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 2

Predmet: Programovanie

Línia: Vlastný odborový kontext informatiky a informatickej výchovy

Operačný program VZDELÁVANIE

EURÓPSKA ÚNIA
Európsky sociálny fond

Európska únia
Európsky sociálny fond

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

Programovanie 2

Identifikácia modulu

Aktivita projektu:	1.3 Ďalšie vzdelávanie kvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ
Línia aktivity:	Vlastný odborový kontext informatiky a informatickej výchovy
Predmet:	Programovanie

Garant predmetu:

RNDr. Ľubomír Salanci, PhD.
KZVI FMFI UK, Bratislava
salanci@fmph.uniba.sk

Autori:

RNDr. Andrej Blaho
KAI FMFI UK, Bratislava
RNDr. Zuzana Kubincová, PhD.
KZVI FMFI UK, Bratislava
RNDr. Ľubomír Salanci, PhD.
KZVI FMFI UK, Bratislava

Zaradenie modulu

Líniu Informatika v tomto vzdelávaní tvorí 8 modulov. 5 z nich je programovanie.

Moduly ***Programovanie 2***, *Programovanie 3* a *Programovanie 4* na seba nadväzujú. Moduly ***Programovanie 2*** a *Programovanie 3* sú povinné a obsahujú učivo, ktoré pokrýva časť Postupy, riešenie problémov, algoritmické myslenie v maturitnom štandarde pre informatiku. Modul *Programovanie 4* je voliteľný.

Abstrakt modulu

V module *Programovanie 2* sa učíme základy programovania v jazyku *Objektový Pascal*. Na zápis a ladenie programov využívame prostredie Delphi alebo Lazarus. Modul je rozdelený na sedem častí, sedem tematických jednotiek. V prvej sa oboznamujeme s prostredím, učíme sa pracovať s komponentmi, spúšťať programy a vytvárať prvé aplikácie s tlačidlami, ktoré spúšťajú príkazy a grafickou plochou, ktorá zobrazuje výstup programu. V druhej časti sa učíme používať grafické príkazy a generátor náhodných čísel. V tretej časti sa oboznamujeme s tým, ako môžeme v programe meniť vlastnosti komponentov. Vo štvrtej časť spoznávame premenné, typy hodnôt a viac sa venujeme príkazu priradenia. Ozrejmujeme si význam lokálnych a globálnych premenných. Naučíme sa aj získavať vstupné údaje od používateľa pomocou editovacieho políčka. V piatej časti vysvetľujeme programovú konštrukciu cyklu s pevným počtom opakovaní (`for`-cyklus) a používame ju na zápis algoritmov s cyklami. V šiestej časti vysvetľujeme podmienený príkaz (`if-then`, `if-then-else`) a ukazujeme algoritmy, ktoré vyžadujú vetvenie programu. V poslednej časti sme sa zamerali na cyklus s podmienkou (`while do`), ktorý používame pri zápise algoritmov s neznámym počtom opakovaní.

Modul má e-learningovú podporu

Obsah

Úvod

V informatike nás zaujímajú najmä také algoritmy, ktoré sa dajú vykonávať na určitom zariadení, napríklad na počítači. Algoritmy môžeme zapisovať rôznym spôsobom. Pre nás sú však dôležité zápisy, ktoré vieme nielen jednoznačne čítať a pochopiť ich, ale vieme o nich aj uvažovať, zisťovať ich vlastnosti, obmedzenia, dokazovať ich správnosť, efektívnosť. Takým zápisom je aj program v programovacom jazyku. Už sme sa stretli s jazykom Logo a korytnačkou, ktorá rozumela našim príkazom, ak boli formálne správne zapísané.

Teraz sa začneme učiť programovací jazyk Pascal (presnejšie Objektový Pascal alebo Object Pascal). V jazyku Pascal sa programuje iným spôsobom, ako v jazyku Logo. Jazyk Pascal bol však navrhnutý s úmyslom, že v ňom budú písať programy začiatočníci. Preto má premyslené pravidlá, programy, ktoré v ňom napíšeme, budú zrozumiteľné a dobre čitateľné. Programy v jazyku Pascal budeme zapisovať a testovať prostredí Delphi alebo Lazarus. Vďaka tomu dokážeme ihneď využívať dobré vlastnosti súčasných počítačov. Od začiatku budeme vytvárať vizuálne príťažlivé aplikácie alebo budeme kresliť obrázky pomocou grafických príkazov

Cieľ modulu

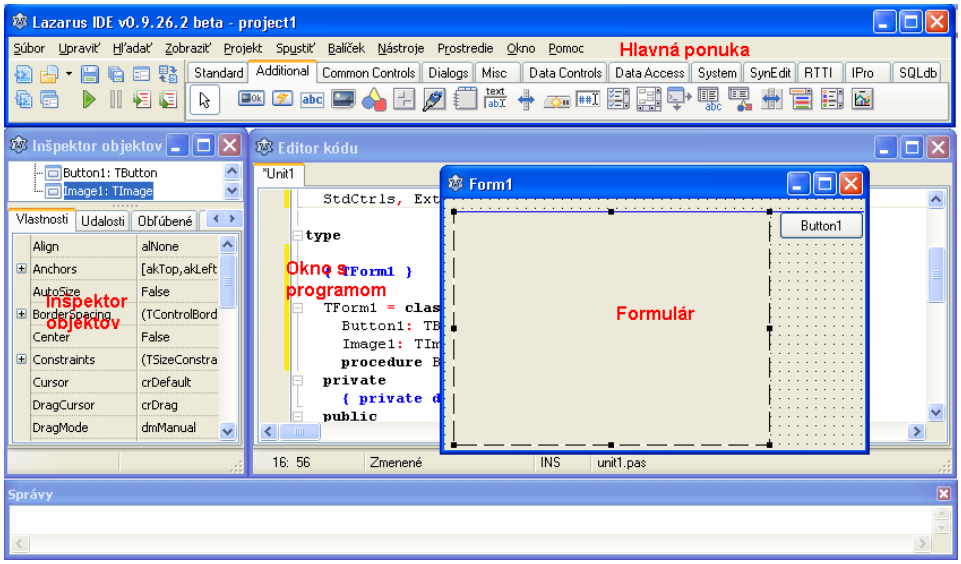
Našim cieľom je naučiť základy programovania v jazyku Pascal s využitím prostredia Delphi alebo Lazarus. Pri zápise vlastných algoritmov chceme naučiť používať grafické príkazy, premenné, programové konštrukcie cyklu s pevným počtom opakovaní a s podmienkou na začiatku, ako aj podmienené príkazy.

Kapitola 1: Začíname

Mnohé programy v súčasných operačných systémoch Windows aj Linux komunikujú s používateľom prostredníctvom **okien**. Napríklad, v okne grafického editora sa zobrazuje ponuka (menu), papier a tlačidlá s nástrojmi na kreslenie. Aj my budeme vytvárať podobné moderné oknové aplikácie.

Vytvorenie a spustenie novej aplikácie

1. Spustíme prostredie Delphi alebo Lazarus.



Prostredie má tieto časti: Hlavná ponuka, Inšpektor, Formulár, Program

2. Vytvoríme nový program (aplikáciu) príkazom z ponuky:
   - Delphi: File ▶ New ▶ Application
   - Lazarus: *Súbor* ▶ *Nový* ▶ *Projekt-Aplikácia*
3. Túto aplikáciu spustíme povelom tlačidlom alebo z ponuky:
   - Delphi: *Run* ▶ *Run*.
   - Lazarus: *Spustiť* ▶ *Spustiť*.
4. Na obrazovke uvidíme prázdne okno s nápisom `Form1`. Na lište bežiacich programov vo Windows uvidíme, že pribudol program s názvom *project1*.
5. Spustený program ukončíme tak, ako iné programy – kliknutím na

Viac o vývojovom prostredí

Vo **vývojovom prostredí** budeme vytvárať aplikácie tak, že:

- navrhneme **vzhľad aplikácie**, teda, ako má vyzerať okno nášho programu,
- budeme programovať v jazyku Pascal (presnejšie Object Pascal),
- budeme spúšťať a **ladiť**, čiže testovať a opravovať, naše programy.

Komponenty sú stavebné prvky, z ktorých vytvárame aplikácie. Slúžia na interakciu používateľa s programom, na zadávanie vstupných a zobrazovanie výstupných hodnôt alebo na riadenie behu aplikácie.

Vzhľad aplikácie vytvoríme tým, že z **palety komponentov** postupne umiestnime **komponenty** do **formulára** (pripomína to skladanie stavby z kociek Lega):

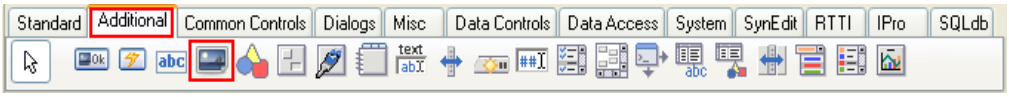
1. Komponent vyberieme z palety kliknutím myšou na ikonu komponentu.
2. Vybraný komponent potom umiestnime do formulára, kliknutím myši alebo ťahaním myši vo formulári.

Najčastejšie budeme používať komponenty tlačidlo (Button), obrázok (Image), editovacie políčko (Edit), textová plocha (Memo), zaškrtávacie políčko (CheckBox).

Formulár predstavuje budúce okno našej aplikácie. Tak, ako teraz komponenty umiestnime do formulára, tak bude vyzerať naša aplikácia po spustení.

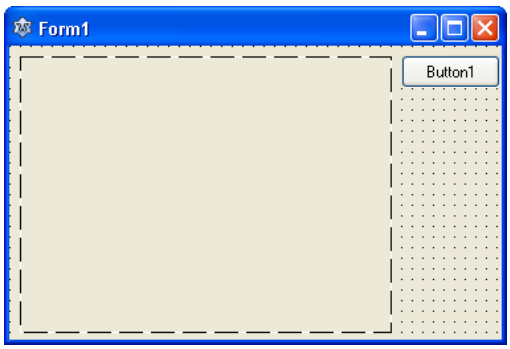
Teraz vložíme do formulára komponenty *Button* (**tlačidlo**) a *Image* (**obrázok**). Tlačidlom budeme spúšťať príkazy programu. Obrázok bude slúžiť ako výstup programu.

Komponent Button (tlačidlo) zo záložky Standard.



Komponent Image (obrázok) zo záložky Additional

Komponenty, ktoré sú už položené vo formulári môžeme pomocou myši presúvať a meniť im veľkosť.



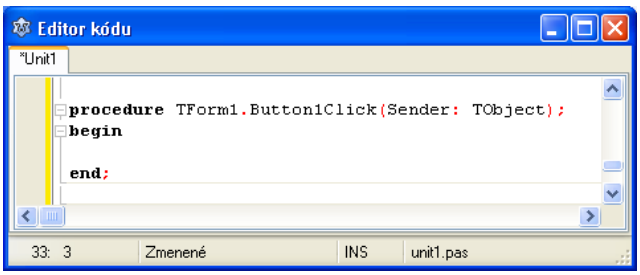
Formulár s tlačidlom a obrázkom

Aplikáciu spustíme. Ako sa správa tlačidlo v spustenom programe? Po spustení programu môžeme na tlačidlo klikať, ale zatiaľ sa nič nestane. Ak chceme, aby program niečo zmysluplné vykonal, musíme začať programovať.

Prvý program

V okne s programom je už napísaný kúsok programu, ktorý vygenerovalo vývojové prostredie automaticky. Postupne sa dozvieme, čo jednotlivé slová znamenajú. Teraz však zrealizujme tieto pokyny:

1. Vo formulári dvojklikneme na tlačidlo (pozor, nie v bežiacom programe).
2. Zobrazí sa **okno s programom**, pričom sa doň automaticky dopísal ďalší text.



Zmena v okne s programom

3. Do riadka, kde bliká kurzor, medzi slová **begin** **end**, napíšeme **príkaz**:

```
Image1.Canvas.TextOut(100, 50, 'Ahoj');
```

↑*bodka* ↑*apostrof* ↑*bodkočiarka*

4. Program spustíme a **v okne bežiaceho programu** stlačíme tlačidlo.

Príkaz, ktorý sme dopisovali do programu, je súčasťou takzvanej **procedúry**:

```
procedure TForm1.Button1Click(Sender: TObject);
```
hlavička

```
begin
  Image1.Canvas.TextOut(100, 50, 'Ahoj');
end;
```
telo

Procedúru môžeme zatiaľ chápať ako zoskupenie príkazov. V zápise procedúry budeme rozlišovať dve časti - **hlavičku** a **telo**:

- Hlavičke procedúry zatiaľ nebudeme úplne rozumieť. Vidíme však, že hlavička začína slovom **procedure**. Ďalej v nej rozpoznáme slovo `Button1Clik` - to je názov procedúry.
- Slová **begin** a **end** vymedzujú telo procedúry (*begin* = *začiatok*, *end* = *koniec*). Do riadkov medzi **begin** a **end** budeme vpisovať aj ďalšie príkazy.

Prázdnu procedúru `Button1Click` vygenerovalo vývojové prostredie automaticky. Stalo sa tak vtedy, keď sme dvojklikli na tlačidlo vo formulári (viď. 1. bod postupu). Týmto úkonom sme zároveň zabezpečili, že príkazy z procedúry `Button1Click` sa vykonajú vtedy, keď v spustenom programe stlačíme tlačidlo.

Komponenty, ktoré vkladáme do formulára dostávajú svoje jednoznačné mená. Napríklad obrázok dostal meno Image1, tlačidlo Button1.

Príkaz `Image1.Canvas.TextOut(100, 50, 'Ahoj')` zobrazí text, preto ho budeme nazývať **príkaz pre výpis textu**:

- Príkaz čítame takto *„obrázok 1, do svojej grafickej plochy napíš na súradnice 100, 50 text 'Ahoj'“*.
- Zátvorky vymedzujú **parametre** príkazu - *x*-ovú, *y*-ovú pozíciu a *text*.
- Text píšeme medzi apostrofy tak, ako píšeme priamu rľeč do úvodzoviek.

Poznámka: pri práci v prostredí Lazarus sa v našom spustenom programe zafarbí plocha na čierno. Ak chceme, aby mal obrázok biele pozadie, musíme ho zmazať:

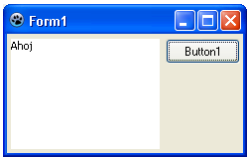
1. Dvojklineme do formulára (keď náš program nie je spustený),
2. V okne s programom vznikne ďalšia procedúra, do ktorej vpíšeme príkaz:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Image1.Canvas.FillRect(Image1.ClientRect);
end;
```

Grafickú plochu budeme takto mazať aj v budúcnosti, keď budeme v prostredí Lazarus vytvárať grafické projekty.

Počítačová súradnicová sústava je iná, ako sme boli zvyknutí v matematike:

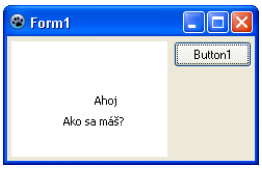
```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.TextOut(0, 0, 'Ahoj');
end;
```



Ak takýto program znovu spustíme, uvidíme, že bod so súradnicami 0, 0 leží v ľavom hornom rohu obrázka. Os *x*-ová rastie smerom vpravo, *y*-ová smerom nadol.

Do grafickej plochy môžeme vypisovať na rôzne miesta aj viac textov:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.TextOut(80, 50, 'Ahoj');
  Image1.Canvas.TextOut(50, 70, 'Ako sa máš?');
end;
```



V jazyku Pascal oddeľujeme dva príkazy `;` (bodkočiarkou). Hoci za posledným príkazom je bodkočiarka zbytočná, budeme ju tam písať pre prípad, že by sme chceli na koniec tela procedúry dopisovať ešte ďalšie príkazy.

Typy hodnôt

Prvé dva parametre v príkaze výpisu musia byť **celé čísla**. Tretí, posledný parameter musí byť text, presnejšie, textový **reťazec**. V zápisoch v programe odlíšime textové reťazce od čísel alebo príkazov tým, že znaky textového reťazca napíšeme medzi apostrofy.

Čísla a texty sú pre počítač rozdielne **typy údajov**. S každým vykonávame iné úkony, inak s nimi manipulujeme. Preto napríklad v príkaze výpisu nemôžeme na mieste textového parametra napísať číslo:

- **nesprávne:** `Image1.Canvas.TextOut(0, 0, 123)` ... `123` je číslo
- **správne:** `Image1.Canvas.TextOut(0, 0, '123')` ...`'123'` je reťazec

Niekedy však budeme chcieť zobraziť výsledky výpočtov:

- **toto je nesprávne:** `Image1.Canvas.TextOut(0, 0, 5*4+1);`
- **zobrazí `5*4+1`:** `Image1.Canvas.TextOut(0, 0, '5*4+1');`
- **vypíše 21:** `Image1.Canvas.TextOut(0, 0, IntToStr(5*4+1));`

`IntToStr` je skratka zo slov *Integer To String* = zmeň *Celé číslo Na Reťazec*. Zmena prebehne v pamäti počítača, kde sa najprv vyhodnotí aritmetický výraz a potom sa z jeho výsledku (čiže z čísla `21`) vytvorí postupnosť cifier, teda reťazec `'21'`. Hodnota niektorých výrazov však môže byť aj **desatinné** číslo:

- **nesprávne:** `Image1.Canvas.TextOut(0, 0, IntToStr(1/2));`
- **správne:** `Image1.Canvas.TextOut(0, 0, FloatToStr(1/2));`

Pomocou `FloatToStr` zmeníme desatinné číslo na text - z čísla `0.5` vznikne `'0.5'`. `FloatToStr` je skratka zo slov *Floating point number To String* = zmeň *Číslo s desatinnou bodkou Na Reťazec*.

Aj desatinné a celé čísla sú dva rozdielne typy. Hodnota výrazu, v ktorom sa delí pomocou `/`, bude vždy desatinné číslo. A to aj v vtedy, keby sme delili `10/2`:

- **nesprávne:** `Image1.Canvas.TextOut(0, 0, IntToStr(10/2));`
- **správne:** `Image1.Canvas.TextOut(0, 0, FloatToStr(10/2));`

V jazyku Pascal sa rozlišuje desatinné a celočíselné delenie. Celočíselné delenie sa realizuje pomocou operátora **div**:

- **vypíše 5:** `Image1.Canvas.TextOut(0, 0, IntToStr(10 div 2));`
- **vypíše 3:** `Image1.Canvas.TextOut(0, 0, IntToStr(10 div 3));`

S celočíselným delením súvisí aj zvyšok, ktorý vznikne po delení dvoch celých čísel. Ten vieme zistiť pomocou operátora **mod**:

- **vypíše 1:** `Image1.Canvas.TextOut(0, 0, IntToStr(3 mod 2));`
- **vypíše 4:** `Image1.Canvas.TextOut(0, 0, IntToStr(14 mod 5));`

V programoch tiež môžeme používať niektoré matematické funkcie. Napríklad `sqrt` počíta **odmocninu** z čísla. Výsledok bude desatinné číslo:

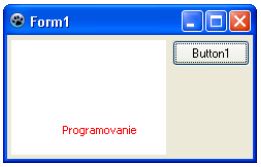
- `Image1.Canvas.TextOut(0, 0, FloatToStr(sqrt(64)));`
- `Image1.Canvas.TextOut(0, 0, FloatToStr(sqrt(5*5-4*4)));`

Kapitola 2: Kreslenie geometrických útvarov

Naučíme sa príkazy, ktoré nám umožnia kresliť rôzne geometrické útvary, akými sú kruhy, štvorce alebo úsečky a naučíme sa pracovať s farbami.

Písma

Texty, ktoré píšeme do grafickej plochy, môžeme písať farebne a rôznymi písmami. Napríklad, takto vypíšeme červený text:




```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Font.Color:=clRed;
  Image1.Canvas.TextOut(50, 80, 'Programovanie');
end;
```

V programe sme použili pre nastavenie farby písma nový **príkaz priradenia**. Tento príkaz obsahuje symboly `:=` (dvojbodka rovná sa). Na pravej strane od `:=` je názov červenej farby `clRed` (`cl` = skratka zo slova **color**, `Red` = červená). Príkaz pre nastavenia farby môžeme čítať nasledovne: *„obrázok 1, tvoja grafická plocha, tvoje písmo, jeho farba nastav na červenú“*. Základné farby sú pomenované: `clBlack`, `clRed`, `clGreen`, `clBlue`, `clWhite`, `clYellow`, ...

Mená ďalších farieb:
`clAqua`, `clCream`, `clDark`, `clDkGray`, `clFuchsia`, `clGray`, `clLime`, `clLtGray`, `clMaroon`, `clMedGray`, `clNavy`, `clOlive`, `clPurple`, `clSilver`, `clSkyBlue`

Ďalšími príkazmi môžeme nastaviť typ alebo veľkosť písma:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Font.Color:=clRed;
  Image1.Canvas.Font.Name:='Times New Roman';
  Image1.Canvas.Font.Height:=24;
  Image1.Canvas.TextOut(30, 80, 'Programovanie');
end;
```

Vlastnosti písma, a neskôr uvidíme, že aj iné vlastnosti, nastavujeme pomocou príkazu priradenia takto:

meno vlastnosti `:=` *nová hodnota*

Poradie príkazov

Poradie príkazov v programe býva veľmi dôležité. Skúsme zmeniť poradie príkazov:

```
  Image1.Canvas.TextOut(200, 100, 'Programovanie');
  Image1.Canvas.Font.Color:=clRed;
```

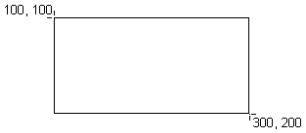
Akou farbou sa vypíše text po prvom a po druhom stlačení tlačidla? Po prvom sa vypíše text čiernou farbou, pretože farbu textu nastavujeme až potom, ako sa vykonal príkaz pre výpis textu. Nastavenie farby sa medzi stlačeniami tlačidla pamätá. Preto sa až po druhom stlačení vypíše text červenou farbou.

Používame viac tlačidiel

Do formulára môžeme vložiť ďalšie tlačidlo. Všimnime si, že prvé tlačidlo má nápis `Button1` a druhé `Button2`. Dvojkliknite na druhé tlačidlo. V texte programu pribudne procedúra `Button2Click`. Do nej budeme dopisovať príkazy, ktoré sa vykonajú, ak počas behu programu stlačíme druhé tlačidlo. Podobným spôsobom môžeme pridávať a reagovať na stlačenie ďalších tlačidiel.

Kreslenie obdĺžnikov a štvorcov

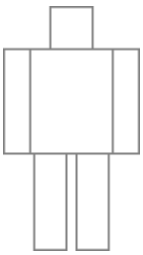
Do grafickej plochy sa dajú kresliť rôzne jednoduché geometrické útvary. Napríklad **obdĺžnik** nakreslíme takto:




```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Image1.Canvas.Rectangle(100, 100, 300, 200);
end;
```

Príkaz má 4 parametre. Sú to súradnice dvoch bodov, medzi ktoré sa nakreslí obdĺžnik. Prvé dve čísla (`100, 100`) sú súradnice prvého vrcholu, tretie a štvrté číslo (`300, 200`) sú súradnice protiľahlého vrcholu obdĺžnika. Obdĺžnik sa nakreslí tak, že bude mať strany rovnobežne so stranami grafickej plochy.

Nakreslite:

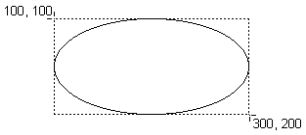


Štvorce tiež kreslíme pomocou príkazu na kreslenie obdĺžnikov. **Štvorec** je špeciálny prípad obdĺžnika, ktorého strany majú zhodnú dĺžku. Napríklad štvorec, so stranou dĺžky 100 nakreslíme takto:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Image1.Canvas.Rectangle(50, 20, 50+100, 20+100);
end;
```

Parametre môžu byť aj výrazy, ktoré sa najskôr vyhodnotia.

Kreslenie elíps a kruhov



Príkaz na kreslenie **elipsy** sa podobá príkazu na kreslenia obdĺžnika:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  Image1.Canvas.Ellipse(100, 100, 300, 200);
end;
```

Elipsa sa vpíše do pomyselného obdĺžnika, ktorého súradnice sme uviedli ako parametre príkazu.

Nakreslite:

Kruh je opäť špeciálny prípad elipsy. Ako nakreslíme kruh so stredom v bode 50, 50 a polomerom 10? Pre príkaz `Ellipse` musíme vypočítať súradnice dvoch protiľahlých vrcholov štvorca, do ktorého sa elipsa - vlastne už kruh, vpíše.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  Image1.Canvas.Ellipse(50-10, 50-10, 50+10, 50+10);
end;
```

Kreslenie úsečiek

Úsečku nakreslíme pomocou dvojice príkazov:

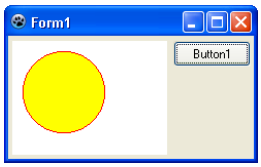
```
procedure TForm1.Button4Click(Sender: TObject);
begin
  Image1.Canvas.MoveTo(100, 50);
  Image1.Canvas.LineTo(200, 80);
end;
```

Príkaz `Image1.Canvas.MoveTo` presunie neviditeľné *grafické pero* do zadaného bodu. Príkaz `Image1.Canvas.LineTo` nakreslí s grafickým perom čiaru.

Farba obrysu a výplne

Pri kreslení obdĺžnikov a kruhov môžeme nastaviť farbu **výplne** (vnútra) a farbu **obrysu** (obvodu). Kruh so žltou výplňou a červeným obrysom nakreslíme takto:

```
procedure TForm1.Button5Click(Sender: TObject);
begin
  Image1.Canvas.Brush.Color:=clYellow;
  Image1.Canvas.Pen.Color:=clRed;
  Image1.Canvas.Ellipse(10, 10, 90, 90);
end;
```

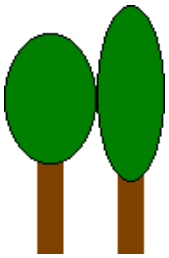


Vysvetlenie:

- `Pen.Color` znamená *farba pera* a určuje farbu čiaru, obrys útvaru
- `Brush.Color` znamená *farba štetca*, ktorou sa vymaľuje, vyplní plocha

Farba pera má vplyv aj pri kreslení úsečiek.

Nakreslite:



Náhodné čísla

Naše programy sa zatiaľ správali veľmi presne - vždy nakreslili rovnaký útvar, pretože súradnice geometrického útvaru sa nezmenili. Pozrime sa ale na tento príklad:

```
procedure TForm1.Button6Click(Sender: TObject);
begin
  Image1.Canvas.LineTo(Random(400), Random(300));
end;
```

Vidíme, že po stlačení tlačidla sa kreslí vždy iná úsečka. *Random* znamená *náhodný*:

- `Random(400)` zvolí niektoré číslo spomedzi čísel od `0` po `399`.
- `Random(300)` zvolí niektoré číslo spomedzi čísel od `0` po `299`.

Obidve náhodne zvolené čísla sa použijú ako súradnice bodu, do ktorého sa nakreslí úsečka (úsečka začína v bode (0,0)). Ak stlačíme tlačidlo druhý krát, vygenerujú sa iné náhodné čísla, takže sa nakreslí úsečka do iného koncového bodu.

Parameter, ktorý uvádzame pre **vygenerovanie náhodného čísla** udáva rozsah náhodných čísel: `Random(n)` náhodne zvolí číslo spomedzi čísel od `0` po `n-1`.

Aké čísla môže vypísať program?

```
procedure TForm1.Button7Click(Sender: TObject);
begin
  Image1.Canvas.TextOut(0, 0, IntToStr(1+Random(6)));
end;
```

`Random(6)` generuje čísla od 0 po 5. Ak k vygenerovanému číslu pripočítame 1, hodnota výrazu bude v rozsahu od 1 po 6.

Aké čísla sa vygenerujú:

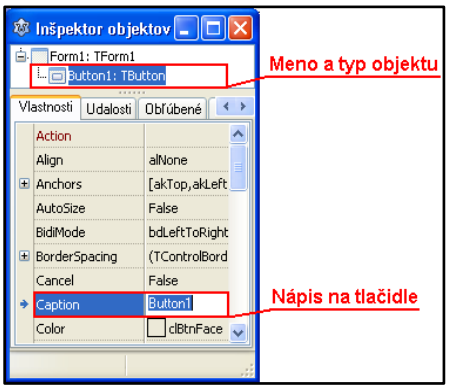
- `1+random(10)`
- `random(21)-10`

Ako vygenerujete čísla:

- od `0` po `100`
- od `20` po `30`
- od `-5` po `+5`?

Kapitola 3: Komponenty a ich vlastnosti

Ak do formulára vložíme tlačidlo, pomocou myši nastavíme jeho polohu alebo veľkosť. Poloha aj veľkosť sú **vlastnosti** tlačidla. Tlačidlo má aj iné vlastnosti (*properties*). Napríklad, aj nápis na tlačidle `Button3` je vlastnosť, ktorá sa dá zmeniť. Môžeme to urobiť v okne *Inšpektor Objektov*.



V okne Inšpektor Objektov sú teraz zobrazené vlastnosti tlačidla `Button1`.

Každá vlastnosť má svoje **meno** a **hodnotu**. Napríklad vlastnosť:

- `Left` a `Top` určujú pozíciu tlačidla vzhľadom na okraje formulára
  - `Left` ... vzdialenosť od ľavého okraja (vodorovná, x-ová súradnica)
  - `Top` ... vzdialenosť od horného okraja (zvislá, y-ová súradnica).
- `Width` a `Height` určujú šírku a výšku tlačidla
- `Caption` je nápis na tlačidle.

Vlastnosť s menom `Caption` má hodnotu `Button1`. Túto hodnotu môžeme v *Inšpektore objektov* prepísať, napríklad na *„Klikni na mňa“*. Všimnime si aj, že pozícia tlačidla sa dá nastaviť dvomi spôsobmi: pomocou myši, ale aj pomocou *Inšpektora objektov*. Ak zmeníme tlačidlu vlastnosť `Left` na hodnotu `0`, tlačidlo sa presunie k ľavému okraju formulára.

Vlastnosti komponentov sa dajú meniť aj počas behu programu, pomocou príkazu priradenia:

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

Alebo aj:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Button2.Left:=random(400);
  Button2.Top:=random(300);
end;
```

K vlastnostiam komponentov pristupujeme pomocou `.` (bodky) takto:

```
komponent.vlastnosť
```

Vlastnosť môžeme nielen nastavovať, ale aj zisťovať jej hodnotu:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  Button3.Left:=Button3.Left+50;
end;
```

Na pravej strane od `:=` sa teraz nachádza zložitejší výraz, ktorý hovorí: *„k súčasnej x-ovej pozícii tlačidla pripočítaj 50“*. Výsledok bude číslo o 50 väčšie, ako je súčasná x-ová pozícia tlačidla. Toto číslo sa použije pre nastavenie vlastnosti `Left`. Preto tlačidlo poskočí a vzdiali sa od ľavého okraja formulára o 50 bodov.

Znamená to, že na pravej strane príkazu priradenia môže byť aj pomerne komplikovaný výraz. Pri vykonaní príkazu priradenia program postupuje nasledovne:

1. Najskôr sa vyhodnotí výraz na *pravej strane* od `:=`.
2. Potom sa výsledná hodnota priradí *ľavej strane* od `:=`.

Na začiatku sme spomínali, že čísla a texty sú rôzne typy údajov. Musíme na to pamätať aj pri práci s vlastnosťami:

- číselné hodnoty: `Button1.Left:=100;`
- textové hodnoty: `Button1.Caption:='Nový nadpis';`
- farby: `Image1.Canvas.Font.Color:=clRed;`

Rôzne typy údajov môžeme chápať ako rôzne množiny. Tie majú v jazyku Pascal svoje pomenovanie:

- celé číslo je typu (je z množiny) `Integer`
- desatinné číslo: `Real`
- textový reťazec: **`string`**
- farba: `TColor`
- obrázok: `TImage` ... aj obrázky a tlačidlá sú pre počítač údaje!
- tlačidlo: `TButton`

Pomocou komponentu **editovacie políčko** *(Edit)* zadávame vstupné údaje programu:

Editovacie políčko na palete komponentov

Editovacie políčko funguje ako veľmi jednoduchý textový editor.

1. Vložme jedno editovacie políčko do formulára.
2. Ak program spustime, môžeme do editovacieho políčka písať krátke texty.
3. Zatvorme a ukončime bežiaci program.

Vidíme, že po spustení programu už je v políčku napísaný text. Ten môžeme zmeniť:

1. Vo formulári označíme editovacie políčko (klikneme naň).
2. V inšpektore objektov nájdeme vlastnosť `Text`
3. Zmeníme text vo vlastnosti `Text` na `Zadaj svoje meno.`

Text, ktorý napísal používateľ do editovacieho políčka, zistíme z vlastnosti `Text`. Tá nadobúda hodnotu typu **`string`** (textový reťazec), ktorú môžeme v programe jednoducho zistiť a vypísať:

```
procedure TForm1.Button4Click(Sender: TObject);
begin
  Image1.Canvas.TextOut(10, 10, 'Dobrý deň');
  Image1.Canvas.TextOut(70, 10, Edit1.Text);
end
```

Kapitola 4: Premenné

Pri počítaní na kalkulačke zvykneme požívať pamäť kalkulačky M+ na to, aby sme do nej odložili dočasný výsledok. Neskôr, keď chceme odloženú hodnotu použiť, vyvoláme ju z pamäti pomocou tlačidla MR. Aj v programoch si budeme potrebovať zapamätať výsledky výpočtov, ktoré neskôr použijeme. A budeme si tiež musieť pamätať aj veľa ďalších údajov, ktoré v programe spracovávame. V programovacích jazykoch sa na tento účel používajú premenné.

Premennou v programovaní rozumieme miesto v pamäti počítača:

- do premennej, čiže do pamäťového miesta, môžeme údaje **zapísať**,
- posledná zapísaná hodnota sa v premennej uchová, **pamätá** sa,
- zapamätanú hodnotu môžeme neskôr využiť, prečítať, **zistiť**.

Vidíme, že v programovaní má pojem *premenná* iný význam, ako v matematike. V matematike používame pojem *premenná* na označenie parametra alebo neznámej hodnoty. Napríklad, dĺžka úsečky je a, obsah štvorca $S=a^2$ alebo hľadáme riešene rovnice $1 + x = 3$. Avšak premenné v matematických výrazoch a v programoch majú spoločné to, že nám umožňujú popísať všeobecnejšie pravidlá alebo platné vzťahy.

V jazyku Pascal môžeme používať veľa premenných (bežná kalkulačka má iba jedno pamäťové miesto M). Pred použitím treba premennú najskôr **zadeklarovať**. Deklarácia premennej je zápis, v ktorom prezradíme:

- **meno premennej** = ako premennú pomenujeme, ako sa bude volať,
- **typ premennej** = aké hodnoty, z akej množiny môže premenná nadobúdať.

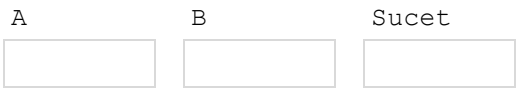
Premenné deklarujeme pred slovom **`begin`**. Takto zadeklarujeme tri premenné:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  A, B, Sucet: Integer;
begin

end;
```

Deklaráciou sme vyhradili v pamäti počítača miesto pre tri celočíselné premenné:

- deklarácia začína slovom **`var`** (skratka zo slova *variable* = *premenná*)
- premenné majú svoje mená `A`, `B`, `Sucet`
- každá z nich, teda aj premenná `A` si dokáže zapamätať jedno ľubovoľné celé číslo - hovoríme, že premenné `A` *je typu celé číslo*.

Obsah premenných budeme znázorňovať pomocou pomenovaných škatuliek:



Premennej nastavíme novú hodnotu pomocou príkazu priradenia:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  A, B, Sucet: Integer;
begin
  A:=2;
  B:=A+3;
  Sucet:=A+B;
  Image1.Canvas.TextOut(0, 0, IntToStr(Sucet));
end;
```

Premenným sme postupne priradili hodnoty:

```
A:=2;
```

A	B	Sucet
2		

```
B:=A+3;
```

A	B	Sucet
2	5	

```
Sucet:=A+B;
```

A	B	Sucet
2	5	7

Do grafickej plochy nakreslíme štvorce so stranou, takej veľkosti, akú zadal používateľ do editovacieho políčka:

```
procedure TForm1.Button2Click(Sender: TObject);
var
  A: Integer;
begin
  A:=StrToInt(Edit1.Text);
  Image1.Canvas.Rectangle(0, 0, A, A);
end;
```

Do celočíselnej premennej smieme priradiť iba celé číslo (t.j. hodnotu typu `Integer`). Preto text z editovacieho políčka zmeníme na číslo pomocou `StrToInt`.

Na náhodnej pozícii v grafickej ploche nakreslíme kruh s polomerom `10`:

```
procedure TForm1.Button3Click(Sender: TObject);
var
  X, Y: Integer;
begin
  X:=random(Image1.Width);
  Y:=random(Image1.Height);
  Image1.Canvas.Brush.Color:=clRed;
  Image1.Canvas.Ellipse(X-10, Y-10, X+10, Y+10);
end;
```

Premenné X a Y nazývame **lokálne premenné**. V pamäti počítača existujú iba počas vykonávania procedúry `Button3Click`. Aj preto príkazy z tejto procedúry nedokážu používať lokálnu premennú `A` z procedúry `Button2Click`. Lokálne premenné sú **viditeľné** iba pre príkazy tej procedúry, v ktorej sú deklarované.

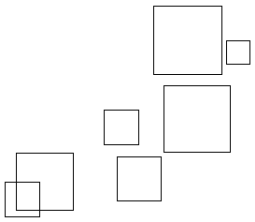
Premennú môžeme deklarovať aj mimo tela procedúry. Nasledujúci program počíta kliknutia na tlačidlo:

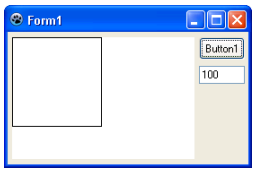
```
var
  N: Integer=0;

procedure TForm1.Button4Click(Sender: TObject);
begin
  N:=N+1;
  Button4.Caption:=IntToStr(N);
end;
```

Premenná `N` je **globálna premenná**. Výhodou globálnej premennej je, že existuje a pamätá si nastavenú hodnotu počas behu celého programu. Globálna premenná je viditeľná aj pre príkazy v procedúrach, ale iba od miesta svojej deklarácie. Navyše, globálnej premennej môžeme pri deklarácii nastaviť počiatočnú hodnotu.

Používaniu globálnych premenných sa však snažíme čo možno najviac vyhýbať. Je to preto, lebo globálne premenné sú do určitej miery nebezpečné - do globálnej premennej môže zasahovať príliš veľa príkazov z rôznych procedúr, a tak môžu jej obsah neprehľadným spôsobom meniť.

Nakreslite na náhodné miesto štvorec náhodnej veľkosti:





Tip: `Width` (šírka) a `Height` (výška) sú vlastnosťami obrázka, ktoré vieme nastavovať aj v programe a vieme sa na ne v programe aj odvolávať.

Preto príkaz:
`X:=random(Image1.Width)`
vygeneruje náhodnú x-ovú súradnicu v rozsahu od 0 po šírku obrázka-1.

Vypíšte na náhodné miesto tieňovaný nápis:

Napis s tienom

Tip: ak chceme, aby sa texty písali bez bieleho pozadia, vypneme výplň:
`Brush.Style:=bsClear`

Tento príkaz použijeme aj vtedy, keď chceme kresliť útvary bez výplne.

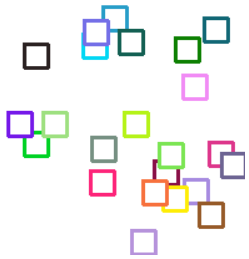
Kapitola 5: Príkaz cyklu

Jednu úsečku s náhodne umiestneným koncom sme kreslili takto:

```
Image1.Canvas.LineTo(Random(400), Random(300));
```

Ak chceme nakresliť viac, napríklad 3 úsečky, môžeme príkaz skopírovať:

Nakreslite na náhodné miesta zadaný počet štvorcov náhodnej farby



```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.LineTo(Random(400), Random(300));
  Image1.Canvas.LineTo(Random(400), Random(300));
  Image1.Canvas.LineTo(Random(400), Random(300));
end;
```

Keby sme chceli kresliť 100 úsečiek, museli by sme veľa kopírovať. Pritom by sme sa nesmeli pomýliť v počte skopírovaných príkazov. Už tušíme, že takýto spôsob vytvárania programov nie je ani elegantný ani efektívny. Pritom potrebujeme dosiahnuť, aby sa v programe opakovane vykonal rovnaký príkaz.

Keby sme chceli nakresliť 3 úsečky, slovne by sme algoritmus zapísali nasledovne:

```
opakuj 3 krát príkaz:
  Image1.Canvas.LineTo(Random(400), Random(300));
```

Cyklus `for`

V jazyku Pascal by sme predchádzajúci algoritmus zapísali takto:

Tip: Vlastnosti grafického pera `Color` môžeme priraďovať aj číselné hodnoty. Na vygenerovanie náhodnej farby môžeme použiť napríklad aj príkaz

```
Image1.Canvas.Pen.color:=random(256*256*256);
```

```
procedure TForm1.Button2Click(Sender: TObject);
var
  I: Integer;
begin
  for I:=1 to 3 do
    Image1.Canvas.LineTo(Random(400), Random(300));
end;
```

V programe najskôr deklarujeme celočíselnú premennú `I`. Tá bude slúžiť ako počítadlo, koľko úsečiek sa nakreslilo. Potom v programe použijeme **konštrukciu cyklu**, ktorá zabezpečí opakované kreslenie úsečky.

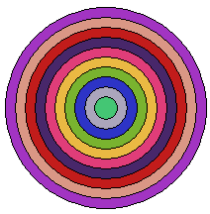
V konštrukcii cyklu **`for`** rozpoznávame dve časti - **hlavičku** a **telo cyklu**:

```
  for I:=1 to 3 do
```
hlavička

```
    Image1.Canvas.LineTo(Random(400), Random(300));
```
telo

Príkaz čítame: *„nastavuj premennú `I` postupne na hodnoty od 1 po 3 a vykonávaj kreslenie náhodnej úsečky“*.

Kreslite na náhodné miesta sústredné kružnice vyplnené náhodnou farbou



Keby sme v hlavičke cyklu zmenili číslo 3 na 100, nakreslilo by sa 100 úsečiek.

Premennú `I` nazývame **riadiacou premennou** cyklu. Hodnota v premennej `I` sa postupne zväčšuje o `1`. Môžeme to vidieť aj na nasledujúcom príklade, v ktorom postupne vypíšeme do grafickej plochy čísla od `1` po `10`.

```
procedure TForm1.Button3Click(Sender: TObject);
var
  I: Integer;
begin
  for I:=1 to 10 do
    Image1.Canvas.TextOut(0, I*20, IntToStr(I));
end;
```

Hlavička cyklu

V hlavičke cyklu napíšeme meno riadiacej premennej, ktorá sa bude postupne zväčšovať o `1`. Uvedieme **hranice cyklu**, teda počiatočnú hodnotu, ktorá sa priradí do riadiacej premennej a koncovú hodnotu, po ktorú sa bude obsah riadiacej premennej zväčšovať.

```
for premenná := počiatočná to koncová do príkaz;
```

Ak by počiatočná hodnota bola väčšia ako koncová hodnota, príkaz z tela cyklu by sa ani raz nevykonal.

Nakreslite pyramídu z rôznofarebných obdĺžničkov

Telo cyklu

Telo cyklu môže obsahovať aj viac príkazov:

```
procedure TForm1.Button4Click(Sender: TObject);
var
  I: Integer;
begin
  for I:=1 to 10 do begin
    Image1.Canvas.TextOut(0, I*20, IntToStr(I));
    Image1.Canvas.TextOut(100, I*20, IntToStr(I*I));
  end;
end;
```

Slová **begin** a **end** fungujú ako jeden **zložený príkaz** alebo programové zátvorky. Označujú skupinu príkazov,, ktoré sa majú spoločne opakovať:

- na začiatku cyklu sa do premennej `I` priradí hodnota `1`
  - vykoná sa príkaz, ktorý vypíše číslo `1`,
  - potom sa vykoná príkaz, ktorý vypíše druhú mocninu čísla `1`,
- obsah premennej `I` sa zväčší o `1`, teda `I=2`
  - opäť sa vykoná príkaz pre výpis `I`, teda sa vypíše číslo 2
  - ďalej sa vykoná príkaz, ktorý vypíše druhú mocninu čísla, teda `4`
- obsah premennej `I` sa zväčši o 1, teda `I=3`, atď.

Skúste upraviť príklad s malou násobilkou tak, aby výpis obsahoval aj vedúci stĺpec a vedúci riadok s hodnotami premenných I a J.

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

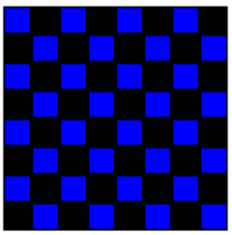
Cyklus v cykle

Ako zobrazíme malú násobilku - teda tabuľku s číslami:

		J=2		4	...	10
	1	2	3	4	...	10
	2	4	6	8	...	20
I=3	3	6	9	12	...	30
				...	...	...
	10	20	30	40	...	100

Tabuľka má 10 riadkov a 10 stĺpcov. Ak je v premennej `I` číslo riadku a v premennej `J` číslo stĺpca, výslednú hodnotu, ktorú vypisujeme, získame súčinom `I*J`.

Dokážete nakresliť šachovnicu?



```
procedure TForm1.Button5Click(Sender: TObject);
var
  I, J: Integer;
begin
  for I:=1 to 10 do
    for J:=1 to 10 do
      Image1.Canvas.TextOut(J*30, I*20, IntToStr(I*J));
end;
```

Telo cyklu môže v sebe obsahovať aj iný príkaz cyklu - tzv. **vnorený cyklus**.

Kapitola 6: Podmienený príkaz

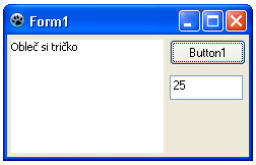
V živote často vyhodnocujeme rôzne situácie a rozhodujeme sa, akú činnosť ďalej vykonáme. Napríklad: ak je vonku teplo, vezmeme si tričko, inak si vezmeme sveter. Podobný spôsob rozhodovania budeme používať aj v našich programoch.

Skúsme napísať program, ktorý nám podľa zadanej teploty `t` poradí, ako sa obliecť. Potrebovali by sme, aby program pracoval nasledovne:

```
ak t>22 potom zobraz text 'obleč si tričko'
inak zobraz text 'obleč si sveter'
```

Podmienený príkaz `if then else`

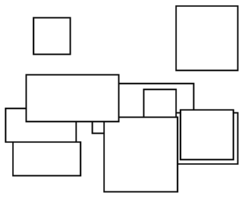
Predchádzajúci slovný zápis vieme priamo prepísať do jazyka Pascal:




```
procedure TForm1.Button1Click(Sender: TObject);
var
  t: Integer;
begin
  t:=StrToInt(Edit1.Text);
  if t>22 then Image1.Canvas.TextOut(0, 0, 'Obleč si tričko')
  else Image1.Canvas.TextOut(0, 0, 'Obleč si sveter');
end;
```

Pozor, pred vyhradeným slovom **`else`** nepíšeme `;` (bodkočiarku).

Kreslite náhodne buď štvorce alebo obdĺžniky



Ak program vyskúšame pre rôzne vstupné hodnoty a uvidíme, že:

- Text *„Obleč si tričko“* sa vypíše pre čísla `23, 24, ...` alebo aj `1 000 000`.
- Naopak, text *„Obleč si sveter“* sa vypíše pre čísla `22, 21, ...` alebo `-100`.

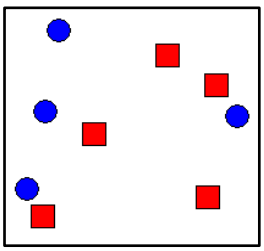
Programovú konštrukciu **`if ... then ... else ...`** nazývame **podmienený príkaz**:

```
if podmienka then príkaz1
else príkaz2;
```

V príkaze **`if then else`** vidíme dve **vetvy** - dve alternatívne možnosti, ktoré sa vykonajú v závislosti na tom, či je *podmienka* splnená alebo nie:

- ak je *podmienka* **splnená**, vykoná sa vetva za slovom **`then`** - $príkaz_1$,
- ak *podmienka* **nie je splnená**, vykoná sa vetva za slovom **`else`** - $príkaz_2$.

Ak potrebujeme v niektorej vetve vykonať viac príkazov, použijeme programové zátvorky **`begin`** a **`end`**. Napríklad, v nasledujúcom programe kreslíme náhodne buď červený štvorec alebo modrý kruh:

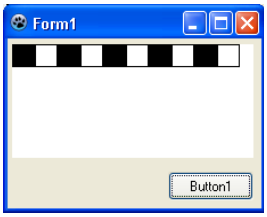


```
procedure TForm1.Button2Click(Sender: TObject);
var
  X, Y: Integer;
begin
  X:=Random(Image1.Width);
  Y:=Random(Image1.Height);
  if Random(2)=0 then begin
    Image1.Canvas.Brush.Color:=clRed;
    Image1.Canvas.Rectangle(X-10, Y-10, X+10, Y+10);
  end else begin
    Image1.Canvas.Brush.Color:=clBlue;
    Image1.Canvas.Ellipse(X-10, Y-10, X+10, Y+10);
  end;
end;
```

Kombinácia podmienených príkazov a cyklov

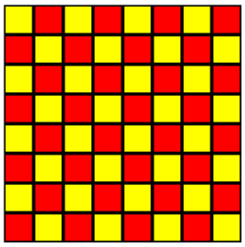
Nakreslíme 10 striedavo čiernych a bielych štvorcov:

```
procedure TForm1.Button3Click(Sender: TObject);
var
  I: Integer;
begin
  for I:=0 to 9 do begin
    if I mod 2=0 then Image1.Canvas.Brush.Color:=clBlack
    else Image1.Canvas.Brush.Color:=clWhite;
    Image1.Canvas.Rectangle(I*20, 0, I*20+20, 20);
  end;
end;
```



Pre párne `I` nastavujeme čiernu farbu výplne a pre nepárne `I` nastavujeme bielu farbu. Následne kreslíme štvorec na súradnice, ktoré odvodíme od hodnoty premennej `I`.

Je jednoduchšie nakresliť šachovnicu, keď poznáme príkaz IF?



Logické výrazy a logické hodnoty

Podmienka, ktorú píšeme v príkaze **`if`**, je v skutočnosti logický výraz. Ten môže nadobudnúť hodnotu buď **`true`** (pravda) alebo **`false`** (nepravda). Splnená podmienka má hodnotu **`true`** a nesplnená podmienka má hodnotu **`false`**.

Hodnoty **`true`** a **`false`** sú typu `Boolean`. `Boolean` je množina, ktorá obsahuje iba tieto dva prvky. Premenná typu `Boolean` si dokáže zapamätať výslednú hodnotu logického výrazu.

Zložené podmienky

Doposiaľ sme používali jednoduché podmienky, ktoré boli založené na porovnávaní hodnôt. Pomocou logických operátorov, ktoré poznáme aj z matematiky, môžeme podmienky skladať a vytvárať zložené podmienky.

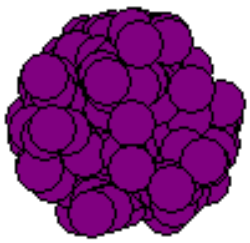
V jazyku Pascal používame operátory **`not`**, **`and`**, **`or`**:

vysvetlenie:	*zápis:*	*príklad:*	*výsledok:*
negácia `V`	**`not`** `V`	**`not`** `(1=2)`	**`= true`**
logické `X` a `Y`	`X` **`and`** `Y`	`(1=2)` **`and`** `(2<3)`	**`= false`**
logické `X` alebo `Y`	`X` **`or`** `Y`	`(1=2)` **`or`** `(2<3)`	**`= true`**

Na obrazovku vypíšeme všetky nepárne čísla do 100, ktoré sú deliteľné 7. Pre také číslo `N` platí predpis, že `(N mod 2<>0) and (N mod 7=0)`

```
procedure TForm1.Button4Click(Sender: TObject);
var
  N, Y: Integer;
begin
  Y:=0;
  for N:=1 to 100 do
    if (N mod 2<>0) and (N mod 7=0) then begin
      Image1.Canvas.TextOut(0, Y, IntToStr(N));
      Y:=Y+20;
    end;
end;
```

Kreslite krúžky na náhodných pozíciách v kruhovej oblasti



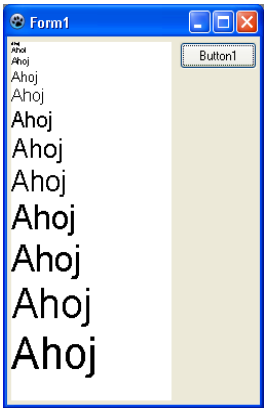
V cykle postupne skúšame čísla `N` od `1` po `100`. Vypíšeme tie, ktoré majú nenulový zvyšok po delení `2`, ale nulový po delení `7`. Ostatné čísla nás nezaujímajú, preto v programe používame príkaz **`if`** bez vetvy **`else`**. V premennej `Y` si pamätáme y-ovú súradnicu, kam vypíšeme výsledky.

Kapitola 7: Cyklus s podmienkou

Príkaz cyklu **for** používame vtedy, keď dopredu poznáme alebo vieme vypočítať hranice cyklu, počet opakovaní. Niekedy ich však nevieme jednoducho vypočítať alebo ani nevieme povedať, dokedy treba príkazy opakovane vykonávať.

Napríklad, chceme písať zväčšujúci sa text. Chcem však napísať iba toľko nápisov, koľko sa zmestí do grafickej plochy:

Budeme potrebovať dve premenné: v premennej `H` si pamätáme výšku písma a v premennej Y súradnicu, kam vypíšeme ďalší text. Algoritmus by sme mohli napísať takto:




```
H:=4;
Y:=0;
kým „sa text zmestí do grafickej plochy“ vykonávaj príkazy
  Image1.Canvas.Font.Height:=H;
  Image1.Canvas.TextOut(0, Y, 'Ahoj');
  Y:=Y+H;
  H:=H+4;
```

Ako zistíme, že sa text *zmestí* do grafickej plochy? Napríklad, podľa toho, že je splnená podmienka Y+H<=Image1.Height.

Potom predchádzajúci algoritmus zapíšeme v jazyku Pascal takto:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  H, Y: Integer;
begin
  Image1.Canvas.Font.Name:='Arial';
  H:=4;
  Y:=0;
  while Y+H<=Image1.Height do begin
    Image1.Canvas.Font.Height:=H;
    Image1.Canvas.TextOut(0, Y, 'Ahoj');
    Y:=Y+H;
    H:=H+4;
  end;
end;
```

Vykresľujte zväčšujúce sa štvorce, kým sa zmestia do grafickej plochy

V programe sme použili novú programovú konštrukciu - cyklus **while**:

```
  while podmienka do príkaz;
```

Cyklus **while** opakuje *príkaz*, kým platí *podmienka*. Telo cyklu, teda *príkaz*, môže byť jediný príkaz alebo skupina príkazov uzavretých medzi **begin** a **end**.

Nesprávne použitie cyklu **while** spôsobí, že cyklus nikdy neskončí. Stane sa tak napríklad vtedy, keď napíšeme za slovom **do** ; (bodkočiarku):

```
while Y+H<=Image1.Height do ; begin
```

V takom prípade by telo cyklu neobsahovalo žiaden príkaz, lebo bodkočiarkou sme ukončili prázdny príkaz. Preto by sa nemenila ani hodnota v premennej Y. Podmienka `Y+H<=Image1.Height` by zostala navždy splnená, takže cyklus by sa vykonával donekonečna. Vznikol by **nekonečný cyklus** a náš program by zamrzol.

Čo sme sa naučili v tomto module

Zhrnutie

V tomto module sme sa učili základy programovania v jazyku Pascal s tým, že sme využívali vývojové prostredie Delphi alebo Lazarus. Naučili sme sa príkazy pre kreslenie geometrických tvarov, vypisovanie textu alebo nastavovanie vlastností pomocou príkazu priradenia. Rozlišovali sme rôzne typy údajov: celé čísla, desatinné čísla, texty, logické hodnoty. Premenné sme používali na to, aby si program pamätal údaje, výsledky rôznych výpočtov alebo počet opakovaní. Okrem postupnosti príkazov sme sa naučili používať aj základné programové konštrukcie, ktorými riadime vykonávanie programu. Pomocou cyklov `for` a `while` sme zabezpečili, aby sa v programe niekoľkokrát opakovane vykonal príkaz alebo skupina príkazov. Podmienený príkaz `if` sme použili pri výbere alternatívy, ktorý z príkazov sa ďalej vykoná. Pritom sme sa snažili predviesť spôsob uvažovania, teda, ako rozmýšľame nad riešením problémov, a ako sa k výslednému riešeniu dopracujeme (často aj po malých krokoch).

Preverenie výstupných vedomostí

Účastník zostaví v prostredí Delphi alebo Lazarus funkčnú aplikáciu takéhoto typu:

Naprogramujte projekt, v ktorom budete pomocou tlačidla a editovacieho políčka kresliť obrázok lesa. Po stlačení tlačidla sa z editovacieho políčka načíta hodnota N a na náhodných pozíciách sa vykreslí N stromov. Korunu stromu vykresľujte ako zelenú elipsu a kmeň ako hnedý obdĺžnik. Veľkosti strán všetkých obdĺžnikov generujte náhodne z intervalu od 30 do 70.

Literatúra a použité zdroje

[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/
[3] Delphi - prednášky na FMFI UK, http://www.pascal.input.sk/
[4] prijímacie pohovory 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/
[8] Lazarus wiki, http://wiki.lazarus.freepascal.org/Main_Page/sk
[9] Free pascal, http://www.freepascal.org/

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

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

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

Podnázov Programovanie 2

Študijný materiál prešiel recenzným pokračovaním.

Recenzenti Mgr. Ján Skalka, PhD.
Mgr. Ján Gunis

Počet strán 20

Náklad 400 ks

Prvé vydanie, Bratislava 2009

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 978-80-8118-007-1