МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»**

Институт ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ЦИФРОВЫЕ ТРАНСФОРМАЦИИ

**Отчет по лабораторной работе № 8**

**по дисциплине «Объектно-ориентированное программирование»**

**Тема: «Классы. Наследование»**

Выполнил: Сидоров Д. С., группа ИТС-122

Проверил: к.т.н., доц. Семёнов А. А.

Москва, 2025г

**Классы. Наследование**

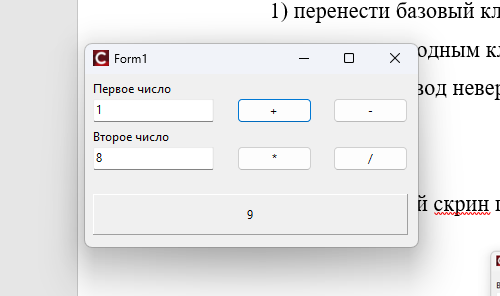
Задание: изучить и повторить проиллюстрированный ниже пример, демонстрирующий реализацию базового класса Float и производного класса FloatPr.

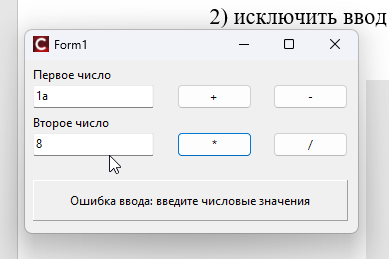
В примере разрабатывается визуальное приложение (Windows VCL Application), в котором на основе стандартного типа float создается базовый класс Float, имеющий два конструктора, метод вывода на экран и метод для перегрузки арифметической операции +. Используя общее наследование, создается производный класс FloatPr, добавляющий возможность использования операций -, \*, /. Далее идет проверка производного класса.

Необходимо доработать приложение:

1) перенести базовый класс в отдельный модуль (unit2), само приложение с производным классом остаются в unit1;

2) исключить ввод неверных данных и обработать деление на ноль через try...catch.

  
Рисунок 1. Работающее приложение

  
Рисунок 2. Обработка ошибочных данных

**Листинг кода**

**Unit1.cpp**

///---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

FloatPr::FloatPr() : Float() {}

FloatPr::FloatPr(float v) : Float(v) {}

FloatPr::FloatPr(Float a) : Float(a) {}

FloatPr FloatPr::operator-(FloatPr b) {

return FloatPr(f - b.f);

}

FloatPr FloatPr::operator\*(FloatPr b) {

return FloatPr(f \* b.f);

}

FloatPr FloatPr::operator/(FloatPr b) {

if (b.f == 0) {

throw Exception("Деление на ноль невозможно!");

}

return FloatPr(f / b.f);

}

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

bool TForm1::ValidateInput()

{

try {

StrToFloat(LabeledEdit1->Text);

StrToFloat(LabeledEdit2->Text);

return true;

}

catch (Exception &e) {

Panel1->Caption = "Ошибка ввода: введите числовые значения";

return false;

}

}

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

try {

if (!ValidateInput()) return;

FloatPr f1, f2, f3;

f1 = StrToFloat(LabeledEdit1->Text);

f2 = StrToFloat(LabeledEdit2->Text);

f3 = f1 + f2;

f3.show(Panel1);

}

catch (Exception &e) {

Panel1->Caption = e.Message;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

try {

if (!ValidateInput()) return;

FloatPr f1, f2, f3;

f1 = StrToFloat(LabeledEdit1->Text);

f2 = StrToFloat(LabeledEdit2->Text);

f3 = f1 - f2;

f3.show(Panel1);

}

catch (Exception &e) {

Panel1->Caption = e.Message;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

try {

if (!ValidateInput()) return;

FloatPr f1, f2, f3;

f1 = StrToFloat(LabeledEdit1->Text);

f2 = StrToFloat(LabeledEdit2->Text);

f3 = f1 \* f2;

f3.show(Panel1);

}

catch (Exception &e) {

Panel1->Caption = e.Message;

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button4Click(TObject \*Sender)

{

try {

if (!ValidateInput()) return;

FloatPr f1, f2, f3;

f1 = StrToFloat(LabeledEdit1->Text);

f2 = StrToFloat(LabeledEdit2->Text);

f3 = f1 / f2;

f3.show(Panel1);

}

catch (Exception &e) {

Panel1->Caption = e.Message;

}

}

**Unit1.h**

/

//---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

#include <System.Classes.hpp>

#include <Vcl.Controls.hpp>

#include <Vcl.StdCtrls.hpp>

#include <Vcl.Forms.hpp>

#include <Vcl.ExtCtrls.hpp>

#include <Vcl.Mask.hpp>

#include "Unit2.h"

//---------------------------------------------------------------------------

class FloatPr : public Float {

public:

FloatPr();

FloatPr(float v);

FloatPr(Float a);

FloatPr operator-(FloatPr b);

FloatPr operator\*(FloatPr b);

FloatPr operator/(FloatPr b);

};

class TForm1 : public TForm

{

\_\_published:

TButton \*Button1;

TButton \*Button2;

TButton \*Button3;

TButton \*Button4;

TLabeledEdit \*LabeledEdit1;

TLabeledEdit \*LabeledEdit2;

TPanel \*Panel1;

void \_\_fastcall Button1Click(TObject \*Sender);

void \_\_fastcall Button2Click(TObject \*Sender);

void \_\_fastcall Button3Click(TObject \*Sender);

void \_\_fastcall Button4Click(TObject \*Sender);

private: // User declarations

bool ValidateInput();

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

**Unit2.cpp**

/

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit2.h"

#include "Unit1.h" // To access Form1

//---------------------------------------------------------------------------

#pragma package(smart\_init)

// Implementation of base class methods

Float::Float() {

f = 0;

}

Float::Float(float d) {

f = d;

}

void Float::show(TPanel\* panel) { // Fully qualify TPanel

panel->Caption = f;

}

Float Float::operator+(Float b) {

return Float(f + b.f);

}

**Unit2.h**

//---------------------------------------------------------------------------

#ifndef Unit2H

#define Unit2H

//---------------------------------------------------------------------------

#include <System.Classes.hpp>

#include <Vcl.ExtCtrls.hpp> // Contains TPanel definition

// Base class

class Float {

protected:

float f;

public:

Float();

Float(float d);

void show(TPanel\* panel); // Fully qualify TPanel

Float operator+(Float b);

};

//---------------------------------------------------------------------------

#endif

**Вывод:** В ходе лабораторной работы были успешно реализованы базовый класс Float и производный класс FloatPr, которые продемонстрировали основные принципы наследования в C++. Базовый класс был перенесен в отдельный модуль Unit2, что улучшило структуру программы и позволило соблюсти принцип разделения ответственности. Были добавлены механизмы обработки исключений с использованием конструкции try-catch для предотвращения ошибок при вводе некорректных данных и при делении на ноль. Для полной реализации потребовалось правильно разрешить проблемы с неоднозначными ссылками на классы VCL, используя полные квалификаторы имен. Данная лабораторная работа продемонстрировала практическое применение принципов ООП: инкапсуляцию, наследование и полиморфизм, а также показала важность правильной обработки исключительных ситуаций в приложениях.