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**

**«МДК 05.02 РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ»**

Выполнила: студентка учебной группы

ИСПк-202-52-00

Широнина Анна Андреевна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2025

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

1. **Формулировка заданий. Вариант 15**

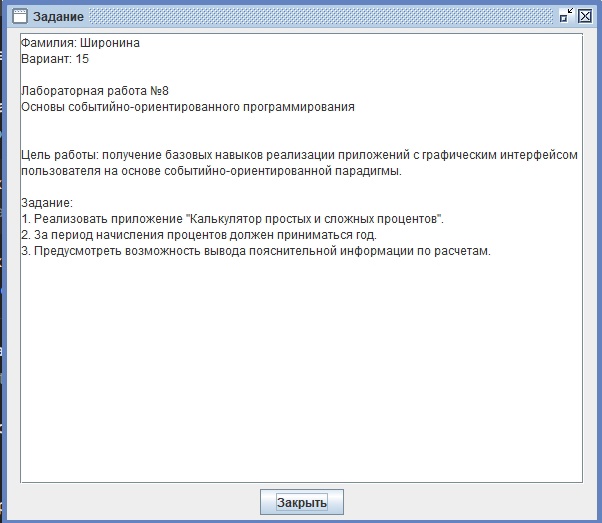


Рисунок 1 – Постановка задачки

1. **Описание приложения**

В данной работе был реализован калькулятор простых и сложных процентов и для его создания были использованы следующие компоненты:

* TForm – основное окно программы;
* TLabel – текстовая надпись;
* TEdit – строка ввода/вывода;
* TSpinEdit – счетчик;
* TButton – кнопка выполнения действия;
* TMemo – многострочное поле для текста.

Текстовые надписи использовались для подписания необходимых входных данных для ввода, а также результата.

Счетчик использовался для установления срока вклада.

TEdit использовался для ввода первоначальной суммы взноса и процентной ставки, а также для вывода результата расчета.

Кнопка TButton использовалась для действия расчета простых или сложных процентов.

Поле TMemo использовался для вывода пояснительной информации по расчетам.

1. **Код программы (файл unit1.pas)**

unit Unit1;

{$mode objfpc}{$H+}

interface

uses

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

Spin, Math, StrUtils;

type

{ TForm1 }

TForm1 = class(TForm)

bg: TImage;

ButtonSlozh: TButton;

ButtonProst: TButton;

LabelResult: TLabel;

LabelSrok: TLabel;

LabelStavka: TLabel;

LabelSumma: TLabel;

MemoRachet: TMemo;

result: TEdit;

srok: TSpinEdit;

stavka: TEdit;

pervSumma: TEdit;

procedure ButtonProstClick(Sender: TObject);

procedure ButtonSlozhClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure MemoRachetKeyPress(Sender: TObject; var Key: char);

procedure pervSummaKeyPress(Sender: TObject; var Key: char);

procedure resultKeyPress(Sender: TObject; var Key: char);

procedure stavkaKeyPress(Sender: TObject; var Key: char);

private

function ValidateInput: Boolean;

function SafeStrToFloat(const S: string): Double;

public

end;

var

Form1: TForm1;

implementation

{$R \*.lfm}

{ TForm1 }

function TForm1.SafeStrToFloat(const S: string): Double;

var

fs: TFormatSettings;

CleanStr: string;

begin

fs := DefaultFormatSettings;

fs.DecimalSeparator := ','; // Используем запятую

fs.ThousandSeparator := ' ';

// Удаляем все точки и лишние пробелы

CleanStr := ReplaceStr(S, '.', ',');

CleanStr := ReplaceStr(CleanStr, ' ', '');

try

Result := StrToFloat(CleanStr, fs);

except

on E: EConvertError do

begin

ShowMessage('Некорректный формат числа. Используйте запятую как разделитель.');

raise;

end;

end;

end;

procedure TForm1.pervSummaKeyPress(Sender: TObject; var Key: char);

var

Edit: TEdit;

PosComma: Integer;

CurrentText: string;

begin

Edit := Sender as TEdit;

CurrentText := Edit.Text;

// Разрешаем: цифры, Backspace, Delete, запятая

if not (Key in ['0'..'9', #8, #127, ',']) then

begin

Key := #0;

Exit;

end;

// Обработка запятой

if Key = ',' then

begin

PosComma := Pos(',', CurrentText);

// Запрещаем несколько запятых

if PosComma > 0 then

begin

Key := #0;

Exit;

end;

end;

end;

procedure TForm1.stavkaKeyPress(Sender: TObject; var Key: char);

var

Edit: TEdit;

PosComma: Integer;

CurrentText: string;

begin

Edit := Sender as TEdit;

CurrentText := Edit.Text;

// Разрешаем: цифры, Backspace, Delete, запятая

if not (Key in ['0'..'9', #8, #127, ',']) then

begin

Key := #0;

Exit;

end;

// Обработка запятой

if Key = ',' then

begin

PosComma := Pos(',', CurrentText);

// Запрещаем несколько запятых

if PosComma > 0 then

begin

Key := #0;

Exit;

end;

end;

end;

procedure TForm1.resultKeyPress(Sender: TObject; var Key: char);

begin

Key := #0;

end;

procedure TForm1.MemoRachetKeyPress(Sender: TObject; var Key: char);

begin

Key := #0;

end;

function TForm1.ValidateInput: Boolean;

var

sum, rate: Double;

begin

Result := False;

// Проверка наличия текста в полях

if Trim(pervSumma.Text) = '' then

begin

ShowMessage('Поле начальной суммы должно быть заполнено!');

pervSumma.SetFocus;

Exit;

end;

if Trim(stavka.Text) = '' then

begin

ShowMessage('Поле ставки должно быть заполнено!');

stavka.SetFocus;

Exit;

end;

try

sum := SafeStrToFloat(pervSumma.Text);

rate := SafeStrToFloat(stavka.Text);

if (sum <= 0) then

begin

ShowMessage('Начальная сумма должна быть положительным числом!');

pervSumma.SetFocus;

Exit;

end;

if (rate <= 0) then

begin

ShowMessage('Ставка должна быть положительным числом!');

stavka.SetFocus;

Exit;

end;

if srok.Value <= 0 then

begin

ShowMessage('Срок вклада должен быть положительным числом!');

srok.SetFocus;

Exit;

end;

Result := True;

except

on E: Exception do

ShowMessage('Ошибка ввода данных: ' + E.Message);

end;

end;

procedure TForm1.ButtonProstClick(Sender: TObject);

var

sum, rate, total, profit: Double;

years: Integer;

begin

if not ValidateInput then Exit;

try

sum := SafeStrToFloat(pervSumma.Text);

rate := SafeStrToFloat(stavka.Text);

years := srok.Value;

if years > 300 then

raise Exception.Create('Срок вклада слишком большой для расчета');

profit := sum \* rate \* years / 100;

total := sum + profit;

if IsInfinite(total) or IsNaN(total) then

raise Exception.Create('Результат расчета слишком большой или неопределенный');

result.Text := FormatFloat('0.00', total);

MemoRachet.Clear;

MemoRachet.Lines.Add('Расчет простых процентов:');

MemoRachet.Lines.Add('');

MemoRachet.Lines.Add('Начальная сумма: ' + FormatFloat('0.00', sum) + ' руб.');

MemoRachet.Lines.Add('Годовая ставка: ' + FormatFloat('0.00', rate) + '%');

MemoRachet.Lines.Add('Срок вклада в годах: ' + IntToStr(years));

MemoRachet.Lines.Add('');

MemoRachet.Lines.Add('Формула: S = P \* (1 + r \* n)');

MemoRachet.Lines.Add('S - итоговая сумма');

MemoRachet.Lines.Add('P - начальная сумма');

MemoRachet.Lines.Add('r - процентная ставка (в долях)');

MemoRachet.Lines.Add('n - срок вклада в годах');

MemoRachet.Lines.Add('');

MemoRachet.Lines.Add(FormatFloat('0.00', sum) + ' \* (1 + ' +

FormatFloat('0,00', rate/100) + ' \* ' + IntToStr(years) + ') = ' +

FormatFloat('0,00', total) + ' руб.');

MemoRachet.Lines.Add('Сумма процентов: ' + FormatFloat('0.00', profit) + ' руб.');

except

on E: EOverflow do

ShowMessage('Ошибка: Слишком большое число для расчетов');

on E: EZeroDivide do

ShowMessage('Ошибка: Деление на ноль');

on E: Exception do

ShowMessage('Ошибка при расчете: ' + E.Message);

end;

end;

procedure TForm1.ButtonSlozhClick(Sender: TObject);

var

sum, rate, total, profit: Double;

years, i: Integer;

yearlySum: array of Double = nil;

begin

if not ValidateInput then Exit;

try

sum := SafeStrToFloat(pervSumma.Text);

rate := SafeStrToFloat(stavka.Text);

years := srok.Value;

if years > 50 then

raise Exception.Create('Срок вклада слишком большой для расчета сложных процентов');

total := sum \* Power(1 + rate / 100, years);

profit := total - sum;

if IsInfinite(total) or IsNaN(total) then

raise Exception.Create('Результат расчета слишком большой или неопределенный');

result.Text := FormatFloat('0.00', total);

MemoRachet.Clear;

MemoRachet.Lines.Add('Расчет сложных процентов:');

MemoRachet.Lines.Add('');

MemoRachet.Lines.Add('Начальная сумма: ' + FormatFloat('0.00', sum) + ' руб.');

MemoRachet.Lines.Add('Годовая ставка: ' + FormatFloat('0.00', rate) + '%');

MemoRachet.Lines.Add('Срок вклада в годах: ' + IntToStr(years));

MemoRachet.Lines.Add('');

MemoRachet.Lines.Add('Формула: S = P \* (1 + r)^n');

MemoRachet.Lines.Add('S - итоговая сумма');

MemoRachet.Lines.Add('P - начальная сумма');

MemoRachet.Lines.Add('r - процентная ставка (в долях)');

MemoRachet.Lines.Add('n - срок вклада в годах');

MemoRachet.Lines.Add('');

MemoRachet.Lines.Add(FormatFloat('0.00', sum) + ' \* (1 + ' +

FormatFloat('0.00', rate/100) + ')^' + IntToStr(years) + ' = ' +

FormatFloat('0.00', total) + ' руб.');

SetLength(yearlySum, years + 1);

yearlySum[0] := sum;

for i := 1 to years do

begin

yearlySum[i] := yearlySum[i-1] \* (1 + rate / 100);

MemoRachet.Lines.Add('Год ' + IntToStr(i) + ': ' +

FormatFloat('0.00', yearlySum[i-1]) + ' -> ' +

FormatFloat('0.00', yearlySum[i]) + ' руб.');

end;

MemoRachet.Lines.Add('Сумма процентов: ' + FormatFloat('0,00', profit) + ' руб.');

except

on E: EOverflow do

ShowMessage('Ошибка: Слишком большое число для расчетов');

on E: EZeroDivide do

ShowMessage('Ошибка: Деление на ноль');

on E: Exception do

ShowMessage('Ошибка при расчете: ' + E.Message);

end;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

pervSumma.Text := '1000,00';

stavka.Text := '10,00';

srok.Value := 1;

MemoRachet.Clear;

result.ReadOnly := True;

MemoRachet.ReadOnly := True;

result.Color := clBtnFace;

MemoRachet.Color := clBtnFace;

pervSumma.MaxLength := 15;

stavka.MaxLength := 7;

end;

end.

1. **Результат выполнения программы**

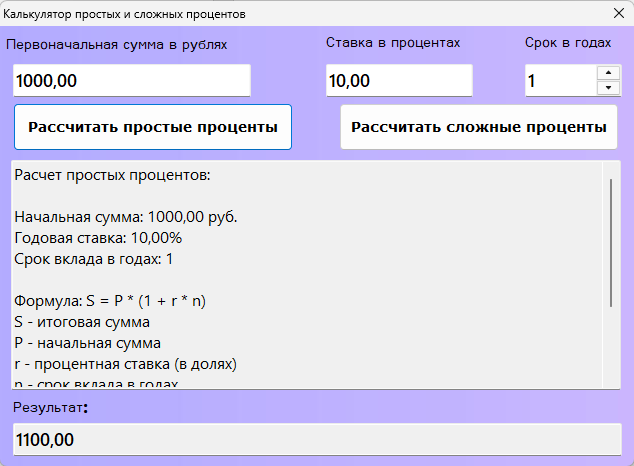


Рисунок 2 – Результат выполнения программы расчет простых процентов



Рисунок 3 – Результат выполнения программы расчет простых процентов

1. **Вывод**

В ходе выполнения домашней контрольной работы был разработан калькулятор для расчета простых и сложных процентов. Цель работы была успешно достигнута. В процессе работы были изучены и применены основные компоненты библиотеки Lazarus, такие как TForm, TLabel, TEdit, TSpinEdit, TButton и TMemo, которые обеспечили интуитивно понятный интерфейс для пользователя.

Калькулятор позволяет вводить начальную сумму вклада, годовую процентную ставку и срок вклада, после чего рассчитывает итоговую сумму с учетом простых или сложных процентов. Для обеспечения корректности ввода данных были реализованы функции обработки ошибок, которые проверяют, что введенные значения являются положительными числами, а также обрабатывают исключительные ситуации, такие как деление на ноль или переполнение. Результаты расчетов выводятся в поле TMemo, где также отображаются пояснения, включая формулы, промежуточные вычисления и итоговую сумму процентов. Это делает процесс расчета более наглядным и понятным для пользователя.

Таким образом, выполнение данной работы позволило не только освоить основы событийно-ориентированного программирования, но и получить практический опыт создания функционального приложения с графическим интерфейсом. Разработанный калькулятор может быть полезен для выполнения финансовых расчетов, а его код и структура могут служить основой для более сложных проектов в будущем.