

# **Основы программирования в Lazarus**

Гвасалия Д.А.

# Объекты

Объекты характеризуются

- *свойствами* (цвет, положение на экране и пр.)
- *методами* (действия или задачи которые выполняет объект)
- *событиями* (на какое событие должен реагировать объект).

# Обработка событий

Многие объекты, в том числе и кнопки, должны как-то реагировать на различные события, например на пользовательское щелканье мышкой. Чтобы сделать их способными на ответные действия, нужно написать **процедуру обработки события**. Самое распространенное событие — Click (Щелчок), пользователь навел курсором на объект и щелкнул кнопкой мыши.

# Структура процедур обработки событий

```
PROCEDURE TForm1.Button1Click( );
```

```
CONST {описание постоянных}
```

```
<имя постоянной> = <значение>;
```

```
VAR {определение лок. переменных}
```

```
<имя переменной> : <тип данных>;
```

```
<имя переменной> : <тип данных>;
```

```
BEGIN
```

```
{код процедуры обработки события}
```

```
END;
```

# Переменные

**Переменная** – представляет собой зарезервированное место в оперативной памяти для временного хранения данных.

Переменная характеризуется **именем и типом данных**

**Имя переменной** – это слово, которое используется для ссылки на значение, которое содержит переменная

# Соглашение об именах

Имена переменных, констант, процедур, объявляемых в программе должны соответствовать следующим требованиям:

- начинаться с латинской буквы
- должны отсутствовать пробелы
- не должны содержать точки
- должны отличаться от ключевых слов языка
- длина не должна превышать 255 символов

Можно: `CurrentNum`, `Total`, `Date_Of_Birth`

Нельзя: `1Time`, `$Total`, `Date of Birth`

# Тип данных

**Типы данных** относятся к самым фундаментальным свойствам любого языка программирования. Данные, с которыми работает программа, хранятся в оперативной памяти. Компилятору необходимо знать, сколько места они занимают, и какие действия с ними можно выполнять.

**Т.О. тип данных** однозначно определяет возможный диапазон значений переменной и допустимые над ней действия.

# Ввод данных

Присвоение значений переменной – используется оператор присвоения (**:=**)

**Переменная := Выражение;**

**Пример:** time = 10;  
Name = 'Иванов';

Ввод данных в программу осуществляется путем изменения свойства объекта **TEdit** в коде.

Например, ввод значения переменной *N* через **Edit1**

**N := StrToInt(Edit1.Text);**

**StrToInt()** – функция преобразует текстовую информацию, введенную в Edit1, в цифровую типа Integer

**StrtoFloat()** – функция преобразует текстовую информацию, введенную в Edit1, в действительное число

# Вывод результатов вычислений на экран

Вывод данных в программе осуществляется также путем изменения свойства объекта **TEdit** в коде. Например, вывод значения переменной *C* через Edit2

```
Edit2.Text := InttoStr(C);
```

функция **InttoStr()** преобразует цифровую информацию в текстовую и данный текст присваивается свойству **Text** объекта **TEdit**

# Математические операторы

Оператор	Выполняемая операторы	Пример
+	сложение	$C := A + B$
-	вычитание	$C := A - B$
*	умножение	$C := A * B$
/	деление	$C := A / B$
mod	остаток от деления нацело	$C := A \text{ mod } B$
div	целочисленное деление	$C := A \text{ div } B$

# Приоритет операций

Операторы	Порядок вычисления
( )	Первыми всегда вычисляются значения в круглых скобках.
-	Создание отрицательного числа (смена знака).
* /	умножение и деление.
div	целочисленное деление.
mod	остаток от деления.
+ -	Последние - это сложение и вычитание.

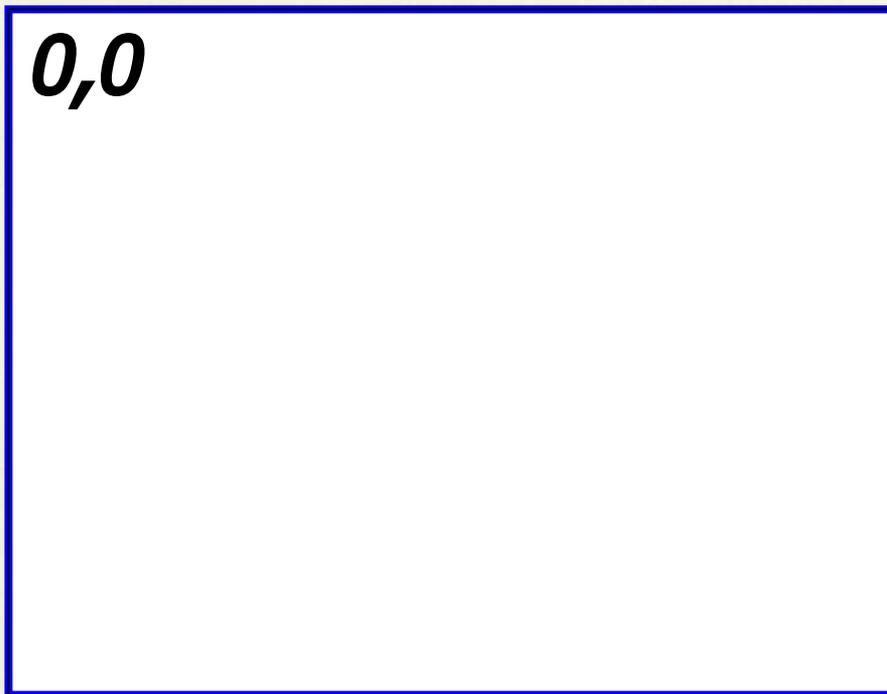
# Математические функции

Метод	Назначение
<b>Abs(n)</b>	абсолютное значение числа n.
<b>ArcTan(n)</b>	арктангенс числа n в радианах.
<b>Cos(n)</b>	косинус угла n. Угол n задается в радианах.
<b>Sin(n)</b>	синус угла n. Угол n задается в радианах.
<b>Exp(n)</b>	константа e, возведенную в степень n.
<b>Ln(x)</b>	натуральный логарифм от числа x.
<b>Sqr(x)</b>	квадрат числа x.
<b>Sqrt(x)</b>	квадратный корень из числа x.
<b>Round(x)</b>	округляет число x

# Графика

- Для рисования рисунков будем использовать объект **TImage**
- Рисование выполняется путем обращения к свойству **Canvas** (графическая канва) этого объекта: **Image1.Canvas**.

# Система координат графического объекта



по умолчанию  
левый верхний  
угол имеет  
координаты  
 $(0,0)$ .  
Координаты  
измеряются в  
пикселях.

# Классы

Классы: позволяют определять положение фигур и задавать различные параметры фигур (тип линий, вид заливки и пр.)

<b>ИМЯ</b>	<b>ОПИСАНИЕ</b>
<b>Pen</b>	определяет параметры простых линий
<b>Brush</b>	определяет параметры области заполнения
<b>Font</b>	Используется для задания шрифта, которым будет нарисован текст. Можно указать имя шрифта, размер и т.д.
<b>Region</b>	Позволяет задать регион (замкнутое пространство). Регионом может быть круг, квадрат или произвольная фигура. Позволяет так же делать дырки в фигурах

# Работа с классами

Определение цвета линии

*Объект.* Canvas.Pen.Color := {цвет};

Определение цвета заливки

*Объект.* Canvas.Brush.Color := {цвет};

Цвета:

clBlack – черный

clRed - красный

clBlue – синий

clWhite - белый

Пример:

Image1.Canvas.pen.Color := clblue;

## Графические функции

- **LineTo** ( $x_1, y_1$ ) - линия и начальной точки в точку с координатами ( $x_1, y_1$ )
- **Rectangle** ( $x_1, y_1, x_2, y_2$ ) - прямоугольник (или квадрат), заполненный цветом текущей кисти и обрамлённый цветом текущего пера; ( $x_1, y_1$ ) – координаты верхнего угла, ( $x_2, y_2$ ) - координаты правого нижнего.
- **Ellipse** ( $x_1, y_1, x_2, y_2$ ) – эллипс вписывается в прямоугольник, с координатами верхнего левого угла ( $x_1, y_1$ ) и правого нижнего ( $x_2, y_2$ ).

# Использование функций рисования

Имя  
объекта

Canvas

функция

(аргументы  
функции)

- **Пример:**

```
Image1.Canvas.pen.Color := clblue;
```

```
Image1.Canvas.LineTo(100,100);
```

