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»**

**ПО ДИСЦИПЛИНЕ «Разработка кода информационных систем»**

Выполнил: студент учебной группы

ИСПк-205-52-00

Порошин А.А.

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

**Цель работы:** получение базовых навыков реализации приложений с графическим интерфейсом пользователя на основе событийно-ориентированной парадигмы.

Вариант 13

**Задание:**

1. ﻿﻿﻿Реализовать приложение "Калькулятор целочисленной арифметики".
2. Требуется реализовать не менее четырех базовых операций. ﻿﻿﻿
3. Результат вычислений всегда должен быть целочисленным.

**Описание алгоритма**

1. После нажатия на клавишу калькулятор считывает название клавиши.
2. Калькулятор обрабатывает название клавиши и переводит в нужную процедуру.
3. В процедуре действия пользователя проверяется на ошибку.
4. После в процедуре калькулятор производит расчёт выбранного действия.
5. После окончания процедуры происходит вывод в эдит.

**Код программы**

unit frm1;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls,

calculatorunit;

type

{ TForm1 }

TForm1 = class(TForm)

btnNumber: TButton;

btnNumber1: TButton;

btnOperation: TButton;

btnNumber2: TButton;

btnNumber3: TButton;

btnNumber4: TButton;

btnNumber5: TButton;

btnNumber6: TButton;

btnNumber7: TButton;

btnNumber8: TButton;

btnNumber9: TButton;

btnOperationSubtract: TButton;

btnOperationSubtract1: TButton;

btnOperationSubtract3: TButton;

btnOperationSubtract4: TButton;

btnOperationSubtract5: TButton;

btnOperationSubtract6: TButton;

btnOperationSubtract7: TButton;

edtDisplay: TEdit;

procedure btnNumberClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormDestroy(Sender: TObject);

private

{ private declarations }

public

{ public declarations }

end;

var

Form1: TForm1;

Calculator: TCalculator;

implementation

{$R \*.lfm}

{ TForm1 }

procedure TForm1.btnNumberClick(Sender: TObject);

begin

Calculator.SendInput( TButton(Sender).Caption );

edtDisplay.Text:=Calculator.GetDisplay;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

Calculator:=TCalculator.Create;

BorderIcons:=Bordericons-[bimaximize];

end;

procedure TForm1.FormDestroy(Sender: TObject);

begin

Calculator.Free;

end;

end.

unit calculatorunit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils;

type

TCalculatorOperation = (None, Equal, Add, Subtract, Multiplication, Division,

Percentage);

{ TCalculator }

TCalculator = class

private

DisplayText: string;

CurrentInput: string;

LastResult: string;

ResultShown:Boolean;

LastOperator:TCalculatorOperation;

public

constructor Create;

procedure SendInput(Input:string);

procedure InputNumber(Number:string);

procedure AddOperator(Operation: TCalculatorOperation);

function GetDisplay:string;

procedure Clear;

procedure AllClear;

procedure Backspace;

end;

implementation

{ TCalculator }

constructor TCalculator.Create;

begin

AllClear;

end;

procedure TCalculator.SendInput(Input: string);

begin

case Input of

'0' .. '9': InputNumber(Input);

'.': InputNumber(Input);

'+': AddOperator(TCalculatorOperation.Add);

'-': AddOperator(TCalculatorOperation.Subtract);

'\*', '×': AddOperator(TCalculatorOperation.Multiplication);

'/', '÷': AddOperator(TCalculatorOperation.Division);

'=': AddOperator(TCalculatorOperation.Equal);

'C': Clear;

'CE': AllClear;

'←': Backspace;

else

//

end;

end;

procedure TCalculator.InputNumber(Number: string);

begin

if (Pos('.', CurrentInput)<>0) and (Number = '.') then Exit;

if CurrentInput = '0' then begin

CurrentInput:=Number;

end else begin

if ResultShown then

CurrentInput:=Number

else

CurrentInput:=CurrentInput+Number;

end;

DisplayText:=CurrentInput;

ResultShown:=False;

end;

procedure TCalculator.AddOperator(Operation: TCalculatorOperation);

begin

if Operation = TCalculatorOperation.Equal then begin

if LastOperator = TCalculatorOperation.Add then

if LastResult='Ноуп' then LastResult:='Ноуп' else

LastResult:= floattostr( StrToFloat(LastResult) + StrToFloat(CurrentInput) )

else if LastOperator = TCalculatorOperation.Subtract then

if LastResult='Ноуп' then LastResult:='Ноуп' else

LastResult:= floattostr( StrToFloat(LastResult) - StrToFloat(CurrentInput) )

else if LastOperator = TCalculatorOperation.Multiplication then

if LastResult='Ноуп' then LastResult:='Ноуп' else

LastResult:= floattostr( StrToFloat(LastResult) \* StrToFloat(CurrentInput) )

else if LastOperator = TCalculatorOperation.Division then

if StrToFloat(CurrentInput)=0 then LastResult:='Ноуп' else

if LastResult='Ноуп' then LastResult:='Ноуп' else

LastResult:= floattostr( StrToFloat(LastResult) / StrToFloat(CurrentInput) )

; ;

CurrentInput:=LastResult;

LastOperator:=TCalculatorOperation.None;

ResultShown:=True;

end else if LastOperator <> TCalculatorOperation.None then begin

AddOperator(TCalculatorOperation.Equal);

AddOperator(Operation);

end else begin

LastOperator := Operation;

LastResult := CurrentInput;

DisplayText := CurrentInput;

CurrentInput := '0';

end;

end;

function TCalculator.GetDisplay: string;

begin

if ResultShown then

Result := LastResult

else

Result := DisplayText;

end;

procedure TCalculator.Clear;

begin

CurrentInput:='0';

DisplayText:='0';

end;

procedure TCalculator.AllClear;

begin

Clear;

LastResult:='0';

ResultShown:=false;

LastOperator:=TCalculatorOperation.None;

end;

procedure TCalculator.Backspace;

begin

if ResultShown then Exit;

if Length(CurrentInput) > 1 then

CurrentInput:=Copy(CurrentInput,0,Length(CurrentInput)-1)

else

CurrentInput:='0';

DisplayText:=CurrentInput;

end;

end.

program proj\_calculator\_class;

{$mode objfpc}{$H+}

uses

{$IFDEF UNIX}{$IFDEF UseCThreads}

cthreads,

{$ENDIF}{$ENDIF}

Interfaces, // this includes the LCL widgetset

Forms, calculatorunit, frm1

{ you can add units after this };

{$R \*.res}

begin

RequireDerivedFormResource:=True;

Application.Initialize;

Application.CreateForm(TForm1, Form1);

Application.Run;

end.

**Результат выполнения**

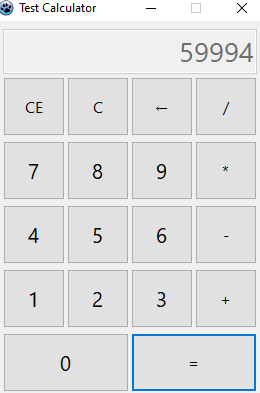


Рисунок.1 – Результат выполнения работы

**Вывод:** в ходе выполнения работы были изучены основы работы с графическим интерфейсом пользователя и событийно-ориентированной парадигмой программирования.