.Width := 10;
  Image1.Canvas.MoveTo(150, 50);
  Image1.Canvas.LineTo(200, 200);
  Image1.Canvas.LineTo(70, 110);
  Image1.Canvas.LineTo(230, 110);
  Image1.Canvas.LineTo(100, 200);
  Image1.Canvas.LineTo(150, 50);
end;
```

Niekedy využijeme kreslenie aj veľmi hrubých čiar. V nasledujúcom príklade dve procedúry pre **Button1** a **Button2** robia presne to isté: kreslia nejaký rovnoramenný trojuholník. Lenže **Button1** to nakreslí veľmi hrubým perom (konkrétne 100) a **Button2** tenkým perom (5):

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Pen.Color := clBlue;
  Image1.Canvas.Pen.Width := 100;
  Image1.Canvas.MoveTo(120, 200);
  Image1.Canvas.LineTo(320, 200);
  Image1.Canvas.LineTo(220, 100);
```

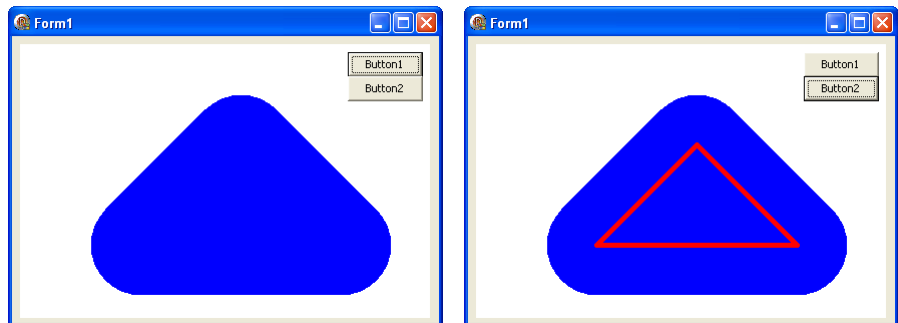
```

Image1.Canvas.LineTo(120, 200);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Image1.Canvas.Pen.Color := clRed;
Image1.Canvas.Pen.Width := 5;
Image1.Canvas.MoveTo(120, 200);
Image1.Canvas.LineTo(320, 200);
Image1.Canvas.LineTo(220, 100);
Image1.Canvas.LineTo(120, 200);
end;

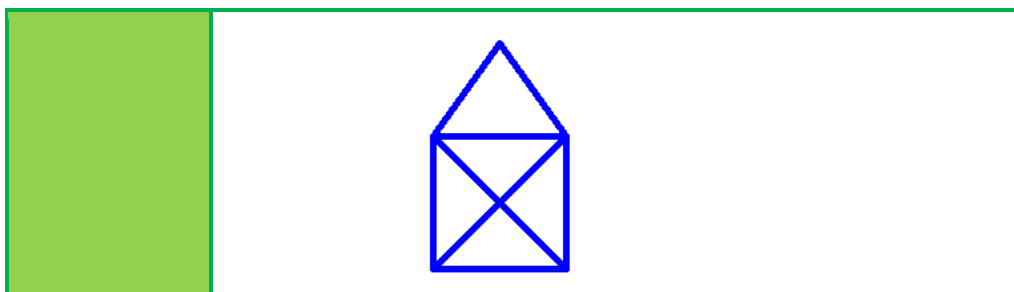
```

Na nasledujúcich dvoch obrázkoch môžete vidieť grafickú plochu po zatlačení prvého tlačidla a potom aj druhého:



Úlohy na precvičenie

Zadanie 1	Pomocou MoveTo a LineTo nakreslite rovnoramenný trojuholník so stranou 200.
Zadanie 2	Nakreslite ľubovoľný obdĺžnik, ktorý nebude mať strany rovnobežné s osami x a y. Na takýto obdĺžnik nevyužijete príkaz Rectangle .
Zadanie 3	Pomocou hrubého pera a príkazov MoveTo a LineTo nakreslite modrý kruh s priemerom 100.
Zadanie 4	<p>V programe vytvorte 6 malých tlačidiel:</p> <ul style="list-style-type: none"> • prvé tlačidlo zmaže grafickú plochu, presunie zdvihnuté pero do bodu (100, 250) a nastaví hrubé modré pero • ďalších 5 tlačidiel bude obsahovať jediný príkaz: presunúť spustené pero do týchto bodov: (100,250), (200,250), (100,150), (200,150), (150,80) <p>Pomocou stlačenia týchto tlačidiel nakreslite takýto domček:</p>



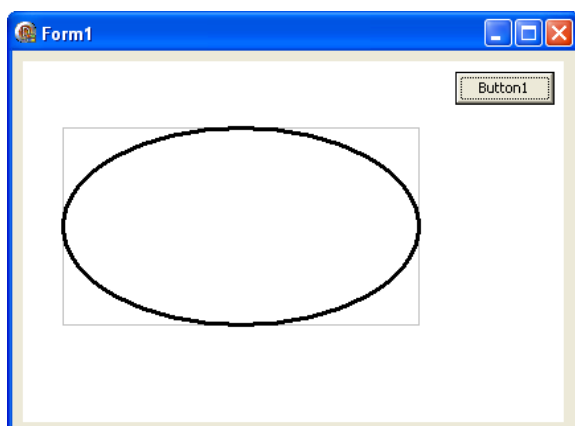
5. Kreslenie elipsy a kruhu

Grafický príkaz **Ellipse** nakreslí elipsu, ale s dobre nastavenými parametrami vie nakresliť aj kružnicu. Má štyri parametre podobne ako **Rectangle** a tieto majú rovnaký význam ako **Rectangle**:

- predstavte si, že nakreslíme nejaký obdĺžnik a chceme do neho vpísať elipsu. Tak použijeme presne tie isté parametre ako na nakreslenie obdĺžnika. V nasledujúcom príklade si všimnite parametre pre **Rectangle** a **Ellipse**:

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
    Image1.Canvas.Pen.Color := clLtGray;  
    Image1.Canvas.Rectangle(30, 50, 300, 200);  
    Image1.Canvas.Pen.Width := 3;  
    Image1.Canvas.Pen.Color := clBlack;  
    Image1.Canvas.Ellipse(30, 50, 300, 200);  
end;
```

A po spustení:



Zrejme, keď budeme chcieť nakresliť kružnicu, tak parametre pre **Ellipse** zadáme tak, aby sme popísali štvorec. Pozrite si nasledujúci príklad a zistite, čo nakreslí:


```

procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Ellipse(50, 130, 150, 230);
  Image1.Canvas.Ellipse(65, 60, 135, 130);
  Image1.Canvas.Ellipse(80, 20, 120, 60);
end;

```

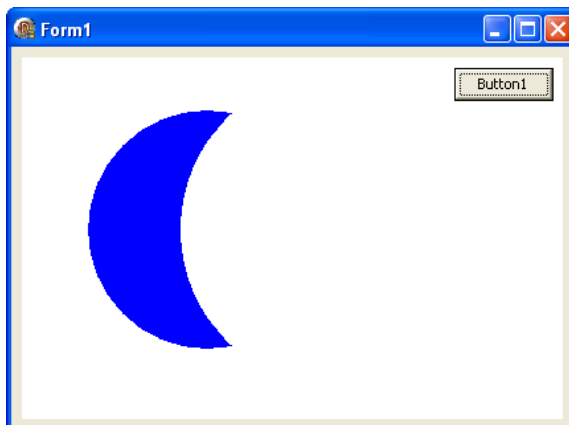
Podobne ako pri obdĺžniku môžeme nastaviť aj farbu výplne. Ukazuje to tento príklad:

```

procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Brush.Color := clBlue;
  Image1.Canvas.Pen.Color := clBlue;
  Image1.Canvas.Ellipse(50, 40, 230, 220);
  Image1.Canvas.Brush.Color := clWhite;
  Image1.Canvas.Pen.Color := clWhite;
  Image1.Canvas.Ellipse(120, 10, 360, 250);
end;

```

a po spustení môžeme vidieť, že prvá kružnica je prekrytá druhou celou bielou:



Úlohy na precvičenie

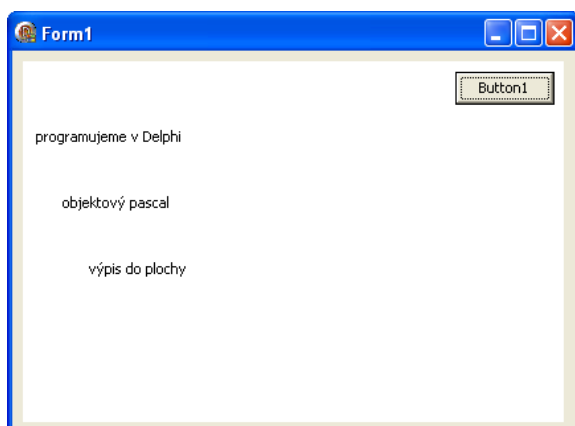
Zadanie 1	Nakreslite bieleho snehuliaka na šedom pozadí.
Zadanie 2	Pomocou obdĺžnikov a elíps nakreslite čo najviac písmen anglickej abecedy.
Zadanie 3	Nakreslite 5 sústredných kružníc.
Zadanie 4	Nakreslite semafor.
Zadanie 5	Nakreslite strom: hnedý hrubý kmeň a na ňom zelená vyplnená elipsa.

6. Písanie textov do grafickej plochy

Na písanie textov do grafickej plochy slúži príkaz **TextOut**. Má tri parametre: prvé dva sú súradnice, kde chceme nejaký text vypísať a tretí je samotný vypisovaný text. Text nesmieme zabudnúť uzavrieť do apostrofov rovnako, ako keď sme nastavovali text pre tlačidlá alebo pre textový komponent (**Label2.Caption := 'Chúďa moje'**);). Pozrite si nasledovný príklad použitia tohto príkazu:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.TextOut(10, 50, 'programujeme v Delphi');
  Image1.Canvas.TextOut(30, 100, 'objektový pascal');
  Image1.Canvas.TextOut(50, 150, 'výpis do plochy');
end;
```

Po spustení dostávame



Takto vypisovaný text môže byť samozrejme väčší a v programe preň môžeme zmeniť rôzne nastavenia. Opäť je mechanizmus veľmi podobný tomu, ako nastavujeme parametre pera. Napr.

```
Image1.Canvas.Pen.Width := 3;
```

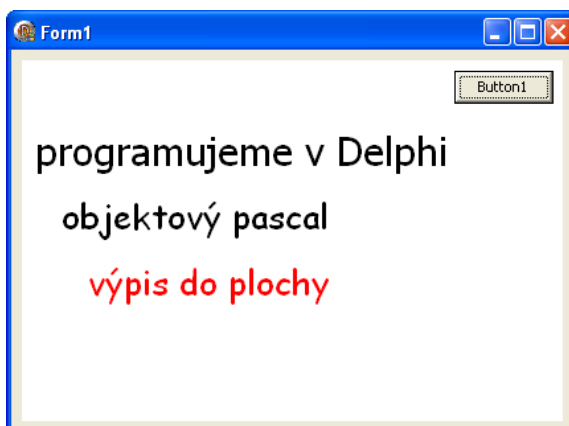
nastavuje hrúbku pera. A pre všetky parametre písma (vnútorný názov **Font**) musíme tiež uviesť **Image1.Canvas**, potom **Font** a za tým názov nastavenia, napr. **Height** (výška). Napr.

```
Image1.Canvas.Font.Height := 36;
```

zmení veľkosť písma. Podobne meníme farbu (**Color**) a meno písma (**Name**), napr. 'Arial', 'Times New Roman', 'Comic Sans Ms' a pod. Ilustruje to nasledujúci príklad:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Font.Height := 36;
  Image1.Canvas.TextOut(10, 50, 'programujeme v Delphi');
  Image1.Canvas.Font.Name := 'Comic Sans MS';
  Image1.Canvas.TextOut(30, 100, 'objektový pascal');
  Image1.Canvas.Font.Color := clRed;
  Image1.Canvas.TextOut(50, 150, 'výpis do plochy');
end;
```

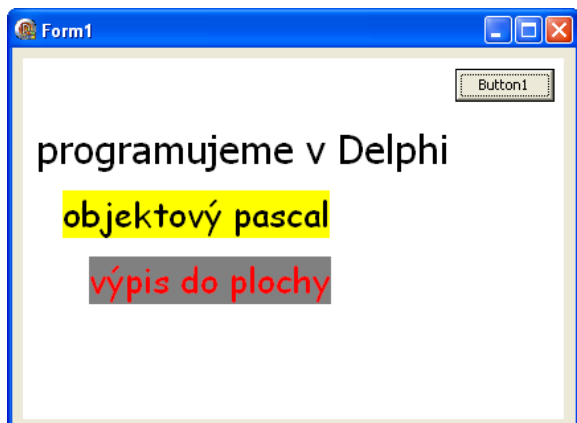
A teraz po spustení dostávame



Pri písaní textov do plochy si treba dávať pozor na to, že každý takýto text dostáva pod seba podklad (farebný obdĺžnik), ktorý je vyplnený momentálnou farbou štetca (**Brush**), napr.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Font.Height := 36;
  Image1.Canvas.TextOut(10, 50, 'programujeme v Delphi');
  Image1.Canvas.Brush.Color := clYellow;
  Image1.Canvas.Font.Name := 'Comic Sans MS';
  Image1.Canvas.TextOut(30, 100, 'objektový pascal');
  Image1.Canvas.Font.Color := clRed;
  Image1.Canvas.Brush.Color := clGray;
  Image1.Canvas.TextOut(50, 150, 'výpis do plochy');end;
```

A teraz po spustení dostávame



Z tohto dôvodu si treba uvedomiť aj to, že ak sa texty prekrývajú, niekú ich časť nemusí byť vidieť.

Úlohy na precvičenie

Zadanie 1	Vypíšte nejaké slovo (napr. DELPHI) veľkými písmenami tak, aby každé bolo inej farby. Nastavte nejaké väčšie písmo.
Zadanie 2	Presne do stredov strán grafickej plochy napíšte slová 'sever', 'východ', 'juh' a 'západ'.
Zadanie 3	Ak nejaký text napíšeme najprv svetlo šedou farbou (cILtGray) a potom čiernou ale trochu ho posunieme o niekoľko bodov, napr. smerom nadol a vľavo, získame dojem tieňa. Vyskúšajte tento efekt s nejakým väčším zaujímavým písmom.

Čo sme sa naučili v tomto module

Účastník vzdelávania po úspešnom absolvovaní tohto modulu

- vie pomenovať časti vývojového prostredia, vie používať elementárne príkazy na zmenu vlastností komponentov a tiež vie zapisovať základné grafické príkazy,
- dokáže analyzovať zadanie a tiež možné chyby v riešení, chyby vie nájsť a opraviť,
- dokáže vytvoriť najjednoduchšie aplikácie v prostredí Delphi (Lazarus),
- rozumie princípu vykonávania bežiaceho programu, rozumie fungovaniu udalosti kliknutia na tlačidlo.

Preverenie výstupných vedomostí

Účastník vzdelávania vie naprogramovať takúto aplikáciu:

Vo formulári je niekoľko tlačidiel a jedna grafická plocha. Každé tlačidlo nakreslí nejakú časť nejakého objektu. Navrhnete projekt, pomocou ktorého sa dá nakresliť jeden z objektov:

- robot (hlava, trup, ruky, nohy, časti tváre, ...)
- dom so záhradou (strecha, okná, dvere, komín, strom, ...)
- nákladné auto (kabína, kolesá, nákladný priestor, náklad, ...)
- bábika
- strašiak do záhrady

Účastník zostaví funkčnú aplikáciu samostatne v prostredí Delphi/Lazarus, iba pomocou týchto materiálov a svojich poznámok.

Základné materiály

- [1] Blaho A., "Informatika pre stredné školy. Programovanie v Delphi", SPN Bratislava, 2006
- [2] Cieľové požiadavky na vedomosti a zručnosti maturantov z informatiky, Štátny pedagogický ústav, <http://www.statpedu.sk/>

Webové stránky pre Delphi sú väčšinou skoro bez zmien použiteľné aj pre Lazarus a naopak.

Internetové zdroje pre prostredie Delphi

Väčšina webových stránok je v anglickom jazyku

- [3] Delphi - vysokoškolské prednášky na FMFI UK, <http://www.delphi.input.sk/>
- [4] prijímacie pohovory z informatiky na FMFI UK, <http://www.prijimacky.input.sk/>
- [5] Delphi Programming, <http://delphi.about.com/>
- [6] Marco Cantu, <http://www.marcocantu.com/>
- [7] Torry's Delphi Pages, <http://www.torry.net/>

Internetové zdroje pre prostredie Lazarus

- [8] Lazarus wiki, http://wiki.lazarus.freepascal.org/Main_Page/sk
- [9] Lazarus Documentation/sk, http://wiki.lazarus.freepascal.org/Lazarus_Documentation/sk
- [10] Free pascal, <http://www.freepascal.org/>

Poznámky

Poznámky

Tento študijný materiál vznikol ako súčasť národného projektu Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika v rámci Aktivity „Vzdelávanie nekvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ“.

Autori © RNDr. Andrej Blaho
RNDr. Ľubomír Salanci, PhD.

Názov Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Podnázov Programovanie 1

Študijný materiál prešiel recenzným pokračovaním.

Recenzenti RNDr. Peter Gurský, PhD.
doc. RNDr. Gabriela Andrejková, CSc.

Počet strán 40

Náklad 300 ks

Prvé vydanie, Bratislava 2009

Všetky práva vyhradené.

Toto dielo ani žiadnu jeho časť nemožno reprodukovat' bez súhlasu majiteľa práv.

Vydal Štátny pedagogický ústav, Pluhová 8, 830 00 Bratislava, v súčinnosti s Univerzitou Pavla Jozefa Šafárika v Košiciach, Univerzitou Komenského v Bratislave, Univerzitou Konštantína Filozofa v Nitre, Univerzitou Mateja Bela v Banskej Bystrici a Žilinskou univerzitou v Žiline

Vytlačil BRATIA SABOVCI, s r.o., Zvolen

ISBN 978-80-89225-54-5