Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №8**

**«Основы событийно-ориентированного программирования»**

**МДК 05.02 «Разработка кода информационных систем»**

Выполнил: студент учебной группы

ИСПк-202-52-00

Гончаров Мирослав Даниилович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

# **Цель работы**

Получение базовых навыков реализации приложений с графическим интерфейсом пользователя на основе событийно-ориентированной парадигмы.

# **Формулировка задания (с вариантом)**

Вариант: 4

1. Реализовать приложение «Калькулятор площади фигур».
2. Предусмотреть выбор из фиксированного набора заданных фигур (не менее четырех)

# **Описание алгоритма**

На главной форме расположены элементы управления для выбора типа фигуры и ввода параметров для расчета её площади:

**Элементы интерфейса:**

**TRadioGroup** - группа радиокнопок для выбора фигуры (**Select\_Figure**).

**TImage** - изображение формулы для выбранной фигуры (**Formula\_Picture**).

**TEdit** - поля ввода для параметров расчета площади (**First\_val\_for\_calcs**, **Second\_val\_for\_calcs**, **Third\_val\_for\_calcs**).

**TButton** - кнопка для расчета площади (**Calculate\_Square**).

**TMemo** - поле для вывода результата расчета (**Answer\_Field**).

**TLabel** - метки для полей ввода параметров (**Label\_for\_first\_val**, **Label\_for\_second\_val**, **Label\_for\_third\_val**).

**Основной алгоритм работы**

1. **Инициализация формы:**

При создании формы инициализируются элементы управления.

1. **Обработка выбора фигуры (Select\_FigureClick):**

Обнуляются поля ввода и вывод результата.

Вызывается метод **UpdateUIForSelectedFigure**, который настраивает видимость элементов управления в зависимости от выбранной фигуры.

1. **Настройка интерфейса в зависимости от выбранной фигуры (UpdateUIForSelectedFigure):**

Скрываются все элементы управления.

В зависимости от выбранной фигуры (круг, прямоугольник, треугольник, трапеция) отображаются соответствующие элементы управления и загружается изображение формулы.

Настраиваются подсказки и метки для полей ввода параметров.

1. **Проверка корректности ввода (IsNumber):**

Проверяется, является ли введенное значение числом (с учетом возможного ввода с запятой или точкой).

1. **Расчет площади фигуры (Calculate\_SquareClick):**

Проверяются значения всех видимых полей ввода.

Если какое-либо значение некорректно, выводится сообщение об ошибке.

Если все значения корректны, выполняется расчет площади в зависимости от выбранной фигуры:

**Круг:** площадь = π \* радиус².

**Прямоугольник:** площадь = длина \* ширина.

**Треугольник:** площадь = 0.5 \* основание \* высота.

**Трапеция:** площадь = 0.5 \* (основание1 + основание2) \* высота.

Выводится результат расчета в поле **Answer\_Field**.

# **Код программы**

unit Unit1;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, StdCtrls, RegExpr;

type

{ TFind\_Figure\_Square }

TFind\_Figure\_Square = class(TForm)

Select\_Figure: TRadioGroup;

Formula\_Picture: TImage;

First\_val\_for\_calcs: TEdit;

Second\_val\_for\_calcs: TEdit;

Third\_val\_for\_calcs: TEdit;

Calculate\_Square: TButton;

Answer\_Field: TMemo;

Label\_for\_first\_val: TLabel;

Label\_for\_second\_val: TLabel;

Label\_for\_third\_val: TLabel;

procedure FormCreate(Sender: TObject);

procedure Answer\_FieldClick(Sender: TObject);

procedure Select\_FigureClick(Sender: TObject);

procedure Calculate\_SquareClick(Sender: TObject);

private

procedure UpdateUIForSelectedFigure;

function IsNumber(const S: string): Boolean;

public

end;

var

Find\_Figure\_Square: TFind\_Figure\_Square;

implementation

{$R \*.lfm}

{ TFind\_Figure\_Square }

procedure TFind\_Figure\_Square.Answer\_FieldClick(Sender: TObject);

begin

end;

procedure TFind\_Figure\_Square.FormCreate(Sender: TObject);

begin

end;

procedure TFind\_Figure\_Square.Select\_FigureClick(Sender: TObject);

begin

First\_val\_for\_calcs.Text := '';

Second\_val\_for\_calcs.Text := '';

Third\_val\_for\_calcs.Text := '';

Answer\_Field.Text := '';

Find\_Figure\_Square.Height := 550;

UpdateUIForSelectedFigure;

end;

procedure TFind\_Figure\_Square.UpdateUIForSelectedFigure;

begin

// Скрываем все компоненты

Formula\_Picture.Visible := False;

First\_val\_for\_calcs.Visible := False;

Second\_val\_for\_calcs.Visible := False;

Third\_val\_for\_calcs.Visible := False;

Calculate\_Square.Visible := False;

Answer\_Field.Visible := False;

Label\_for\_first\_val.Visible := False;

Label\_for\_second\_val.Visible := False;

Label\_for\_third\_val.Visible := False;

// Обновляем UI в зависимости от выбранной фигуры

case Select\_Figure.ItemIndex of

0: // Круг

begin

Formula\_Picture.Picture.LoadFromFile('square\_circles.png');

Formula\_Picture.Visible := True;

First\_val\_for\_calcs.Hint := 'Радиус';

First\_val\_for\_calcs.Visible := True;

Calculate\_Square.Visible := True;

Label\_for\_first\_val.Caption := 'R =';

Label\_for\_first\_val.Visible := True;

end;

1: // Прямоугольник

begin

Formula\_Picture.Picture.LoadFromFile('square\_rectangles.png');

Formula\_Picture.Visible := True;

First\_val\_for\_calcs.Hint := 'Длина';

Second\_val\_for\_calcs.Hint := 'Ширина';

First\_val\_for\_calcs.Visible := True;

Second\_val\_for\_calcs.Visible := True;

Calculate\_Square.Visible := True;

Label\_for\_first\_val.Caption := 'a =';

Label\_for\_second\_val.Caption := 'b =';

Label\_for\_first\_val.Visible := True;

Label\_for\_second\_val.Visible := True;

end;

2: // Треугольник

begin

Formula\_Picture.Picture.LoadFromFile('square\_triangle.png');

Formula\_Picture.Visible := True;

First\_val\_for\_calcs.Hint := 'Основание';

Second\_val\_for\_calcs.Hint := 'Высота';

First\_val\_for\_calcs.Visible := True;

Second\_val\_for\_calcs.Visible := True;

Calculate\_Square.Visible := True;

Label\_for\_first\_val.Caption := 'a =';

Label\_for\_second\_val.Caption := 'h =';

Label\_for\_first\_val.Visible := True;

Label\_for\_second\_val.Visible := True;

end;

3: // Трапеция

begin

Formula\_Picture.Picture.LoadFromFile('square\_trapezoid.png');

Formula\_Picture.Visible := True;

First\_val\_for\_calcs.Hint := 'Основание 1';

Second\_val\_for\_calcs.Hint := 'Основание 2';

Third\_val\_for\_calcs.Hint := 'Высота';

First\_val\_for\_calcs.Visible := True;

Second\_val\_for\_calcs.Visible := True;

Third\_val\_for\_calcs.Visible := True;

Calculate\_Square.Visible := True;

Label\_for\_first\_val.Caption := 'a =';

Label\_for\_second\_val.Caption := 'b =';

Label\_for\_third\_val.Caption := 'h =';

Label\_for\_first\_val.Visible := True;

Label\_for\_second\_val.Visible := True;

Label\_for\_third\_val.Visible := True;

end;

end;

end;

function TFind\_Figure\_Square.IsNumber(const S: string): Boolean;

var

TempStr: string;

Regex: TRegExpr;

begin

TempStr := StringReplace(S, ',', '.', [rfReplaceAll]); // Заменяем запятые на точки для проверки

Regex := TRegExpr.Create;

try

Regex.Expression := '^[0-9]+([.,][0-9]+)?$'; // Регулярное выражение для проверки числового значения с точкой или запятой

Result := Regex.Exec(TempStr);

finally

Regex.Free;

end;

end;

procedure TFind\_Figure\_Square.Calculate\_SquareClick(Sender: TObject);

var

Area, Value1, Value2, Value3: Double;

begin

if First\_val\_for\_calcs.Visible then

begin

if (First\_val\_for\_calcs.Text = '') or not IsNumber(First\_val\_for\_calcs.Text) or (StrToFloat(StringReplace(First\_val\_for\_calcs.Text, '.', ',', [rfReplaceAll])) < 0) then

begin

ShowMessage('Пожалуйста, введите корректное значение для ' + First\_val\_for\_calcs.Hint);

Exit;

end;

end;

if Second\_val\_for\_calcs.Visible then

begin

if (Second\_val\_for\_calcs.Text = '') or (not IsNumber(Second\_val\_for\_calcs.Text)) or (StrToFloat(StringReplace(Second\_val\_for\_calcs.Text, '.', ',', [rfReplaceAll])) < 0) then

begin

ShowMessage('Пожалуйста, введите корректное значение для ' + Second\_val\_for\_calcs.Hint);

Exit;

end;

end;

if Third\_val\_for\_calcs.Visible then

begin

if (Third\_val\_for\_calcs.Text = '') or not IsNumber(Third\_val\_for\_calcs.Text) or (StrToFloat(StringReplace(Third\_val\_for\_calcs.Text, '.', ',', [rfReplaceAll])) < 0) then

begin

ShowMessage('Пожалуйста, введите корректное значение для ' + Third\_val\_for\_calcs.Hint);

Exit;

end;

end;

Value1 := StrToFloatDef(StringReplace(First\_val\_for\_calcs.Text, '.', ',', [rfReplaceAll]),0);

Value2 := StrToFloatDef(StringReplace(Second\_val\_for\_calcs.Text, '.', ',', [rfReplaceAll]),0);

Value3 := StrToFloatDef(StringReplace(Third\_val\_for\_calcs.Text, '.', ',', [rfReplaceAll]),0);

case Select\_Figure.ItemIndex of

0: // Круг

Area := Pi \* Sqr(Value1);

1: // Прямоугольник

Area := Value1 \* Value2;

2: // Треугольник

Area := 0.5 \* Value1 \* Value2;

3: // Трапеция

Area := 0.5 \* (Value1 + Value2) \* Value3;

end;

Find\_Figure\_Square.Height := 700;

Answer\_Field.Visible := True;

Answer\_Field.Lines.Add('Рассчитанная площадь: ' + FloatToStr(Area));

end;

end.

# **Результат выполнения программы**

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 1 – Результат выполнения программы

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 2 – Результат выполнения программы

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рис. 3 – Результат выполнения программы

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис. 4 – Результат выполнения программы

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 5 – Результат выполнения программы

# **Вывод**

В процессе выполнения данной лабораторной работы была достигнута основная цель: получение базовых навыков реализации приложений с графическим интерфейсом пользователя на основе событийно-ориентированной парадигмы. Работа включала в себя разработку программы на языке Delphi в среде Lazarus, которая позволяла пользователю выбирать различные геометрические фигуры и вычислять их площади. Программа предоставляла возможность ввода параметров для выбранной фигуры, после чего происходил расчет и вывод результата.

Программа была реализована в одной форме без использования отдельных модулей, что упростило структурирование кода и взаимодействие между компонентами. В процессе работы были преодолены различные технические сложности, связанные с обработкой пользовательского ввода, корректностью вычислений и управлением графическим интерфейсом.

Особое внимание было уделено проверке корректности вводимых данных и визуализации формул для каждой из выбранных фигур. Это обеспечило интуитивно понятное взаимодействие пользователя с приложением и правильность вычислений.

Результаты выполнения программы демонстрируют успешную реализацию поставленных задач.