PROGRAMOVANIE

1.Vysvetlite pojem algoritmus, algoritmizácia, popíšte vlastnosti algoritmu.

Algoritmus = postup, ktorého realizáciou získame zo zadaných vstupných údajov po konečnom počte činností v konečnom čase správne výsledky.

- presný a jednoznačný predpis postupu riešenia úlohy,
- proces transformácie vstupných údajov na výsledok,
- postupnosť elementárnych krokov (operácií), ktoré sa vykonávajú v určitom poradí

Algoritmizácia - schopnosť aktívne vytvárať algoritmy určené pre *nemysliace zariadenia*

Vlastnosti algoritmu:

- **elementárnosť** - postup je zložený z činností, ktoré sú pre realizátora elementárne, zrozumiteľné (Napr. počítač nevie urobiť 4. mocninu, treba mu povedať aby urobil dvakrát druhú mocninu)
- **determinovanosť** - postup je zostavený tak, že v každom momente jeho vykonávania je jednoznačne určené, aká činnosť má nasledovať, alebo či sa postup už skončil
- **rezultatívnosť** - postup dáva pre rovnaké vstupné údaje vždy rovnaké výsledky (ak skončí).
- **konečnosť** - Postup skončí vždy v konečnom čase a po vykonaní konečného počtu operácií (Niektoré metódy sú teoreticky konečné a algoritmicky správne, môžu však trvať tak dlho, že sú prakticky nerealizovateľné).
- **hromadnosť** - postup je aplikovateľný pre celú triedu prípustných vstupných údajov (Program na výpočet priemeru z n čísel pracuje nad množinou čísel, nie nad konkrétnymi číslami.)
- **efektívnosť** - postup sa uskutočňuje v čo najkratšom čase a s využitím čo najmenšieho počtu prostriedkov

Zadanie 1:

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 správnu odpoveď.

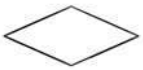
```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label2.Caption := 'Chúďa moje :(';
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Label2.Caption := 'Správna odpoveď, GRATULUJEM :)';
end;
```

2. Vysvetlite spôsoby vyjadrenia algoritmu, opíšte základné symboly vývojových diagramov.

Algoritmus môže byť vyjadrený viacerými spôsobmi:
verbálne, text, štruktúrogram, vývojový diagram, samotný programovací jazyk, rozhodovacie tabuľky, ...

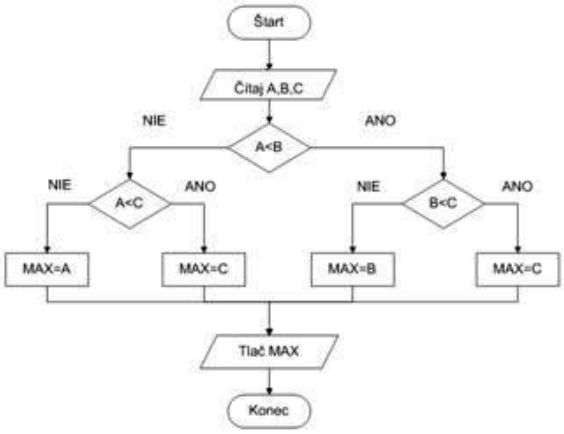
Zápis algoritmov:
a) prirodzeným jazykom, ale s kľúčovými (vyhradenými) slovami kvôli lepšej zrozumiteľnosti
b) vývojové diagramy

Symboly vývojových diagramov sú popísané v norme a majú nasledujúci tvar:

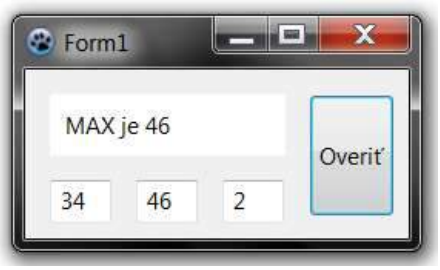
	Symbol predstavuje vstup (výstup) z vonkajšieho prostredia (do vonkajšieho prostredia). Definuje **začiatok** a **koniec** algoritmu.
	Symbol predstavuje akýkoľvek druh **spracovania** alebo **vykonania definovanej operácie** (skupiny operácií).
	Symbol predstavuje **rozhodovaciu** alebo **prepínaciu funkciu**. Má jeden vstup a dva alebo viac alternatívnych výstupov.
	Symbol predstavuje **vstupno-výstupné operácie** s údajmi.
	Symbol predstavuje volanie **čiastkového algoritmu** (podprogramu), ktorý môže napr. vykonávať nejaký výpočet.
	Symbol predstavuje viacnásobné vykonávanie operácií - **cyklus**.
	Tento symbol predstavuje **návestie** - prechod z jednej časti vývojového diagramu na inú časť. Používa sa na prerušenie spojnice a k jej pokračovaniu na inom mieste.
	Symbol v tvare zvislej alebo vodorovnej čiary predstavuje **tok údajov** alebo **riadenie** a slúži na spojenie jednotlivých symbolov

Zadanie 2:

Vysvetlite vývojový diagram a prepíšte ho do vášho programovacieho prostredia.




```
procedure TForm1.Button1Click(Sender: TObject);
var
  A,B,C,MAX : Integer;
begin
  A := StrToInt(Edit1.Text);
  B := StrToInt(Edit2.Text);
  C := StrToInt(Edit3.Text);

  if A<B then if B<C then MAX := C
                     else MAX := B
         else if A<C then MAX := C
                     else MAX := A;

  Image1.Canvas.TextOut( 10, 10, 'MAX je '+ IntToStr(MAX) );
end;
```



3. Vysvetlite pojmy: program, programovanie, programovací jazyk, popíšte typy programovacích jazykov.

Počítačový program je algoritmus prepísaný do programovacieho jazyka.

Programovacie jazyky slúžia na opis riešenia problému, ktorý umožní jeho vykonanie pomocou programovateľného technického zariadenia.

Programovanie je činnosť, pri ktorej programátor zadáva počítaču inštrukcie v jazyku, ktorý počítač pozná a dokáže ich vykonať.

Typy jazykov:

- ***strojovo orientované jazyky*** - program napísaný v strojovom jazyku je postupnosť elementárnych príkazov, ktoré nazývame inštrukcie a ktoré môže počítač priamo vykonávať (inštrukcia zodpovedá príkazu jazyka)
- ***vyššie programovacie jazyky*** - snažia sa odstrániť neštrukturovanosť, zvyšujú zrozumiteľnosť programu a zjednodušujú programovanie
- ***prekladače*** - umožňujú vykonávanie programov zapísaných v programovacom jazyku

Niektoré jazyky sú navrhnuté na ľubovoľné problémy, iné sú špecializované na problémy nejakej oblasti alebo len pre istý typ používateľov

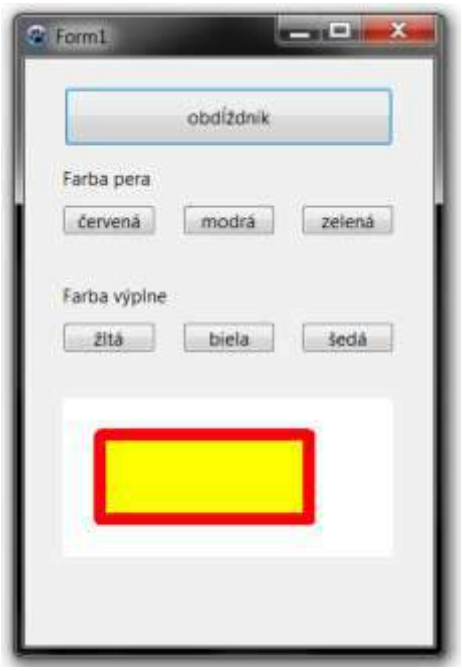
Rok	Jazyk	Charakteristika
1950	Jazyk assemblera	nižší programovací jazyk - symbolické pomenovanie strojových inštrukcií a adries
1957	Fortran	(Formula Translation) - jazyk na technické výpočty
1958	Algol	(Algorithmic language) - základ pre štruktúrované jazyky
1958	Lisp	(List Processing) - jazyk pre vedcov pracujúcich v oblasti umelej inteligencie a symbolických výpočtov
1964	Basic	(Beginers All Purpose Instruction Code) - úvodný jazyk inžinierov ako príprava na Fortran
1967	Logo	Jazyk pre deti - korytnačia grafika, práca so zoznamami
1970	Pascal	(pomenovaný na počesť B. Pascala) - určený ako úvodný jazyk na vyučovanie programovania
1972	C	Jazyk systémových programátorov
1980	Smalltalk 80	Čisto objektovo orientovaný jazyk - idea okien, práca s myšou
1983	Pascal	Štandard jazyka Pascal - neskôr sa rozšíril skoro na všetky univerzity sveta ako základný jazyk informatiky
1985	C++	Objektovo orientovaný jazyk založený na C. Momentálne najrozšírenejším jazykom systémových programátorov.
1991	Visual Basic	"= vizuálny (""windovskovský"") Basic. Je vhodný na tvorbu aplikácií pod Windows. Programy sú pomalé a preto je nevhodný pre náročné oblasti."
1992	Comenius Logo	korytnačia geometria
1994	Java	Jazyk Internetu. Je spoľahlivý a prenositeľný pod rôzne typy operačných systémov
1995	Delphi	"= ""windowsovský"" Pascal. Vizuálny objektovo orientovaný Pascal"
2001	Imagine	"""windowsovké objektové Logo"" - riadené udalosťami"

Zadanie 3:

V programe vytvorte 7 malých tlačidiel:

- **tri menia farbu pera na červenú, modrú a zelenú**
- **tri 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.**

```
procedure TForm1.Button1Click(Sender: TObject);
begin
   Image1.Canvas.Pen.Width := 10;
   Image1.Canvas.Pen.Color:=ClRed;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
   Image1.Canvas.Pen.Width := 10;
   Image1.Canvas.Pen.Color := clBlue;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
   Image1.Canvas.Pen.Width := 10;
   Image1.Canvas.Pen.Color:=ClGreen;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
   Image1.Canvas.Brush.Color:=ClYellow;
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
   Image1.Canvas.Brush.Color:=ClWhite;
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
   Image1.Canvas.Brush.Color:=ClGray;
end;

procedure TForm1.Button7Click(Sender: TObject);
begin
   Image1.Canvas.Rectangle(30, 30, 200, 100);
end;
```



4. Opíšte programátorské prostredie Lazarusu, ako objektového Pascalu.

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.
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, 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.
Súčasné moderné programátorské 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 bežiacej 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..

Editovacie okno (v pravo) slúži na zápis algoritmov.

Formulár 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. Do tohto formuláru sa rozmiestňujú a upravujú najrôznejšie súčiastky (komponenty).

Paleta komponentov (nad editovacím oknom) obsahuje predpripravené "súčiastky", z ktorých sa bude skladať okno našej vytváranej aplikácie. Táto paleta sa skladá z viacerých záložiek, napr.:

- zo štandardnej palety **Standard**

o súčiastka **jednoduchý text** – vnútorné meno TLabel
o súčiastka **editovací riadok** – vnútorné meno TEdit
o súčiastka **textová plocha** – vnútorné meno TMemo
o súčiastka **tlačidlo** – vnútorné meno TButton
o súčiastka **posúvač** – vnútorné meno TScrollBar

- z prídavnej palety **Additional**

o súčiastka **grafická plocha** – vnútorné meno TImage

- z prídavnej palety **System**

o súčiastka **časovač** – vnútorné meno TTimer

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

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

Zadanie 4:

Napíšte program s tlačidlami:

- **malé, veľké – menia veľkosť okna aplikácie (šírku a výšku okna formulára),**
- **nadpis1, nadpis2 – menia nadpis okna formulára,**
- **farba1, farba2 – menia farbu formulára,**
- **poloha1, poloha2 – menia polohu umiestnenia formulára.**

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Caption:='Form.1';
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Caption:='Formulár';
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  Color:= ClRed;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  Color:= ClYellow;
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
  Height:= 215;
  Width:= 231;
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
  Height:= 450;
  Width:= 500;
end;

procedure TForm1.Button7Click(Sender: TObject);
begin
  Left:= 500;
  Top:= 500;
end;

procedure TForm1.Button8Click(Sender: TObject);
begin
  Left:= 300;
  Top:= 400;
end;
```

5. Popíšte grafickú plochu v Lazaruse, vysvetlite syntax grafických príkazov (Rectangle, Ellipse, MoveTo, LineTo).

Grafická plocha, na ktorú môžeme kresliť, sa nachádza na komponente Image1, ktorý umiestníme vo formulári. Image1 je názov jedného konkrétneho objektu triedy TImage. Takých obrázkov by sme mohli mať na formulári aj viac (Image2, Image3, ...). Obrázok Image1 má v sebe plátno **Canvas**, na ktoré sa pomocou príslušných príkazov môže kresliť alebo písať. Preto, ak sme chceli kresliť obdĺžnik, museli sme programu oznámiť, že sa bude kresliť na obrázok Image1 a na jeho Canvas.
Grafická plocha má svoju výšku aj šírku. Tieto hodnoty sú uložené vo vlastnostiach Image1.Height (výška) a Image1.Width (šírka). Grafická plocha má súradnice, aby sme vedeli, kam vykresľovať. V ľavom hornom rohu je vždy bod [0,0]. X-ová aj Y-ová os sú kladné.

Image1.Canvas.Rectangle(10,100,150,200);
Príkaz **Rectangle** má 4 vstupné parametre, sú to celé čísla a označujú súradnice dvoch bodov. Prvé dve súradnice patria bodu v ľavom hornom rohu obdĺžnika a druhé dve čísla sú súradnice pravého dolného rohu obdĺžnika. Týmito dvoma bodmi je obdĺžnik jednoznačne určený.

Image1.Canvas.Ellipse(x1,y1,x2,y2);
Podobne funguje príkaz **Ellipse**, ktorý má rovnako 4 parametre, ktoré označujú ľavý horný a pravý dolný bod obdĺžnika, ktorý je elipse opísaný.

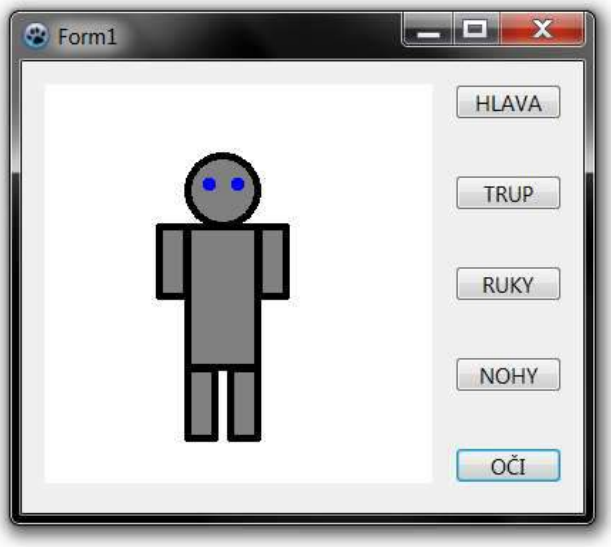
Image1.Canvas.MoveTo(x-ova_suradnica, y-ova_suradnica);
Image1.Canvas.LineTo(x-ova_suradnica, y-ova_suradnica);
Pomocou MoveTo presúvame po grafickej ploche pomyselné pero do zadaného bodu, bez toho aby sme kreslili. Príkaz LineTo slúži na kreslenie rovnej čiary z aktuálneho bodu do zadaného bodu. Oba príkazy majú ako vstup súradnice jedného bodu (tj. dve celé čísla).

- **farba pera**: Image1.Canvas.Pen.Color
- **hrúbka pera**: Image1.Canvas.Pen.Width
- **farba výplne:** Image1.Canvas.Brush.Color

Zadanie 5:

Pomocou tlačidiel (hlava, trup, ruky, nohy, časti tváre, ...) nakreslite robota.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
   Image1.Canvas.Pen.Width := 5;
   Image1.Canvas.Pen.Color := ClBlack;
   Image1.Canvas.Brush.Color := ClGray;
   Image1.Canvas.Ellipse( 100, 50, 100+50, 50+50);
end;

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

procedure TForm1.Button3Click(Sender: TObject);
begin
   Image1.Canvas.Rectangle( 80, 100, 80+20, 100+50);
   Image1.Canvas.Rectangle( 150, 100, 150+20, 100+50);
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
   Image1.Canvas.Rectangle( 100, 200, 100+20, 200+50);
   Image1.Canvas.Rectangle( 130, 200, 130+20, 200+50);
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
   Image1.Canvas.Pen.Width := 10;
   Image1.Canvas.Pen.Color := ClBlue;
   Image1.Canvas.MoveTo( 115, 70);
   Image1.Canvas.LineTo( 115, 70);
   Image1.Canvas.MoveTo( 135, 70);
   Image1.Canvas.LineTo( 135, 70);
end;
```



6. Vysvetlite na čo slúžia identifikátory, aké pravidlá spĺňajú a ako definujeme konštanty v Pascale.

Pomenovávanie konštánt, premenných, procedúr a aj vlastných typov realizujeme pomocou **identifikátorov**.
Identifikátor v jazyku Pascal slúži na pomenovávanie rôznych prvkov v programoch a má tieto pravidlá:

- skladá sa z písmen (len 26 písmen anglickej abecedy bez diakritiky), číslic a znaku _ podtržník
- nesmie začínať číslicou
- identifikátor sa musí líšiť od rezervovaných slov v pascale takých ako **begin**, **end**, **procedure**

Správne zapísané identifikátory:

XovaSuradnica,
Maximum,
Text1,
DlhyText,
VelkostTextu,

Nesprávne zapísané identifikátory:

Prvy Text,
2Text,
Polomer.Kruhu,
Veľkosť

Definovanie konštánt zapisujeme pred telo procedúry medzi riadky **procedure** a **begin**.
Konštanty môžeme zadefinovať menom konštanty a jej hodnotou pomocou slova **const**:

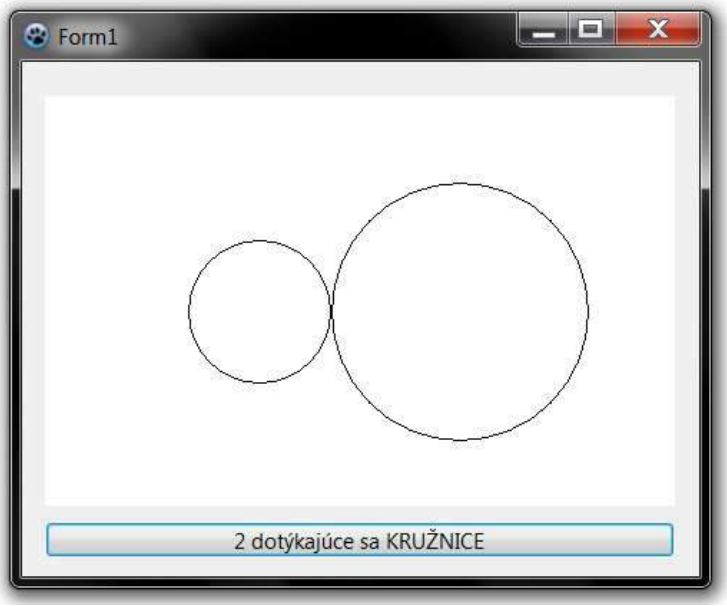
```
const
 Hrube = 10;
 Tenke = 1;
 Modra = clBlue;
 Velkost = 120;
```

Definície konštánt sú v rôznych procedúrach úplne nezávislé.

Zadanie 6:

Nakreslite dve dotýkajúce sa kružnice s polomermi R1 a R2, pričom stred prvej je v X, Y a druhá má rovnakú Y súradnicu (R1, R2, X, Y sú konštanty).

```
procedure TForm1.Button1Click(Sender: TObject);
const
  R1 = 50;
  R2 = 90;
  X = 150;
  Y = 150;
var
  X2 : Integer;
begin
  X2 := X+R1+R2;
  Image1.Canvas.Ellipse( X-R1, Y-R1, X+R1, Y+R1);
  Image1.Canvas.Ellipse(X2-R2, Y-R2, X2+R2, Y+R2);
end;
```



7. Definujte pojem premenná, vysvetlite pravidlá a spôsoby deklarovania premennej.

Premenné v programovaní sú nejaké vyhradené časti pamäte počítača, ktoré pomenujeme a môžeme si v nich uchovávať rôzne hodnoty. Vytvorenie premennej v pamäti počítača sa nazýva **deklarácia premennej**. Na špeciálnom mieste programu povieme, ako sa bude premenná nazývať a akého bude typu. Počítač podľa údajového typu premennej vyhradí miesto o určitej veľkosti a bude strážiť aké
operácie s danou premennou vykonávame.
Napr.:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  a: integer;
begin
...
end;
```

Po spustení takéhoto programu sa vytvorí v pamäti počítača miesto, ktoré je pomenované podľa názvu premennej a má veľkosť, ktorá vyplýva z typu premennej.

- Do premennej vieme **zapisovať** hodnotu,
- tá sa v premennej **uchováva**,
- a z premennej vieme hodnotu **prečítať**.

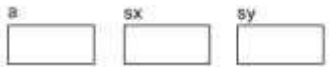
Slovom **var** označujeme miesto v programe, kde budú nasledovať deklarácie premenných. Var je skratka od variable (angl. premenná). Nachádza sa za hlavičkou procedúry, pred slovom begin, ktoré označuje začiatok tela procedúry.

Premenné, ktoré tu deklarujeme, môžeme používať v tele procedúry, ale ich existencia zanikne, keď procedúra skončí (toto miesto označuje slovo end). Ak by sme mali vo formulári viacero tlačidiel, premenná **a** je prístupná iba pre tlačidlo Button1.

Deklarácia premennej sa skladá z názvu premennej a, z dvojbodky a z typu premennej - v tomto prípade je to integer (celé číslo). Na konci je bodkočiarka.

Deklarácia premennej: názov_premennej: typ_premennej;

Premenné rovnakého typu môžeme písať za sebou. Oddelené sú čiarkami.

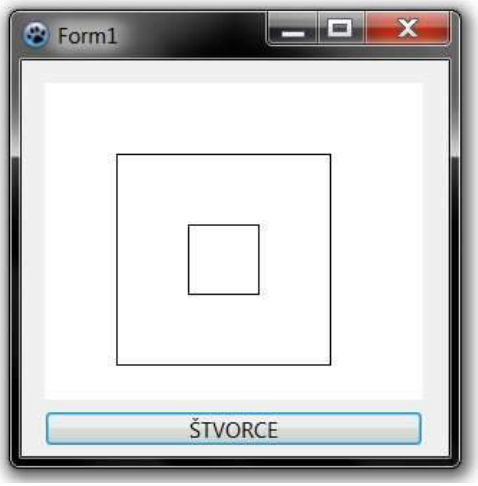


Deklarácia viacerých premenných:
názov_premennej1, názov_premennej2: typ_premennej;
Napr.: a, sx, sy: integer;

Zadanie 7:

Pre zadané premenné X, Y a A nakreslite štvorec, ktorého ľavý horný roh je v X,Y a veľkosť strany je A; potom v jeho vnútri presne v strede nakreslite štvorec s tretinovou veľkosťou strany – vypočítajte pritom premenné X3,Y3 (ľavý horný roh nového štvorca) a A3 veľkosť strany nového štvorca.
Overte zmenou hodnôt premenných X, Y, A.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  X, Y, A, A1, X1, Y1 : Integer;
begin
  X := 50;
  Y := 50;
  A := 150;
  A1 := ROUND(A/3);
  X1 := X+ROUND(A/3);
  Y1 := Y+ROUND(A/3);
  Image1.Canvas.Rectangle( X, Y, X+A, Y+A);
  Image1.Canvas.Rectangle( X1, Y1, X1+A1, Y1+A1);
end;
```



8. Vysvetlite ako priraďujeme premennej hodnotu, Popíšte syntax priraďovacieho príkazu v Pascale. Vysvetlite prioritu operácií vo výraze.

Priradenie hodnoty do premennej sa robí pomocou dvojice znakov **:=**.

Meno premennej je na ľavej strane, priraďovaná hodnota na pravej strane.

Priradenie do premennej: názov_premennej:= hodnota;

```
Image1.Canvas.Pen.Color:=clBlue;
Image1.Canvas.Pen.Width:=3;
Image1.Canvas.Font.Name:='Courier';
Image1.Canvas.Font.Color:=clBlue;
Image1.Canvas.Font.Size:=20;
```

Všetky z nich sú priradenia - do niektorej vlastnosti (property) priraďujeme konkrétnu hodnotu. Vlastnosti ako farba písma, alebo hrúbka pera sú už naprogramované a v pamäti počítača existuje pre ne už vopred vyhradené miesto. Nemusíme ich teda deklarovať, ale rovno s nimi pracujeme.

Ukážeme, ako budeme používať svoje vlastné premenné.
Zadeklarujeme premenné a priradíme do nich príslušné hodnoty.
napr.:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  a, sx, sy: integer;
begin
  a:=200;
  sx:=Image1.Width div 2;
  sy:=Image1.Height div 2;
end;
```

Premennej môžeme priradiť aj aritmetický výraz, alebo výsledok funkcie.
Pascal má presné pravidlá, ako vyhodnocuje výrazy:

- najprv sa vyhodnocujú časti, ktoré sú v okrúhlych zátvorkách
- potom sa vyhodnotia operácie s vyššou prioritou: *, div, mod
- nakoniec operácie s nižšou prioritou +, -
- ak sú vedľa seba operácie s rovnakou prioritou, tak sa vyhodnocujú zľava doprava

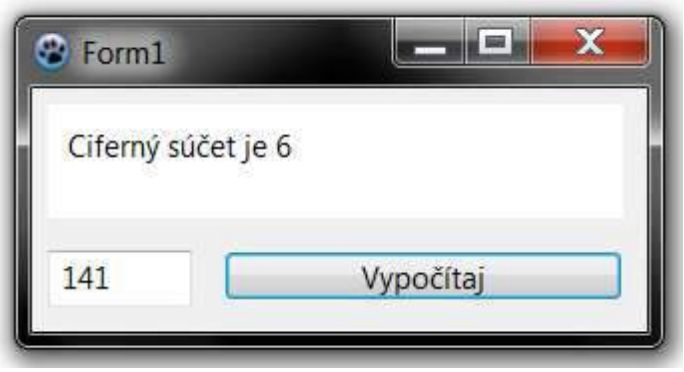
Napr.: -5 + 6 * 7 mod 4 = (-5) + ((6*7) mod 4) = -5 + 2 = -3

Výnimku ešte predstavuje *unárne mínus*, ktoré sa, samozrejme, uplatní bezprostredne k svojmu operandu.

Zadanie 8:

Pre zadané číslo zo vstupného riadka vypočítajte jeho ciferný súčet.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  CISLO,CISLICA,SUCET : Integer;
begin
  Image1.Canvas.FillRect(Image1.ClientRect);
  CISLO := StrToInt(Edit1.Text);
  SUCET := 0;
  repeat CISLICA := CISLO mod 10;
         SUCET := SUCET + CISLICA;
         CISLO := CISLO div 10
  until CISLO = 0;

  Image1.Canvas.TextOut(10,10, 'Ciferný súčet je '+ IntToStr(SUCET) );
end;
```



9. Definujte celočíselný typ premennej, popíšte operácie a štandardné funkcie na tomto type, vysvetlite riziká chýb v celočíselnej aritmetike.

Celočíselný typ je ***ordinálny*** (údajový typ medzi hodnotami ktorého je definované usporiadanie)

Typy celých čísel:

- *byte* - čísla z rozsahu 0 až 255
- *shortint* - čísla z rozsahu -128 až 127
- *word* - čísla z rozsahu 0 až 65535
- *integer* - čísla z rozsahu -32768 až 32767
- *longint* čísla z rozsahu -2147483648 až 2147483647

Pre prácu s výrazmi používame:
operácie porovnávania: *<, >, =, <=, >=, <>,*
celočíselné aritmetické operácie: **+**, **-**, *****, **div (**celočíselné delenie), **mod (**zvyšok pri celočíselnom delení).

14 **div** 5 = 2
14 **mod** 5 = 4

Napr. 14 **mod** 7 = 0 znamená, že 7 delí 14.

Štandardné funkcie pre celočíselný typ: **Abs** a **Sqr**.
Funkcia **Abs** počíta absolútnu hodnotu, napr. Abs(7-5) = Abs(5-7) = 2.
Funkcia **Sqr** počíta druhú mocninu čísla, napr. Sqr(3+4) = $(3+4)^2$ = 49.

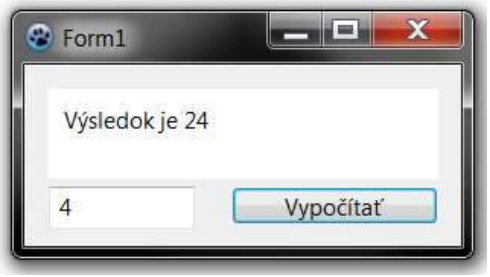
Konverzná funkcia **IntToStr** - dokáže prerobiť celé číslo na text
Parametrom konverznej funkcie **IntToStr** nemusí byť len premenná, ale aj nejaký výpočet.
Napr.
Image1.Canvas.TextOut(100, 100, **IntToStr(Sucet)**);
Image1.Canvas.TextOut(100, 130, **IntToStr(1+2+3+4+5)**);

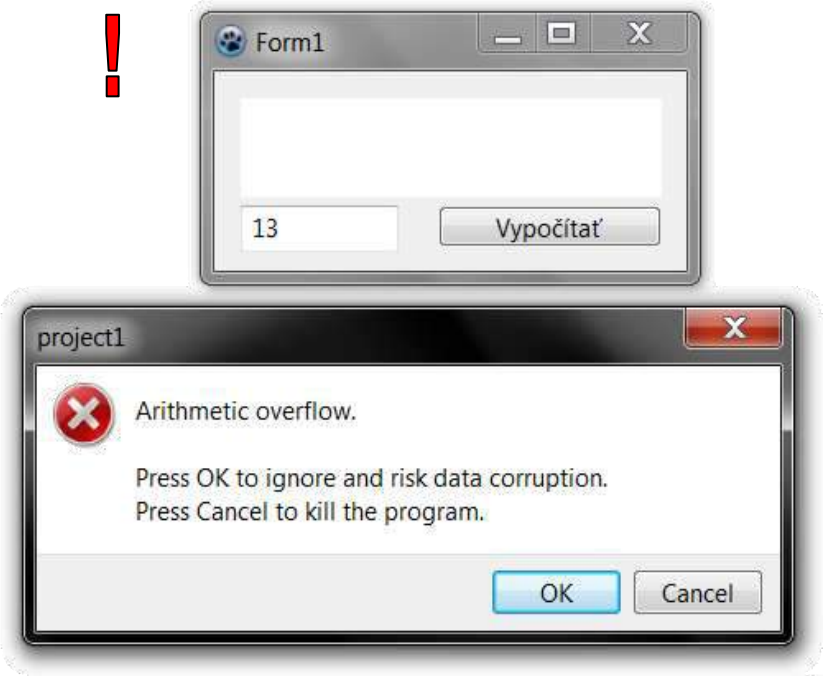
Chyby pri celočíselnej aritmetike:
Delenie nulou, ktoré spôsobí spadnutie programu:
Hodnota1 := 17 div (16 mod 4);
Pretečenie - vyplýva z obmedzení celočíselnej aritmetiky. Celé čísla sa v Delphi ukladajú do 4 bajtov (32 bitov) <- 2 147 483 648, 2 147 483 647>, približne ±2 miliardy.

Zadanie 9:

Naprogramujte výpočet N! a túto hodnotu vypíšte do grafickej plochy. (13! je už viac ako 6 miliárd, preto tento výpočet bude nesprávny). Nastavte automatickú kontrolu pretečenia a výpis chybovej správy.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  N,I,FAKT : Integer;
begin
  Image1.Canvas.FillRect(Image1.ClientRect);
  N := StrToInt(Edit1.Text);
  FAKT := 1;
  for I := 1 to N do
  FAKT := FAKT * I;

  Image1.Canvas.TextOut(10,10, 'Výsledok je '+ IntToStr(FAKT) );

end;
```





10. Vysvetlite spôsob generovania náhodných čísel pomocou funkcie Random, uveďte príklady jej využitia.

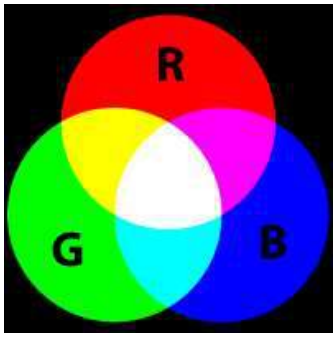
Niekedy potrebujeme, aby počítač vygeneroval **náhodné číslo**. V Pascale je pre tento účel pripravená funkcia **Random**.

Random je špeciálna funkcia, ktorá vytvára náhodné hodnoty. Funkcia umožňuje, aby počítač zakaždým, keď ju zavoláme, zvolil nejaké náhodné číslo.
Random(n) **vracia náhodné číslo z intervalu <0,n-1>**.
Môžeme vygenerovať **číslo z ľubovoľného intervalu <n,m>:** n + Random (m - n + 1)
Keď túto funkciu zavoláme viackrát, dostaneme rôzne výsledky.

Random(2)	{0, 1}
Random(3)-1	{-1, 0, 1}
Random(6)+1	{1, 2, 3, 4, 5, 6}
2*Random(2)-1	{-1, 1}
2*Random(4)+2	{2, 4, 6, 8}
Random(21)-10	<-10, 10>
Random(181)-90	<-90, 90>
Random(B-A+1)+A	interval <A, B>
Random(2)*(B-A)+A	dvojprvková množina {A, B}

Ak chceme generovať náhodné pozície grafickej plochy tak, aby sa zmestili do grafickej plochy, použijeme výpočet súradníc bodu:
X:=Random(Image1.Width);
Y:=Random(Image1.Height);

Farby sme zadávali takto: clRed, clYellow, clBlue, clBlack ...
Farba sa však dá vyjadriť aj inak - pomocou zložiek RGB (R=red - červená, G=green - zelená, B=blue - modrá). Každá farba sa dá napísať ako trojica čísel z intervalu <0,255>. Čísla vyjadrujú pomer základných farieb, z ktorých sa každá farba skladá.
Napríklad (0,0,0) vyjadruje čiernu farbu, (255,255,255) bielu farbu, (255,0,0) je červená, (100,100,100) je šedá.
Ak predpokladáme, že rôznych farieb v počítači je 256*256*256 = 16777216, tak by sme náhodnú farbu mohli generovať aj takýmto zápisom:
Random(256*256*256);



Pomocou funkcie RGBToColor môžeme ľahko vyrobiť v programe ľubovoľnú, hoci aj náhodnú farbu:
RGBToColor(Random(256),Random(256),Random(256))

Zadanie 10:

Nakreslite náhodný trojuholník, ktorého vrcholy sú od okrajov plochy vzdialené aspoň na 20. Vykreslite ho náhodnou hrúbkou od 5 do 10 a náhodnou farbou.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  X1,X2,X3,Y1,Y2,Y3: Integer;
begin
  Image1.Canvas.Pen.Width:=5+Random(5);
  Image1.Canvas.Pen.Color:=RGBToColor(Random(256),Random(256),Random(256));
  Image1.Canvas.Brush.Color:=RGBToColor(Random(256),Random(256),Random(256));
  X1:=20+Random(Image1.Width-40);
  X2:=20+Random(Image1.Width-40);
  X3:=20+Random(Image1.Width-40);
  Y1:=20+Random(Image1.Height-40);
  Y2:=20+Random(Image1.Height-40);
  Y3:=20+Random(Image1.Height-40);
  Image1.Canvas.Polygon ( [Point(X1,Y1), Point(X2,Y2), Point(X3,Y3) ] );
end;
```

11. Vysvetlite pojem cyklus a popíšte cyklus s pevným počtom opakovaní.

Cyklus je mechanizmus, ktorý jeden príkaz (alebo postupnosť príkazov) vie zopakovať viac krát bez toho, aby sme to museli veľakrát zapisovať.
Ak dopredu vieme zadať, koľkokrát sa budú príkazy opakovať, použijeme **cyklus s pevným počtom opakovaní .**

Konštrukciu môžeme zapísať takto:

for *premenná* **:=** *dolná_hranica* **to** *horná_hranica* **do** *príkaz*;

kde **premenná** je názov premennej cyklu (počítadlo cyklu – riadiaca premenná) a nesmieme ju zabudnúť zadeklarovať (musí byť ordinálneho typu). **Dolná a horná hranica cyklu** sú dve celočíselné hodnoty definujúce interval hodnôt, pre ktorý sa bude postupne (pre každú hodnotu z tohto intervalu) vykonávať zadaný príkaz. To znamená, že najprv sa do počítadla priradí dolná hranica cyklu a vykoná sa **príkaz** (telo cyklu). Potom sa počítadlo zvýši o jednotku a opäť sa vykoná príkaz. Toto sa opakuje, až kým počítadlo nepresiahne hornú hranicu. Vtedy sa cyklus zastaví a pokračuje sa v príkazoch nasledujúcich za cyklom.

Prevrátený cyklus (počítadlo sa bude znižovať):

for premenná **:=** horná_hranica **downto** dolná_hranica **do** príkaz;

V programovacom jazyku pascal musíme špeciálne označiť blok príkazov, ktorý chceme, aby sa celý opakoval v príkaze for-cyklu. Začiatok bloku príkazov zapisujeme rezervovaným slovom **begin** a koniec bloku rezervovaným slovom **end**.

Vo vnútri cyklu môžeme používať hodnotu počítadla (môžeme ju priradiť, môže byť v nejakom výraze, môže byť parametrom príkazu), ale **nesmieme** do tejto premennej v tele cyklu nič priraďovať.
V tele cyklu môžeme používať aj ďalšie premenné, ktoré sa vypočítajú nanovo a nezávisia od počítadla cyklu.

Zadanie 11:

Nakreslite 10 sústredných farebných kruhov tak, že sa ich priemery líšia o zvolenú konštantu.

```
procedure TForm1.Button1Click(Sender: TObject);
var
   X,Y,I,PRIEMER : Integer;
const
   p=20;
begin
   X:=30;
   Y:=30;
   PRIEMER:=400;
   Image1.Canvas.Pen.Width:=5;
   for I:=1 to 10 do
   begin
     Image1.Canvas.Brush.Color:=RGBToColor(Random(255),Random(255),Random(255));
     Image1.Canvas.Ellipse ( X, Y, X+PRIEMER, Y+PRIEMER );
     X:=X+p;
     Y:=Y+p;
     PRIEMER:=PRIEMER-2*p;
   end;
end;
```

12. Vysvetlite vetvenie v programe, popíšte úplný a neúplný podmienený príkaz.

Niektoré príkazy sa budú vykonávať len pri splnení istých podmienok.
Podmienkou budeme rozumieť taký **výraz**, ktorého výsledkom je buď **pravda alebo nepravda**.
Veľmi často to budú podmienky, v ktorých sa nejaké hodnoty navzájom porovnávajú.
V Pascale na porovnávanie hodnôt máme k dispozícii tieto tzv. relačné operátory:
= rovnosť,
<> nerovnosť,
<, <=, >, >=.

Zápisy niekoľkých jednoduchých podmienok:

X > 100	A > B
I mod 5 = 0	Random(6) = 1
I div 10 >= 1	Abs(X) < 0

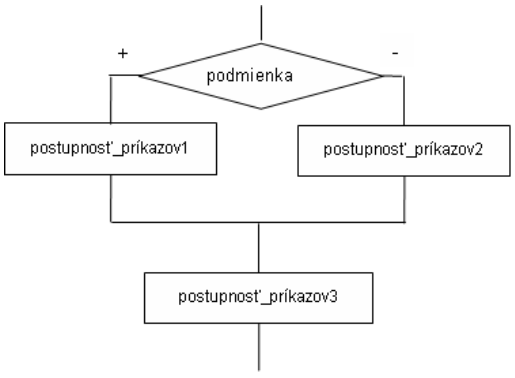
Výsledok podmienky väčšinou závisí od nejakých premenných (napr. X > 100).

Zostavovanie zložených podmienok (logické operácie):
podm1 **and** *podm2* – otestovanie, či súčasne platia obe podmienky ***podm1*** aj ***podm2***
podm1 **or** *podm2* – otestovanie, či platí aspoň jedna z podmienok ***podm1*** a ***podm2***
not *podm1* – otestovanie, či neplatí podmienka ***podm1***, to znamená znegovanie podmienky
Operandy týchto logických operácií by mali byť výrazy (najčastejšie relačné), ktorých hodnota je **pravda** alebo **nepravda**.

Parametre logických operácií väčšinou budeme dávať do zátvoriek:
Napr.:(Hodin < 0) **or** (Hodin > 24)

Príkaz vetvenia (hovoríme mu aj **podmienený príkaz**):

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



Povedané inak: **ak** je splnená *podmienka*,**tak** vykonaj *príkaz1*,**inak** vykonaj *príkaz2*;
Pred **else** sa **nesmie** písať bodkočiarka.
Ak potrebujeme v niektorej vetve podmieneného príkazu zapísať dva alebo viac príkazov, použijeme zložený príkaz **begin** ... **end.**

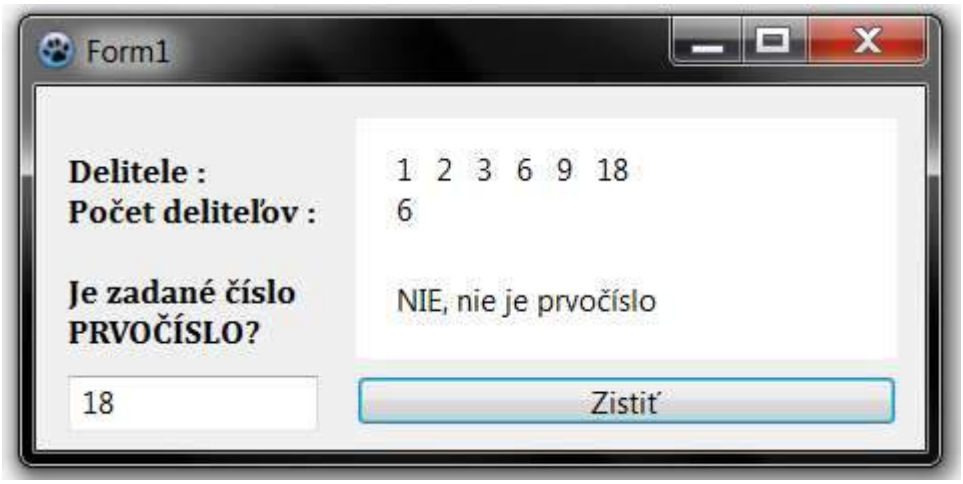
Neúplný podmienený príkaz nemá vetvu else. Jeho zápis je:

```
if podmienka then
príkaz;
```

Zadanie 12:

Vypíšte všetky prirodzené delitele zadaného prirodzeného čísla a podľa ich počtu zistite, či je číslo prvočíslo.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  I,N,D:Integer;
begin
  Image1.Canvas.FillRect(Image1.ClientRect);
  N:=StrToInt(Edit1.Text);
  I:=1;
  D:=0;

  repeat
    if N mod I=0 then
    begin
      D:= D+1;
      Image1.Canvas.TextOut(D*20,15,IntToStr(I));
    end;
  I:= I+1;
  until I>N;
  Image1.Canvas.TextOut(20,35,IntToStr (D));

  if D=2 then Image1.Canvas.TextOut(20,80,'ÁNO, je prvočíslo')
         else Image1.Canvas.TextOut(20,80,'NIE, nie je prvočíslo');
end;
```



13. Vysvetlite syntax zloženého podmieneného príkazu (case), uveďte príklady jeho použitia.

Pascal poskytuje príkaz vetvenia, ktorý netestuje hodnotu nejakej podmienky na pravda/nepravda, ale podľa hodnoty nejakého číselného výrazu, vykoná jeden z možných výrazov.

```
case výraz of
    hodnota1: príkaz1;
    hodnota2: príkaz2;
    hodnota3: príkaz3;
    ...
else
    príkazE;
end;
```

Príkaz **case** nemusí mať **else**-vetvu.

Príklady:

```
Case znak of
'a','e','i','o','u','y' : Image1.Canvas.TextOut(50, 50, 'samohláska');
 'ô' : Image1.Canvas.TextOut(50, 50,'dvojhláska');
else
  Image1.Canvas.TextOut(50, 50,'spoluhláska');
end;
```

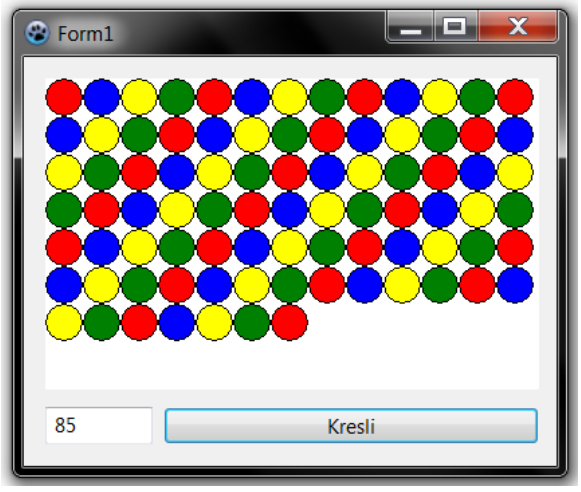
Do príkazu case je možné zadať aj interval hodnôt, nesmú mať však spoločný prienik:

```
Case Cislo of
 1..10   : Image1.Canvas.TextOut(50, 50,'Malé číslo');
 11..100 : Image1.Canvas.TextOut(50, 50,'Stredne veľké číslo');
else
 Image1.Canvas.TextOut(50, 50,'Veľké číslo');
end;
```

Zadanie 13:

Nakreslite do radu N kružníc, pričom sa striedajú 4 farby výplne kruhov: červená, modrá, žltá, zelená.

```
procedure TForm1.Button1Click(Sender: TObject);
const
  R=14;
var
  I,X,Y,N: Integer;
begin
  Image1.Canvas.Brush.Color:=ClWhite;
  Image1.Canvas.FillRect(Image1.ClientRect);
  N:=StrToInt(Edit1.Text);
  X:=R;
  Y:=R;
  for I:=1 to N do
   begin
     case I mod 4 of
     1: Image1.Canvas.Brush.Color:=ClRed;
     2: Image1.Canvas.Brush.Color:=ClBlue;
     3: Image1.Canvas.Brush.Color:=ClYellow;
     0: Image1.Canvas.Brush.Color:=ClGreen;
     end;
      if X+R > Image1.Width
      then begin
             X:=R;
             Y:=Y+2*R;
             Image1.Canvas.Ellipse(X-R,Y-R,X+R,Y+R);
             X:=X+2*R;
           end
      else begin
             Image1.Canvas.Ellipse(X-R,Y-R,X+R,Y+R);
             X:=X+2*R;
           end;
   end;
end;
```

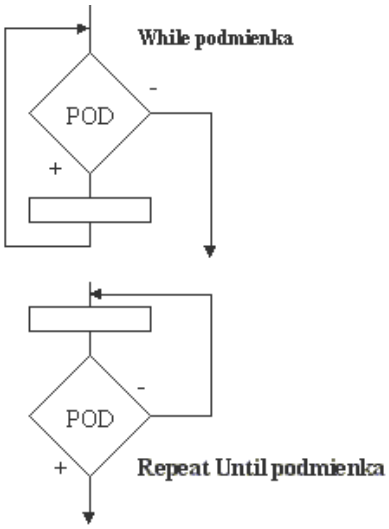


14. Zapíšte syntax cyklu s podmienkou na začiatku a na konci, vysvetlite rozdiel medzi nimi.

Ak chceme nejaké príkazy opakovať, ale nevieme dopredu určiť počet opakovaní, použijeme cyklus s podmienkou.

1.Cyklus s podmienkou na začiatku:

while *podmienka* **do**
begin
príkaz (y)
end;



2.Cyklus s podmienkou na konci:

repeat
príkaz (y)
until *podmienka;*

Ak použijeme cyklus s podmienkou na začiatku, príkazy cyklu sa nemusia vykonať ani raz, pri cykle s podmienkou na konci sa vykonajú aspoň raz.

Zadanie 14:

Vysvetlite Euklidov algoritmus postupného odčítania na výpis najväčšieho spoločného deliteľa dvoch prirodzených čísel a naprogramujte túto úlohu pomocou Euklidovho algoritmu postupného delenia.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  A, B, ZV: Integer;
begin
  Image1.Canvas.FillRect(Image1.ClientRect);
  A := StrToInt(Edit1.Text);
  B := StrToInt(Edit2.Text);
  repeat ZV:= A mod B;
         A:=B;
         B:=ZV;
  until ZV = 0;
  Image1.Canvas.Font.Height := 30;
  Image1.Canvas.TextOut(20,20,'NSD = ' + IntToStr(A));
end;
```

15. Definujte logickú premennú a jej deklaráciu, popíšte logické operácie a funkcie na type Boolean.

Premenné ***logického typu*** (typ **Boolean)** v pamäti zaberajú 1 bajt. (typ Integer zaberá v pamäti 4 bajty).
Logický typ má preddefinované dve svoje konštanty pre pravdu **True** a pre nepravdu **False**.

Deklarujeme:

```
var
    B, B1: Boolean;
begin
    B := True;
```

Logické operácie na type Boolean: **not , and, or.**

a AND b..... konjunkcia hodnôt *a*, *b*,
a OR b.....disjunkcia hodnôt *a*, *b*,
NOT a.....negácia hodnoty *a*

Platí: False < True

Užitočné **funkcie** na prácu s logickými hodnotami:
Ord(*logická_hodnota*)- konvertuje logickú hodnotu na číselnú (vráti 0 alebo 1, podľa toho, či logická hodnota bola **False** alebo **True).**
Boolean(*číslo*)- konvertuje číselnú hodnotu na logickú (vráti **True** alebo **False** podľa toho či číslo je 0 alebo 1).

Priorita aritmetických, logických a relačných operácií:

1. **not**
2. *** / div mod and**
3. **+ - or**
4. **= < <= > >= <>**

Zadanie 15:

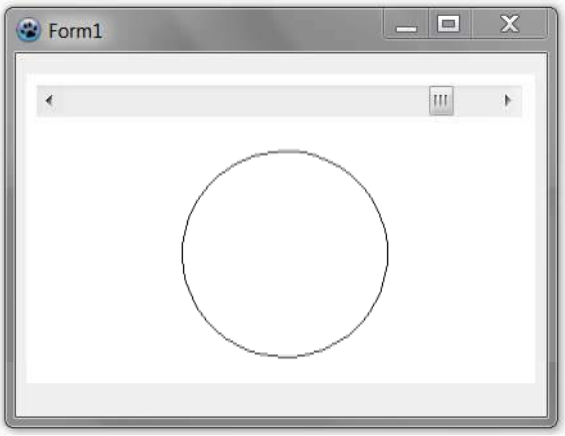
Zistite, či sa v načítanom prirodzenom čísle aspoň raz vyskytuje číslica 7.

```
procedure TForm1.Button1Click(Sender: TObject);
var
 Vyskyt7: boolean;
 Cislo: Integer;
begin
  Cislo := StrToInt(Edit1.Text);
  Vyskyt7 := False;
  while Cislo > 0 do
        begin if   Cislo mod 10 = 7
              then Vyskyt7 := True;
                   Cislo := Cislo div 10;
        end;
  if Vyskyt7 then Image1.Canvas.TextOut(20,50,'v čísle je číslica 7')
  else Image1.Canvas.TextOut(20,50,'v čísle nie je 7');
end;
```

Zadanie 16:

Nakreslite kružnicu so stredom v (200, 100) a s polomerom 80 bodov pomocou parametrických rovníc kružnice: x = x0 + r * cos t y = y0 + r * sin t
kde r je polomer kružnice, (x0, y0) je jej stred a t je parameter funkcie z intervalu <0, 2π>.
Realizujte zmenu polomeru kružnice pomocou posuvníka.

```
procedure TForm1.ScrollBar1Change(Sender: TObject);

//   !!! NEZABUDNI na zmenenie "min" a "max" v inšpektore objektov !!!

var
  R: Integer;
  X, Y, Uhol: Real;
begin
  Image1.Canvas.FillRect(Image1.ClientRect);
  Uhol := 0;
  R:=ScrollBar1.Position;
  while Uhol <= 6.3 do
  begin
    X := 200 + R * Cos(Uhol);
    Y := 100 + R * Sin(Uhol);
    if Uhol = 0 then Image1.Canvas.MoveTo(Round(X), Round(Y))
                else Image1.Canvas.LineTo(Round(X), Round(Y));
  Uhol := Uhol + 0.1;
  end;
end;
```



17. Vysvetlite pojmy: podprogram, procedúra. Vysvetlite mechanizmus, ktorý prebehne pri volaní podprogramu.
Popíšte, ako definujeme procedúru bez parametrov.

Často pri tvorbe väčších programov si celú našu úlohu najprv rozdelíme na menšie podprogramy, ktoré potom **voláme** menom.

Procedúra je podprogram - algoritmus, nejaká časť programu s tým, že túto časť môžeme vyvolať pomocou jej mena aj viackrát.
Doteraz sme používali napr. procedúru na spracovanie nejakej udalosti (kliknutie na tlačidlo) - **procedure** TForm1.Button1Click, niektoré hotové procedúry: napr. grafické príkazy (**Rectangle**, **MoveTo**, **TextOut),** podprogramy pre textovú plochu (**Clear**, **Append**), ale aj rôzne iné podprogramy, napr. **Sleep**, **Repaint**, **Close** a **Click**.

Mechanizmus, ktorý prebehne pri volaní každého podprogramu:
1. **zapamätá sa návratové miesto**, t.j. riadok, kam sa bude treba vrátiť po skončení procedúry (zapamätá sa adresa príkazu v pamäti, kde sa bude pokračovať vo vykonávaní po návrate z procedúry);
2. **vytvoria sa lokálne premenné** procedúry (so zatiaľ nedefinovanou hodnotou)
3. prenesie sa **riadenie** programu **do tela podprogramu** (za **begin**);
4. vykonajú sa všetky príkazy podprogramu (až po koncový **end**);
5. **zrušia sa lokálne premenné** - "zabudnú" sa;
6. **riadenie sa vráti** za miesto v programe, odkiaľ bol podprogram volaný – t.j. **na návratové miesto**.

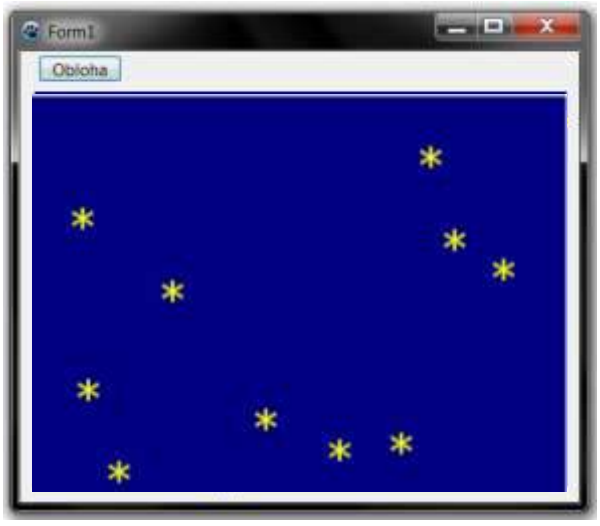
Definovanie vlastnej procedúry (v deklaračnej časti procedúry **Button1Click**):

```
procedure meno;
// deklarácie lokálnych premenných, konštánt a podprogramov pre danú procedúru
begin
// telo procedúry
end;
```

Zadanie 17:

Zatlačenie tlačidla zmaže plochu na modro a nakreslí nočnú oblohu s 10 malými žltými hviezdičkami na náhodných pozíciách. Hviezdičky kreslite pomocou procedúry Hviezda.

```
procedure TForm1.Button1Click(Sender: TObject);
var
   I:Integer;
  procedure Hviezda;
   var
     X,Y: Integer;
   begin
    X:= Random (Image1.Width);
    Y:= Random (Image1.Height);
    Image1.Canvas.Font.Height:=50;
    Image1.Canvas.Font.Color:=clYellow;
    Image1.Canvas.TextOut(X,Y, '*');
    end;
begin
  Image1.Canvas.Brush.Color:=clNavy;
  Image1.Canvas.FillRect(Image1.ClientRect);
 for I:= 1 to 10 do
 Hviezda;
end;
```



18. Vysvetlite rozdiel medzi lokálnou a globálnou premennou procedúry.

Procedúra môže mať svoje vlastné ***lokálne premenné***.

Samotná procedúra **Button1Click,** ktorá obsahuje podprogramy**,** môže mať svoje lokálne premenné. Sú **viditeľné** aj vo všetkých procedúrach vo vnútri tejto procedúry. Podmienkou je ale to, aby premenné boli definované skôr, ako procedúry a zároveň by tieto procedúry nemali mať definované svoje lokálne premenné s rovnakým menom.

Globálne premenné – premenné *zadefinované na 1. úrovni:*

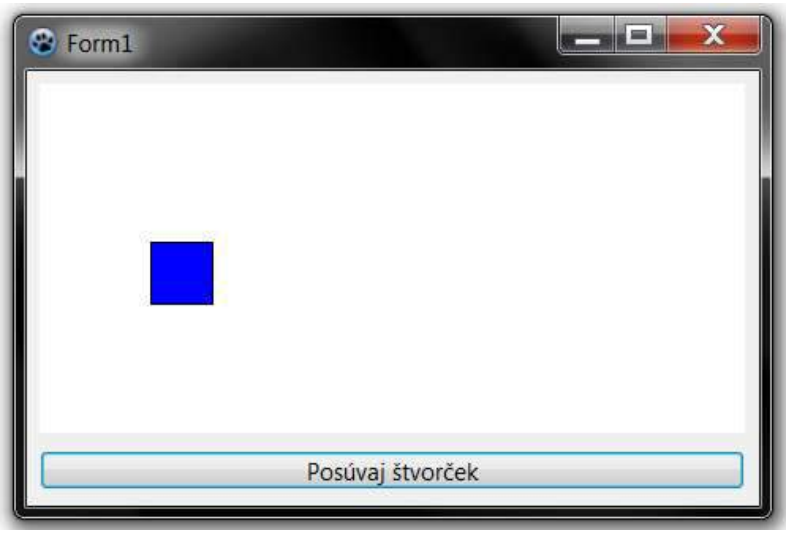
- vznikajú pri štarte programu,
- existujú so svojimi hodnotami počas celého behu programu,
- môžeme im nastaviť ich počiatočné hodnoty už pri deklarácii – ak to nespravíme, majú nulové hodnoty.

Tieto globálne premenné sú viditeľné vo všetkých procedúrach, ktoré sú v programe definované neskôr ako tieto premenné.

Zadanie 18:

Keď budeme postupne klikať na tlačidlo, nakreslí sa modrý štvorček najprv na súradniciach 10, 100, pri každom ďalšom kliknutí sa nakreslí posunutý o 2 vpravo.

```
var
  X : Integer = 10;
  Y : Integer = 100;

procedure TForm1.Button1Click(Sender: TObject);

    procedure Zmaz;
    begin
      Image1.Canvas.Brush.Color := clWhite;
      Image1.Canvas.FillRect(Image1.ClientRect);
    end;

    procedure Stvorec;
    begin
      Image1.Canvas.Brush.Color := clBlue;
      Image1.Canvas.Rectangle(X, Y, X+40, Y+40);
    end;

begin
   Zmaz;
   Stvorec;
   X := X + 2;
end;
```



19. Popíšte, ako definujeme procedúru s parametrami, vysvetlite mechanizmus volania procedúry s parametrami, vysvetlite pojmy formálny a skutočný parameter.

procedure meno(*deklarácia parametrov);*
// deklarácie lokálnych premenných, konštánt a podprogramov pre danú procedúru
begin
// telo procedúry
end;

Ak používame v procedúre parametre rôznych typov, deklarujeme napr.
procedure Bodky(N: Integer; Farba: TColor);

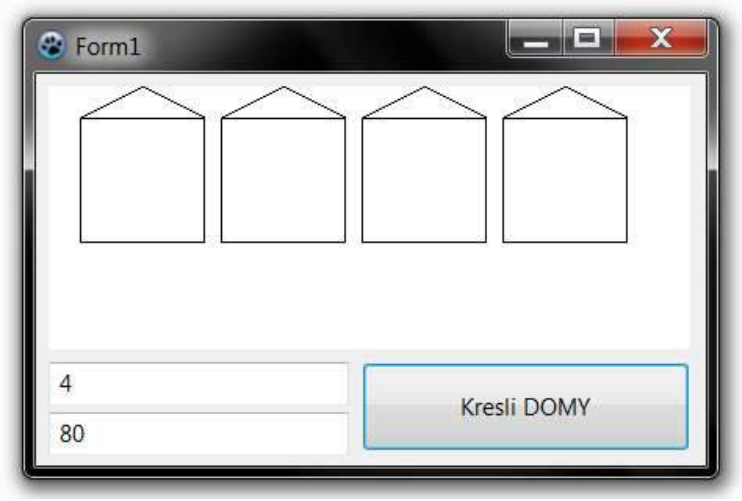
Mechanizmus volania procedúry s parametrami:
1. **zapamätá sa návratové miesto**, t.j. riadok, kam sa bude treba vrátiť po skončení procedúry;
2. **vytvoria sa všetky zadeklarované lokálne premenné** procedúry . aj parametre sú lokálne premenné, preto sa na tomto mieste vytvoria aj všetky **parametre** (premenné), do nich sa priradia vstupné hodnoty);
3. prenesie sa **riadenie** programu **do tela podprogramu** ;
4. vykonajú sa všetky príkazy podprogramu (až po koncový **end**);
5. **zrušia sa lokálne premenné** – a teda sa zrušia aj parametre;
6. **riadenie sa vráti** za miesto v programe, odkiaľ bol podprogram volaný – t.j. **na návratové miesto**.

Deklarované parametre procedúry sú **formálnymi parametrami**. Hodnotám parametrov ktoré zadávame pri volaní procedúry hovoríme **hodnoty vstupných parametrov (skutočné parametre)**. Tieto môžu byť zadané konštantou, hodnotou premennej alebo ako výsledok aritmetického výrazu. Parametre procedúr sú rovnaké ako obyčajné lokálne premenné, len už majú priradenú počiatočnú hodnotu, ktorá je rovná vstupnej hodnote parametra.
Zadefinovanú procedúru musíme volať so všetkými jej parametrami. Pri volaní procedúry musíme dodržať poradie vstupných hodnôt pre parametre.

Zadanie 19:

Zadefinujte procedúru Dom (A,N), ktorá nakreslí vedľa seba N domov, kde A je veľkosť strany štvorca. Parametre A,N načítajte z editovacieho políčka.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  A,N : Integer;

  procedure Dom (A,N : Integer);
  var
    X,Y,I : Integer;
  begin
    X := 20;
    Y := 20;
    for I :=1 to N do
    begin
    Image1.Canvas.Rectangle (X, Y, X+A, Y+A);
    Image1.Canvas.MoveTo(X,Y);
    Image1.Canvas.LineTo(X+A div 2,Y-20);
    Image1.Canvas.MoveTo(X+A div 2,Y-20);
    Image1.Canvas.LineTo(X+A,Y);
     X := X+A+10;
    end;
  end;
begin
  Image1.Canvas.FillRect(Image1.ClientRect);
  N := StrToInt(Edit1.Text);
  A := StrToInt(Edit2.Text);
  Dom (A,N);
end;
```



20. Vysvetlite rozdiel medzi lokálnou a globálnou procedúrou.

Všetky procedúry, ktoré sme mohli použiť len na mieste, kde sú vnorené sú **lokálne procedúry**.

Niekedy chceme tú istú procedúru využívať vo viacerých procedúrach pre stláčanie tlačidiel. Ak by sme ju zadefinovali mimo procedúr tlačidiel, bola by **globálna**.

Ak globálna procedúra nemá označenie, že patrí konkrétnemu komponentu, Pascal nevie rozpoznať prvky, ktoré sú v komponente.

Ak potrebujeme vytvoriť globálnu procedúru, ktorá pracuje s prvkami (komponentmi) formuláru, tak pred tieto prvky musíme písať slovo **Form1.**

Napr.

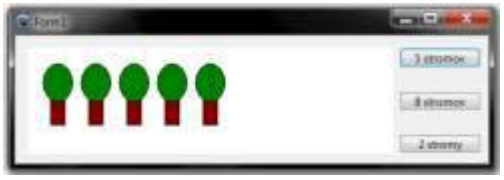
```
procedure Stvorec(X, Y: Integer; Farba: TColor);
begin
    Form1.Image1.Canvas.Brush.Color := Farba;
    Form1.Image1.Canvas.Rectangle(X, Y, X+45, Y+45);
end;
```

Zadanie 20:

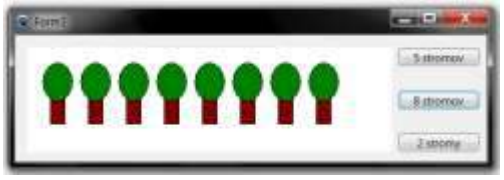
Napíšte globálnu procedúru na kreslenie stromu.
Vo formulári budú tri tlačidlá: každé nakreslí rad stromov s iným počtom stromov.

```
procedure Strom (X, Y: Integer);
begin
  Form1.Image1.Canvas.Brush.Color := clMaroon;
  Form1.Image1.Canvas.Rectangle (X+10, Y+40, X+30, Y+80);
  Form1.Image1.Canvas.Brush.Color := clGreen;
  Form1.Image1.Canvas.Ellipse (X, Y, X+40, Y+50);
end;
```

```
procedure TForm1.Button1Click(Sender: TObject);
var I,J,Z: Integer;
begin
  Image1.Canvas.Brush.Color:=clWhite;
  Image1.Canvas.FillRect(Image1.ClientRect);
  I := 20;
  J := 20;
  for Z:=1 to 5 do
    begin
      Strom (I,J);
      I := I + 50;
    end;
end;
```



```
procedure TForm1.Button2Click(Sender: TObject);
var I,J,Z: Integer;
begin
  Image1.Canvas.Brush.Color:=clWhite;
  Image1.Canvas.FillRect(Image1.ClientRect);
  I := 20;
  J := 20;
  for Z:=1 to 8 do
    begin
      Strom (I,J);
      I := I + 50;
    end;
end;
```



```
procedure TForm1.Button3Click(Sender: TObject);
var I,J,Z: Integer;
begin
  Image1.Canvas.Brush.Color:=clWhite;
  Image1.Canvas.FillRect(Image1.ClientRect);
  I := 20;
  J := 20;
  for Z:=1 to 2 do
    begin
      Strom (I,J);
      I := I + 50;
    end;
end;
```

21. Charakterizujte znakový typ údajov (char) a popíšte základné funkcie na tomto type.

Typ znak sa v Pascale označuje slovom **Char**.
Premenná tohto typu si môže zapamätať práve jeden znak, napr. písmeno, číslicu ale aj iné znaky:
A B C … a b c … 0 1 2 … á ä č ď … @ # $ % … + - / * = < > ( ) [ ] { } …

Znakové konštanty zapisujeme medzi dva apostrofy, napr.
'a', 'B', '3', '#', ' ', '.', 'ž', 'ô'.

Na rozdiel od celočíselného aj logického typu s týmto typom nemôžeme robiť žiadne operácie. Môžeme nejakú znakovú hodnotu priradiť do premennej, môžeme ju porovnať s inou hodnotou a môžeme ju vypísať napr. do grafickej plochy.

V pamäti počítača je pre každú takúto premennú vyhradený jeden bajt (8 bitov) a preto sa do tejto premennej "zmestí" 256 rôznych hodnôt. Tieto hodnoty sú navzájom usporiadané, preto ich môžeme navzájom porovnávať nielen na rovnosť, ale aj reláciami menší a väčší. Znaky sú kódované tzv. ASCII kódovaním (kód - číslo od 0 do 255).

Funkcie, ktoré vedia pracovať so znakmi:

- funkcia **Ord(znak)** - vráti vnútorný kód znaku (ASCII kód: <0, 255>)
- funkcia **Char(číslo)** alebo **Chr(číslo)**- ak je celé číslo z intervalu <0, 255>, tak vráti prislúchajúci znak
- funkcia **UpCase** z malého písmena abecedy spraví zodpovedajúce veľké

Napr.:

Ord('a') = 97
Char(65) = 'A'
Char(Ord('1')+1) = '2'
Ord(' ') = 32

UpCase('e') = 'E'
UpCase('M') = 'M'
UpCase('%') = '%'
UpCase(#97) = 'A'

znakovú konštantu môžeme zapísať aj pomocou #číslo – kde číslo je nejaký ASCII kód, napr. #65 označuje znak 'A', #32 je medzera;

Zadanie 21:

Napíšte program, v ktorom bude na tlačidle Button1 jednoznakový text 'A'. Po každom zatlačení tlačidla sa na tlačidle objaví nasledujúci znak abecedy. Po stlačení tlačidla so znakom 'Z' program skončí (Close).

```
var
  Z: Char = 'A';

procedure TForm1.Button1Click(Sender: TObject);

begin
  Button1.Caption := Z;
  Inc(Z);
  if Ord(Z)= 92 then Close;
end;
```

22. Vysvetlite pojem ordinálny typ údajov, rozhodnite o ordinalite typov: Integer, Boolean, Char, Real.
Popíšte príkazy: Inc a Dec, funkcie: Low a High na týchto typoch.

Ordinálne typy – každá konštanta tohto typu **má svojho predchodcu aj nasledovníka** (okrem minimálnej a maximálnej konštanty).
Napr. celočíselný typ I**nteger**, aj logický typ **Boolean** a aj znakový typ **Char** sú ordinálne typy.
Reálny typ **Real** nie je ordinálnym typom.

Ordinálne typy majú v Pascale isté špeciálne postavenie:

- počítadlo (riadiaca premenná) for-cyklu musí byť ordinálneho typu,
- hodnota, na základe ktorej sa príkaz case rozhodne, ktorú vetvu vykoná, musí byť ordinálneho typu,
- existujú štandardné príkazy **Inc** a **Dec**, pomocou ktorých vieme zvýšiť alebo znížiť obsah ordinálnej premennej,
  - ak má celočíselná premenná I hodnotu 17
    Inc(I); ju zvýši o 1 (v premennej I je teraz číslo 18)
    Inc(I, 5); ju zvýši o 5 (v premennej I je teraz číslo 23)
  - ak má celočíselná premenná J hodnotu 22
    Dec(J); ju zníži o 1 (v premennej J je teraz číslo 21)
    Dec(J, 3); ju zníži o 3 (v premennej J je teraz číslo 18)
- štandardné funkcie **Low** a **High** vrátia konštanty pre minimálnu a maximálnu povolenú hodnotu príslušného ordinálneho typu, napr.

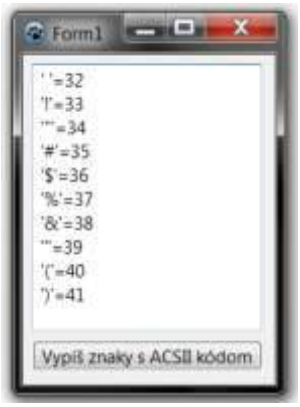
  Low(Integer) = -2147483648
  High(Integer) = 2147483647
  Low(Char) = Char(0)
  High(Char) = Char(255)
  Low(Boolean) = False
  High(Boolean) = True

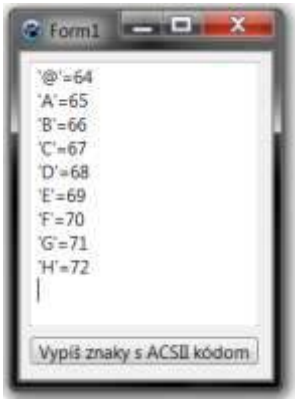
- indexom polí musí byť ordinálny typ,
- bázový typ množín musí byť ordinálnym typom.

Zadanie 22:

Napíšte program, ktorý vypíše znaky od prázdneho znaku(' ') po znak 'H' a ich ASCII kódy.

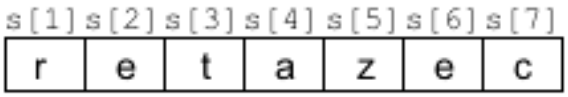
```
procedure TForm1.Button1Click(Sender: TObject);
var
  Z : Char;
begin
  Memo1.Clear;
  for Z := ' ' to 'H' do
  Memo1.Lines.Append ('''' +Z + '''=' + IntToStr(Ord(Z)))
  end;
```





23. Charakterizujte údajový typ String (reťazec znakov). Popíšte základné operácie, funkcie a procedúry na tomto type.

Textový reťazec je údajový typ veľmi podobný poľu. Obsahuje znaky (teda typ char), ktoré sú očíslované indexmi od 1 po dĺžku reťazca.



Length(retazec) - funkcia, ktorá vracia dĺžku reťazca. Napr: s:= 'retazec';
dlzka:= Lenght(s); //dlzka=7

Môžeme pristupovať k jednotlivým prvkom reťazca: s[1]:= 'r';
Priamo môžeme pristupovať iba k tým prvkom reťazca, ktoré existujú. Ak má teda dĺžku 7, nesmieme priraďovať napríklad do prvku s[10], takýto program by skončil chybovou správou.
Niekedy sa môže hodiť vyrobiť reťazec s nulovou dĺžkou, ktorý je prázdny: s:= '';

Znakové reťazce môžeme porovnávať pomocou **relačných operácií** (=, <>, <, >, <=, >=). Reťazce sú usporiadané pomocou tzv. lexikografického usporiadania.

Napr.:
'abc' > 'ABC' 'jana' < 'jano'
'Adam' < 'Eva' 'Jana' < 'jana'

Porovnávanie dvoch znakov sa robí podľa pravidiel Char: t.j. menší je ten znak, ktorý má menší ASCII-kód.
Na zreťazovanie reťazcov slúži **operácia +**. Plus má v tomto prípade iný význam než pri sčitovaní čísel.
s:= 'abcd' + 'efgh'; // s='abcdefgh'

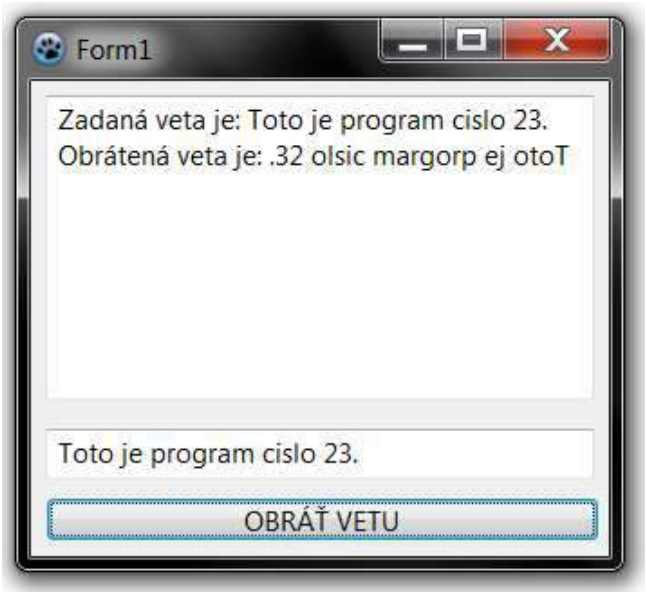
Ďalšie užitočné **funkcie a procedúry**:

- **TextOut**(x, y, text); // vypíše text do plochy
- **IntToStr**(číslo) // konvertuje číslo na reťazec
- **StrToInt**(text) // konvertuje reťazec na číslo
- **UpperCase**(text) // vráti reťazec, v ktorom sú všetky písmená veľké
- **LowerCase**(text) // vráti reťazec, v ktorom sú všetky písmená malé
- **IntToHex**(číslo, počet_cifier) // vráti číslo v 16-ovej sústave
- **SetLength**(meno_reťazca, nová_dĺžka) - nastaví novú dĺžku reťazca, ak je kratší, znaky sa stratia
- **Copy**(reťazec, od, koľko) - vráti reťazec, ktorý je podreťazcom prvého a to taký, ktorý má 'koľko' znakov a začína v pôvodnom reťazci na pozícii 'od'
  Retazec := Copy('abcde', 2, 3); // Retazec='bcd'
- **Pos**(podreťazec, reťazec) - vráti počiatočný index prvého výskytu podreťazca v danom reťazci, alebo 0 ak sa tam taký nenachádza
  I := Pos('d', 'abcde'); // I=4
- **Delete** (reťazec, od, koľko) – z daného reťazca vymaže podreťazec od príslušného indexu danej dĺžky.
  s := 'abrakadabra'; Delete(s, 1, 1); // s = 'brakadabra'
- **Insert** (podreťazec, premenná, index) - slúži na vkladanie podreťazca do reťazcovej premennej a to od konkrétneho indexu.
  s := 'abxy'; Insert(s, s, 3); // s = 'ababxyxy'

Zadanie 23:

Napíšte program, ktorý zo zadaného textu vyrobí reťazec, v ktorom budú znaky v opačnom poradí (na výpis využite textovú plochu).

```
procedure TForm1.Button1Click(Sender: TObject);
var
   Retazec, R1: String;
   I, Dlzka : Integer;
begin
   Memo1.Clear;
   Retazec := Edit1.Text;
   Memo1.Lines.Append ('Zadaná veta je: ' + Retazec);
   Dlzka := Length(Retazec);
   R1 := '';
   for I := 1 to Dlzka do  R1 := Retazec[I] + R1;
   Memo1.Lines.Append ('Obrátená veta je: ' + R1);
end;
```



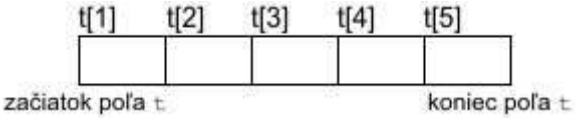
24. Charakterizujte jednorozmerné pole ako štruktúrovaný typ údajov. Popíšte jeho deklaráciu a spôsob pristupovania k jednotlivým prvkom poľa.

Jednorozmerné pole je štruktúrovaný typ premennej, umožňuje pracovať s veľkým množstvom hodnôt rovnakého typu - napríklad zaznamenať denné teploty v priebehu celého mesiaca a vypočítať z nich priemer.

Pole (po anglicky **array**) deklarujeme tak, že určíme koľko bude mať prvkov a ako budú indexované a určíme typ prvkov.

Deklarácia poľa: meno : **array [od .. do] of typ_prvkov**;
Napr.: **var**
t : array [1..5] of integer;

Pri deklarácií sa vytvorí v pamäti počítača toľko premenných daného typu, koľko má pole prvkov.



K jednotlivým prvkom poľa potom pristupujeme pomocou indexov.
Napr.: tretí prvok poľa: t[3].

Indexom poľa môže byť:
- konštanta
- aritmetický výraz
- premenná alebo výraz s premennou

Inicializácia poľa- už počas deklarácie vymenujeme všetky hodnoty prvkov poľa.
Napr.: Teplota: array [1..7] of Real =(20.3, 22.0, 17.7, 18.1, 16.7, 17.3, 19.0);

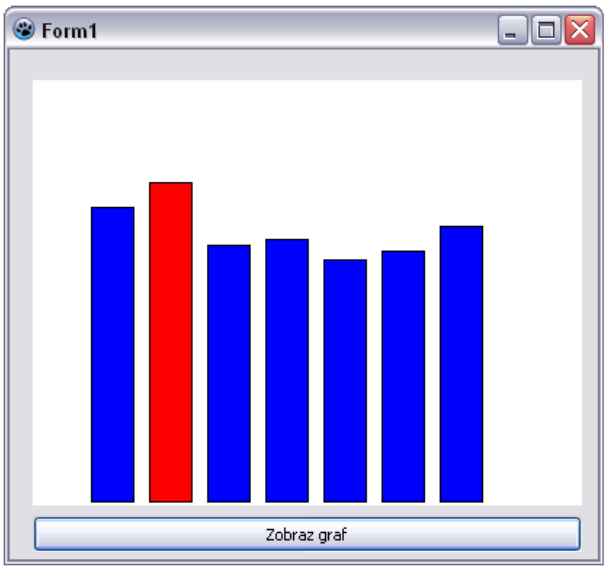
Často sa nám môže hodiť meniť veľkosť poľa. Pre veľkosť poľa je dobré zaviesť v programe konštantu, obvykle ju pomenujeme max. Ak budeme neskôr chcieť zmeniť počet prvkov poľa, budeme to robiť iba na jednom mieste - pomocou konštanty max.

```
const
  max = 100;
var
  t :  Array [1..max] of Integer;
  i :   Integer;
begin
  ...
  for I: = 1 to max do
  ...
end;
```

Zadanie 24:

Inicializujte jednorozmerné pole siedmich nameraných teplôt: 20.3, 22.0, 17.7, 18.1, 16.7, 17.3, 19.0. Znázornite tieto teploty pomocou stĺpcového grafu tak, že stĺpec maximálnej teploty bude červený, ostatné budú modré.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Teplota: array [1..7] of Real = (20.3, 22.0, 17.7, 18.1, 16.7, 17.3, 19.0);
  I: Integer;
  Max: Real;
begin
  Max := Teplota[1];
  for I := 2 to 7 do
      begin
        if Teplota[I] > Max then Max := Teplota[I];
      end;
  for I := 1 to 7 do
      begin
        if Teplota[I] = Max
        then Image1.Canvas.Brush.Color := clRed
        else Image1.Canvas.Brush.Color := clBlue;
             Image1.Canvas.Rectangle(40*I, 290, 40*I+30, 290-Round(Teplota[I]*10));
      end;
end;
```



25. Charakterizujte dvojrozmerné pole ako štruktúrovaný typ údajov. Popíšte jeho deklaráciu a spôsob pristupovania k jednotlivým prvkom poľa.

Dvojrozmerné pole sa dá inak nazvať aj **tabuľka**, či **matica**.
Hovoríme, že dvojrozmerné pole je aj vektor vektorov, alebo pole jednorozmerných polí.

Syntax **deklarácie** viacrozmerného poľa:
var identifikátor : array [dh1..hh1,dh2..hh2,dhn..hhn] of typ ;
napr.: vek: array[1..10,1..10] of integer;

Deklarovali sme pole s veľkosťou 10 x 10 prvkov, tj. 100 prvkov.
Spodná a horná hranica musí byť ordinálneho typu – celočíselný typ alebo znakový typ - char.

Štvorcová matica typu NxN – má rovnaký počet riadkov a stĺpcov, napríklad matica A typu 3x3 bude mať prvky:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

Obdĺžniková matica typu MxN – má M riadkov a N stĺpcov, napríklad matica B typu 4x2 bude mať prvky:

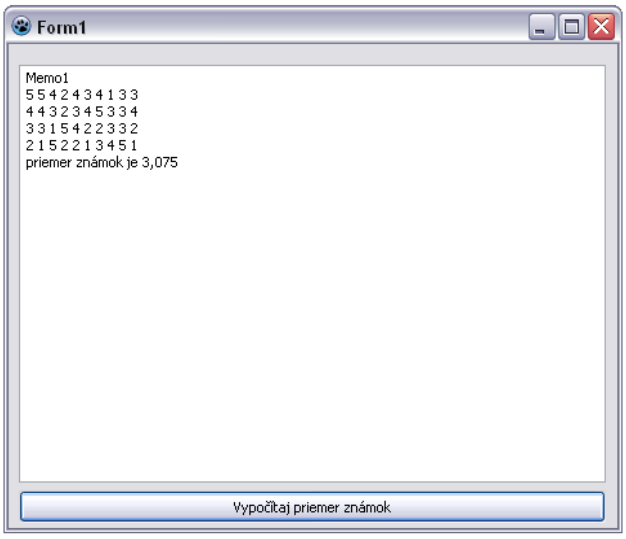
$$\begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} \\ b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} \end{pmatrix}$$

Pristupovanie k prvkom poľa: A[2,3], B[1,2],…

Zadanie 25:

V dvojrozmernom poli náhodne vygenerujte známky pre štyroch žiakov z 10 predmetov (od 1 do 5). Vypíšte túto maticu a priemer zo všetkých známok.

```
procedure TForm1.Button1Click(Sender: TObject);
type
  matica = array[1..100,1..100] of Integer;
var
  A: matica;
  I,J,M,N, sucet: Integer;
  riadok: String;
  priemer: Real;
begin
  M:=4;
  N:=10;
  sucet:=0;
  Randomize;
  for I := 1 to M do
      for J := 1 to N do
          A[I,J] := 1 + Random(5);
          for I := 1 to M do
              begin
                riadok := '';
                for J := 1 to N do
                    riadok := IntToStr(A[I,J]) + ' ' + riadok;
                    Memo1.Lines.Add(riadok);
              end;
          for I:= 1 to M do
              for J:= 1 to N do
                  sucet:= sucet + A[I, J];
                  priemer:=sucet/(M*N);
                  Memo1.Lines.Append('priemer známok je ' + FloatToStr(priemer));
end;
```



26. Vysvetlite spôsob vytvorenia textového súboru a zápisu do neho (deklarácia premennej typu súbor, procedúry na vytvorenie súboru, zápis do súboru, zatvorenie súboru). Vysvetlite použitie formátovacieho parametra.

Var Subor**: TextFile**; ..... deklarácia premennej s menom Súbor typu súbor.

AssignFile (premenná, meno súboru); ..... procedúra premennej typu TextFile priradí konkrétny súbor na disku. Meno súboru zadávame ako absolútnu cestu (napr. 'c:\udaje\mojsubor.txt'), alebo ako relatívnu cestu (napr. 'mojsubor.txt') pre súbor v tom istom priečinku, ako aplikácia. Napríklad: AssignFile(Subor, 'mojsubor.txt');

Rewrite(Subor); ..... vytvorenie súboru, otvorenie pre zápis.

Znamená to, že súbor sa najprv na disku vytvorí ako úplne prázdny (ak už existoval, tak sa vyprázdni) a nastaví sa režim, v ktorom môžeme do súboru zapisovať nejaké hodnoty.

Write(premenná, hodnota); .....zapisovanie do súboru.
WriteLn(premenná, hodnota); .....zapisovanie do súboru a prechod na nový riadok.

Napr.:
WriteLn(Subor, 37); .... vypíšeme do súboru hodnotu 37 a prejdeme na nový riadok.
WriteLn(Subor, 10, 12); .....vypíše: 1012 ..... jedno číslo
WriteLn(Subor, 10, ' ', 12); .....vypíše: 10 12 ..... dve čísla
WriteLn(Subor, I**:5**); .... **formátovací parameter 5** určuje rezervované miesto pre výpis čísla, ktorý bude zarovnaný doprava. Ak to miesto nepostačuje, vypisuje sa aj za toto rezervované miesto.

Napr. príkaz **WriteLn(Subor, I:5);** pre rôzne celé číslo **I** zapíše:
pre I = 123 *zapíše* ' 123'
pre I = 7 *zapíše* ' 7'
pre I = 65535 *zapíše* '65535'
pre I = 1234567 *zapíše* '1234567'

Pre reálne číslo R = 3,14 môžeme použiť dva formátovacie parametre
(druhý parameter udáva počet desatinných miest):
Writeln(Subor, R); zapíše '3.14000000000000E+000'
Writeln(Subor, R:7); zapíše '3.1E+000'
Writeln(Subor, R:7:3); zapíše ' 3.140'
Writeln(Subor, R:7:1); zapíše ' 3.1'
Writeln(Subor, R:0:2); zapíše '3.14'

CloseFile(Subor); ..... zatvorenie súboru.
Súbor musíme zatvoriť, aby sa korektne prerušilo naše napojenie pomocou súborovej premennej a aby operačný systém potom vykonal všetky nevyhnutné činnosti na disku.

Memo1.Lines.**LoadFromFile**(meno súboru); .... príkaz, ktorým vypíšeme obsah ľubovoľného textového súboru do textovej plochy.

Zadanie 26:

Vytvorte program, ktorý do súboru postupne zapíše 100 čísel (od 1 do 100) tak, že v každom riadku bude 10 čísel. Použite formátovací parameter, aby boli čísla vypísané na rovnakej pozícii pod sebou.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Subor: TextFile;
  I: Integer;
begin
  AssignFile(Subor, 'cisla100.txt');
  Rewrite(Subor);
  for I := 1 to 100 do
  begin
    Write(Subor, I:4);
    if I mod 10 = 0 then
    WriteLn(Subor);      // writeLn - skočí na ďalší riadok !
  end;
  CloseFile(Subor);
  Memo1.Lines.LoadFromFile('cisla100.txt');
end;
```

27. Vysvetlite spôsob spracovania dát z textového súboru (otvorenie súboru, čítanie zo súboru). Vysvetlite použitie logických funkcií pre koniec riadku a koniec súboru.

Reset(Subor) ..... otvorenie súboru na čítanie.
Súbor musí na disku už existovať a mal by byť pripravený korektne. Preto, ak má napr. číselný súbor obsahovať nejakých 20 celých čísel, tak ich tam nesmie byť menej a samozrejme, že v súbore nesmú byť žiadne nevhodné znaky.

Existujú dva varianty príkazov na čítanie zo súboru.
Read(Subor, premenná); ..... načíta dáta zo súboru do premennej.
Prvým parametrom musí byť súborová premenná, ktorá odkazuje už na otvorený súbor. Druhým parametrom musí byť premenná (Integer, Real, string, char, podľa toho, čo sa nachádza v súbore).
ReadLn(Subor, premenná); ..... načítanie dát zo súboru do premennej a prechod na nový riadok.

Napr.:
ReadLn(Subor, retazec); ..... načítanie slova alebo celého riadku zo súboru a prechod na nový riadok.
ReadLn(Subor, znak); ..... načítanie znaku zo súboru a prechod na nový riadok.

Príkaz **ReadLn(Subor)**; ... nastaví ďalšie čítanie na začiatok nasledujúceho riadka.

V Pascale máme dve pomocné logické funkcie, ktoré nám o otvorenom súbore prezradia, či v súbore, resp. v momentálnom riadku máme ešte čo čítať, alebo sme už na konci.

Funkcia **SeekEof** (Eof je - "end of file", t.j. "koniec súboru") ..... ak sme už na konci súboru vráti True, alebo vráti False, ak ešte nie sme na konci súboru.

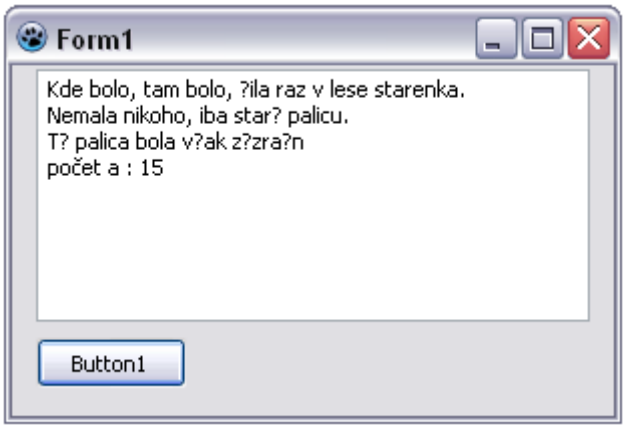
Funkcia **Eof** nepreskakuje prázdne riadky (napr. ak chceme počet všetkých riadkov).

Funkcia **SeekEoln** (Eoln - "end of line", t.j. "koniec riadka") ..... táto funkcia vráti True vtedy, keď v momentálne čítanom riadku už nie sú ďalšie hodnoty (možno sú tam už len medzery).
Na rozdiel od SeekEof, táto funkcia už nepreskakuje na nový riadok, ale čítanie ostane nastavené na konci tohto riadka.

Zadanie 27:

Vytvorte program, ktorý zistí, koľkokrát sa v texte v súbore (*príloha_27.txt)* vyskytuje písmeno „a“. Do textovej plochy Memo vypíšte text súboru, aj počet výskytov.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  subor: textfile;
  znak:char;
  znaky:Integer;
begin
  AssignFile(subor, 'priloha_27.txt');
  Reset(subor);
  znaky:=0;
  while not Eof(subor) do
  begin
    read(subor, znak);
  if znak = 'a' then Znaky:= znaky +1;
  end;
  closeFile(subor);
  Memo1.Lines.loadfromfile('priloha_27.txt');
  memo1.lines.Append('počet a : ' + inttostr(znaky));
end;
```



28. Vysvetlite pojem funkcia, deklarujte funkciu bez parametrov. Popíšte mechanizmus, ktorý prebehne pri volaní funkcie.

Funkcia tak ako procedúra sa používa ako podprogram. Funkcia sa od procedúry líši tým, že okrem vykonania sady príkazov **vráti nejakú hodnotu**. Hodnotu, ktorá sa má vrátiť, určuje špeciálna premenná ***Result***. Táto premenná sa správa rovnako ako obyčajná lokálna premenná:

- je rovnakého typu ako typ funkcie,
- na začiatku má nedefinovanú hodnotu,
- hodnota, ktorú má Result pri skončení podprogramu sa zapamätá a teda vráti ako hodnota funkcie.

Funkcia bez parametrov sa deklaruje kľúčovým slovom **function**:

```
function Meno_funkcie:typ výsledku;
begin
...
end;
```

napr.:

```
function HodKockou: Integer;
begin
   Result := Random(6) + 1;
end;
```

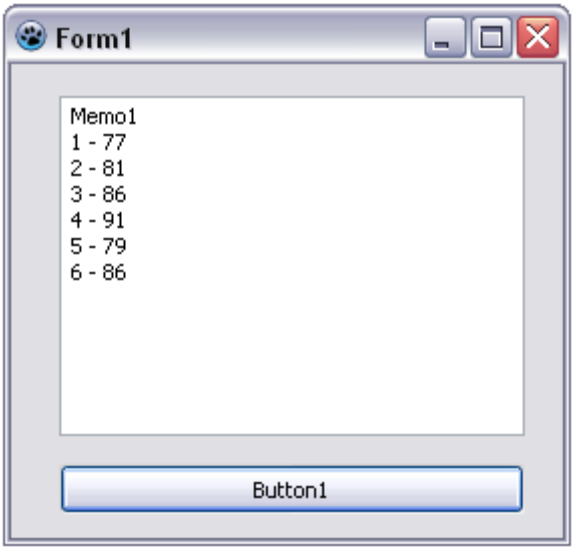
Mechanizmus volania funkcie:

1. zapamätá sa návratové miesto (riadok programu, kam sa vrátime po skončení funkcie);
2. vytvoria sa premenné:
   - všetky zadeklarované lokálne premenné funkcie (prázdne),
   - parametre (sú tiež lokálne premenné - do nich sa priradia príslušné vstupné hodnoty,
   - automaticky sa vytvorí špeciálna lokálna premenná Result, (rovnakého typu ako typ funkcie) – zatiaľ má nedefinovanú hodnotu;
3. prenesie sa riadenie programu do tela podprogramu (za príslušný begin) vykonajú sa všetky príkazy podprogramu (až po koncový end);
4. zrušia sa lokálne premenné – a teda sa zrušia aj parametre; pri tom sa zapamätá hodnota špeciálnej premennej Result;
5. riadenie sa vráti na návratové miesto – sem sa dosadí zapamätaný výsledok funkcie.

Zadanie 28:

Simulujte hod kockou (500 krát) s použitím funkcie HodKockou. Počet padnutí čísel (1 až 6) ukladajte do poľa a vypíšte.

```
function HodKockou: Integer;
begin
  Result := Random(6) + 1;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
 Pole: array [1..6] of Integer;
 I, J: Integer;
begin
  for I := 1 to 6 do
  Pole[I] := 0;
  for I := 1 to 500 do
  begin
    J := HodKockou;
    Inc(Pole[J]);          // alebo toto ... Pole[J] := Pole[J] + 1
  end;
  for J := 1 to 6 do
  Memo1.Lines.Append(IntToStr(J) + ' - ' + IntToStr(Pole[J]));
end;
```



29. Popíšte spôsob deklarácie funkcie s parametrami. Vysvetlite, s akými parametrami môžu funkcie pracovať, uveďte príklady štandardných funkcií.

Typ parametrov a aj typ výsledku treba zadefinovať v hlavičke funkcie:
function meno_funkcie(parametre): typ_výsledku;

Funkcie môžu mať:

- **jeden jednoduchý parameter** (Integer, Real, Char, string), výsledný typ funkcie je opäť rôzny, napr. preddefinované funkcie: IntToStr, RGBToColor, Red, Green, Round, Sqr, Sqrt, Sin, Cos, Random, Ord, Char, ...,
- **jeden štruktúrovaný parameter** – napr. premenná textový súbor (napr. funkcie Eof, Eoln), ale aj ľubovoľné pole (funkcie Length, High),
- **viac parametrov** (napr. Copy, Pos) - záleží na ich poradí.

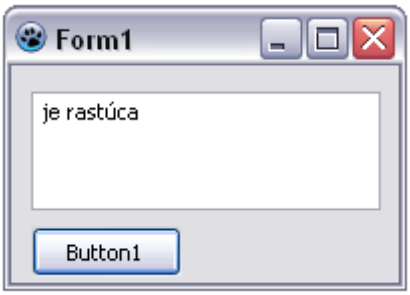
Parametre funkcií aj procedúr musia mať v hlavičke uvedené len identifikátory už predtým definovaných typov. Nové definované typy zvykneme v Pascale pomenovávať tak, že začínajú veľkým písmenom T, za ktoré píšeme upresňujúci text.
Napr:

```
type
  TPole = array [1..100] of Integer;
```

Zadanie 29:

Zapíšte logickú funkciu JeRastuca, ktorá o prvkoch poľa zistí, či tvoria rastúcu postupnosť.

```
function JeRastuca(P:array of integer) :boolean;
var
   I:Integer;
   begin
    result := true;
    for I:= 1 to 9 do
    if P[I]> P[I+1] then result:= false;
    end;
procedure TForm1.Button1Click(Sender: TObject);
var
   pole:array[1..10] of integer = (1,2,3,4,5,6,7,8,9,10);
begin
  Memo1.Clear;
   if JeRastuca(pole) then memo1.Lines.Append('je rastúca')
                      else memo1.lines.Append(' nie je rastúca');
end;
```



30. Vysvetlite pojmy priama a nepriama rekurzia. Vysvetlite mechanizmus rekurzívneho výpočtu faktoriálu prirodzeného čísla.

Rekurzia v programovaní znamená, že podprogram najčastejšie zavolá sám seba, t.j. že podprogram je definovaný pomocou samého seba.

· **rekurzia priama** – nastáva u tej procedúry alebo funkcie, ktorá má vo svojom tele príkaz vyvolania samej seba (u procedúry) alebo zápis samej seba (u funkcie). K rekurzívnemu vyvolaniu prichádza tak, že procedúra alebo funkcia vyvolá samu seba.

· **rekurzia nepriama** (vzájomná), nastáva vtedy, keď dochádza k vzájomnému vyvolávaniu sa procedúr alebo funkcií. Prvá procedúra vyvolá druhú, druhá tretiu, atď, (n-1)–vá vyvolá n–tú a n–tá vyvolá prvú.

- Príkazová časť rekurzívnej procedúry (funkcie) musí obsahovať vetvu pre triviálny prípad niektorého základného parametra alebo vstupnej hodnoty.
- Ľubovoľné volanie procedúry (funkcie) vo vnútri jej príkazovej časti musí mať vhodne redukovaný argument (alebo globálnu hodnotu). Pri tejto redukcii sa očakáva, že raz dosiahne triviálny prípad po konečnom počte opakovaní.

Počítame **Faktoriál** prirodzeného čísla **N**, pričom vieme, že

- 0! = 1 ... triviálny prípad
- n! = (n-1)!*n ... rekurzívne volanie

Triviálnym prípadom je tu úloha, ako vyriešiť **0!**. Toto vieme aj bez rekurzie, lebo je to 1. Ostatné prípady sú už rekurzívne: na to, aby sme vyriešili zložitejší problém (N faktoriál), najprv vypočítame jednoduchší (N-1 faktoriál) - zrejme pomocou rekurzie - a z neho skombinujeme (násobením) požadovaný výsledok.

Zadanie 30:

Pomocou rekurzívnej funkcie Mocnina počítajte n-tú mocninu prirodzeného čísla.

```
procedure TForm1.Button1Click(Sender: TObject);
var
 X,N: Integer;
function Mocnina(X,N: byte):Longint;
 begin
  if N=0 then Mocnina:=1
         else Mocnina:=X*Mocnina(X,N-1);
  end;
begin
 X:=StrToInt(Edit1.Text);
 N:=StrToInt(Edit2.Text);
 Memo1.Lines.Append(IntToStr(mocnina(X,N)));
 end;
```