Algoritmus

- konečná postupnosť presne definovaných inštrukcií na splnenie určitej úlohy/skupiny úloh
- recept, návod na použitie

Vlastnosti algoritmov

- **Rezultatívnosť** (konečnosť) - skončí po vykonaní *konečného* počtu krokov
- **Efektívnosť** - požadujeme, aby algoritmus bol efektívny, aby každá operácia bola dostatočne jednoduchá t.j. aby bola vykonateľná v konečnom čase dostupnými prostriedkami
- **Determinovanosť** - každý ďalší krok musí byť jednoznačne a presne definovaný
- **Hromadnosť** (všeobecnosť) - rieši všeobecnú triedu obdobných problémov (ako vypočítať súčin dvoch celých čísel)

Etapy algoritmizácie úloh

- formulácia úlohy a presné vymedzenie problému
- analýza úlohy
- zostavenie riešiaceho algoritmu – zápis jednotlivých krokov (príkazov) algoritmu (algoritmický jazyk, vývojový diagram)
- realizácia – prepis algoritmu do programovacieho jazyka
- ladenie a testovanie
- dokumentácia a údržba

Lazarus - začíname

Otvorenie novej aplikácie:

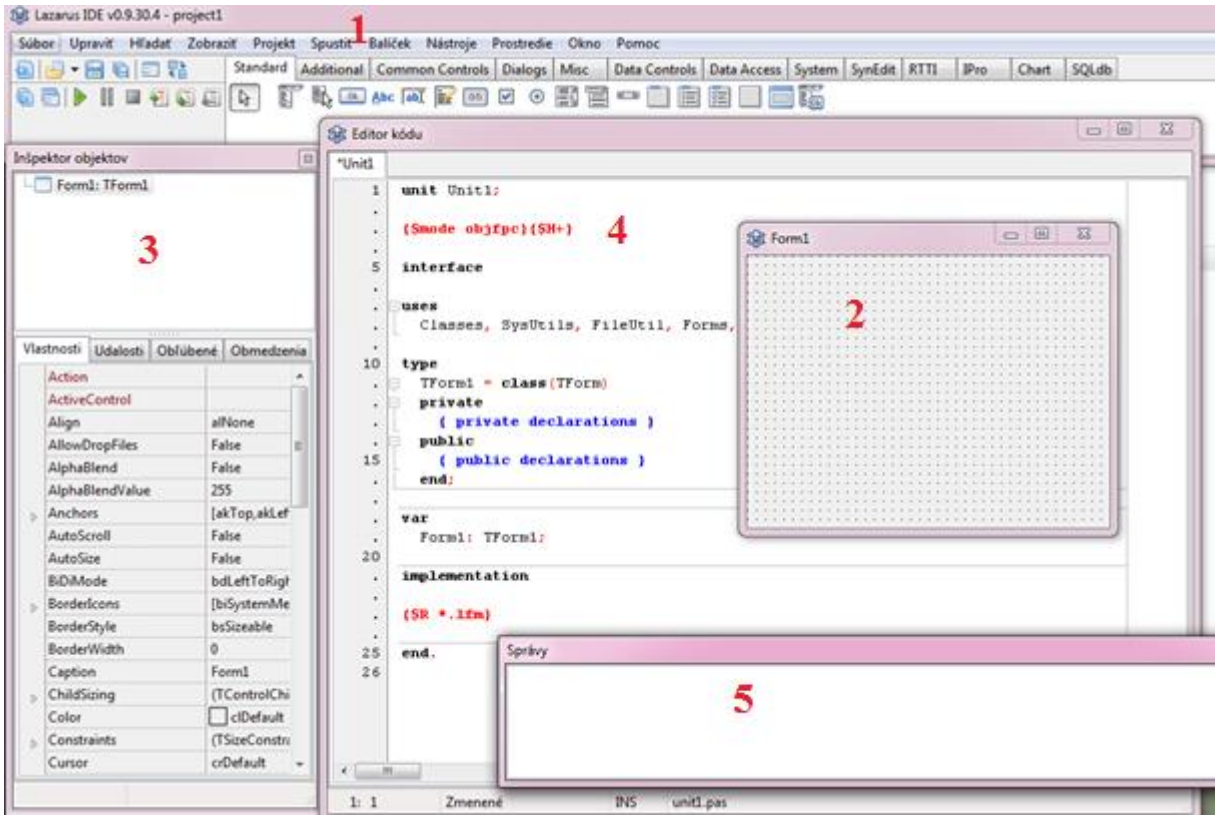
- súbor → nový → aplikácia

Uloženie programu:

- súbor → uložiť ako
- každý program ukladáme vždy do samostatného priečinka (2krát sa nás program opýta či ho chceme uložiť)

Základné pravidlá:

- z vygenerovaného kódu, ktorý sa zobrazí po vytvorení novej aplikácie nič nemažeme. Program musí mať na konci end s bodkou t.j. **`end.`** ktorý hovorí, že tu program končí.
- všetky príkazy píšeme medzi `begin` a `end` tlačidiel (buttonov)
- každý príkaz píšeme na nový riadok a musí byť ukončený bodkočiarkou **`;`**
-  spustenie programu (zelený trojuholník)
-  zastavenie programu (červený štvorček)



1: PALETA KOMPONENTOV

2: FORMULÁR – vkladám doň komponenty

3: INŠPEKTOR KOMPONENTOV – v ňom nadstavujem vlastnosti komponentov

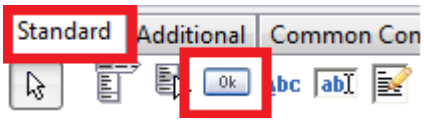
4: EDITOVACIE OKNO – miesto, kde píšem kód/program/to čo sa má vykonať

5: SPRÁVY – chybové hlásenia

Komponenty

`ButtonClick` (tlačidlo)

- príkazy, ktoré napíšeme medzi `begin` a `end` `ButtonClick`-u sa vykonajú, keď program spustíme (vytvoríme aplikáciu) a klikneme na tlačidlo.



Keď do formulára vložím Button, dvakrát naň kliknem a v editovacom okne sa vygeneruje kód:

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

    //tu píšem to, čo sa má urobiť, keď stlačím button po spustení aplikácie (príkazy, kód).

    //tento príkaz čítam: keď sa vytvorí aplikácia a kliknem na tlačidlo (button) vykonaj/urob to, čo je napísané medzi begin a end.

end;
```

Vlastnosti tlačidla:

- šírka **`Width`**
- výška **`Height`**
- popis na tlačidle **`Caption`**
- umiestnenie v rámci formulára vo vodorovnom smere (x-ová súradnica) **`Top`**
- umiestnenie v rámci formulára vo zvislom smere (y-ová súradnica) **`Left`**

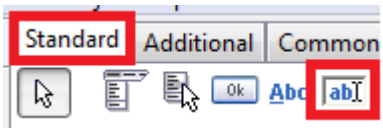
Nadstavenie vlastností tlačidla:

- v inšpektore objektov
- v programe → v `ButtonClick`-u: musím osloviť `Button` jeho menom a cez bodku mu povedať, ktorú vlastnosť nadstavujem a pomocou príkazu priradenia **`:=`** nadstavím konkrétnu hodnotu.
  - `Button1.Width:=300;`[1]
  - `Button1.Height:=150;`
  - `Button1.Caption:='môj text';`
  - `Button1.Top:=10;`
  - `Button1.Left:=40;`

[1] Príkaz `Button1.Width:=300;` čítame: Tlačidlu jedna (`buttonu 1`), jeho šírke (`width`) priraď/ nadstav/ vlož (`:=`) hodnotu `300` koniec príkazu (`;`).

`Edit` (editovacie okno)

- po spustení programu sa stane interaktívnym prvkom programu – môžeme mu meniť hodnotu a s touto hodnotou pracovať.



Vlastnosti `Edit`-u:

- Aktuálna hodnota v Edite/to čo je v Edite napísané **`Text`**
- umiestnenie v rámci formulára vo vodorovnom smere (x-ová súradnica) **`Top`**
- umiestnenie v rámci formulára vo zvislom smere (y-ová súradnica) **`Left`**

Nadstavenie vlastností `Edit`-u:

- v inšpektore objektov
- v programe → v ButtonClick-u: musím osloviť `Edit` jeho menom a cez bodku mu povedať, ktorú vlastnosť nadstavujem a pomocou príkazu priradenia **`:=`** nadstavím konkrétnu hodnotu.
  - `Edit1.Text:='môj text';`
  - `Edit1.Top:=10;`
  - `Edit1.Left:=40;`

Text v EDITE: čokoľvek čo napíšeme do Editu sa považuje za text. Príkazy `Edit1.Text:='Ahoj';` a `Edit1.Text:='1236';` sú pre Lazarus/program rovnocenné (v oboch prípadoch vidí iba text/slovo/postupnosť znakov). Ak chceme, aby sa na číslo 1236 program nepozeral ako na text, ale ako na číslo, musíme použiť konverziu medzi týmito formátmi (použijeme čarovné slovíčko, aby sa text stal číslom):

`StrToInt``(Edit1.Text)`

Použitie uvidíme na hodine, je to skratka od: string to integer (reťazec na celé číslo).

Zakomentovanie príkazu – občas potrebujeme skryť nejaký riadok kódu – nevieme čo robí alebo je v ňom chyba, ktorú nevieme odstrániť, ale aj napriek tomu chceme program spustiť a príkaz nezmazať. Aby sme nemuseli riadok vymazať, môžeme ho zakomentovať t.j. zneviditeľniť pre program.

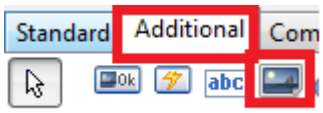
Použijeme na to dve lomítka, ktoré napíšeme pred začiatok riadka, ktorý chceme zakomentovať. Zakomentovaný riadok zmodrá a pri spúšťaní programu akoby neexistoval.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  //ShowMessage('Ahoj');
  Button1.Caption:=Button2.Caption; //iba tento príkaz sa vykoná po spustení programu
  //Button2.Caption:=Button1.Caption;
end;
```

Ak chceme zakomentovať viac riadkov pod sebou môžeme použiť na zakomentovanie zložené zátvorky. Zakomentované riadky opäť zmodrajú.

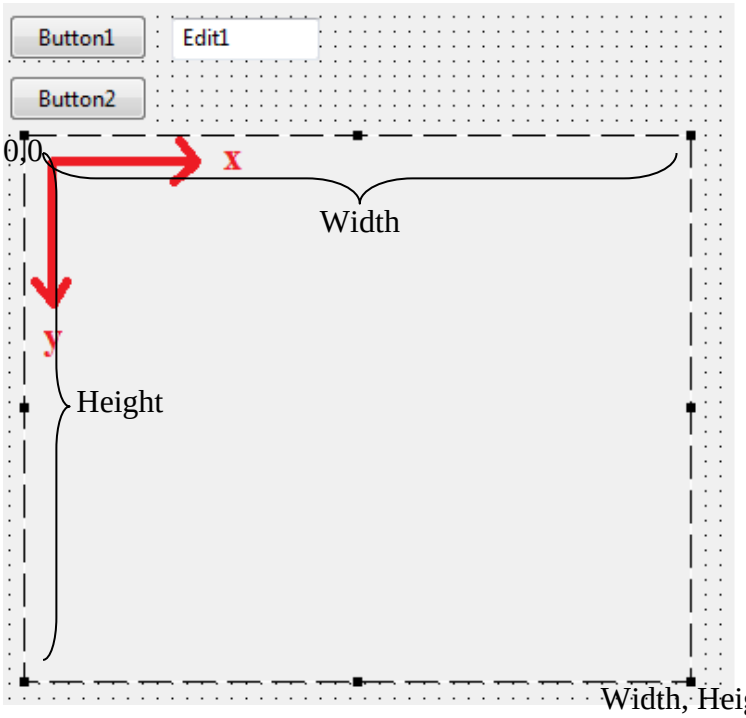
```
procedure TForm1.Button1Click(Sender: TObject);
begin
  {ShowMessage('Ahoj');
  Button1.Caption:=Button2.Caption;}
  Button2.Caption:=Button1.Caption;  //iba tento príkaz sa vykoná po spustení programu
end;
```

Image



- grafická plocha, do ktorej môžeme kresliť ale aj písať text.

V `Image`-i sa orientuje pomocou súradníc, na ktoré vykresľujeme text a grafické príkazy. V ľavom hornom rohu je súradnica `0,0` a v pravom dolnom rohu je súradnica `Image1.Width`, `Image1.Height`. Nepohybujeme sa v cm/mm, ale jednotka pre meranie sú pixle.



Vlastnosti `Image`:

- šírka **`Width`**
- výška **`Height`**
- umiestnenie v rámci formulára vo vodorovnom smere (x-ová súradnica) **`Top`**
- umiestnenie v rámci formulára vo zvislom smere (y-ová súradnica) **`Left`**

Nadstavenie vlastností `Image`-u:

- v inšpektore objektov
- v programe → v ButtonClick-u: musím osloviť `Image` jeho menom a cez bodku mu povedať, ktorú vlastnosť nadstavujem a pomocou príkazu priradenia **`:=`** nadstavím konkrétnu hodnotu.
  - `Image1.Width:=250;`
  - `Image1.Height:=500;`
  - `Image1.Top:=10;`
  - `Image1.Left:=40;`

Príkazy v `Image1`: Aby sme mohli do `Image` písať alebo kresliť musíme ho osloviť jeho menom `Image1`, povedať, že ideme kresliť na jeho plátno `Canvas` a cez ďalšiu bodku povieme, ktorý príkaz sa má použiť a nadstavíme príslušné hodnoty.

Hneď po vytvorení aplikácie je pre:

- pozadie Image nastavená čierna farba
- štetec nastavená biela nepriehľadná farba
- pero nastavená čierna farba a hrúbka 1
- písmo nastavená čierna farba

Vyfarbenie (vybielenie) Image-u (po spustení programu je `Image` čierny, preto ho môžeme vybieliť (vyfarbiť farbou, ktorú máme nastavenú v štetci - `brush`))

```
Image1.Canvas.FillRect(Image1. ClientRect);
```

Text v Image (na súradniciach `x=10` a `y=30` sa vypíše slovo v apostrofoch):

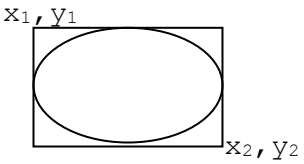
```
Image1.Canvas.TextOut(10,30, 'Ahoj');
```

Vlastnosti písma, s ktorým píšem:

```
Image1.Canvas.Font.Color:=clBlack;

Image1.Canvas.Font.Size:=20;

Image1.Canvas.Font.Style:=[fsBold, fsItalic];

Image1.Canvas.Font.Name:='Arial';

//Times New Roman, Comic Sans MS, Verdana, Tahoma, Courier
```

Obdĺžnik

```
Image1.Canvas.Rectangle(x1,y1,x2,y2);
```



Elipsa

```
Image1.Canvas.Ellipse(x1,y1,x2,y2);
```

Čiara (čiaru kreslíme perom, ktoré je umiestnené v ľavom hornom rohu).

- Ak chceme nakresliť čiaru, ktorá vychádza z ľavého horného rohu a končí v bode so súradnicami `x=50` a `y=80` stačí nám príkaz:
  ```
  Image1.Canvas.LineTo(50,80);
  ```
- Ak chceme nakresliť čiaru, ktorá vychádza z bodu `x=100` a `y=80` a končí v bode `x=200` a `y=190` musíme použiť dvojicu príkazov:
  ```
  Image1.Canvas.MoveTo(100,80);//presunie pero na novú pozíciu
  Image1.Canvas.LineTo(200,190);//nakreslí čiaru z novej pozície
  ```

Vlastnosti pera, s ktorým kreslíme čiary (obrysy):

Farba pera

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

Hrúbka pera

```
Image1.Canvas.Pen.Width:=10;
```

Typ čiary pera

```
Image1.Canvas.Pen.Style:=psDot;
```

- `psDash`
- `psDashDot`
- `psDashDotDot`
- `psClear`
- `psSolid`

♥ **clBlue**	modrá	**clWhile**	biela
♥ **clRed**	červená	♥ **clBlack**	čierna
♥ **clGreen**	zelená	♥ **clGray**	šedá
♥ **clYellow**	žltá	♥ **clSilver**	strieborná
♥ **clNavy**	námornícka modrá	♥ **clOlive**	olivová
♥ **clMaroon**	hnedá	♥ **clTeal**	modrozelená
♥ **clAqua**	azúrová	♥ **clLime**	limetková
♥ **clPurple**	fialová	♥ **clFuchsia**	svetlofialová

Vlastnosti štetca, s ktorým vyfarbujeme:

Farba štetca

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

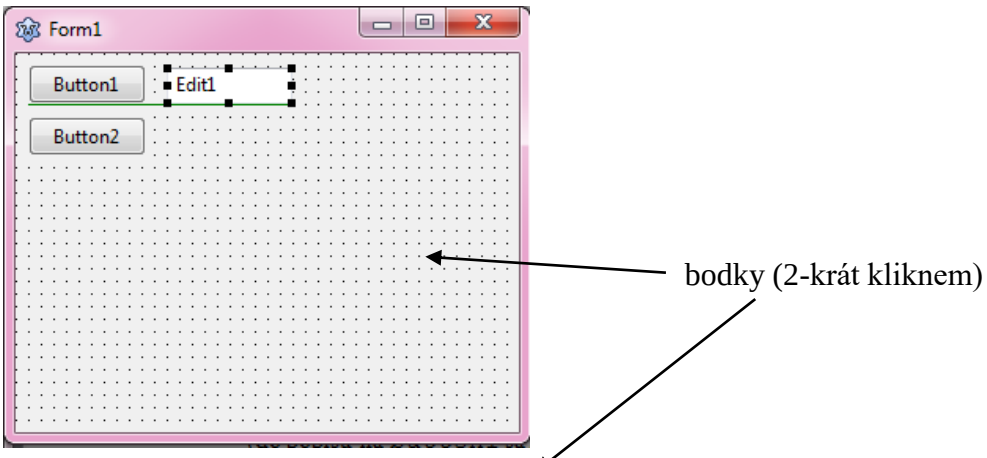
Typ (vzor) výplne

```
Image1.Canvas.Brush.Style:=bsCross;
```

- `bsCross`
- `bsDiagCross`
- `bsFDiagonal`
- `bsHorizontal`
- `bsVertical`
- `bsClear //urobí farbu v štetci priehľadnú (vypne farebnosť)`
- `bsSolid //vypne priehľadnosť štetca`

Procedúra `FormCreate` – doteraz sme všetky hodnoty nadstavovali až po kliknutí na tlačidlo. Ak vytvoríme procedúru `FormCreate`, všetky **príkazy** v nej **sa vykonajú práve raz**, a to **hneď po vybudovaní aplikácie** t.j. ešte predtým ako klikneme na `button`.

Procedúru `FormCreate` vytvoríme tak, že 2-krát klikneme na bodky vo formulári.




```
procedure TForm1.FormCreate(Sender: TObject);
begin
```

akýkoľvek príkaz, ktorý tu napíšem sa vykoná ešte predtým ako stihneme stlačiť `button`

```
end;
```

`Label` (jednoduchý text)

- text, ktorý napíšeme do popisu `Label`-u → `Caption` sa v ňom zobrazí.

Vlastnosti `Label`-u:

- popis **`Caption`**
- farba písma popisu **`Font.Color`**
- veľkosť písma popisu **`Font.Size`**
- umiestnenie v rámci formulára vo vodorovnom smere (x-ová súradnica) **`Top`**
- umiestnenie v rámci formulára vo zvislom smere (y-ová súradnica) **`Left`**

Nadstavenie vlastností `Label`-u:

- v inšpektore objektov
- v programe → v ButtonClick-u: musím osloviť `Label` jeho menom a cez bodku mu povedať, ktorú vlastnosť nadstavujem a pomocou príkazu priradenia **`:=`** nadstavím konkrétnu hodnotu.
  - `Label1.Font.Size:=30;`
  - `Label1.Font.Color:=clBlue;`
  - `Label1.Caption:='môj text';`
  - `Label1.Top:=10;`
  - `Label1.Left:=40;`

Vlastnosti komponentov sme nadstavovali (komponentom sme nadstavili nejakú hodnotu) pomocou príkazu priradenia `:=`. Avšak komponentu možno priradiť (nadstaviť) aj hodnotu, ktorú má iný komponent. Napr.:

- `Button1.Caption:=Button2.Caption;`
  (do popisu na `button1` sa nadstaví to, čo má na sebe napísané `button2`)
- `Button1.Caption:=Label1.Caption;`
  (do popisu na `button1` sa napíše popis `label1`)

Náhoda

Ak chceme, aby si program vymyslel (vybral) náhodné číslo použijeme funkciu **random**. Funkcia má jeden parameter, je ním číslo:

```
random(250)
```

Znamená to, že počítač si vyberie ľubovoľné číslo z intervalu 0 – 249, napr. 58.

POZOR: po opakovanom spustení aplikácie si môžeme všimnúť, že random si vyberá stále to isté náhodné číslo/farbu, preto je potrebné uviesť do činnosti procedúru

```
randomize;
```

ktorá zabezpečí, že čísla sa budú naozaj vyberať náhodne. Použitie uvidíme na hodine.

Použitie:

```
Image1.Canvas.Font.Size:=random(28);
Image1.Canvas.Rectangle(random(70),150, random(200),390);
```

Náhodná farba

Všetky farby sú zložené z 3 základných farieb – červená, zelená, modrá (RGB model) a výsledná farba závisí od toho, ako je v nej zastúpená každá z týchto troch farieb. Zastúpenie vyjadrujeme pomocou čísel 0 - 255.

Žltú vieme namiešať ako červenú a zelenú, teda RGB model vyzerá (255, 255, 0). Ak zvolíme iné hodnoty pre červenú a zelenú dostaneme iné odtiene žltej.

čierna: (0, 0, 0)
červená: (255, 0, 0)
biela: (255, 255, 255)

Použitie:

```
Image1.Canvas.Brush.Color:=random(256* 256*256); //úplne náhodná farba

Image1.Canvas.Pen.Color:=RGBToColor(random(256),0,0); //náhodné odtiene červenej

Image1.Canvas.Font.Color:=RGBToColor(0,random(256),0); //náhodné odtiene zelenej

Image1.Canvas.Brush.Color:=RGBToColor(random(256),0,random(256));
                                                    //náhodné odtiene fialovej

Image1.Canvas.Pen.Color:=RGBToColor(159,45,214);
```

Premenná

Občas potrebujeme niektoré hodnoty uchovať, uložiť, odložiť si niekam „nabok“. Vtedy použijeme premenné.

Premenná je **pamäťové miesto**, miesto na ukladanie, košík, **do ktorého vieme vkladať** (ukladať, ale aj z neho čítať) **hodnoty** (my sa zatiaľ budeme rozprávať len o celých číslach/celočíselných hodnotách).

V programe môžeme používať neobmedzený počet premenných, avšak každá premenná musí mať svoje meno/názov. Názov premennej musí spĺňať isté kritériá:

- skladá sa len z písmen americkej abecedy (nepoužívame diakritiku), číslic a podtrhovníka _
- nerozlišujú sa veľké a malé písmená
  `strana`, `Strana`, `StrANa` sú tie isté slová
- nesmie začínať číslicou
  správne: `strana1`, `strana2` nesprávne: `1strana`, `3vysky`
- nesmie sa zhodovať s niektorých rezervovaným slovom programovacieho jazyka (begin, procedure...)
- mal by čo najvýstižnejšie vyjadrovať hodnotu, ktorú bude obsahovať
  správne: `stranaA`, `vyska`, `x`, `y` nesprávne: `Jozko`, `hodnota1`, `hodnota3`

Premenné musíme v programe **zadeklarovať**. Deklaráciu píšeme medzi `begin` a `end` procedúr.

variable (premenná) typ premennej (v našom prípade je to celé číslo)

názvy našich premenných

```
procedure TForm1.Button1Click(Sender: TObject);
var  stranaA, vyskaTroj, x, y, z : integer;
begin
  stranaA:= 50;
end;
```

premenné pozná iba tlačidlo, v ktorom sú deklarované

Aké čísla môžem uložiť do integeru? Celé čísla sa uchovávajú v 32 bitoch, t.j. do premennej typu integer môžeme vložiť čísla z intervalu $\langle -2\,147\,483\,648,\ 2\,147\,483\,647\rangle = \langle -2^{31}, 2^{31}\rangle$.

Výpis premennej: Ak chceme premennú vypísať pomocou `TextOut` alebo vložiť ju do `Edit1.Text` treba použiť konverziu, ktorá číslo premení na text

```
IntToStr(stranaB)
```

Použitie uvidíme na hodine, je to skratka od: integer to string (celé číslo na reťazec).

Delenie celých čísel

Ak chceme deliť celé čísla, môžeme:

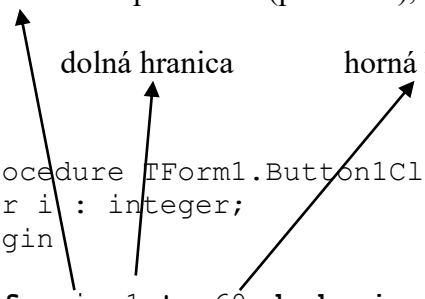
- podiel zaokrúhliť (zaokrúhľuje sa stále smerom nadol) funkciou **`trunc`**
  `vysledok:=trunc(a/b);`
- deliť so zvyškom (7:4 = 1 zvyšok 3) funkciami **`div`** a **`mod`**
  div dáva celú časť čísla po celočíselnom delení
  mod dáva zvyšok po celočíselnom delení

```
var a, b, celaCast, zvysok : integer;
begin
  a:=7;
  b:=4;
  celaCast:= a div b; //celaCast=1
  zvysok:= a mod b; //zvysok=3
end;
```

FOR cyklus – cyklus s pevným počtom opakovaní

Ak chceme niektorý kus kódu opakovať viackrát po sebe použijem cyklus.

`i` je riadiaca premenná (počítadlo), ktorá sa stará o to, aby sa cyklus opakoval koľkokrát chceme




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

  for i:=1 to 60 do begin
     //príkazy, ktoré sa majú vo for cykle opakovať
  end;
end;
```

pre cyklus `for i:=A to B do` platí:

- ak `A<B`, for cyklus sa vykoná `A-B+1krát`
- ak `A=B`, for cyklus sa vykoná raz
- ak `A>B`, for cyklus sa nevykoná ani raz

OPAČNÝ VARIANT FOR CYKLU: cyklus prebehne s klesajúcim počítadlom.

```
  for i:=10 downto 1 do begin
     //príkazy, ktoré sa majú vo for cykle opakovať
  end;
```

While cyklus – cyklus s neznámym počtom opakovaní, cyklus s podmienkou

While cyklus používame vtedy, ak chceme nejakú časť programu opakovať, ale nevieme koľkokrát sa má daný príkaz zopakovať, ale poznáme podmienku kedy sa má opakovanie skončiť.

Čítame: pokiaľ platí *podmienka* urob (príkazy)...

```
while podmienka do begin
  //príkazy, ktoré sa majú vo while cykle opakovať, ak platí podmienka
end;
```

podmienky môžu vyzerať, napr. takto:

- y < 200
- strana < > 250+x
- (x > 300) and (x < 450)
- (stranaA < 30) or (stranaB > 60)

Rozhodovanie sa – príkazy vetvenia

Občas sa potrebuje rozhodnúť, urobiť niečo ak sú splnené nejaké podmienky, resp. na základe nejakej podmienky robiť jednu vec, ak podmienka platí, alebo robiť inú vec ak podmienka neplatí. Na to slúži príkaz vetvenia – **`if`** alebo **`if else`**

Čítame: ak *podmienka* platí urob (príkazy)...

```
if podmienka then begin
  //príkazy ak podmienka platí;
end;
```

Čítame: ak *podmienka* platí urob (príkazy1), inak (ak podmienka neplatí) urob (príkazy2)...

(POZOR !!PRED ELSE NIE JE NIKDY BODKOČIARKA!!)

```
if podmienka then begin
   //príkazy1 ak podmienka platí;
end
else begin
  //príkazy2 ak podmienka neplatí;
end;
```

Ak chceme riešiť viac ako tri možnosti je lepšie použiť príkaz **case** (aby sme sa v ifoch a else vetvách nestratili). Podmienku delíme (trháme) na 2 časti: za case píšeme výraz, ktorý sa má vyhodnotiť, do vetiev píšeme hodnoty, ktoré môže daný výraz nadobudnúť.

```
case x of
1 : begin
      //príkazy, ktoré sa majú vykonať, ak je hodnota x rovná 1
end;
2 : begin
      //príkazy, ktoré sa majú vykonať, ak je hodnota x rovná 2
end;
3 : begin
      //príkazy, ktoré sa majú vykonať, ak je hodnota x rovná 3
end;
4 : begin
      //príkazy, ktoré sa majú vykonať, ak je hodnota x rovná 4
end;
```

Príkaz case môžeme použiť aj s else vetvou, opäť pred else nie je bodkočiarka:

```
case x of
1 : begin
      //príkazy, ktoré sa majú vykonať, ak je hodnota x rovná 1
end;
2 : begin
      //príkazy, ktoré sa majú vykonať, ak je hodnota x rovná 2
end;
3 : begin
      //príkazy, ktoré sa majú vykonať, ak je hodnota x rovná 3
end
else begin
      //príkazy, ktoré sa majú vykonať, ak je hodnota x rovná 4
end;
```