Міністерство освіти і науки України
Відокремлений структурний підрозділ
«Фаховий коледж ракетно-космічного машинобудування
Дніпровського національного університету імені Олеся Гончара»

ЗВІТ

з Навчальної практики з розробки інформаційних систем

Спеціальність 121

Група ПЗ-22-1

Виконав Бєдін Т. В.

Перевірила Гапоненко Н.В.

2025

ЗМІСТ

ЗАВДАННЯ № 1

Створення додатку з елементами управління графічного інтерфейсу. Проєктування структури бази даних та реалізація підключення. Використання в програмі компонентів "надпис", "однорядковий текстовий редактор". Використання в програмі обробників подій натискання на кнопку, зміни текста. Обробка повідомлень в додатку

Мета: закріпити навички з використання елементів управління графічного інтерфейсу в додатку, проєктування структури бази даних та реалізації підключення, використання компонентів "надпис", "однорядковий текстовий редактор", обробників подій натискання на кнопку, зміни текста, оброблення повідомлень в додатку

Хід роботи

1.1 Постановка задачі

Спроектувати таблиці БД MySQL, створити проект, виконати підключення, забезпечити введення даних та розрахунок результату за варіантом.

Програма повинна надати можливість користувачу забезпечувати контроль введених даних (відповідно до варіанту). Контроль повинен здійснюватись засобами, які найбільше відповідають характеру даного.

1.2 Елементи розробки інтерфейсу наведено у таблиці 1.1

Таблиця 1.1 – Перелік та призначення елементів інтерфейсу

Елемент	Призначення
EditPIB	Для вводу ПІБ студента
EditUniversity	Для вводу назви навчального закладу
EditCourse	Для вводу курсу
EditGroup	Для вводу групи
EditRecordBook	Для вводу номеру залікової книжки

Продовження таблиці 1.1

EditScholarship	Для вводу розміру стипендії
TLabel (6 шт)	Пояснювальні надписи біля полів
ComboBenefit	Для вибору пільгової категорії
ButtonAdd	Для додавання запису студента до бази даних
ButtonShow	Для виводу пільговиків з БД за категорією з ComboBenefit
ButtonShowAll	Для виведення усіх записів з бази даних
ADOConnection1	Підключення до бази
ADOQuery1	Для виконання SQL-запитів
DataSource1	Для прив’язки до візуального виводу
DBGrid1	Візуальний вивід

1.3 Елементи розробки функцій програми наведено у таблиці 1.2

Таблиця 1.2 – Перелік та призначення обробників подій

	Функція	Обробник подій, що її реалізує	Модуль, у якому розміщується обробник подій
1.	Перевірка введення ПІБ (тільки літери та пробіли)	EditPIBChange	Unit1
2.	Перевірка правильності курсу (1–6)	EditCourseChange	Unit1
3.	Перевірка стипендії (додатне число)	EditScholarshipChange	Unit1
4	Додавання нового студента до бази даних	ButtonAddClick	Unit1
5	Виведення списку студентів з обраною пільговою категорією	ButtonShowClick	Unit1
6	Вибір пільгової категорії зі списку	ComboBenefitChange	Unit1
7	Створення пунктів у ComboBenefit	FormCreate	Unit1
8	Показати усі записи таблиці	ButtonShowAllClick	Unit1

1.4 Вікно розробленої програми наведено на рисунку 1.1

Вікно Form 1 – Головне та єдине вікно програми, у якому відбуваються всі події. Забезпечує введення таких даних як ПІБ студента, учбовий заклад, курс, група, номер залікової книжки, розмір стипендії, пільгова категорія.

Вікно забезпечує виведення переліку та кількості студентів, які мають вказану пільгову категорію, або всіх студентів які знаходяться у базі даних.

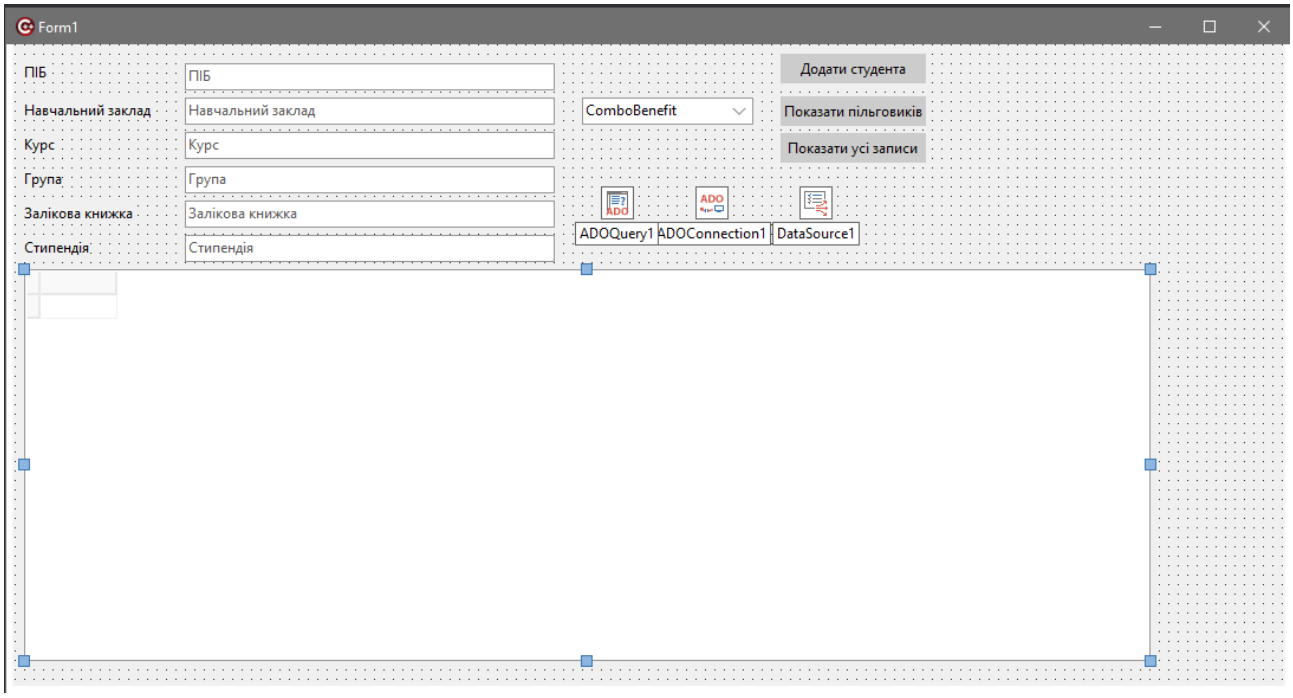


Рисунок 1.1 – Головне вікно програми

1.5 База даних була створена за допомогою SQL запиту наведеного в лістингу 1.1.

Лістинг 1.1 – SQL запит для створення структури таблиці.

```
CREATE DATABASE student_db CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;

USE student_db;

CREATE TABLE students (
    pib VARCHAR(100) NOT NULL,
    university VARCHAR(100) NOT NULL,
    course INT NOT NULL,
    group_name VARCHAR(10) NOT NULL,
    record_book VARCHAR(20) NOT NULL,
```

```
    scholarship DECIMAL(10,2) NOT NULL,
    benefit_category VARCHAR(50) NOT NULL
);
```

Створена й заповнена база даних зображена на рисунку 1.2.

pib	university	course	group_name	record_book ▲ 1	scholarship	benefit_category
Іваненко Олексій Сергійович	КНУ	1	ПЗ-21	ЗК100001	1500.00	Сирота
Петренко Марія Олександрівна	КПІ	2	КП-32	ЗК100002	3000.00	Інвалід
Шевченко Ігор Дмитрович	НУЛП	3	ЮФ-43	ЗК100003	4500.00	Багатодітна родина
Коваленко Аліна Андріївна	КНУ	4	ПЗ-21	ЗК100004	2000.00	
Сидорчук Віктор Миколайович	КПІ	1	КП-32	ЗК100005	7000.00	
Ткаченко Олена Ігорівна	НУЛП	2	ЮФ-43	ЗК100006	1800.00	Сирота
Мельник Андрій Павлович	КНУ	3	ПЗ-21	ЗК100007	3200.00	Інвалід
Кравець Наталія Сергіївна	КПІ	4	КП-32	ЗК100008	2700.00	
Бондаренко Вадим Юрійович	НУЛП	1	ЮФ-43	ЗК100009	3500.00	
Лисенко Інна Богданівна	КНУ	2	ПЗ-21	ЗК100010	2200.00	Багатодітна родина
Гнатюк Олександр Євгенович	КПІ	3	КП-32	ЗК100011	4000.00	Інвалід
Федоренко Катерина Романівна	НУЛП	4	ЮФ-43	ЗК100012	5000.00	

Рисунок 1.2 – Створена й заповнена база даних

1.6 Тексти програми наведено у лістингах 1.2—1.4

Код модуля Unit1.cpp наведеного в лістингу 1.2.

Лістинг 1.2 – Головний файл

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::EditPIBChange(TObject *Sender)
{
    String text = EditPIB->Text;
    for (int i = 1; i <= text.Length(); ++i)
    {
        if (!IsCharAlpha(text[i]) && text[i] != ' ')
        {
            ShowMessage("ПІБ повинен містити тільки літери та пробіли.");
            EditPIB->Text = "";
            break;
```

```
        }
    }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::EditCourseChange(TObject *Sender)
{
       static bool isChanging = false; // Захист від повторного виклику

       if (isChanging)
              return;

       isChanging = true;

       int course = StrToIntDef(EditCourse->Text, -1);
       if (course < 1 || course > 6)
       {
              ShowMessage("Курс має бути цифра від 1 до 6.");
              EditCourse->Text = "";
       }

    isChanging = false;
}

//---------------------------------------------------------------------------
void __fastcall TForm1::EditScholarshipChange(TObject *Sender)
{

       try {
              double scholarship = StrToFloat(EditScholarship->Text);
              if (scholarship < 0)
              {
             ShowMessage("Стипендія не може бути від'ємною.");
                     EditScholarship->Text = "";
        }
    } catch (...) {
              ShowMessage("Стипендія повинна бути числом.");
              EditScholarship->Text = "";
       }

}

//---------------------------------------------------------------------------
void __fastcall TForm1::ButtonAddClick(TObject *Sender)
{
    String pib = EditPIB->Text.Trim();
    String univ = EditUniversity->Text.Trim();
    int course = StrToInt(EditCourse->Text);
       String group_name = EditGroup->Text.Trim();
    String record = EditRecordBook->Text.Trim();
    double scholarship = StrToFloat(EditScholarship->Text);
    String benefit = ComboBenefit->Text;

       if (pib == "" || univ == "" || group_name == "" || record == "" ) {
         ShowMessage("Заповніть усі поля.");
         return;
    }

    ADOQuery1->SQL->Clear();
ADOQuery1->SQL->Add(
    "INSERT INTO students (PIB, University, Course, group_name, Record_Book, Scholarship,
Benefit_category) "
       "VALUES (:pib, :univ, :course, :group_name, :record, :scholarship, :benefit)");
//ShowMessage(ADOQuery1->SQL->Text);
ADOQuery1->Parameters->ParamByName("pib")->Value = pib;
ADOQuery1->Parameters->ParamByName("univ")->Value = univ;
ADOQuery1->Parameters->ParamByName("course")->Value = course;
ADOQuery1->Parameters->ParamByName("group_name")->Value = group_name;
ADOQuery1->Parameters->ParamByName("record")->Value = record;
ADOQuery1->Parameters->ParamByName("scholarship")->Value = scholarship;
ADOQuery1->Parameters->ParamByName("benefit")->Value = benefit;

ADOQuery1->ExecSQL();
```

```
    ShowMessage("Студента додано.");
}

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


void __fastcall TForm1::FormCreate(TObject *Sender)
{
       ComboBenefit->Items->Add("");
       ComboBenefit->Items->Add("Сирота");
       ComboBenefit->Items->Add("Інвалід");
       ComboBenefit->Items->Add("багатодітна родина");
       ComboBenefit->ItemIndex = 0; // вибрати перший пункт
}
//---------------------------------------------------------------------------


void __fastcall TForm1::ButtonShowClick(TObject *Sender)
{

    String selectedCategory = ComboBenefit->Text;

    // Підготовка SQL-запиту з параметром
    ADOQuery1->Close(); // Закриваємо попередній запит
    ADOQuery1->SQL->Clear(); // Очищаємо SQL
       ADOQuery1->SQL->Add("SELECT * FROM Students WHERE Benefit_Category = :cat");
    ADOQuery1->Parameters->ParamByName("cat")->Value = selectedCategory;
       ADOQuery1->Open(); // Виконуємо запит і оновлюємо DBGrid

    int count = ADOQuery1->RecordCount;
    ShowMessage("Кількість студентів з пільгою '" + selectedCategory + "': " +
IntToStr(count));
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ButtonShowAllClick(TObject *Sender)
{
         ADOQuery1->Close(); // закриваємо попередній запит
       ADOQuery1->SQL->Clear();
       ADOQuery1->SQL->Add("SELECT * FROM students");
       ADOQuery1->Open(); // відкриваємо результат запиту

       // Дані автоматично з’являться в DBGrid1, якщо все правильно з'єднано
}
//---------------------------------------------------------------------------
```

Лістинг 1.3 – Специфікація TForm

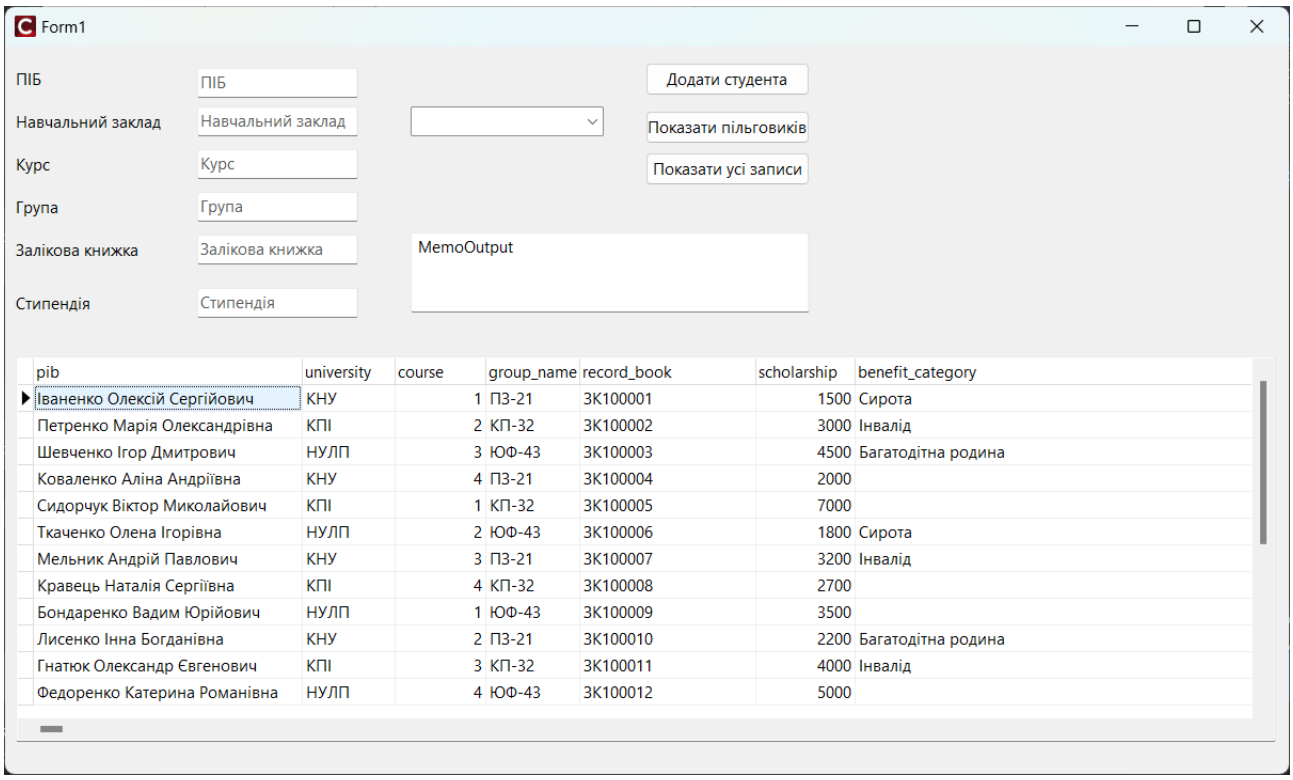
```
//---------------------------------------------------------------------------

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

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Data.DB.hpp>
#include <Data.Win.ADODB.hpp>
#include <Vcl.DBGrids.hpp>
#include <Vcl.Grids.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published: // IDE-managed Components
```

```
        TLabel *LabelPIB;
        TLabel *LabelUniversity;
        TLabel *LabelCourse;
        TLabel *LabelGroup;
        TLabel *LabelRecordBook;
        TLabel *Scholarship;
        TComboBox *ComboBenefit;
        TEdit *EditPIB;
        TEdit *EditUniversity;
        TEdit *EditCourse;
        TEdit *EditRecordBook;
        TEdit *EditScholarship;
        TButton *ButtonAdd;
        TButton *ButtonShow;
        TADOConnection *ADOConnection1;
        TEdit *EditGroup;
        TADOQuery *ADOQuery1;
        TDataSource *DataSource1;
        TDBGrid *DBGrid1;
        TButton *ButtonShowAll;
        void __fastcall EditPIBChange(TObject *Sender);
        void __fastcall EditCourseChange(TObject *Sender);
        void __fastcall EditScholarshipChange(TObject *Sender);
        void __fastcall ButtonAddClick(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall ButtonShowClick(TObject *Sender);
        void __fastcall ButtonShowAllClick(TObject *Sender);
private:     // User declarations
public:             // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

1.7 Тексти програми наведено у лістингах 1.2—1.9



pib	university	course	group_name	record_book	scholarship	benefit_category
Іваненко Олексій Сергійович	КНУ	1	ПЗ-21	ЗК100001	1500	Сирота
Петренко Марія Олександрівна	КПІ	2	КП-32	ЗК100002	3000	Інвалід
Шевченко Ігор Дмитрович	НУЛП	3	ЮФ-43	ЗК100003	4500	Багатодітна родина
Коваленко Аліна Андріївна	КНУ	4	ПЗ-21	ЗК100004	2000	
Сидорчук Віктор Миколайович	КПІ	1	КП-32	ЗК100005	7000	
Ткаченко Олена Ігорівна	НУЛП	2	ЮФ-43	ЗК100006	1800	Сирота
Мельник Андрій Павлович	КНУ	3	ПЗ-21	ЗК100007	3200	Інвалід
Кравець Наталія Сергіївна	КПІ	4	КП-32	ЗК100008	2700	
Бондаренко Вадим Юрійович	НУЛП	1	ЮФ-43	ЗК100009	3500	
Лисенко Інна Богданівна	КНУ	2	ПЗ-21	ЗК100010	2200	Багатодітна родина
Гнатюк Олександр Євгенович	КПІ	3	КП-32	ЗК100011	4000	Інвалід
Федоренко Катерина Романівна	НУЛП	4	ЮФ-43	ЗК100012	5000	

Рисунок 1.2 – Показати усі записи.

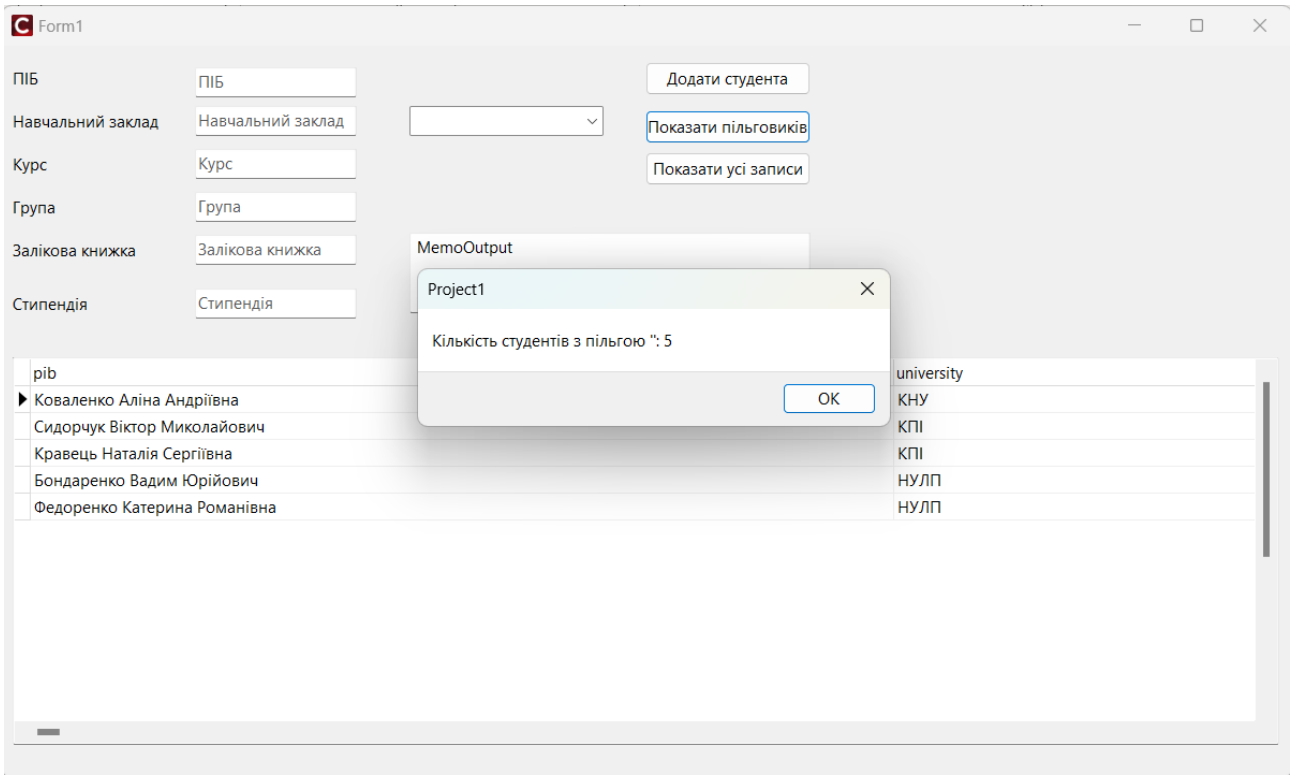


Рисунок 1.3 – Показати к-сть пільговиків у яких зараз немає пільги.

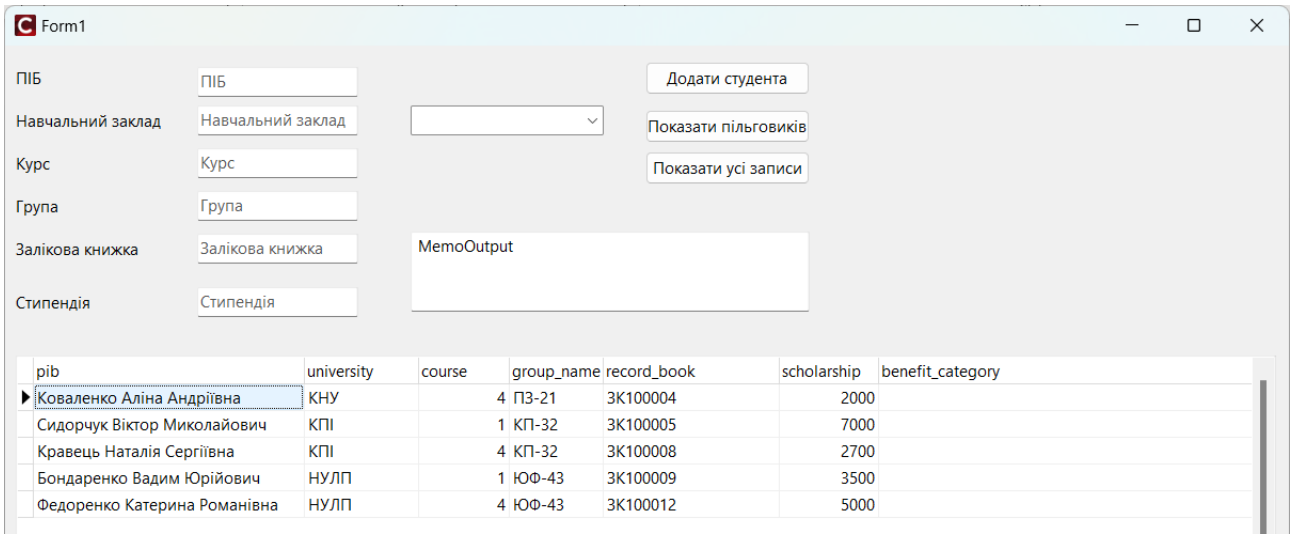


Рисунок 1.4 – Показати пільговиків у яких зараз немає пільги.

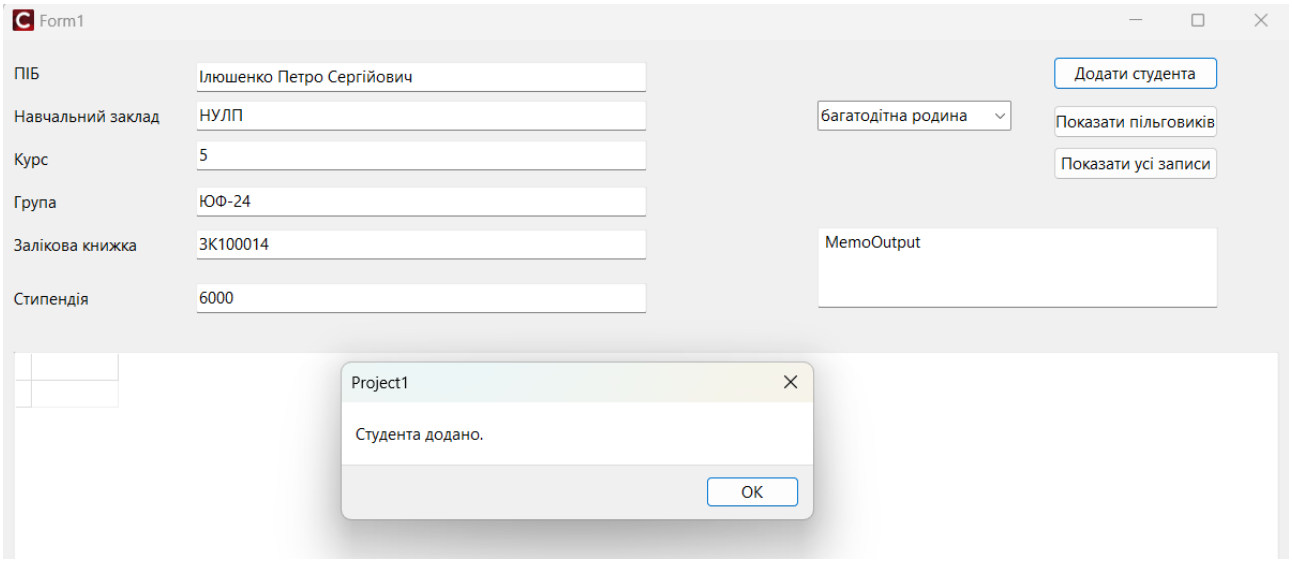


Рисунок 1.5 – Додати студента.

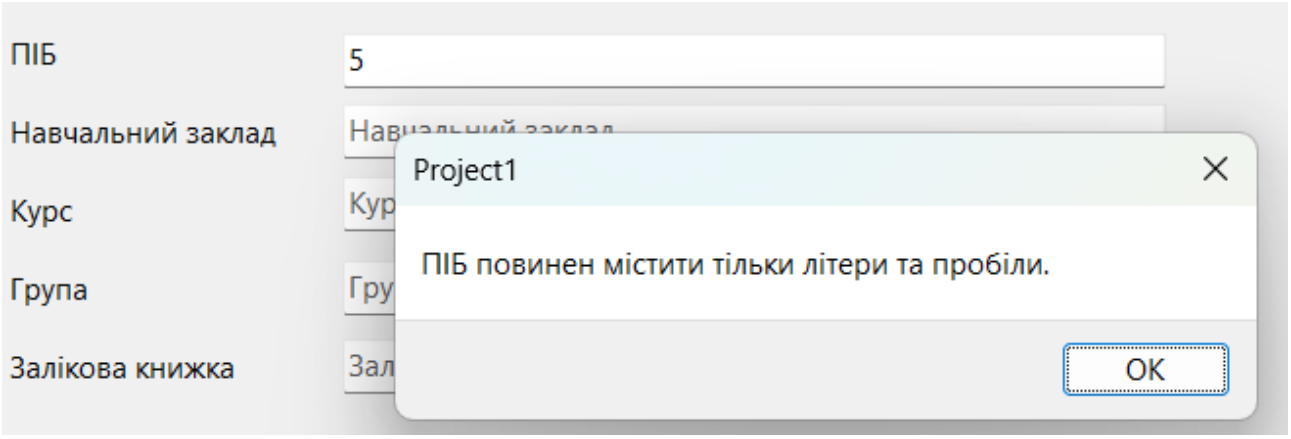


Рисунок 1.6 – Неправильне ведення ПІБ.

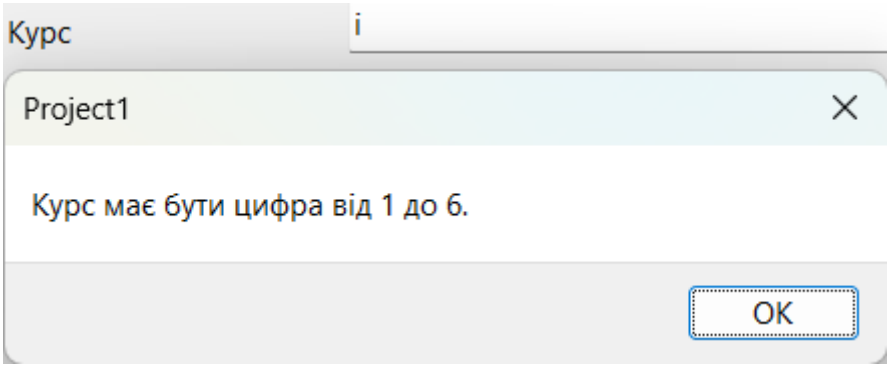


Рисунок 1.7 – Неправильне ведення курсу.

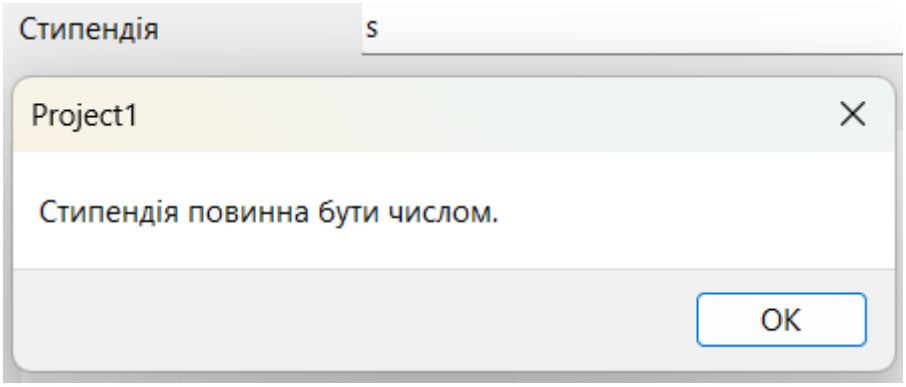


Рисунок 1.8 – Неправильне ведення числа.

Висновки: Було закріплено навички з використання елементів управління графічного інтерфейсу в додатку, проєктування структури бази даних та реалізації підключення, використання компонентів "надпис", "однорядковий текстовий редактор", обробників подій натискання на кнопку, зміни текста, оброблення повідомлень в додатку.

ЗАВДАННЯ № 2

Аналіз предметної області, визначення актуальності та призначення. Формулювання постановки задачі. Використання в програмі компонентів-списків. Використання в програмі компонентів-випадаючих списків. Використання в програмі обробників подій вибору в списку

Мета: закріпити навички з проведення аналізу предметної області, визначення актуальності та призначення, формулювання постановки задачі; навички з використання в програмі компонентів-списків, компонентів-випадаючих списків та обробників подій вибору в списку

Хід роботи

2.1 Постановка задачі

Відповідно до варіанту визначити функції програми та розробити інтерфейс програми. Окрім візуальної форми обов'язково спроектувати власний клас, в який винести логіку роботи. Надати можливість користувачу обирати необхідні дані, використовуючи в програмі списки та повторювати виконання програми.

Актуальність та призначення програми

У сучасних умовах розвитку торгівлі та постійного зростання обсягів товарообігу між виробниками та торговими точками виникає потреба в ефективному обліку закупівельної діяльності. Особливо це стосується підприємств харчової промисловості, таких як хлібозаводи, продукція яких є масового споживання і постачається до великої кількості торговельних точок. Чіткий і точний контроль за обсягами закупівель, цінами та асортиментом продукції дозволяє як виробнику, так і роздрібним продавцям оптимізувати витрати, планувати запаси та аналізувати прибутковість своєї діяльності.

Запропонована програма є актуальною з кількох причин. По-перше, вона дозволяє автоматизувати процес обліку закупівель продукції чотирма торговими точками, що значно спрощує розрахунки і мінімізує можливість

помилок, пов’язаних із людським фактором. По-друге, програма враховує важливу особливість — різницю у вартості продукції в залежності від сорту борошна (перший чи вищий гатунок), що відображає реальні умови ринку і дозволяє отримати більш точні економічні показники. По-третє, аналітичні можливості програми, зокрема розрахунок мінімальної та максимальної вартості закупівель, а також визначення переваг у виборі сорту борошна, дають змогу керівникам торгових точок та хлібозаводу приймати обґрунтовані рішення щодо планування закупівель та маркетингової політики.

Призначення програми полягає в розрахунку загальної вартості закупівель кожною з чотирьох торгових точок на основі обраного асортименту хлібобулочних виробів, визначенні вартості продукції, яку реалізував хлібозавод, а також аналізі обсягів закупівлі продукції залежно від сорту борошна. Програма має забезпечити зручний інтерфейс для вибору найменувань продукції та введення кількості одиниць, автоматично здійснити обчислення відповідно до вказаних умов (у тому числі з урахуванням 25% різниці у вартості між сортами борошна), та надати користувачу підсумкові результати в наочному вигляді.

Таким чином, реалізація цієї програми дозволить покращити точність обліку, підвищити ефективність обробки інформації, а також створити основу для подальшої автоматизації облікових процесів у сфері виробництва та реалізації хлібобулочної продукції.

2.2 Елементи розробки інтерфейсу наведено у таблиці 2.1

Таблиця 2.1 – Перелік та призначення елементів інтерфейсу

Компонент або його властивість	Призначення
ComboBox1	Для вибору сорту хліба
ListBox1	Для вибору магазину

Продовження таблиці 2.1

CheckBox1	Для вибору чи хліб вищого гатунку
TLabel	Виведення пояснювальних написів
TEdit	Введення кількості штук хліба
Button1	Додавання запису до масиву
Button2	Рахування вимаганої в завданні інформації та записів у масиві
Memo1	Виведення інформації
Memo2	Виведення історії

2.3 Елементи розробки функцій програми наведено у таблиці 2.2

Таблиця 2.2 – Перелік та призначення обробників подій

	Функція	Обробник подій, що її реалізує	Модуль, у якому розміщується обробник подій
1.	Додати запис до масиву	TForm1::Button1Click	Unit1
2.	Вивести інформацію	TForm1::Button2Click	Unit1
3	Створення найменувань хліба та торгових точок	TForm1::FormCreate	Unit1

2.4 Вікно розробленої програми наведено на рисунку 2.1

Вікно Form 1 – Головне та єдине вікно програми, у якому відбуваються всі події. Забезпечує введення таких даних як сорт хліба, гатунок хліба, кількість одиниць товару, магазин.

Вікно надає список продукції хлібозаводу, дозволяє вибрати і вказати кількість продукції, розраховувати вартість закупівель, виробленої кожної з

чотирьох торгових точок, вартість мінімальної і максимальної закупівлі, а також визначати, з якого сорту борошна продукції закуповується більше. Програма визначає вартість продукції, проданої хлібозаводом. Результати розрахунків виводяться на екран.

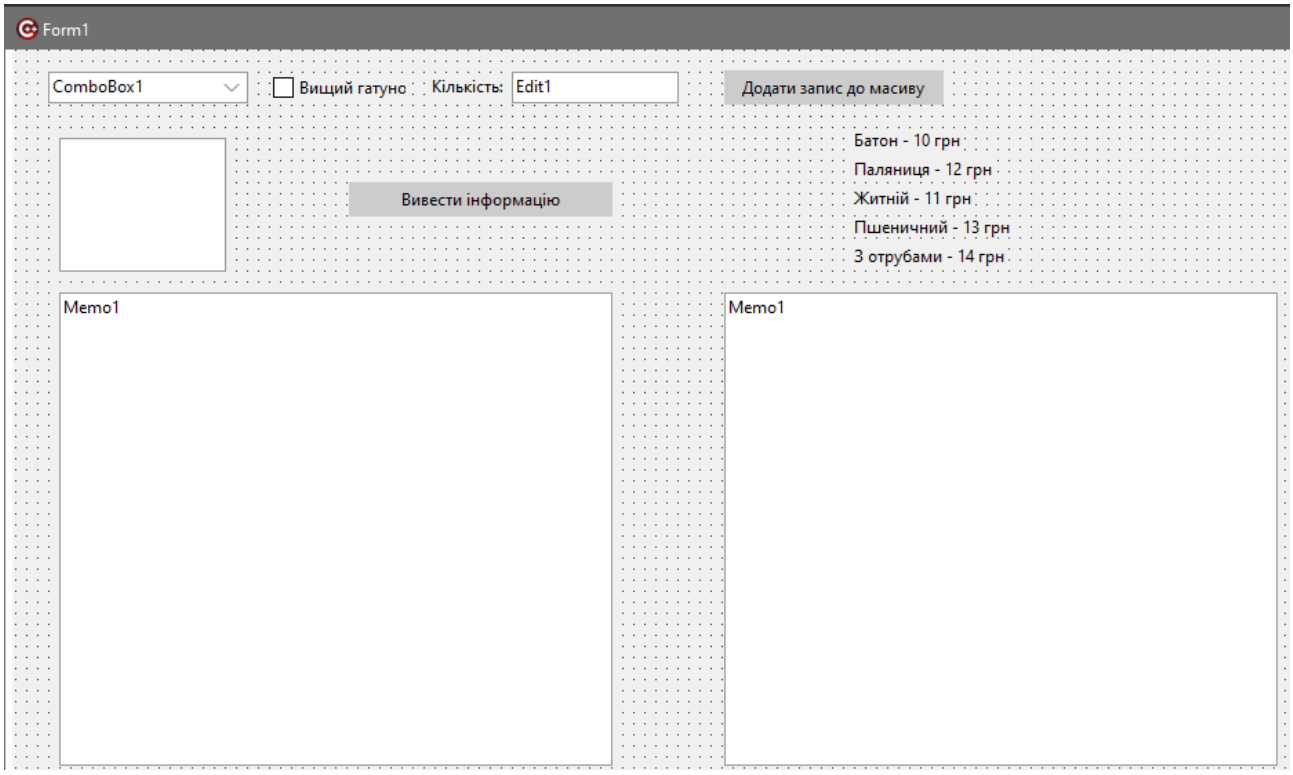


Рисунок 2.1 – Головне вікно програми

2.5 Текст розробленої програми наведено у лістингах 2.1—2.4

Лістинг 2.1—Специфікація класу ProductSale

```
#ifndef ProductSaleH
#define ProductSaleH

#include <string>

class ProductSale {
private:
    std::string product;
    std::string store;
    bool premiumFlour;
    int quantity;

public:
    ProductSale(std::string prod, std::string st, bool premium, int qty);

    std::string getProduct() const;
    std::string getStore() const;
```

```
    bool isPremium() const;
    int getQuantity() const;
    double getUnitPrice() const;
    double getTotalPrice() const;
};

#endift
```

Лістинг 2.2—Реалізація класу ProductSale

```
#include "ProductSale.h"

ProductSale::ProductSale(std::string prod, std::string st, bool premium, int
qty)
    : product(prod), store(st), premiumFlour(premium), quantity(qty) {}

std::string ProductSale::getProduct() const {
    return product;
}

std::string ProductSale::getStore() const {
    return store;
}

bool ProductSale::isPremium() const {
    return premiumFlour;
}

int ProductSale::getQuantity() const {
    return quantity;
}

// Базові ціни для хліба
double ProductSale::getUnitPrice() const {
    // Можна винести в мапу, але тут спрощено
    double basePrice = 0.0;
    if (product == "Батон") basePrice = 10.0;
    else if (product == "Паляниця") basePrice = 12.0;
    else if (product == "Житній") basePrice = 11.0;
    else if (product == "Пшеничний") basePrice = 13.0;
    else if (product == "З отрубами") basePrice = 14.0;

    if (premiumFlour)
        basePrice *= 1.25; // +25%

    return basePrice;
}

double ProductSale::getTotalPrice() const {
    return getUnitPrice() * quantity;
}e
```

Лістинг 2.3—Головний файл

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "ProductSale.h" // Підключення класу
```

```
#include <vector>        // Потрібен для std::vector
#include <string>
#include <limits>
#include <map>
#include <sstream>
#include <algorithm>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
// Масив усіх продажів
std::vector<ProductSale> sales;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    // Перевірка кількості
    if (Edit1->Text.IsEmpty())
    {
        MessageBox(Handle, L"Введіть кількість!", L"Помилка", MB_OK |
MB_ICONERROR);
        return;
    }

    int quantity = Edit1->Text.ToIntDef(-1);
    if (quantity <= 0)
    {
        MessageBox(Handle, L"Кількість має бути цілим числом більше 0!",
L"Помилка", MB_OK | MB_ICONERROR);
        return;
    }

    // Зчитування інших значень
    String breadName = ComboBox1->Text;
    String storeName = ListBox1->Items->Strings[ListBox1->ItemIndex];
    bool isPremium = CheckBox1->Checked;

    // Перетворення в std::string
    std::string prod = AnsiString(breadName).c_str();
    std::string st = AnsiString(storeName).c_str();

      // Створення об'єкта
      ProductSale sale(prod, st, isPremium, quantity);

      // Додавання до масиву
      sales.push_back(sale);

    // Повідомлення
    MessageBox(Handle, L"Запис успішно додано!", L"Готово", MB_OK |
MB_ICONINFORMATION);

    // Очистити поле вводу
    Edit1->Text = L"";
}

//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    // 5 найменувань хліба
    ComboBox1->Items->Add(L"Батон");
```

```
    ComboBox1->Items->Add(L"Паляниця");
    ComboBox1->Items->Add(L"Житній");
      ComboBox1->Items->Add(L"Пшеничний");
    ComboBox1->Items->Add(L"З отрубами");
    ComboBox1->ItemIndex = 0;

    // 4 торгові точки
    ListBox1->Items->Add(L"Магазин №1");
    ListBox1->Items->Add(L"Магазин №2");
    ListBox1->Items->Add(L"Магазин №3");
    ListBox1->Items->Add(L"Магазин №4");
    ListBox1->ItemIndex = 0;

    // За замовчуванням встановимо порожній рядок у Edit1
    Edit1->Text = L"";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    if (sales.empty()) {
        MessageBox(Handle, L"Немає жодного запису для аналізу.", L"Помилка",
MB_OK | MB_ICONERROR);
        return;
    }

    Memo1->Clear();

    std::map<std::string, double> storeTotals;
    double totalSum = 0;
    double minPurchase = std::numeric_limits<double>::max();
    double maxPurchase = std::numeric_limits<double>::min();
    std::string storeMin, storeMax;
    int premiumTotalQty = 0, regularTotalQty = 0;

    for (const auto& sale : sales) {
        double totalPrice = sale.getTotalPrice();
        storeTotals[sale.getStore()] += totalPrice;
        totalSum += totalPrice;

        if (totalPrice < minPurchase) {
            minPurchase = totalPrice;
            storeMin = sale.getStore();
        }
        if (totalPrice > maxPurchase) {
            maxPurchase = totalPrice;
            storeMax = sale.getStore();
        }

        if (sale.isPremium())
            premiumTotalQty += sale.getQuantity();
        else
            regularTotalQty += sale.getQuantity();
    }

    // Вивід даних у Memo1
    Memo1->Lines->Add(L"Загальна вартість по кожній торговій точці:");
    for (const auto& pair : storeTotals) {
        String line = String(pair.first.c_str()) + L": " +
FloatToStrF(pair.second, ffFixed, 10, 2) + L" грн";
        Memo1->Lines->Add(line);
    }

    Memo1->Lines->Add(L"");
    Memo1->Lines->Add(L"Мінімальна закупівля:");
```

```
    Memo1->Lines->Add(String(storeMin.c_str()) + L": " +
FloatToStrF(minPurchase, ffFixed, 10, 2) + L" грн");

    Memo1->Lines->Add(L"Максимальна закупівля:");
    Memo1->Lines->Add(String(storeMax.c_str()) + L": " +
FloatToStrF(maxPurchase, ffFixed, 10, 2) + L" грн");

    Memo1->Lines->Add(L"");

    if (premiumTotalQty > regularTotalQty)
        Memo1->Lines->Add(L"Більше закуплено продукції з ВИЩОГО сорту
борошна.");
    else if (premiumTotalQty < regularTotalQty)
        Memo1->Lines->Add(L"Більше закуплено продукції з ПЕРШОГО сорту
борошна.");
    else
        Memo1->Lines->Add(L"Однакова кількість продукції з обох сортів
борошна.");

    Memo1->Lines->Add(L"");

    Memo1->Lines->Add(L"Загальна вартість усієї продукції, проданої
хлібозаводом:");
      Memo1->Lines->Add(FloatToStrF(totalSum, ffFixed, 10, 2) + L" грн");

    Memo2->Clear();

      for (const auto& sale : sales)
      {
            String product = String(sale.getProduct().c_str());
            String store = String(sale.getStore().c_str());
            String flourType = sale.isPremium() ? "Вищий ґатунок" : "Перший
ґатунок";
            int quantity = sale.getQuantity();
            double unitPrice = sale.getUnitPrice();
            double totalPrice = sale.getTotalPrice();

            String line = "Товар: " + product +
                          ", Точка: " + store +
                          ", Сорт: " + flourType +
                          ", Кількість: " + String(quantity) +
                          ", Ціна за одиницю: " +
String().sprintf(L"%.2f", unitPrice) +
                          ", Загальна вартість: " +
String().sprintf(L"%.2f", totalPrice);

            Memo2->Lines->Add(line);
      }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
      Memo2->Clear();
      // Memo2->Lines->Add(L"Мінімальна закупівля: " +
String(storeMinGlobal.c_str()));
      // Memo2->Lines->Add(L"Максимальна закупівля: " +
String(storeMaxGlobal.c_str()));
}
//---------------------------------------------------------------------------
```

Лістинг 2.4—Специфікація TForm

```
//---------------------------------------------------------------------------

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <vector>
#include <string>
#include "ProductSale.h"
//#include "ProductSale.h"
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
      TComboBox *ComboBox1;
      TListBox *ListBox1;
      TLabel *Label1;
      TEdit *Edit1;
      TButton *Button1;
      TMemo *Memo1;
      TCheckBox *CheckBox1;
      TButton *Button2;
      TLabel *Label2;
      TLabel *Label3;
      TLabel *Label4;
      TLabel *Label5;
      TLabel *Label6;
      TMemo *Memo2;
      void __fastcall Button1Click(TObject *Sender);
      void __fastcall FormCreate(TObject *Sender);
      void __fastcall Button2Click(TObject *Sender);
      void __fastcall Button3Click(TObject *Sender);
private:    // User declarations
      std::vector<ProductSale> sales; // Оголошення масиву записів

public:           // User declarations
      __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

2.6 Результат роботи програми наведено на рисунках 2.2—2.4

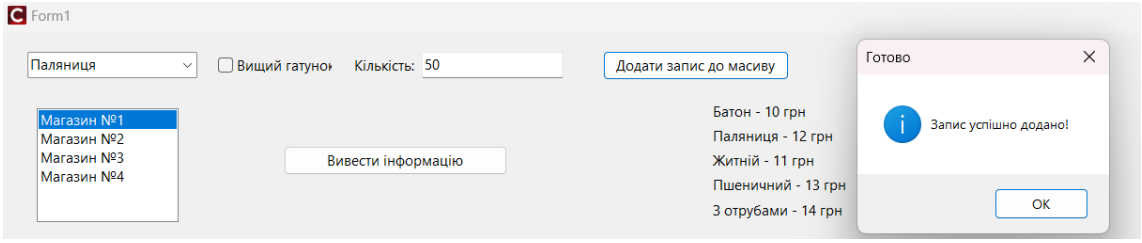


Рисунок 2.2 – Додавання запису.

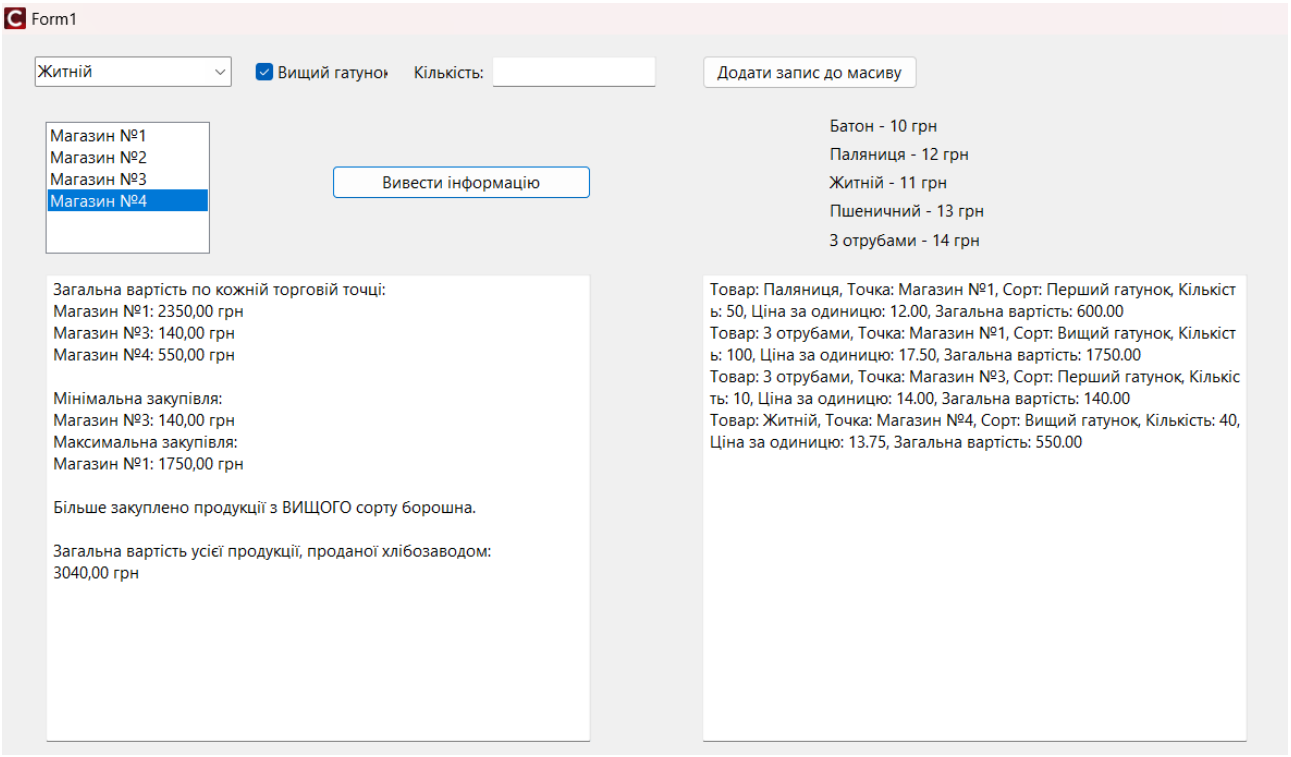


Рисунок 2.3 – Вивід інформації.

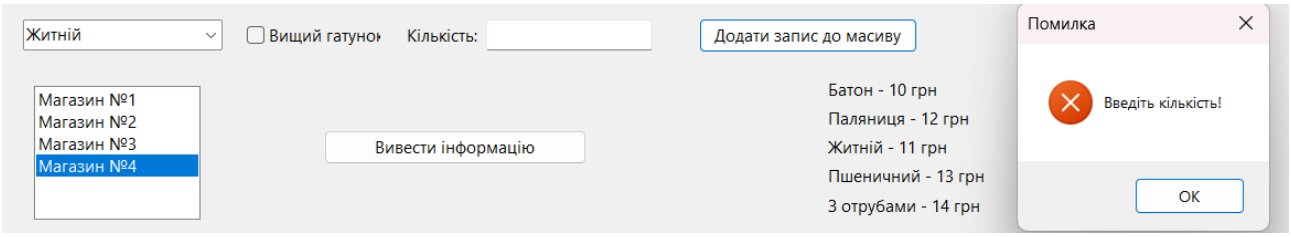


Рисунок 2.4 – Помилка введення кількості.

Висновки: закріпив навички з проведення аналізу предметної області, визначення актуальності та призначення, формулювання постановки задачі; навички з використання в програмі компонентів-списків, компонентів-випадаючих списків та обробників подій вибору в списку.

ЗАВДАННЯ № 3

Розробка алгоритму математичного розрахунку. Проектування та розробка інтерфейсу додатку. Перевірка коректності введених даних. Використання властивостей та методів компоненту, що відтворює прогрес

Мета: закріпити навички з розробки алгоритму математичного розрахунку, проектування та розробки інтерфейсу додатку, перевірки коректності введених даних, навчитися використовувати властивості та методи компоненту, що відтворює прогрес.

Хід роботи

3.1 Постановка задачі

При довготривалих обчисленнях для відображення ходу виконання програми може бути використаний компонент TProgressBar.

Розробити програму, яка обчислює й виводить на екран у табличному вигляді значення функції, заданої за допомогою ряду Тейлора, на інтервалі від $x_{нач}$ до $x_{кон}$.

Крок, з котрим розраховується функція на інтервалі від $x_{нач}$ до $x_{кон}$., дорівнює dx.

Точність розрахунку функції в кожній точці інтервалу дорівнює Epsil.

Кожний рядок таблиці повинен містити значення аргументу, значення функції й кількість просумованих членів ряду.

Таблицю постачити заголовком і шапкою.

Значення Хнач., Хкін., кількість точок розрахунку, на основі якої розраховується крок dx, а також точність Epsil повинні вводитися із клавіатури.

Обов'язково спроектувати в програмі власний клас (окрім форми), в який винести логіку та метод розрахунку в точці.

3. $$\frac{1}{1+x} = \sum_{n=0}^{\infty} (-1)^n x^n = 1 - x + x^2 - x^3 + \ldots \qquad (-1 < x < 1)$$

3.2 Елементи розробки інтерфейсу наведено у таблиці 3.1

Таблиця 3.1 – Перелік та призначення елементів інтерфейсу

Елемент	Призначення
Edit1	Для вводу початкового x
Edit2	Для вводу кінцевого x
Edit3	Для вводу кількості точок розрахунку
Edit4	Для вводу кількості ітерацій в точці
Edit5	Для вводу точності розрахунку
Label1, …Label5	Для підказки введення
Memo1	Для виведення результатів обчислення
ProgressBar1	Для відображення прогресу обчислення
Button1	Для обчислення формули за вхідними даними

3.3 Елементи розробки функцій програми наведено у таблиці 3.2

Таблиця 3.2 – Перелік та призначення обробників подій

	Функція	Обробник подій, що її реалізує	Модуль, у якому розміщується обробник подій
1	Обчислення формули за варіантом та виведення результату, а також для відображеня прогресу обчислення через ProgressBar1.	TForm1::Button1Click	Unit1

3.4 Вікно розробленої програми наведено на рисунку 3.1

Вікно TForm1—Головне та єдине вікно програми, у якому відбуваються всі події, такі як введення даних, виведення результату обчислення та відображення прогресу обчислення.

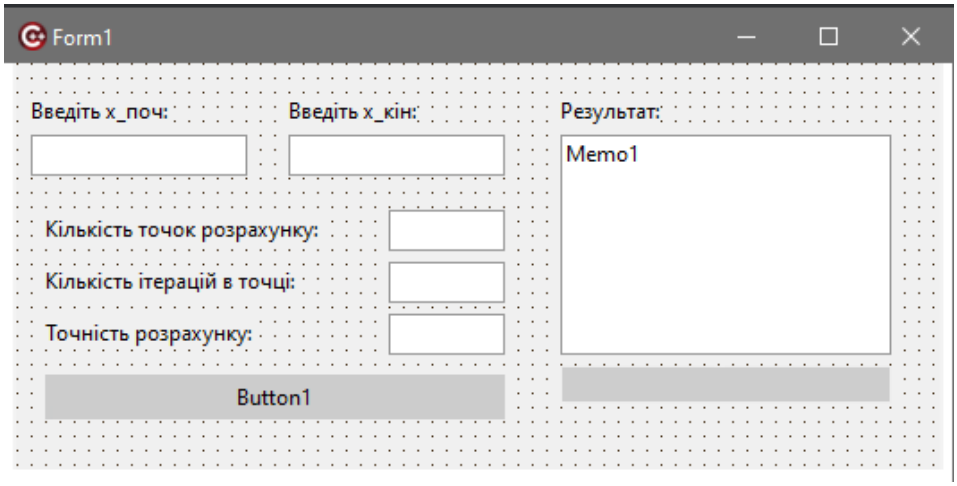


Рисунок 3.1 — Головне вікно програми

3.5 Текст розробленої програми наведено у лістингах 3.1—3.4

Лістинг 3.1—Специфікація класу Formula

```
#ifndef FormulaH
#define FormulaH

class Formula {
private:
    double x;              // аргумент
    double epsilon;        // точність обчислення
    int maxIterations;     // максимальна кількість ітерацій
    int iterationsUsed;    // кількість просумованих членів

public:
    Formula(double x, double eps, int maxIter);

    double calculate();    // обчислює значення функції в точці x
    int getIterationsUsed() const; // повертає кількість просумованих членів
};

#endif
```

Лістинг 3.2—Реалізація класу Formula

```
#include "Formula.h"
#include <cmath>

Formula::Formula(double x, double eps, int maxIter) {
    this->x = x;
    this->epsilon = eps;
    this->maxIterations = maxIter;
    this->iterationsUsed = 0;
}

double Formula::calculate() {
            double sum = 0.0;
        double term;
            iterationsUsed = 0;
```

```
        for (int n = 0; n < maxIterations; ++n) {
        term = pow(-1, n) * pow(x, n);
        sum += term;
        iterationsUsed++;
        if (fabs(term) < epsilon) break;
      }

        return sum;
}

int Formula::getIterationsUsed() const {
    return iterationsUsed;
}
```

Лістинг 3.3—Головний файл

```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
#include "Formula.h"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
try {
      if (Edit1->Text.IsEmpty() || Edit2->Text.IsEmpty() || Edit3-
>Text.IsEmpty() || Edit4->Text.IsEmpty() || Edit5->Text.IsEmpty()) {
            ShowMessage("Всі поля мають бути заповнені!");
            return;
            }


            double xStart = StrToFloat(Edit1->Text);
            double xEnd = StrToFloat(Edit2->Text);
            int points = StrToInt(Edit3->Text);
            int maxIter = StrToInt(Edit4->Text);
            double eps = StrToFloat(Edit5->Text);

            if (points < 2) {
                  MessageBox(Handle, L"Кількість точок повинна бути більше 1",
L"Помилка", MB_OK | MB_ICONERROR);
                  return;
            }
            if (xStart <= -1 || xEnd >= 1) {
                  MessageBox(Handle, L"Вихід за межі збіжності ряду (-1 < x <
1)", L"Помилка", MB_OK | MB_ICONERROR);
                  return;
            }
        if (xEnd <= xStart || maxIter <= 0 || eps > 0.1 ) {
            ShowMessage("Перевірте правильність введених даних!");
            return;
            }
```

```
            double dx = (xEnd - xStart) / (points - 1);
            Memo1->Lines->Clear();
            Memo1->Lines->Add("  x\t\tf(x)\t\tN");

        ProgressBar1->Min = 0;
            ProgressBar1->Max = points;
            ProgressBar1->Position = 0;

            for (int i = 0; i < points; ++i) {
                  double x = xStart + i * dx;

                  if (x <= -1 || x >= 1) {
                        Memo1->Lines->Add(FormatFloat("0.000", x) + "\tOut of
range");
                        continue;
                  }

                  Formula formula(x, eps, maxIter);
                  double fx = formula.calculate();
                  int used = formula.getIterationsUsed();

                  String line = FormatFloat("0.000", x) + "\t" +
FormatFloat("0.000000", fx) + "\t" + IntToStr(used);
                  Memo1->Lines->Add(line);

            ProgressBar1->Position = i + 1; // Оновлення прогресу
                  Application->ProcessMessages(); // Дозволяє оновити інтерфейс
            }
      }
      catch (...) {
            MessageBox(Handle, L"Помилка введення даних", L"Помилка", MB_OK |
MB_ICONERROR);
      }

}
//---------------------------------------------------------------------------
```

Лістинг 3.4—Специфікація TForm

```
//---------------------------------------------------------------------------

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Vcl.ComCtrls.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
      TEdit *Edit1;
      TEdit *Edit2;
      TEdit *Edit3;
      TMemo *Memo1;
      TButton *Button1;
      TProgressBar *ProgressBar1;
      TLabel *Label1;
      TLabel *Label2;
```

```
        TLabel *Label3;
        TLabel *Label4;
        TEdit *Edit4;
        TEdit *Edit5;
        TLabel *Label5;
        TLabel *Label6;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

3.6 Результат роботи програми наведено на рисунках 3.2—3.3

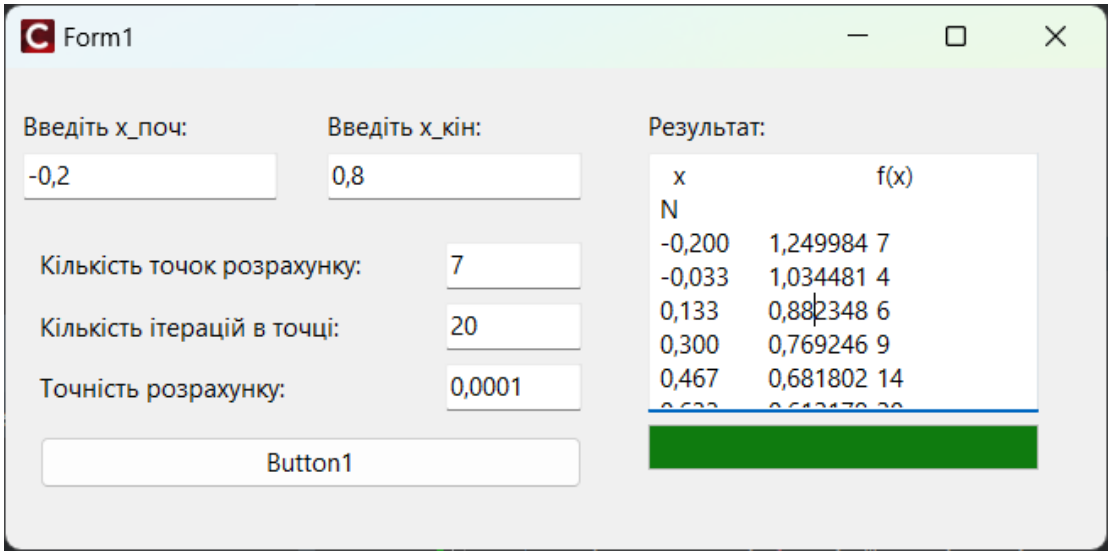


Рисунок 3.2 — Обчислення формули

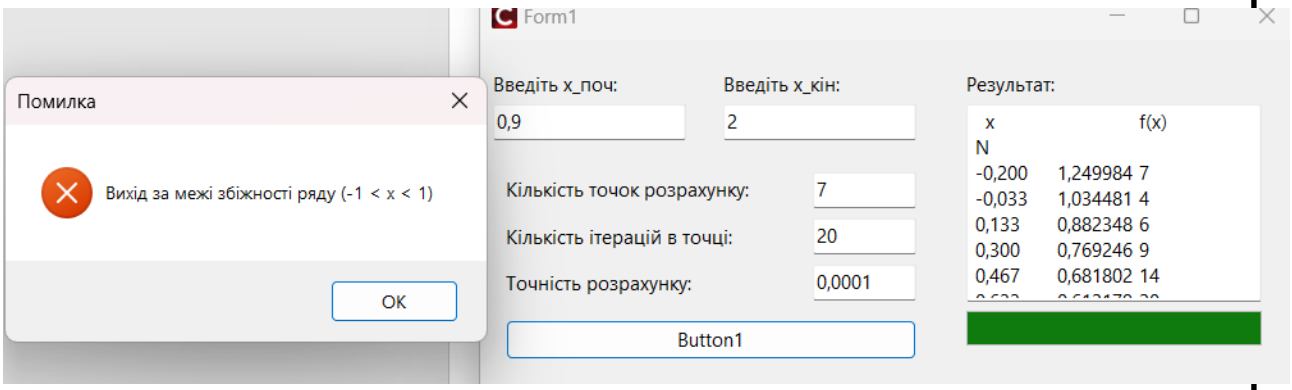


Рисунок 3.3 — Приклад помилки коректності введення даних

Висновки: закріплено навички з розробки алгоритму математичного розрахунку, проектування та розробки інтерфейсу додатку, перевірки коректності введених даних, отримано навички з використання властивостей та методів компоненту, що відтворює прогрес.

ЗАВДАННЯ № 4

Розробка програми з використанням компонентів-залежних перемикачів. Робота зі списками, які формуються під час роботи програми. Робота з текстовими файлами в програмі

Мета: закріпити навички з використання компонентів-залежних перемикачів, формування списків під час виконання програми, роботи з текстовими файлами в програмі.

Хід роботи

4.1 Постановка задачі

Створити програму, яка виконає вибір варіанта розрахунку функції, збереження результатів розрахунку в файл. Обов'язково передбачити створення власного класу (окрім форми), в якому буде логіка роботи, виконання розрахунку.

Варіант 3

$$F = \begin{cases} ax^2 + bx + c & при\ a < 0\ и\ c \neq 0 \\ \dfrac{-a}{x - c} & при\ a > 0\ и\ c = 0 \\ a(x + c) & в\ інших\ випадках \end{cases}$$

4.2 Елементи розробки інтерфейсу наведено у таблиці 4.1

Таблиця 4.1 – Перелік та призначення елементів інтерфейсу

Елемент	Призначення
Edit1	Для вводу a
Edit2	Для вводу b
Edit3	Для вводу c
Edit4	Для вводу початкового x
Edit5	Для вводу кінцевого x
Edit6	Для вводу dX
Edit7	Для вводу x
Label1, …Label7	Для підказки введення

Продовження таблиці 4.1

Елемент	Призначення
Memo1	Для виведення результатів обчислення
RadioButton1	Для вибору обчислення функції на інтервалі
RadioButton2	Для вибору обчислення функції в точці
Button1	Для обчислення функції
Button2	Для збереження результату у файл
GroupBox1,	Для позначення заголовків груп компонентів

4.3 Елементи розробки функцій програми наведено у таблиці 4.2

Таблиця 4.2 – Перелік та призначення обробників подій

	Функція	Обробник подій, що її реалізує	Модуль, у якому розміщується
1	Обчислення функції за варіантом та обраним методом обчислення, а також для виведення результату.	TForm1::Button1Click	Unit1
2	Збереженя до файлу результатів обчислення.	TForm1::Button2Click	Unit1
3	Приховання компонентів введеня даних для методу розрахунку в точці та відкриття компонентів введення даних для методу розрахунку на інтервалі.	TForm1::RaiodButton1Click	Unit1
4	Приховання компонентів введеня даних для методу розрахунку на інтервалі та відкриття компонентів введення даних для методу розрахунку в точці.	TForm1::RaiodButton2Click	Unit1

4.4 Вікно розробленої програми наведено на рисунку 4.1

Вікно TForm1—Головне та єдине вікно програми, у якому відбуваються всі події, такі як введення даних, виведення результату обчислення та можливість обрати метод обчислення.

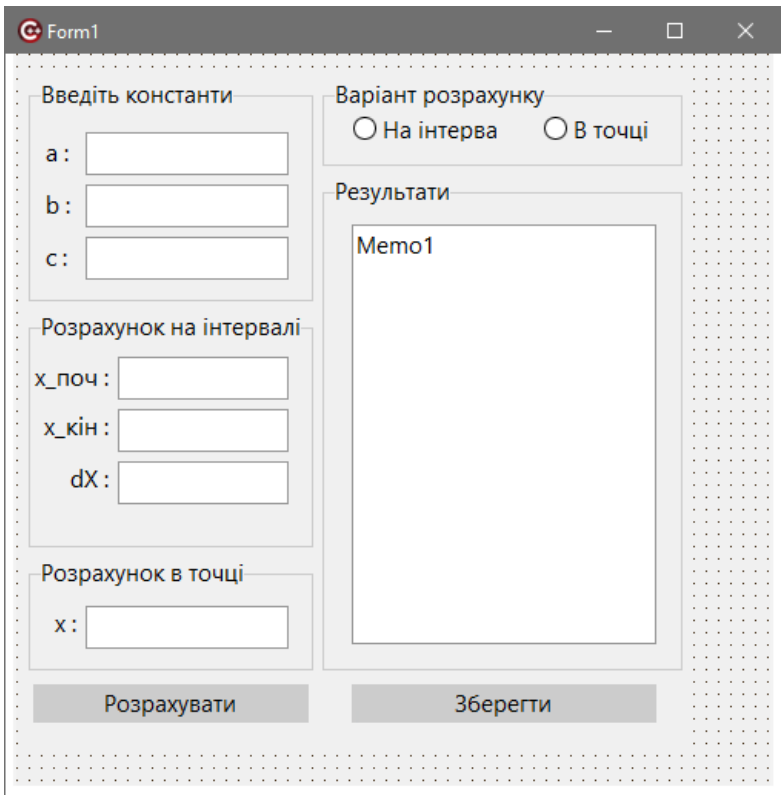


Рисунок 4.1 — Головне вікно програми

4.5 Текст розробленої програми наведено у лістингах 4.1—4.4

Лістинг 4.1—Специфікація класу Func

```
#ifndef FunctionF_H
#define FunctionF_H

class FunctionF {
private:
    double a, b, c;

public:
    FunctionF(double a_, double b_, double c_);
    double calculate(double x);
};

#endif
```

Лістинг 4.2—Реалізація класу Func

```
#include "FunctionF.h"

FunctionF::FunctionF(double a_, double b_, double c_) {
    a = a_;
```

```
    b = b_;
    c = c_;
}

double FunctionF::calculate(double x) {
    if (a < 0 && c != 0)
        return a * x * x + b * x + c;
    else if (a > 0 && c == 0 && x != c)  // захист від ділення на 0
        return -a / (x - c);
    else
        return a * (x + c);
}
```

Лістинг 4.3—Головна функція

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "FunctionF.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
if (Edit1->Text.IsEmpty() || Edit2->Text.IsEmpty() || Edit3->Text.IsEmpty()) {
            ShowMessage("Всі поля мають бути заповнені!");
            return;
      }

try {
        double a = StrToFloat(Edit1->Text);
        double b = StrToFloat(Edit2->Text);
        double c = StrToFloat(Edit3->Text);

        FunctionF func(a, b, c);

        Memo1->Lines->Clear();
        Memo1->Lines->Add("   X\t\t|\tF(X)");
        Memo1->Lines->Add("---------------------------");

            if (RadioButton1->Checked) {

        if (Edit4->Text.IsEmpty() || Edit5->Text.IsEmpty() || Edit6-
>Text.IsEmpty()) {
                  ShowMessage("Всі поля мають бути заповнені!");
                  return;
            }

            double xStart = StrToFloat(Edit4->Text);
            double xEnd = StrToFloat(Edit5->Text);
            double dx = StrToFloat(Edit6->Text);

            if (dx <= 0 || xEnd < xStart) {
                ShowMessage("Помилка: dx повинен бути > 0, xEnd ≥ xStart");
                return;
```

```
            }

            for (double x = xStart; x <= xEnd; x += dx) {
                double fx = func.calculate(x);
                Memo1->Lines->Add(FormatFloat("0.000", x) + "\t|\t" +
FormatFloat("0.000", fx));
            }

            } else if (RadioButton2->Checked) {

        if (Edit7->Text.IsEmpty()) {
                ShowMessage("Всі поля мають бути заповнені!");
                return;
            }

            double x = StrToFloat(Edit7->Text);
            double fx = func.calculate(x);
            Memo1->Lines->Add(FormatFloat("0.000", x) + "\t|\t" +
FormatFloat("0.000", fx));
        }
    }
    catch (...) {
        ShowMessage("Помилка: Невірне введення даних.");
    }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{

    if (Memo1->Lines->Count == 0)
      {
            ShowMessage("Немає результатів для збереження.");
            return;
      }

      String fileName = "result.txt";

      Memo1->Lines->SaveToFile(fileName);

      ShowMessage("Результати збережено у файл: " + fileName);

}
//---------------------------------------------------------------------------
void __fastcall TForm1::RadioButton1Click(TObject *Sender)
{
    GroupBox3->Enabled = true;  // інтервал
    GroupBox4->Enabled = false; // одна точка
}
//---------------------------------------------------------------------------
void __fastcall TForm1::RadioButton2Click(TObject *Sender)
{
    GroupBox3->Enabled = false;
    GroupBox4->Enabled = true;
}
//---------------------------------------------------------------------------
```

Лістинг 4.4—Специфікація TForm

```
//---------------------------------------------------------------------------

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

#ifndef Unit1H
```

```
#define Unit1H
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
      TGroupBox *GroupBox1;
      TRadioButton *RadioButton1;
      TRadioButton *RadioButton2;
      TGroupBox *GroupBox2;
      TEdit *Edit1;
      TEdit *Edit2;
      TEdit *Edit3;
      TGroupBox *GroupBox3;
      TEdit *Edit4;
      TEdit *Edit5;
      TEdit *Edit6;
      TGroupBox *GroupBox4;
      TEdit *Edit7;
      TGroupBox *GroupBox5;
      TMemo *Memo1;
      TGroupBox *GroupBox6;
      TButton *Button1;
      TGroupBox *GroupBox7;
      TButton *Button2;
      TLabel *Label1;
      TLabel *Label2;
      TLabel *Label3;
      TLabel *Label4;
      TLabel *Label5;
      TLabel *Label6;
      TLabel *Label7;
      void __fastcall Button1Click(TObject *Sender);
      void __fastcall Button2Click(TObject *Sender);
      void __fastcall RadioButton1Click(TObject *Sender);
      void __fastcall RadioButton2Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
      __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

4.6 Результат виконання програми наведено на рисунках 4.2—4.6

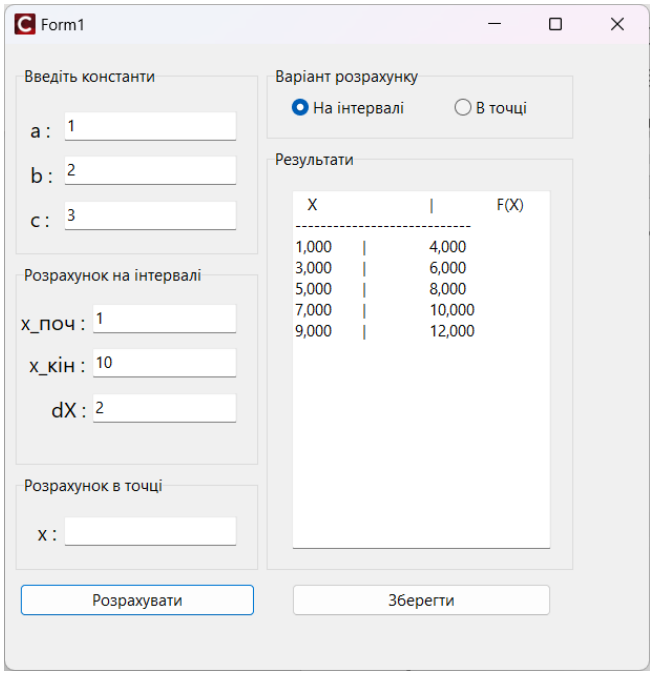


Рисунок 4.2 — Розрахунок на інтервалі

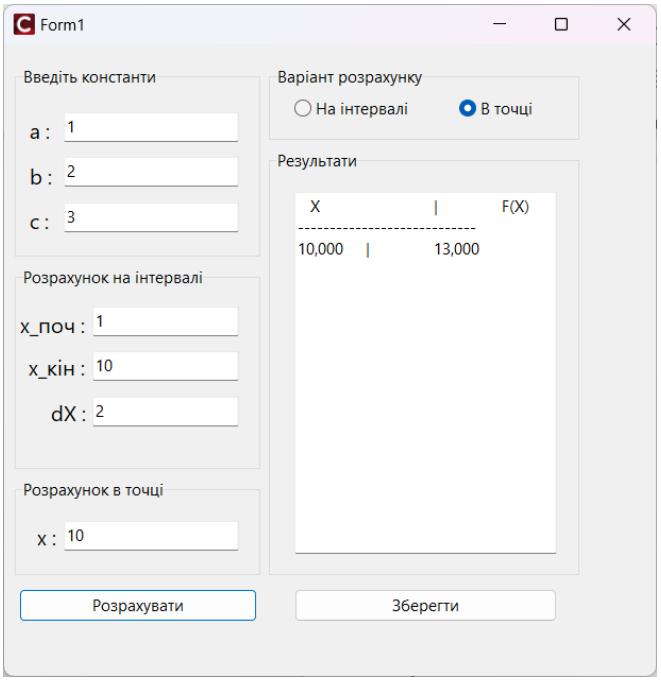


Рисунок 4.3 — Розрахунок в точці

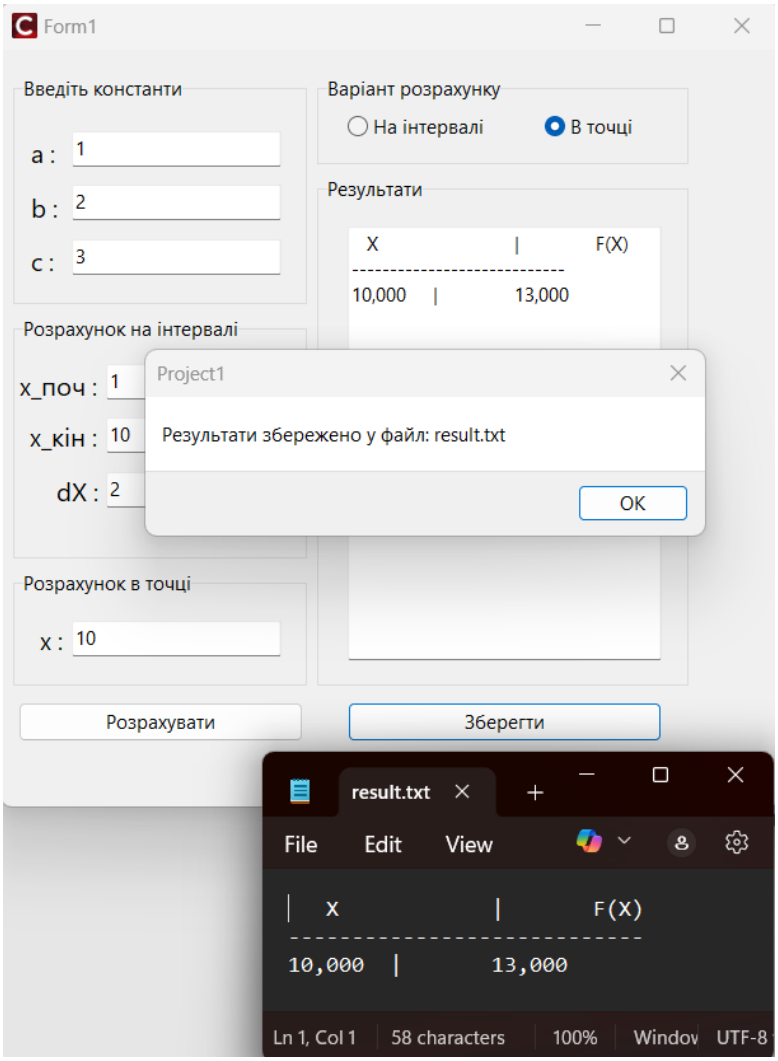


Рисунок 4.4 — Збереження до файлу результату обчислення

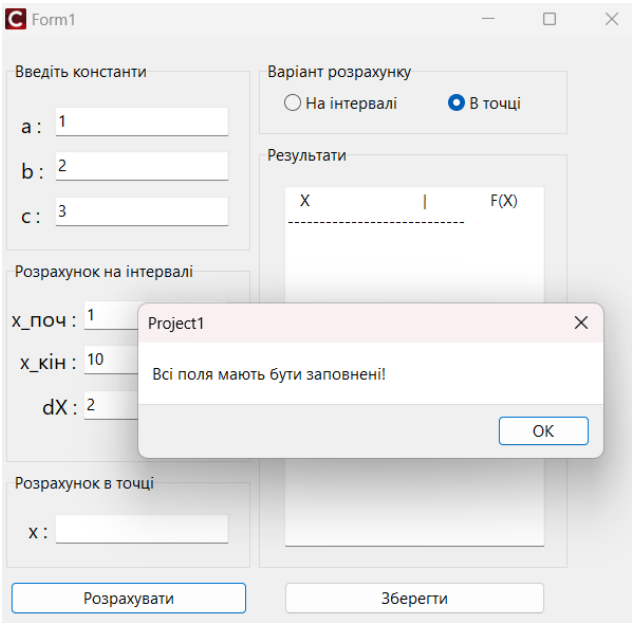


Рисунок 4.5 — Контроль вхідних даних при розрахунку

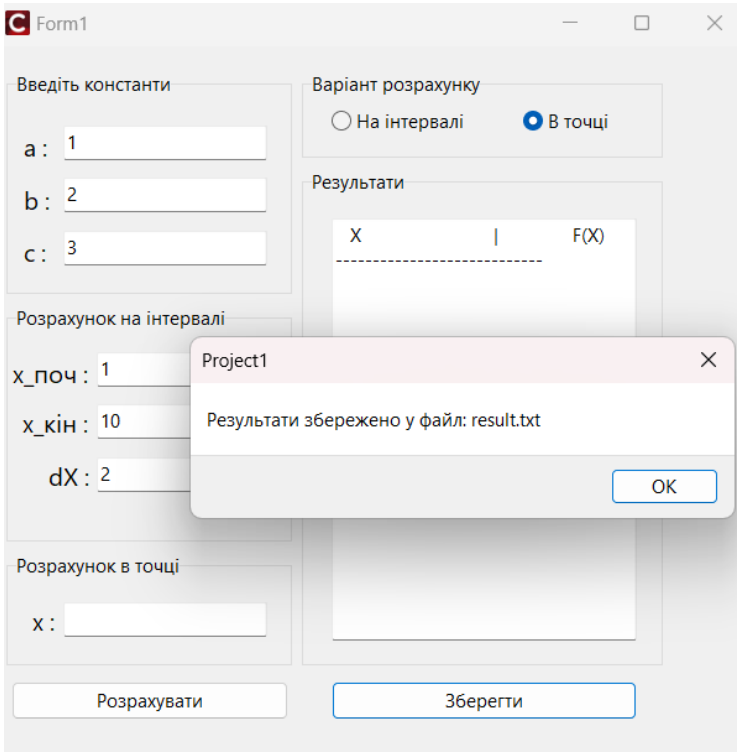


Рисунок 4.6 — Контроль вхідних даних при збереженні

Висновки: закріплено навички з використання компонентів-залежних перемикачів, формування списків під час виконання програми, роботи з текстовими файлами в програмі.

ЗАВДАННЯ № 5

Проєктування шаблону . Організація доступу до елементів керування вікна. Формування елементів масиву на основі списку. Проектування додатку на основі об'єктно-орієнтованого підходу. Створення програми з проектуванням полів та методів власного класу

Мета: закріпити навички з проєктування шаблону, організації доступу до елементів керування вікна, формування елементів масиву на основі списку, проектування додатку на основі об'єктно-орієнтованого підходу, полів та методів власного класу.

Хід роботи

5.1 Постановка задачі

Задано одномірний динамічний масив. Створити програму, яка на основі заданого масиву розраховує довжину нового масиву і його елементи. Спроектувати власний клас, винести в клас логіку і відповідні розрахунки.

Варіант 3.

1 Замінити всі негативні елементи нового масиву їх квадратами і перенести в кінець масиву.

2 Обчислити добуток елементів масиву з парними номерами.

3 Обчислити суму елементів масиву, розташованих між першим і останнім нульовими елементами.

4 Перетворити масив таким чином, щоб спочатку розташовувалися всі позитивні елементи і 0, а потім все негативні (не сортування, а перестановка).

5.2 Елементи розробки інтерфейсу наведено у таблиці 5.1

Таблиця 5.1 – Перелік та призначення елементів інтерфейсу

Елемент	Призначення
Edit1	Для вводу масиву
Memo1	Для виводу обробленого масиву та результатів обчислень
Label1, Label2	Для підказки введення
Button1	Для обробки та виводу масиву
Edit2	Для виводу довжини нового масиву

5.3 Елементи розробки функцій програми наведено у таблиці 5.2

Таблиця 5.2 – Перелік та призначення обробників подій

	Функція	Обробник подій, що її реалізує	Модуль, у якому розміщується
1	Обробка та виведення введеного масиву.	TForm1::Button1Click	Unit1

5.4 Вікно розробленої програми наведено на рисунку 5.1

Вікно TForm1—Головне та єдине вікно програми, у якому відбуваються всі події, такі як заповнення масиву, його обробка та виведення.

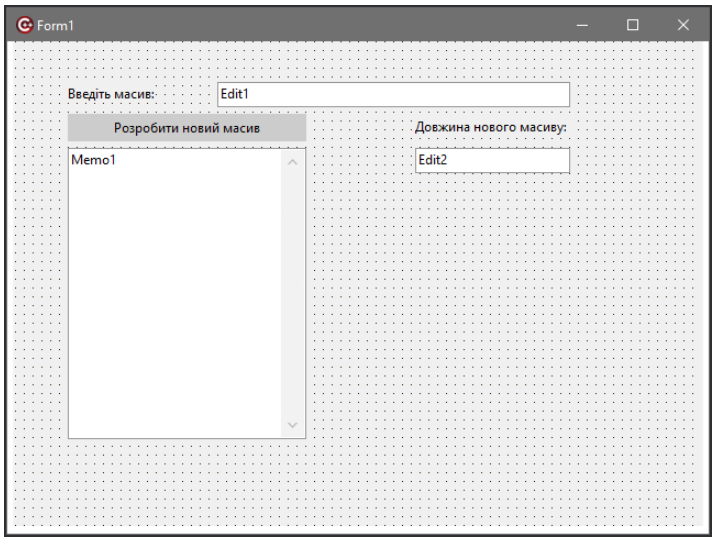


Рисунок 5.1 — Головне вікно програми

5.5 Текст розробленої програми наведено у лістингах 5.1—5.4

Лістинг 5.1 — Специфікація класу MyArray

```
#ifndef MyArrayH
#define MyArrayH

class MyArray
{
private:
    int* arr;        // динамічний масив
    int length;      // довжина масиву

public:
    MyArray(int* source, int len);   // конструктор, копіює елементи з source
    ~MyArray();                      // деструктор

    void processArray();             // виконує всі кроки: заміна, перестановка
    int productEvenIndices() const; // добуток елементів з парними індексами
    int sumBetweenZeros() const;    // сума між першим і останнім нулями

    void getArray(int*& outArr, int& outLen) const; // повертає масив і довжину

private:
    void replaceNegativesAndMoveToEnd();
    void rearrangePositiveZeroNegative();
};

#endif
```

Лістинг 5.2 — Реалізація класу MyArray

```
#include "MyArray.h"

MyArray::MyArray(int* source, int len)
{
    length = len;
    arr = new int[length];
    for (int i = 0; i < length; i++)
        arr[i] = source[i];
}

MyArray::~MyArray()
{
    delete[] arr;
}

void MyArray::replaceNegativesAndMoveToEnd()
{
    // Заміна негативних елементів квадратами
    for (int i = 0; i < length; i++)
        if (arr[i] < 0)
            arr[i] = arr[i] * arr[i];

    // Перенесення негативних (тепер це квадрати) в кінець
    // Але квадрат будь-якого від'ємного числа завжди >= 0,
    // отже їх потрібно відокремити як "негативні" за індексами ДО заміни.
    // Оскільки у пункті 4 потрібно розташувати спочатку позитивні та нулі,
    // а потім негативні, тут маємо після квадрування усі >= 0.
    // Тож логіку тут зробимо в rearrangePositiveZeroNegative.
}

void MyArray::rearrangePositiveZeroNegative()
{
```

```
    // Спочатку розташуємо усі позитивні елементи та нулі,
    // потім усі негативні (якщо такі залишилися, але в нашому випадку їх немає
після квадрування).

    // Тож в цій задачі після replaceNegativesAndMoveToEnd негативних немає,
    // але щоб дотриматися умови, переставимо масив так:

    int* temp = new int[length];
    int posIndex = 0;

    // Спочатку позитивні та нулі
    for (int i = 0; i < length; i++)
        if (arr[i] >= 0)
            temp[posIndex++] = arr[i];

    // Потім негативні (в нашій задачі після заміни їх немає)
    for (int i = 0; i < length; i++)
        if (arr[i] < 0)
            temp[posIndex++] = arr[i];

    for (int i = 0; i < length; i++)
        arr[i] = temp[i];

    delete[] temp;
}

void MyArray::processArray()
{
    replaceNegativesAndMoveToEnd();
    rearrangePositiveZeroNegative();
}

int MyArray::productEvenIndices() const
{
    int product = 1;
    bool hasEvenIndex = false;
    for (int i = 0; i < length; i += 2)
    {
        product *= arr[i];
        hasEvenIndex = true;
    }
    if (!hasEvenIndex) return 0; // якщо немає елементів з парними індексами
    return product;
}

int MyArray::sumBetweenZeros() const
{
    int firstZero = -1;
    int lastZero = -1;

    for (int i = 0; i < length; i++)
    {
        if (arr[i] == 0)
        {
            if (firstZero == -1) firstZero = i;
            lastZero = i;
        }
    }

    if (firstZero == -1 || lastZero == -1 || firstZero == lastZero)
        return 0; // немає двох нулів, або вони одні й ті ж

    int sum = 0;
    for (int i = firstZero + 1; i < lastZero; i++)
```

```
        sum += arr[i];

    return sum;
}

void MyArray::getArray(int*& outArr, int& outLen) const
{
    outLen = length;
    outArr = new int[length];
    for (int i = 0; i < length; i++)
        outArr[i] = arr[i];
}
```

Лістинг 5.3—Головний файл

```
//---------------------------------------------------------------------------
#include <vcl.h>
#include <sstream>
#include <vector>
#pragma hdrstop
#include "MyArray.h"
#include <string>
#include <cctype>
#include <cstdlib>
#include <limits>
#include "Unit1.h"
#include <cwctype> // для iswdigit
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Memo1->Clear();
    Edit2->Text = "";

    // Читаємо вхідний рядок і перевіряємо
    UnicodeString input = Edit1->Text.Trim();
      if (input.IsEmpty())
      {
            ShowMessage("Будь ласка, введіть елементи масиву через пробіл.");
            return;
      }

    // Додаткова перевірка: лише цифри, пробіли і знак «-» (тільки перед числом)
for (int i = 1; i < input.Length(); i++)
{
      if (!iswdigit(input[i]) && input[i] != ' ' && input[i] != '-')
      {
            ShowMessage("Вхідні дані містять недопустимі символи.");
            return;
      }
}

// Перевірка, що знак «-» стоїть тільки перед цифрою
for (int i = 1; i <= input.Length(); i++)
{
```

```
      if (input[i] == '-')
      {
            if (i == input.Length() || !iswdigit(input[i + 1]))
            {
                  ShowMessage("Неправильне розміщення знака мінус.");
                  return;
            }
      }
}

    // Розбиваємо рядок на числа
    TStringList *list = new TStringList();
    list->Delimiter = ' ';
    list->StrictDelimiter = true;
    list->DelimitedText = input;

    std::vector<int> inputVec;

    for (int i = 0; i < list->Count; i++)
    {
        try
        {
            int val = StrToInt(list->Strings[i]);
            inputVec.push_back(val);
        }
        catch(...)
        {
            ShowMessage("Неправильне введення: " + list->Strings[i]);
            delete list;
            return;
        }
    }
    delete list;

    // Виводимо довжину в Edit2 (ReadOnly)
    Edit2->Text = IntToStr((int)inputVec.size());

    // Створюємо об’єкт класу, який опрацьовує масив
    class ArrayProcessor
    {
        std::vector<int> arr;
        std::vector<int> newArr;

    public:
        ArrayProcessor(const std::vector<int>& input) : arr(input)
        {
            newArr = arr; // початково копіюємо
        }

        void process()
        {
            // 1. Заміна негативних на квадрат та перенесення у кінець
            std::vector<int> positivesZeros;
            std::vector<int> negSquares;

            for (int val : newArr)
            {
                if (val >= 0)
                    positivesZeros.push_back(val);
                else
                    negSquares.push_back(val * val);
            }
            // Об’єднуємо: спочатку позитивні + 0, потім квадрати негативних
            newArr.clear();
```

```
            newArr.insert(newArr.end(), positivesZeros.begin(),
positivesZeros.end());
            newArr.insert(newArr.end(), negSquares.begin(), negSquares.end());
        }

        int productEvenIndices() const
        {
            int prod = 1;
            bool hasEven = false;
            for (size_t i = 0; i < newArr.size(); i += 2)
            {
                prod *= newArr[i];
                hasEven = true;
            }
            return hasEven ? prod : 0;
        }

        int sumBetweenFirstLastZero() const
        {
            int firstZero = -1, lastZero = -1;
            for (size_t i = 0; i < newArr.size(); i++)
            {
                if (newArr[i] == 0)
                {
                    if (firstZero == -1)
                        firstZero = i;
                    lastZero = i;
                }
            }
            if (firstZero == -1 || lastZero == -1 || lastZero <= firstZero + 1)
                return 0; // немає нулів або немає елементів між ними

            int sum = 0;
            for (int i = firstZero + 1; i < lastZero; i++)
                sum += newArr[i];
            return sum;
        }

        const std::vector<int>& getNewArray() const
        {
            return newArr;
        }
    };

    ArrayProcessor processor(inputVec);
    processor.process();

    // Виводимо новий масив
    Memo1->Lines->Add("Новий масив:");
    std::vector<int> newArray = processor.getNewArray();
    for (int val : newArray)
        Memo1->Lines->Add(IntToStr(val));

    // Виводимо добуток елементів з парними індексами
    int prodEven = processor.productEvenIndices();
    Memo1->Lines->Add("Добуток елементів з парними індексами: " +
IntToStr(prodEven));

    // Виводимо суму між першим і останнім нулями
    int sumBetweenZeros = processor.sumBetweenFirstLastZero();
    Memo1->Lines->Add("Сума між першим та останнім нульовими елементами: " +
IntToStr(sumBetweenZeros));
}
//---------------------------------------------------------------------------
```

Лістинг 5.4—Специфікація TForm

```
//---------------------------------------------------------------------------

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
      TLabel *Label1;
      TEdit *Edit1;
      TButton *Button1;
      TMemo *Memo1;
      TLabel *Label2;
      TEdit *Edit2;
      void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
      __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

5.6 Результат виконання програми наведено на рисунках 5.2—5.5

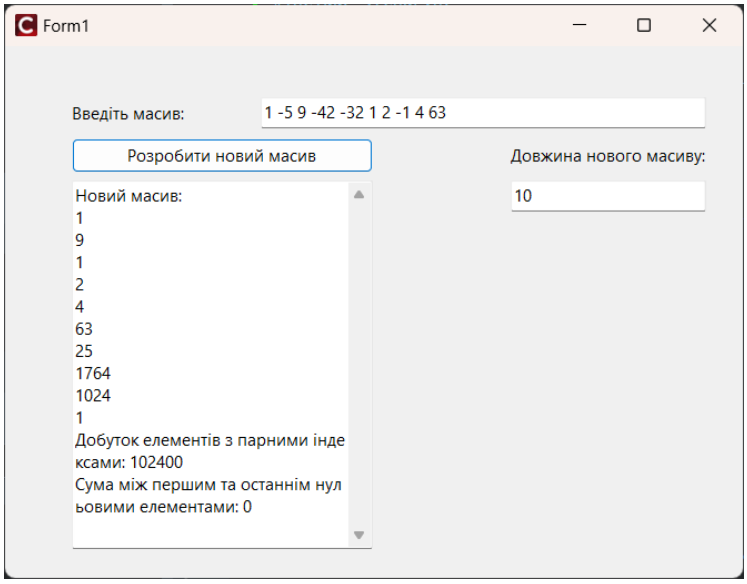


Рисунок 5.2 — Обробка масиву без нулів

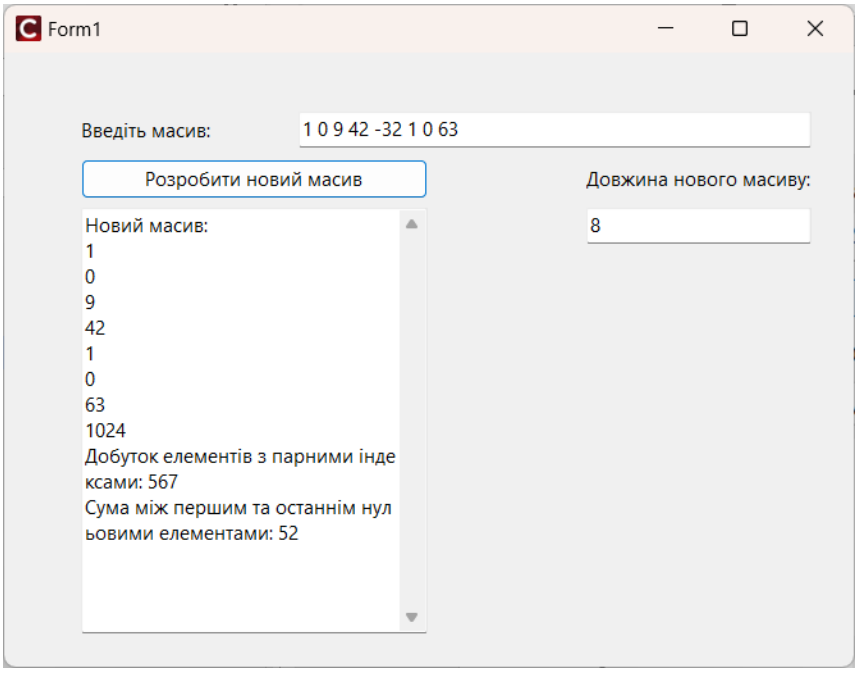


Рисунок 5.3 — Обробка масиву з нулями

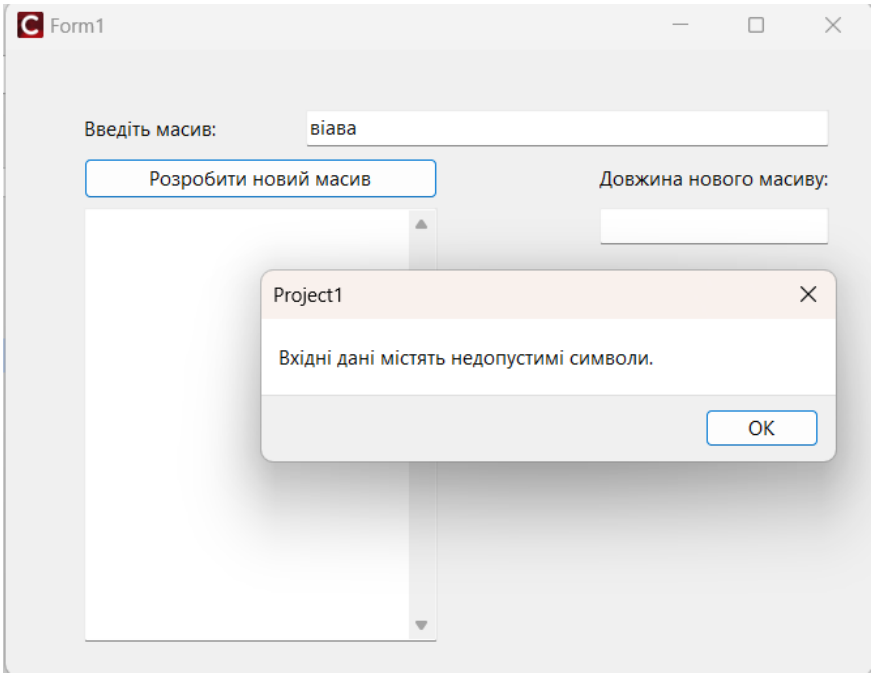


Рисунок 5.4 — Помилка при спробі некоректного введення

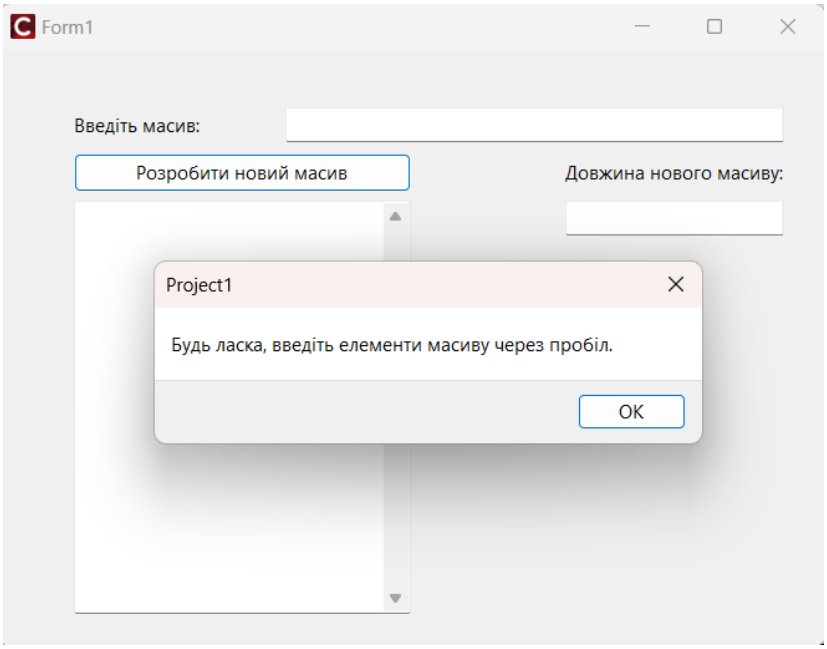


Рисунок 5.5 — Помилка при обробці порожнього масиву

Висновки: закріплено навички з проєктування шаблону, організації доступу до елементів керування вікна, формування елементів масиву на основі списку, проектування додатку на основі об'єктно-орієнтованого підходу, полів та методів власного класу.

ЗАВДАННЯ № 6

Створення та знищення об'єкту класу в межах додатку. Ініціалізація полів класу в конструкторі класу. Використання статичних даних в програмі. Використання компонентів-перемикачів. Відлагодження та тестування програми

Мета: закріпити навички зі створення та знищення об'єкту класу та ініціалізації полів, використання статичних даних та компонентів-перемикачів, відлагодження та тестування програми на прикладі обчислення площ та об’ємів геометричних тіл.

Хід роботи

6.1 Постановка задачі

Є дві групи геометричних тіл: багатогранники та круглі тіла. Розрахувати відповідно до варіанту для різних груп геометричних тіл їх характеристики: об'єм, загальну площу поверхні (в загальному випадку це сума площ бічної поверхні і основ). Розрахунок вести за сторонами (при необхідності кутами, радіусами тощо), які вводяться користувачем. У кожну з груп відповідно до варіанту входять конкретні геометричні тіла.

№ варіанту	Геометричні тіла, що входять у варіант	
	Багатогранники	Круглі тіла
3	Усічена пряма чотирикутна піраміда, куб	Усічений конус, циліндр

6.2 Елементи розробки інтерфейсу наведено у таблиці 6.1

Таблиця 6.1 – Перелік та призначення елементів інтерфейсу

Елемент	Призначення
MemoResultPolyhedra	Для виводу результатів обчислень багатогранників

Продовження таблиці 6.1

Елемент	Призначення
MemoResultRound	Для виводу результатів обчислень круглих тіл
LabelParam1, ... LabelParam4	Для підказки введення
ButtonCalc	Для обробки та виводу об’єму та площі фігури, формули.
ButtonCompare	Для порівняння сум об’ємів та площ багатогранників з круглими тілами.
ButtonClear	Для очистки TMemo, TEdit.
EditParam1, ... EditParam4	Для вводу даних
ComboBoxGroup	Для вибору групи геометричних тіл
ListBoxShapes	Для вибору фігури

6.3 Елементи розробки функцій програми наведено у таблиці 6.2

Таблиця 6.2 – Перелік та призначення обробників подій

	Функція	Обробник подій, що її реалізує	Модуль, у якому розміщується
1	Обчислення та виведення характеристик обраної фігури.	TForm1::ButtonCalcClick	Unit1
2	Порівняння сум об’ємів та площ багатогранників з круглими тілами.	TForm1::ButtonCompareClick	Unit1
3	Очистка . TMemo, TEdit	TForm1::ButtonClearClick	Unit1

4	Обробник OnChange для EditParam1, EditParam2, EditParam3	TForm1::EditParam1Change	Unit1
5	Функція у якій прописана поведінка TEdit та TLabel.	TForm1::UpdateParameterFields	Unit1
6	Функція у якій прописана поведінка ComboBoxGroup.	TForm1::ComboBoxGroupChange	Unit1

6.4 Вікно розробленої програми наведено на рисунку 6.1

Вікно TForm1—Головне та єдине вікно програми, у якому відбуваються всі події, такі як введення параметрів фігури та виведення результатів обчислення за формулами, а також можливість обрати групу геометричних тіл та фігуру.

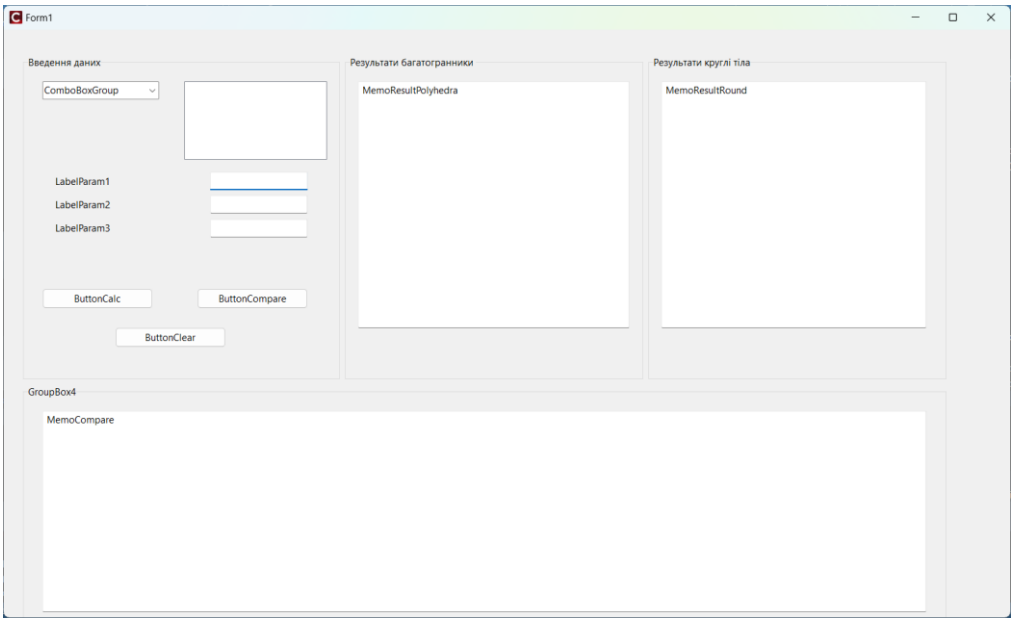


Рисунок 6.1 — Головне вікно програми

6.5 Текст розробленої програми наведено у лістингах 6.1—6.6

Лістинг 6.1—Специфікація класу Polyhedron

```
class Polyhedron {
public:
    static double totalSurfaceArea;
    static double totalVolume;
    virtual double surfaceArea() const = 0;
    virtual double volume() const = 0;
    virtual String formulaInfo() const = 0;
    virtual String inputData() const = 0;
    virtual ~Polyhedron() {}
};
```

Лістинг 6.2—Реалізація класу Polyhedron

```
#include "Polyhedron.h"

double Polyhedron::totalVolume = 0;
double Polyhedron::totalSurfaceArea = 0;
```

Лістинг 6.3—Специфікація класу Cube

```
#ifndef CUBE_H
#define CUBE_H

#include "Polyhedron.h"
#include <cmath>

class Cube : public Polyhedron {
private:
    double edge;  // ребро куба
public:
      Cube(double edge);
    double volume() const override;
    double surfaceArea() const override;
    String formulaInfo() const override;
    String inputData() const override;
};

#endif
```

Лістинг 6.4—Реалізація класу Cube

```
#include "Cube.h"
#include <cmath>

Cube::Cube(double edge) : edge(edge) {
    totalVolume += volume();
      totalSurfaceArea += surfaceArea();
}

double Cube::volume() const {
    return pow(edge, 3);
}

double Cube::surfaceArea() const {
    return 6 * pow(edge, 2);
}

String Cube::formulaInfo() const {
    return "Об'єм: a^3\r\nПлоща: 6 * a^2";
}

String Cube::inputData() const {
    return "a = " + FloatToStr(edge);
}
```

Лістинг 6.5—Специфікація класу Cylinder

```
#ifndef CYLINDER_H
#define CYLINDER_H

#include "RoundBody.h"
#include <cmath>

class Cylinder : public RoundBody {
private:
    double R, h;
public:
    Cylinder(double R, double h);
    double volume() const override;
    double surfaceArea() const override;
    String formulaInfo() const override;
    String inputData() const override;
};

#endif
```

Лістинг 6.6—Реалізація класу Cylinder

```
#include "Cylinder.h"
#include <cmath>

Cylinder::Cylinder(double R, double h) : R(R), h(h)
{
    totalVolume += volume();
    totalSurfaceArea += surfaceArea();
}
```

```
double Cylinder::volume() const {
    return M_PI * pow(R, 2) * h;
}

double Cylinder::surfaceArea() const {
    return 2 * M_PI * R * (R + h);
}

String Cylinder::formulaInfo() const {
    return "Об'єм: π * R² * h\r\n"
           "Площа: 2 * π * R * (R + h)";
}

String Cylinder::inputData() const {
    return "R = " + FloatToStr(R) + ", h = " + FloatToStr(h);
}
```

Лістинг 6.7 — Специфікація класу Figure

```
#ifndef FIGURE_H
#define FIGURE_H

#include <System.hpp> // для String (якщо потрібно)

class Figure {
public:

    virtual double volume() const = 0;
    virtual double surfaceArea() const = 0;
    virtual String formulaInfo() const = 0;
    virtual String inputData() const = 0;
    virtual ~Figure() {}
};

#endif
```

Лістинг 6.8 — Реалізація класу Polyhedron

```
#include "Polyhedron.h"

double Polyhedron::totalVolume = 0;
double Polyhedron::totalSurfaceArea = 0;
```

Лістинг 6.9 — Специфікація класу Polyhedron

```
#ifndef POLYHEDRON_H
#define POLYHEDRON_H

#include "Figure.h"
```

```
class Polyhedron : public Figure {
    // Тут можна додати загальні методи/поля для багатогранників, якщо потрібно
public:
    static double totalVolume;
    static double totalSurfaceArea;

    virtual double volume() const = 0;
    virtual double surfaceArea() const = 0;
    virtual String formulaInfo() const = 0;
      virtual String inputData() const = 0;
    virtual ~Polyhedron() {}
};

#endif
```

Лістинг 6.10 — Специфікація класу RoundBody

```
#ifndef ROUNDBODY_H
#define ROUNDBODY_H

#include "Figure.h"

class RoundBody : public Figure {
    // Загальні методи/поля для круглих тіл
public:
      static double totalVolume;
      static double totalSurfaceArea;

      virtual double volume() const = 0;
      virtual double surfaceArea() const = 0;
      virtual String formulaInfo() const = 0;
      virtual String inputData() const = 0;

    virtual ~RoundBody() {}
};

#endif
```

Лістинг 6.11 — Реалізація класу RoundBody

```
#include "RoundBody.h"

double RoundBody::totalVolume = 0;
double RoundBody::totalSurfaceArea = 0;
```

Лістинг 6.12 — Реалізація класу TruncatedCone

```
#include "TruncatedCone.h"
#include <cmath>
```

```
TruncatedCone::TruncatedCone(double R, double r, double h)
    : R(R), r(r), h(h)
{
    totalVolume += volume();
    totalSurfaceArea += surfaceArea();
}

double TruncatedCone::volume() const {
    return M_PI * h / 3.0 * (pow(R, 2) + R * r + pow(r, 2));
}

double TruncatedCone::surfaceArea() const {
    double l = sqrt(pow(R - r, 2) + pow(h, 2));
    return M_PI * (R + r) * l + M_PI * pow(R, 2) + M_PI * pow(r, 2);
}

String TruncatedCone::formulaInfo() const {
    return "Об'єм: (π * h / 3) * (R² + Rr + r²)\r\n"
           "Площа: π(R + r)√((R - r)² + h²) + πR² + πr²";
}

String TruncatedCone::inputData() const {
    return "R = " + FloatToStr(R) + ", r = " + FloatToStr(r) + ", h = " +
FloatToStr(h);
}
```

Лістинг 6.13—Специфікація класу TruncatedCone

```
#ifndef TRUNCATEDCONE_H
#define TRUNCATEDCONE_H

#include "RoundBody.h"
#include <cmath>

class TruncatedCone : public RoundBody {
private:
    double R, r, h;
public:
    TruncatedCone(double R, double r, double h);
    double volume() const override;
    double surfaceArea() const override;
    String formulaInfo() const override;
    String inputData() const override;
};

#endif
```

Лістинг 6.14—Специфікація класу TruncatedPyramid

```
#ifndef TRUNCATED_PYRAMID_H
#define TRUNCATED_PYRAMID_H

#include "Polyhedron.h"

class TruncatedPyramid : public Polyhedron {
```

```
private:
    double a, b, h; // сторони основ (a, b) і висота (h)
public:
    TruncatedPyramid(double a, double b, double h);
    double volume() const override;
    double surfaceArea() const override;
    String formulaInfo() const override;
    String inputData() const override;
};

#endif
```

Лістинг 6.15 — Специфікація класу TruncatedPyramid

```
#include "TruncatedPyramid.h"
#include <cmath>

TruncatedPyramid::TruncatedPyramid(double a, double b, double h)
    : a(a), b(b), h(h)
{
    totalVolume += volume();
    totalSurfaceArea += surfaceArea();
}

double TruncatedPyramid::volume() const {
    return h / 3 * (pow(a, 2) + a * b + pow(b, 2));
}

double TruncatedPyramid::surfaceArea() const {
    return 2 * (a + b) * sqrt(pow((a - b) / 2, 2) + pow(h, 2)) + pow(a, 2) +
pow(b, 2);
}

String TruncatedPyramid::formulaInfo() const {
    return "Об'єм: h/3 * (a^2 + ab + b^2)\r\n"
           "Площа: 2*(a+b)*sqrt(((a-b)/2)^2 + h^2) + a^2 + b^2";
}

String TruncatedPyramid::inputData() const {
    return "a = " + FloatToStr(a) + ", b = " + FloatToStr(b) + ", h = " +
FloatToStr(h);
}
```

Лістинг 6.16 — Специфікація TForm1

```
//---------------------------------------------------------------------------

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <vector>
```

```
//---------------------------------------------------------------------------

class TForm1 : public TForm
{
__published:      // IDE-managed Components
      TGroupBox *GroupBox1;
      TEdit *EditParam1;
      TEdit *EditParam2;
      TEdit *EditParam3;
      TButton *ButtonCalc;
      TGroupBox *GroupBox2;
      TMemo *MemoResultPolyhedra;
      TButton *ButtonCompare;
      TButton *ButtonClear;
      TListBox *ListBoxShapes;
      TGroupBox *GroupBox3;
      TMemo *MemoResultRound;
      TComboBox *ComboBoxGroup;
      TLabel *LabelParam1;
      TLabel *LabelParam2;
      TLabel *LabelParam3;
      TGroupBox *GroupBox4;
      TMemo *MemoCompare;
      void __fastcall ComboBoxGroupChange(TObject *Sender);
      void __fastcall ListBoxShapesClick(TObject *Sender);
      void __fastcall EditParam1Change(TObject *Sender);
      void __fastcall EditParam2Change(TObject *Sender);
      void __fastcall EditParam3Change(TObject *Sender);
      void __fastcall ButtonCalcClick(TObject *Sender);
      void __fastcall ButtonCompareClick(TObject *Sender);
      void __fastcall ButtonClearClick(TObject *Sender);
private:    // User declarations
      bool __fastcall ValidateInputs(const std::vector<int>& neededEdits,
AnsiString& errorMsg);
      double totalVolumePolyhedrons = 0.0;
      double totalSurfacePolyhedrons = 0.0;

    double totalVolumeRoundBodies = 0.0;
    double totalSurfaceRoundBodies = 0.0;
public:           // User declarations
      __fastcall TForm1(TComponent* Owner);
      void UpdateParameterFields();
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Лістинг 6.17 — Головний файл

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Cube.h"
#include "TruncatedPyramid.h"
#include "TruncatedCone.h"
```

```
#include "Cylinder.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------

// Допоміжна функція перевірки, чи рядок є валідним числом (float або int)
bool IsValidNumber(const UnicodeString &text)
{
      if (text.IsEmpty())
            return true; // дозвіл порожнього поля (користувач ще вводить)

      try
      {
            double val = StrToFloat(text);
            (void)val; // просто щоб компілятор не "жалівся"
            return true;
      }
      catch(...)
      {
            return false;
      }
}

// Метод для перевірки потрібних параметрів
bool __fastcall TForm1::ValidateInputs(const std::vector<int>& neededEdits,
AnsiString& errorMsg)
{
    for (int idx : neededEdits)
    {
        TEdit* edit = nullptr;
        switch (idx)
        {
            case 1: edit = EditParam1; break;
            case 2: edit = EditParam2; break;
            case 3: edit = EditParam3; break;
        }
        if (!edit) continue;

        if (edit->Text.IsEmpty())
        {
            errorMsg = "Please fill all required parameters.";
            edit->SetFocus();
            return false;
        }

        try
        {
            double val = StrToFloat(edit->Text);
            if (val <= 0) // додатні значення для розмірів
            {
                errorMsg = "Parameters must be positive numbers.";
                edit->SetFocus();
                return false;
            }
        }
        catch (...)
```

```
        {
            errorMsg = "Please enter valid numeric values.";
            edit->SetFocus();
            return false;
        }
    }
    return true;
}

void __fastcall TForm1::ComboBoxGroupChange(TObject *Sender)
{
      ListBoxShapes->Items->Clear();

      if (ComboBoxGroup->ItemIndex == 0) // Багатогранники
      {
            ListBoxShapes->Items->Add("Усічена пряма чотирикутна піраміда");
            ListBoxShapes->Items->Add("Куб");
      }
      else if (ComboBoxGroup->ItemIndex == 1) // Круглі тіла
      {
            ListBoxShapes->Items->Add("Усічений конус");
            ListBoxShapes->Items->Add("Циліндр");
      }

      ListBoxShapes->ItemIndex = 0; // вибрати перший елемент
    UpdateParameterFields();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ListBoxShapesClick(TObject *Sender)
{
      UpdateParameterFields();
}
//---------------------------------------------------------------------------

void TForm1::UpdateParameterFields()
{
    // Ховаємо всі спочатку
    EditParam1->Visible = false;
    EditParam2->Visible = false;
    EditParam3->Visible = false;

    LabelParam1->Visible = false;
    LabelParam2->Visible = false;
    LabelParam3->Visible = false;

    if (ComboBoxGroup->ItemIndex == 0) // Багатогранники
    {
        if (ListBoxShapes->ItemIndex == 0) // Усічена пряма чотирикутна піраміда
        {
            LabelParam1->Caption = "a (нижня основа):";
            LabelParam2->Caption = "b (верхня основа):";
            LabelParam3->Caption = "h (висота):";

            EditParam1->Visible = true;
            EditParam2->Visible = true;
            EditParam3->Visible = true;

            LabelParam1->Visible = true;
            LabelParam2->Visible = true;
            LabelParam3->Visible = true;
        }
        else if (ListBoxShapes->ItemIndex == 1) // Куб
        {
            LabelParam1->Caption = "a (довжина ребра):";
```

```
            EditParam1->Visible = true;
            LabelParam1->Visible = true;
        }
    }
    else if (ComboBoxGroup->ItemIndex == 1) // Круглі тіла
    {
        if (ListBoxShapes->ItemIndex == 0) // Усічений конус
        {
            LabelParam1->Caption = "R (радіус більшої основи):";
            LabelParam2->Caption = "r (радіус меншої основи):";
            LabelParam3->Caption = "h (висота):";

            EditParam1->Visible = true;
            EditParam2->Visible = true;
            EditParam3->Visible = true;

            LabelParam1->Visible = true;
            LabelParam2->Visible = true;
            LabelParam3->Visible = true;
        }
        else if (ListBoxShapes->ItemIndex == 1) // Циліндр
        {
            LabelParam1->Caption = "R (радіус основи):";
            LabelParam2->Caption = "h (висота):";

            EditParam1->Visible = true;
            EditParam2->Visible = true;

            LabelParam1->Visible = true;
            LabelParam2->Visible = true;
        }
    }
}
// Обробник OnChange для EditParam1, EditParam2, EditParam3
void __fastcall TForm1::EditParam1Change(TObject *Sender)
{
      TEdit* edit = dynamic_cast<TEdit*>(Sender);
      if (!edit) return;

      if (IsValidNumber(edit->Text))
            edit->Color = clWindow; // білий фон, все добре
      else
            edit->Color = (TColor)0x00FFC0CB; // рожевий колір (light pink)
}
//---------------------------------------------------------------------------
void __fastcall TForm1::EditParam2Change(TObject *Sender)
{
      TEdit* edit = dynamic_cast<TEdit*>(Sender);
      if (!edit) return;

      if (IsValidNumber(edit->Text))
            edit->Color = clWindow; // білий фон, все добре
      else
            edit->Color = (TColor)0x00FFC0CB; // рожевий колір (light pink)
}
//---------------------------------------------------------------------------
void __fastcall TForm1::EditParam3Change(TObject *Sender)
{
      TEdit* edit = dynamic_cast<TEdit*>(Sender);
      if (!edit) return;

      if (IsValidNumber(edit->Text))
            edit->Color = clWindow; // білий фон, все добре
```

```
      else
            edit->Color = (TColor)0x00FFC0CB; // рожевий колір (light pink)
}
//---------------------------------------------------------------------------



void __fastcall TForm1::ButtonCalcClick(TObject *Sender)
{
      AnsiString errorMsg;
    std::vector<int> neededParams;

    AnsiString selectedShape = ListBoxShapes->Items->Strings[ListBoxShapes-
>ItemIndex];
      String selectedGroup = ComboBoxGroup->Text;

    if (selectedShape == "Куб")
    {
        neededParams = {1}; // лише EditParam1 (a)
    }
      else if (selectedShape == "Усічена пряма чотирикутна піраміда")
    {
        neededParams = {1, 2, 3}; // EditParam1,2,3 (a,b,c)
    }
    else if (selectedShape == "Циліндр")
    {
        neededParams = {1, 2}; // R, h
    }
    else if (selectedShape == "Усічений конус")
    {
        neededParams = {1, 2, 3}; // R, r, h
    }
    else
    {
        ShowMessage("Please select a figure.");
        return;
    }

    if (!ValidateInputs(neededParams, errorMsg))
    {
        ShowMessage(errorMsg);
        return;
    }

    // Якщо всі параметри валідні — парсимо їх і рахуємо
    double p1 = neededParams.size() >= 1 ? StrToFloat(EditParam1->Text) : 0;
    double p2 = neededParams.size() >= 2 ? StrToFloat(EditParam2->Text) : 0;
    double p3 = neededParams.size() >= 3 ? StrToFloat(EditParam3->Text) : 0;
    double volume = 0.0;
      double surface = 0.0;
      // код розрахунку залежно від фігури
    if (selectedShape == "Куб") {
            Polyhedron* cube = new Cube(p1); // p1 — довжина ребра
            MemoResultPolyhedra->Lines->Add("Куб:");
        MemoResultPolyhedra->Lines->Add("Вхідні дані: " + cube->inputData());
            MemoResultPolyhedra->Lines->Add("Об'єм = " + FloatToStr(cube-
>volume()));

            MemoResultPolyhedra->Lines->Add("Площа = " + FloatToStr(cube-
>surfaceArea()));
        MemoResultPolyhedra->Lines->Add("Формули: ");
            MemoResultPolyhedra->Lines->Add(cube->formulaInfo());

            delete cube;
```

```
      }


      if (selectedShape == "Усічена пряма чотирикутна піраміда") {
            Polyhedron* pyramid = new TruncatedPyramid(p1, p2, p3);
            MemoResultPolyhedra->Lines->Add("Усічена пряма чотирикутна
піраміда:");
            MemoResultPolyhedra->Lines->Add("Вхідні дані: " + pyramid-
>inputData());
            MemoResultPolyhedra->Lines->Add("Об'єм = " + FloatToStr(pyramid-
>volume()));
            MemoResultPolyhedra->Lines->Add("Площа = " + FloatToStr(pyramid-
>surfaceArea()));
        MemoResultPolyhedra->Lines->Add("Формули: \r\n" + pyramid-
>formulaInfo());

            delete pyramid;
      }


    if (selectedShape == "Усічений конус") {
            Figure* cone = new TruncatedCone(p1, p2, p3); // p1 = R, p2 = r, p3
= h
            MemoResultRound->Lines->Add("Усічений конус:");
            MemoResultRound->Lines->Add("Вхідні дані: " + cone->inputData());
            MemoResultRound->Lines->Add("Об'єм = " + FloatToStr(cone-
>volume()));
            MemoResultRound->Lines->Add("Площа = " + FloatToStr(cone-
>surfaceArea()));
            MemoResultRound->Lines->Add("Формули: \r\n" + cone->formulaInfo());
            delete cone;
      }


    else if (selectedShape == "Циліндр") {
            Figure* shape = new Cylinder(p1, p2);
            MemoResultRound->Lines->Add("Циліндр:");
            MemoResultRound->Lines->Add("Вхідні дані: " + shape->inputData());
            MemoResultRound->Lines->Add("Об'єм = " + FloatToStr(shape-
>volume()));
            MemoResultRound->Lines->Add("Площа = " + FloatToStr(shape-
>surfaceArea()));
            MemoResultRound->Lines->Add("Формули:\r\n" + shape->formulaInfo());
            delete shape;
      }

      if (selectedGroup == "Багатогранники") {
            Polyhedron::totalVolume += volume;
            Polyhedron::totalSurfaceArea += surface;
      } else if (selectedGroup == "Круглі тіла") {
            RoundBody::totalVolume += volume;
            RoundBody::totalSurfaceArea += surface;
      }


}
//---------------------------------------------------------------------------



void __fastcall TForm1::ButtonCompareClick(TObject *Sender)
{
MemoCompare->Lines->Add("=== Порівняння сумарних характеристик ===");
```

```
      bool polyhedronsCalculated = (Polyhedron::totalVolume > 0 ||
Polyhedron::totalSurfaceArea > 0);
      bool roundBodiesCalculated = (RoundBody::totalVolume > 0 ||
RoundBody::totalSurfaceArea > 0);

    if (!polyhedronsCalculated && !roundBodiesCalculated)
    {
            MemoCompare->Lines->Add("Ще не обчислено характеристики жодної
групи.");
        return;
    }
    if (!polyhedronsCalculated)
    {
            MemoCompare->Lines->Add("Ще не обчислено характеристики
багатогранників.");
        return;
    }
    if (!roundBodiesCalculated)
    {
            MemoCompare->Lines->Add("Ще не обчислено характеристики круглих
тіл.");
        return;
    }

      MemoCompare->Lines->Add("Об'єм:");
      MemoCompare->Lines->Add(" - Багатогранники: " +
FloatToStr(Polyhedron::totalVolume));
      MemoCompare->Lines->Add(" - Круглі тіла: " +
FloatToStr(RoundBody::totalVolume));
    if (Polyhedron::totalVolume > RoundBody::totalVolume)
            MemoCompare->Lines->Add("   => Більший об'єм у багатогранників");
      else if (Polyhedron::totalVolume < RoundBody::totalVolume)
            MemoCompare->Lines->Add("   => Більший об'єм у круглих тіл");
    else
            MemoCompare->Lines->Add("   => Об'єми рівні");

      MemoCompare->Lines->Add("Площа поверхні:");
      MemoCompare->Lines->Add(" - Багатогранники: " +
FloatToStr(Polyhedron::totalSurfaceArea));
      MemoCompare->Lines->Add(" - Круглі тіла: " +
FloatToStr(RoundBody::totalSurfaceArea));
    if (Polyhedron::totalSurfaceArea > RoundBody::totalSurfaceArea)
            MemoCompare->Lines->Add("   => Більша площа у багатогранників");
    else if (Polyhedron::totalSurfaceArea < RoundBody::totalSurfaceArea)
            MemoCompare->Lines->Add("   => Більша площа у круглих тіл");
    else
            MemoCompare->Lines->Add("   => Площі рівні");

    MemoCompare->Lines->Add("=======================================");
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ButtonClearClick(TObject *Sender)
{
      EditParam1->Text = "";
    EditParam2->Text = "";
      EditParam3->Text = "";


      Polyhedron::totalVolume = 0.0;
    Polyhedron::totalSurfaceArea = 0.0;
    RoundBody::totalVolume = 0.0;
      RoundBody::totalSurfaceArea = 0.0;
```

```
MemoResultPolyhedra->Clear();
MemoResultRound->Clear();
MemoCompare->Clear();

}
```

6.6 Результат виконання програми наведено на рисунках 6.2—6.5

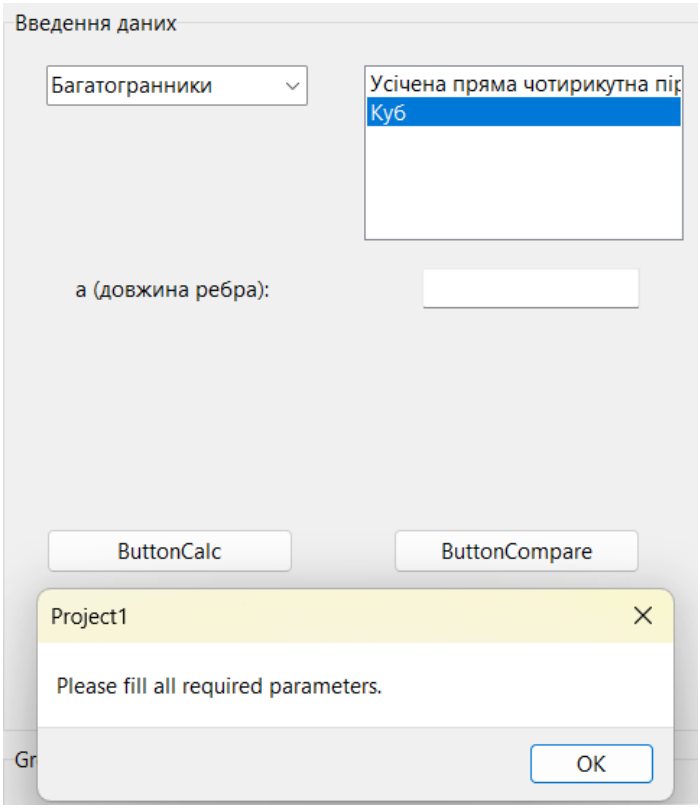


Рисунок 6.2 — Перевірка на порожні поля введення

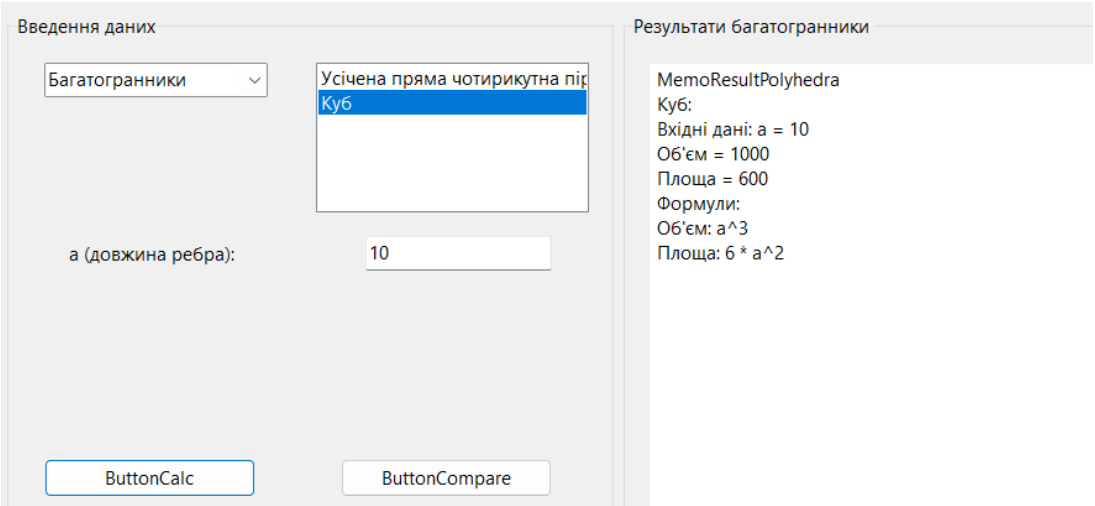


Рисунок 6.3 — Обчислення характеристик кубу

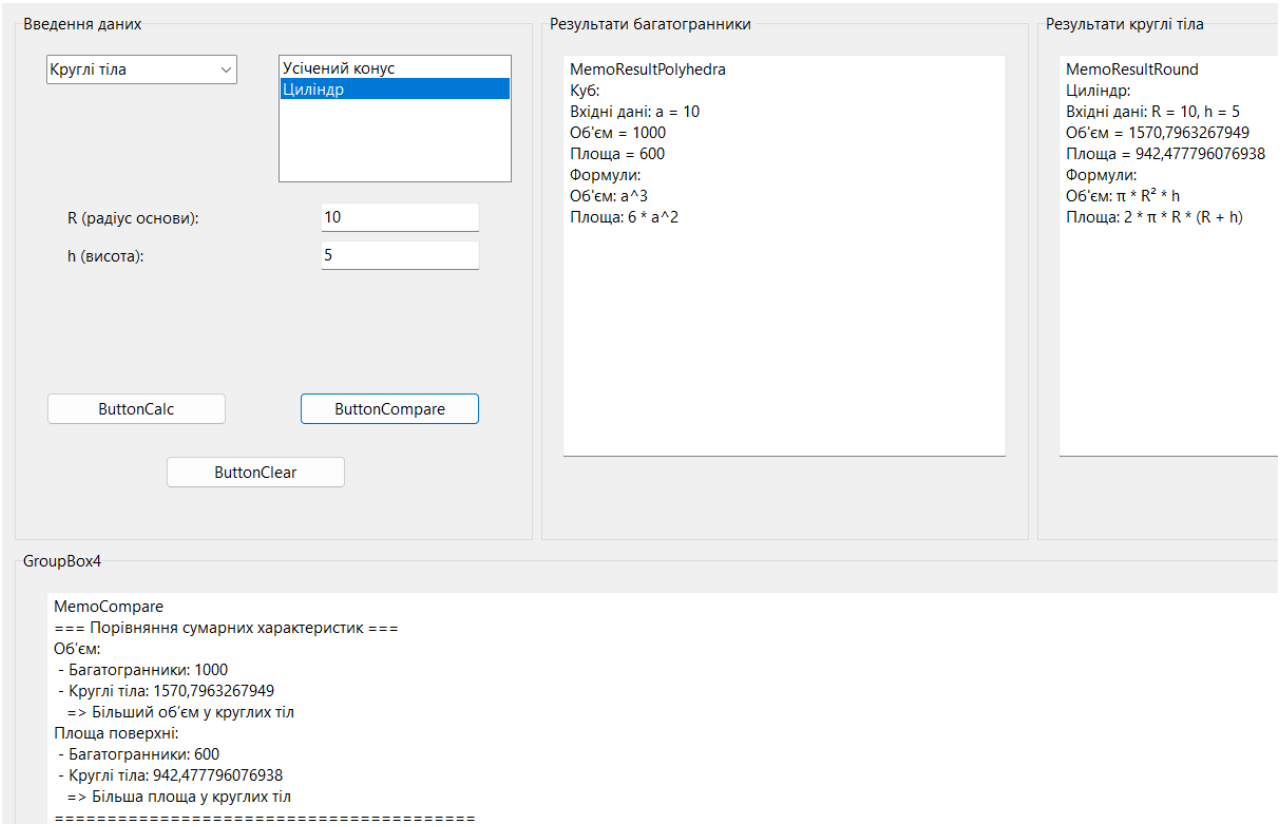


Рисунок 6.4 — Порівняння сум характеристик кожного виду фігур

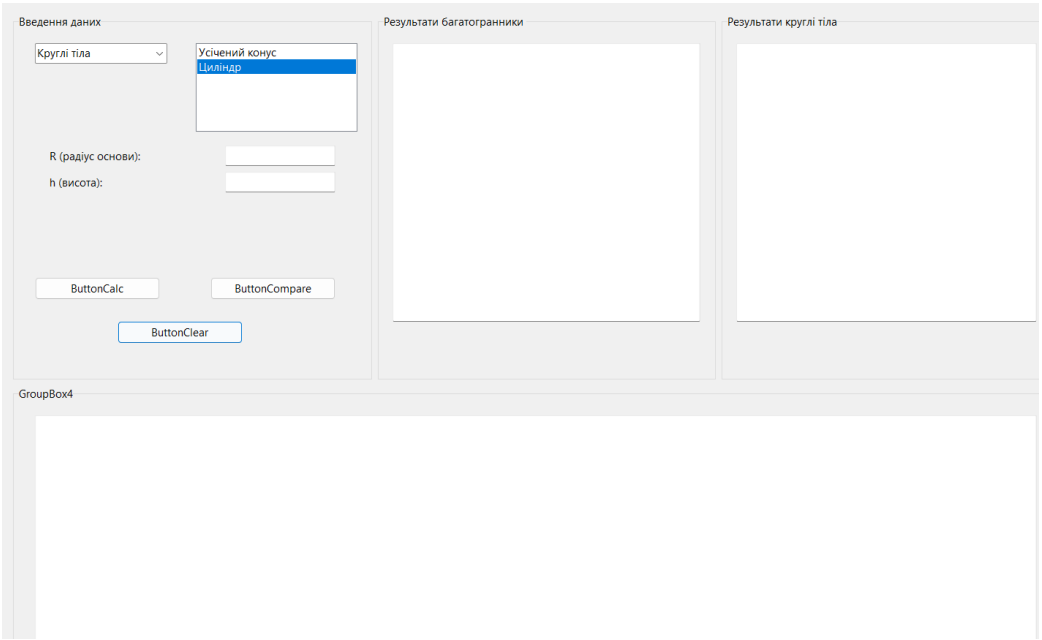


Рисунок 6.5 — Використання ButtonClear

Висновки: закріплено навички зі створення та знищення об'єкту класу та ініціалізації полів, використання статичних даних та компонентів-перемикачів, відлагодження та тестування програми на прикладі обчислення площ та об’ємів геометричних тіл.

ЗАВДАННЯ № 7

Проєктування класу для роботи з динамічною матрицею. Використання полів класу для отримання введених за допомогою інтерфейсу даних. Виведення результатів обчислення на екран

Мета: закріпити навички з проєктування класу для роботи з динамічною матрицею, використання полів класу для отримання введених за допомогою інтерфейсу даних та виведення результатів обчислення на екран.

Хід роботи

7.1 Постановка задачі

Задано динамічну матрицю (двомірний масив). Створити програму, яка дає можливість вводити елементи матриці, а також розраховувати необхідні характеристики.

Варіант 3

Дана цілочисельна квадратна матриця. Визначити:

1) Різниця сум елементів заданого рядка й стовпця.

2) Добуток елементів у тих рядках, які не містять негативних елементів;

3) Максимум серед сум елементів діагоналей, паралельних головній діагоналі матриці.

7.2 Елементи розробки інтерфейсу наведено у таблиці 7.1

Таблиця 7.1 – Перелік та призначення елементів інтерфейсу

Елемент	Призначення
Memo1	Для виводу матриці та результатів її обробки
Label1, ...Label4	Для підказки введення
Edit1	Для вводу розміру матриці
Edit2	Для вводу елементів матриці
Edit3	Для вводу рядку
Edit4	Для вводу стовпця
Button1	Для додавання елемента матриці

Button2	Для створення матриці
Button3	Для виводу матриці та результатів її обробки

7.3 Елементи розробки функцій програми наведено у таблиці 7.2

Таблиця 7.2 – Перелік та призначення обробників подій

	Функція	Обробник подій, що її реалізує	Модуль, у якому розміщується
1	Додавання введенного елемента до матриці та її виведення.	TForm1::Button1Click	Unit1
2	Створення квадратної матриці із зазначеним розміром.	TForm1::Button2Click	Unit1
3	Виведення результатів обробки матриці.	TForm1::Button3Click	Unit1

7.4 Вікно розробленої програми наведено на рисунку 7.1

Вікно TForm1—Головне та єдине вікно програми, у якому відбуваються всі події, такі як заповнення матриці та її виведення, а також виведення результатів її обробки.

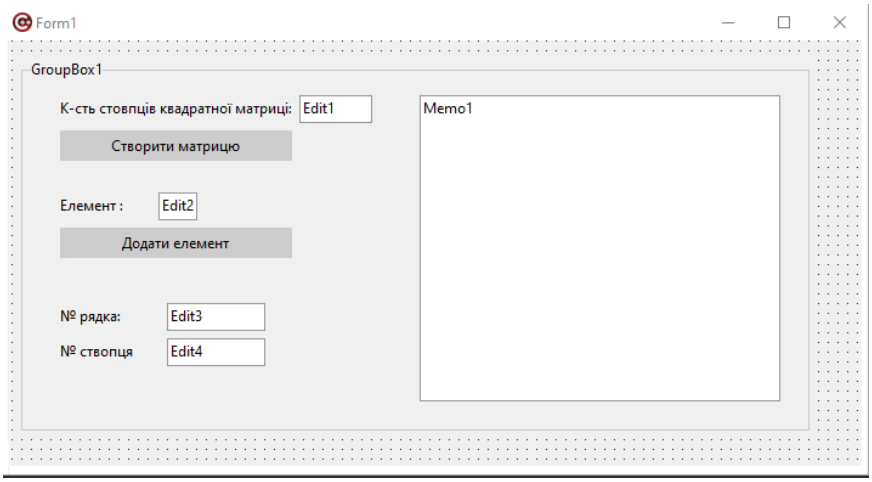


Рисунок 7.1 — Головне вікно програми

7.5 Текст розробленої програми наведено у лістингах 7.1—7.4

Лістинг 7.1—Специфікація класу Matrix

```
#ifndef MatrixH
#define MatrixH

class Matrix {
private:
	int size;
	int** data;

public:
	Matrix(int n);
	~Matrix();

	void SetElement(int i, int j, int value);
	int GetElement(int i, int j);
	int GetSize();

	int RowColSumDifference(int row, int col);
	int ProductOfRowsWithoutNegatives();
	int MaxSumOnDiagonalParallels();
};

#endif
```

Лістинг 7.2—Реалізація класу Matrix

```
#include "Matrix.h"
#include <algorithm>

Matrix::Matrix(int n) : size(n) {
	data = new int*[size];
	for (int i = 0; i < size; ++i)
		data[i] = new int[size]();
}

Matrix::~Matrix() {
	for (int i = 0; i < size; ++i)
		delete[] data[i];
	delete[] data;
}

void Matrix::SetElement(int i, int j, int value) {
	if (i >= 0 && i < size && j >= 0 && j < size)
		data[i][j] = value;
}

int Matrix::GetElement(int i, int j) {
	if (i >= 0 && i < size && j >= 0 && j < size)
		return data[i][j];
	return 0;
}

int Matrix::GetSize() {
```

```
	return size;
}

int Matrix::RowColSumDifference(int row, int col) {
	int rowSum = 0, colSum = 0;
	for (int i = 0; i < size; ++i) {
		rowSum += data[row][i];
		colSum += data[i][col];
	}
	return rowSum - colSum;
}

int Matrix::ProductOfRowsWithoutNegatives() {
	int product = 1;
	bool found = false;
	for (int i = 0; i < size; ++i) {
		bool hasNegative = false;
		int rowProduct = 1;
		for (int j = 0; j < size; ++j) {
			if (data[i][j] < 0) {
				hasNegative = true;
				break;
			}
			rowProduct *= data[i][j];
		}
		if (!hasNegative) {
			product *= rowProduct;
			found = true;
		}
	}
	return found ? product : 0;
}

int Matrix::MaxSumOnDiagonalParallels() {
	int maxSum = INT_MIN;
	for (int d = 0; d < size; ++d) {
		int sum = 0;
		for (int i = 0; i < size - d; ++i)
			sum += data[i][i + d];
		maxSum = std::max(maxSum, sum);
	}
	for (int d = 1; d < size; ++d) {
		int sum = 0;
		for (int i = 0; i < size - d; ++i)
			sum += data[i + d][i];
		maxSum = std::max(maxSum, sum);
	}
	return maxSum;
}
```

Лістинг 7.3—Головний файл

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
```

```
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
    matrix = nullptr;
      inputIndex = 0;

}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
if (Edit1->Text.IsEmpty()) {
            ShowMessage("Введіть розмір матриці.");
            return;
      }
      Label2->Enabled=true;
      Edit2->Enabled=true;
      Button2->Enabled=true;
      Label2->Caption="Елемент [1][1]:";
      int n = StrToInt(Edit1->Text);
      if (n <= 0) {
            ShowMessage("Розмір має бути > 0.");
            return;
      }

      if (matrix) delete matrix;
      matrix = new Matrix(n);
      inputIndex = 0;

      Memo1->Clear();
      Memo1->Lines->Add("Матриця " + IntToStr(n) + "x" + IntToStr(n) + "
створена.");
      Memo1->Lines->Add("Введіть елементи по черзі.");



}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
      if (Edit3->Text.IsEmpty() || Edit4->Text.IsEmpty()) {
            ShowMessage("Введіть рядок і стовпець матриці.");
            return;
      }
      int row = StrToInt(Edit3->Text);
      int col = StrToInt(Edit4->Text);
      int n = matrix->GetSize();

      if (row <= 0 || row >= n+1 || col <= 0 || col >= n+1) {
            ShowMessage("Недопустимі індекси.");
            return;
      }

      int diff = matrix->RowColSumDifference(row-1, col-1);
      int prod = matrix->ProductOfRowsWithoutNegatives();
      int maxSum = matrix->MaxSumOnDiagonalParallels();

      Memo1->Lines->Add("");
      Memo1->Lines->Add("Результати:");
      Memo1->Lines->Add("1) Різниця суми рядка " + IntToStr(row) + " і суми
стовпця " + IntToStr(col) + ": " + IntToStr(diff));
      Memo1->Lines->Add("2) Добуток рядків без від'ємних елементів: " +
IntToStr(prod));
```

```
      Memo1->Lines->Add("3) Максимальна сума діагоналей, паралельних головній: "
+ IntToStr(maxSum));

}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    if (Edit2->Text.IsEmpty()) {
            ShowMessage("Введіть елемент матриці.");
            return;
      }
      int val = StrToInt(Edit2->Text);
      int n = matrix->GetSize();

      if (inputIndex >= n * n) {
            ShowMessage("Усі елементи вже введено.");
            return;
      }

      int i = inputIndex / n;
      int j = inputIndex % n;
      matrix->SetElement(i, j, val);
      inputIndex++;
      Edit2->Clear();

      Memo1->Clear();
      Memo1->Lines->Add("Матриця:");
      for (int r = 0; r < n; r++) {
            String line = "";
            for (int c = 0; c < n; c++) {
                  line += IntToStr(matrix->GetElement(r, c)) + "\t";
            }
            Memo1->Lines->Add(line);
      }
      Memo1->Lines->Add("Введено: " + IntToStr(inputIndex) + " з " + IntToStr(n
* n));

      if (inputIndex == n * n) {
            Button3->Enabled = true;
            Label3->Enabled = true;
            Label4->Enabled = true;
            Edit3->Enabled = true;
        Edit4->Enabled = true;
      }
      else {
            int next_i = inputIndex / n;
            int next_j = inputIndex % n;
            Label2->Caption = "Елемент [" + IntToStr(next_i+1) + "][" +
IntToStr(next_j+1) + "]";
      }


}
//---------------------------------------------------------------------------
```

Лістинг 7.4—Специфікація TForm

```
//---------------------------------------------------------------------------

#ifndef Unit1H
#define Unit1H
```

```
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include "Matrix.h"
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
      TGroupBox *GroupBox1;
      TLabel *Label1;
      TEdit *Edit1;
      TButton *Button1;
      TEdit *Edit2;
      TLabel *Label2;
      TButton *Button2;
      TEdit *Edit3;
      TLabel *Label3;
      TLabel *Label4;
      TEdit *Edit4;
      TMemo *Memo1;
      TButton *Button3;
      void __fastcall Button1Click(TObject *Sender);
      void __fastcall Button3Click(TObject *Sender);
      void __fastcall Button2Click(TObject *Sender);
private:    // User declarations
      Matrix* matrix;
      int inputIndex;
public:           // User declarations
      __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

7.6 Результат виконання програми наведено на рисунках 7.2—7.8

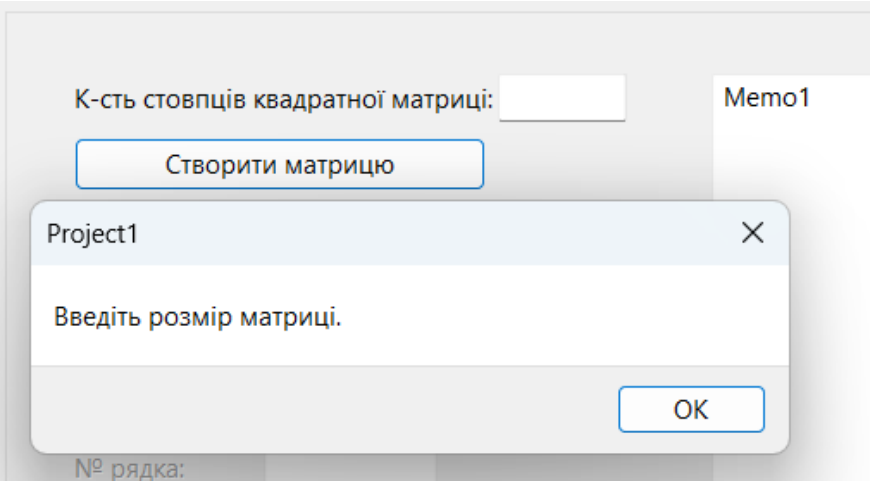


Рисунок 7.2 — Перевірка на порожні поля введення

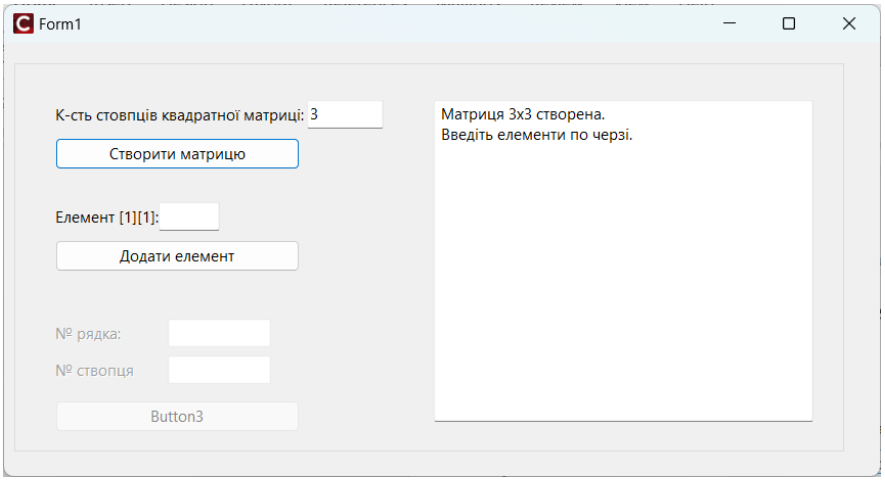


Рисунок 7.3 — Створення матриці

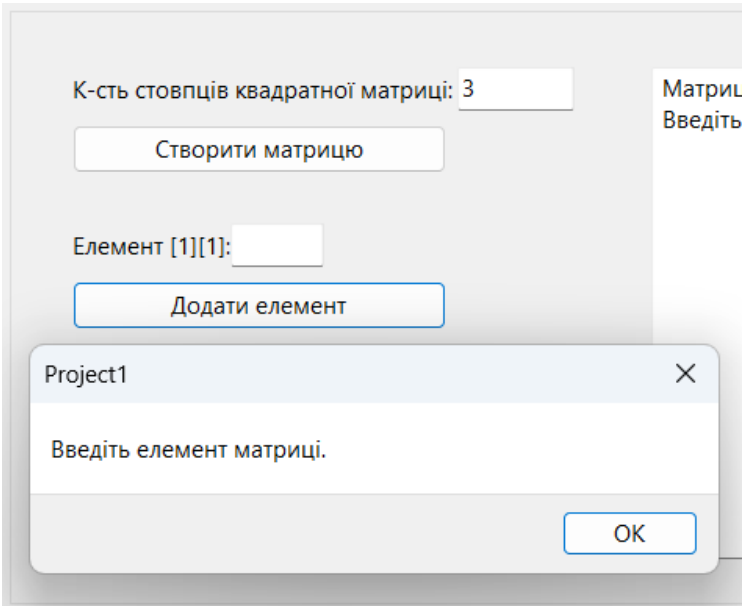


Рисунок 7.4 — Перевірка на порожні поля введення

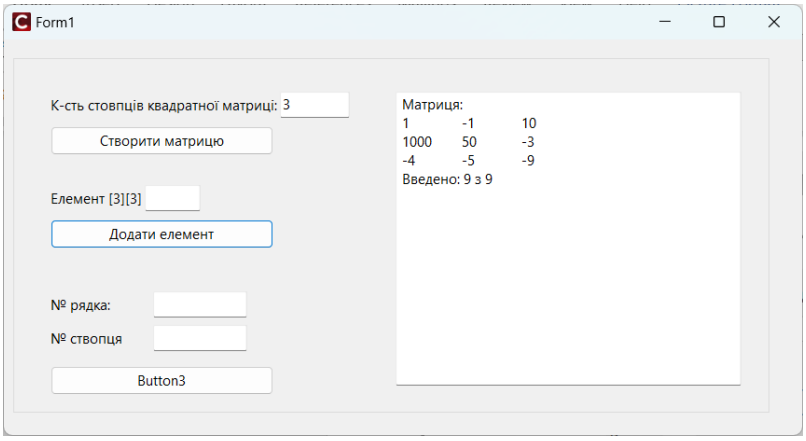


Рисунок 7.5 — Заповнення матриці

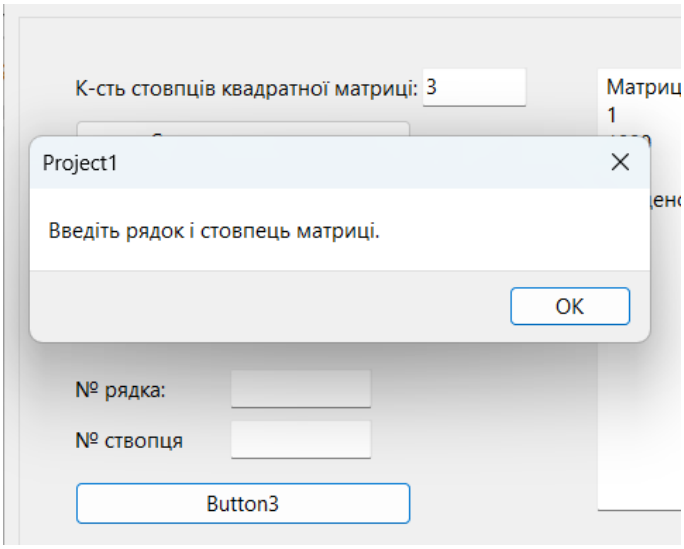


Рисунок 7.6 — Перевірка на порожні поля введення

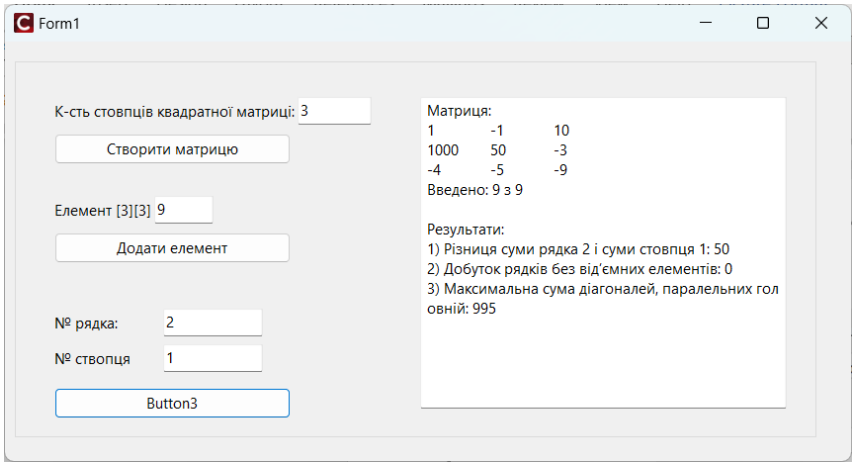


Рисунок 7.7 — Результати обробки матриці

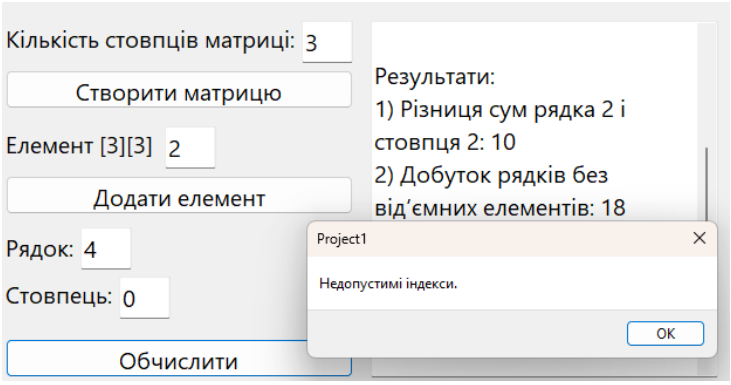


Рисунок 7.8 — Помилка при недопустимих індексах матриці

Висновки: закріплено навички з проєктування класу для роботи з динамічною матрицею, використання полів класу для отримання введених за допомогою інтерфейсу даних та виведення результатів обчислення на екран.

ЗАВДАННЯ № 8

Проєктування структури бази даних, ERD. Перевірка введених даних на допустимість. Реалізація додавання даних в таблиці БД. Реалізація пошуку за запитом. Використання компонентів-списків та обробка повідомлень

Мета: закріпити навички з проєктування структури бази даних, ERD, перевірки введених даних на допустимість, реалізації додавання даних в таблиці БД та пошуку за запитом, використання компонентів-списків та обробки повідомлень

8.1 Постановка завдання.

Спроектувати БД згідно завдання, рекомендовано з двох таблиць, пов'язаних між собою ключом. У звіті навести структуру таблиць БД. Створити проект, виконати підключення, забезпечити введення даних та розрахунок.

Потрібно розробити програму, яка виконує додавання інформації та пошук. Перелік даних та дій, які повинна виконувати програма, приведені у варіантах завдання.

1 Дані про студента містять:

– прізвище й ініціали;

– дата народження;

– рік вступу до коледжу;

– успішність (оцінки даного студента).

2 Написати програму, що виконує наступні дії:

– введення даних про студента;

– введення даних про успішність студента;

– виведення на екран інформації про студентів, які мають хоча б одну оцінку 2;

– виведення на екран списку студентів, які поступили до коледжу після вказаного року.

8.2 Елементи розробки інтерфейсу наведено у таблиці 8.1

Таблиця 8.1 – Перелік та призначення елементів інтерфейсу

Елемент	Призначення
Label1, ...Label5	Для підказки введення
EditName	Прізвище й ініціали студента
EditBirth	Дата народження студента
EditYear	Рік вступу до коледжу студента
EditGrade	Оцінка студента
EditSearchYear	Рік
ButtonAddStudent	Додати студента, (введення даних про студента)
ButtonAddGrade	Додати оцінку студента (введення даних про успішність студента)
ButtonFindBySearchYear	Виведення на екран списку студентів, які поступили до коледжу після вказаного року.
ButtonFindBadGrades	Виведення на екран інформації про студентів, які мають хоча б одну оцінку 2;
DBGrid1	Для виведення даних про студента

8.3 Елементи розробки функцій програми наведено у таблиці 8.2

Таблиця 8.2 – Перелік та призначення обробників подій

	Функція	Обробник подій, що її реалізує	Модуль, у якому розміщуєтьс
1	Виведення студентів з оцінкою 2.	TForm1::ButtonFindBadGradesClick	Unit1
2	Студенти вступивші після вказаного року	TForm1::ButtonFindBySearchYearClick	Unit1
3	Додавання студента до БД	TForm1::ButtonAddStudentClick	Unit1
4	Додавання оцінок до студента в БД	TForm1::ButtonAddGradeClick	Unit1

8.4 Вікно розробленої програми наведено на рисунку 8.1

Вікно TForm5—Головне та єдине вікно програми, у якому відбуваються всі події, такі як додавання записів до таблиці бази даних, її виведення, сортування та виконання запитів.

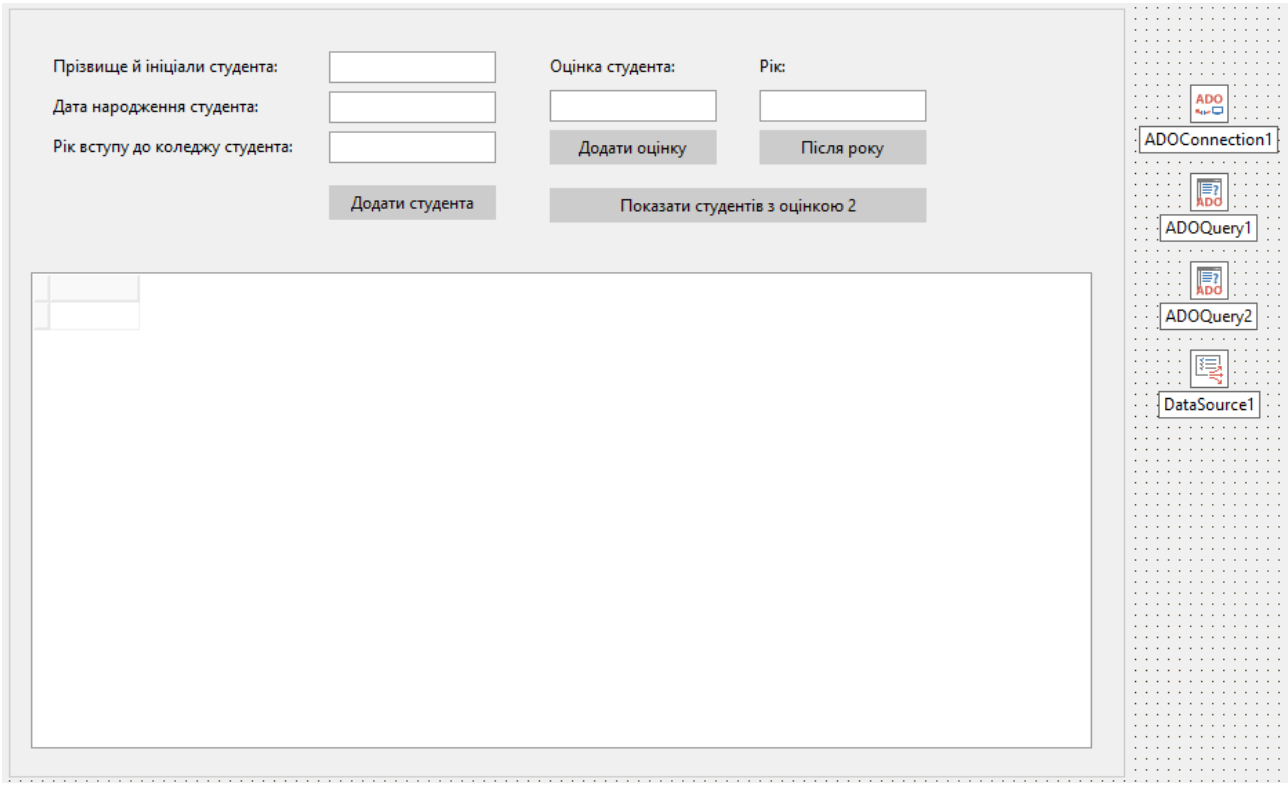


Рисунок 8.1 — Головне вікно програми

8.5 Текст розробленої програми наведено у лістингах 8.1—8.2

Лістинг 8.1—Головний файл

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------


void __fastcall TForm1::FormCreate(TObject *Sender)
{
      ADOQuery1->Open();
}
//---------------------------------------------------------------------------


void __fastcall TForm1::ButtonAddStudentClick(TObject *Sender)
{
String name = EditName->Text.Trim();
    String birth = EditBirth->Text.Trim();
    String year = EditYear->Text.Trim();

    if (name.IsEmpty() || birth.IsEmpty() || year.IsEmpty())
    {
        ShowMessage("Будь ласка, заповніть усі поля.");
        return;
```

```
    }


    try {
        ADOQuery1->Close();
        ADOQuery1->SQL->Text = "INSERT INTO Students (name, birth_date,
admission_year) "
                               "VALUES (:name, :birth, :year)";
        ADOQuery1->Parameters->ParamByName("name")->Value = name;
        ADOQuery1->Parameters->ParamByName("birth")->Value = birth;
        ADOQuery1->Parameters->ParamByName("year")->Value = year.ToInt();


        ADOQuery1->ExecSQL();


        ShowMessage("Студента додано!");


        // Оновлюємо таблицю
        ADOQuery1->SQL->Text = "SELECT * FROM Students";
        ADOQuery1->Open();
    }
    catch (Exception &e) {
            ShowMessage("Помилка: " + e.Message);
}
}
//---------------------------------------------------------------------------


void __fastcall TForm1::ButtonAddGradeClick(TObject *Sender)
{
if (ADOQuery1->IsEmpty())
    {
        ShowMessage("Немає вибраного студента.");
        return;
    }


    String gradeStr = EditGrade->Text.Trim();
    if (gradeStr.IsEmpty())
    {
        ShowMessage("Введіть оцінку.");
        return;
    }
```

```
    int grade = gradeStr.ToIntDef(-1);
    if (grade < 2 || grade > 5)
    {
        ShowMessage("Оцінка має бути від 2 до 5.");
        return;
    }

    int student_id = ADOQuery1->FieldByName("id")->AsInteger;

    try {
        TADOQuery *q = new TADOQuery(this);
        q->Connection = ADOConnection1;
        q->SQL->Text = "INSERT INTO Grades (student_id, grade) VALUES (:id,
:grade)";
        q->Parameters->ParamByName("id")->Value = student_id;
        q->Parameters->ParamByName("grade")->Value = grade;
        q->ExecSQL();
        delete q;

        ShowMessage("Оцінку додано.");
    }
    catch (Exception &e) {
        ShowMessage("Помилка: " + e.Message);
    }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ButtonFindBadGradesClick(TObject *Sender)
{
      ADOQuery1->Close();
    ADOQuery1->SQL->Text =
        "SELECT DISTINCT Students.id, name, birth_date, admission_year "
        "FROM Students "
        "JOIN Grades ON Students.id = Grades.student_id "
        "WHERE grade = 2";
    ADOQuery1->Open();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ButtonFindBySearchYearClick(TObject *Sender)
```

```
{
      String yearStr = EditSearchYear->Text.Trim();
    if (yearStr.IsEmpty())
    {
        ShowMessage("Введіть рік для пошуку.");
        return;
    }

    int year = yearStr.ToIntDef(-1);
    if (year < 1900 || year > 2100)
    {
        ShowMessage("Некоректний рік.");
        return;
    }

    ADOQuery1->Close();
    ADOQuery1->SQL->Text =
        "SELECT id, name, birth_date, admission_year "
        "FROM Students "
        "WHERE admission_year > :year";
    ADOQuery1->Parameters->ParamByName("year")->Value = year;
      ADOQuery1->Open();
}
//---------------------------------------------------------------------------
```

Лістинг 8.2—Специфікація TForm

```
//---------------------------------------------------------------------------

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Data.DB.hpp>
```

```
#include <Data.DBXMySQL.hpp>
#include <Data.FMTBcd.hpp>
#include <Data.SqlExpr.hpp>
#include <Vcl.DBGrids.hpp>
#include <Vcl.ExtCtrls.hpp>
#include <Vcl.Grids.hpp>
#include <Vcl.Mask.hpp>
#include <Vcl.WinXPickers.hpp>
#include <Data.Win.ADODB.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
      TDBGrid *DBGrid1;
      TButton *ButtonAddStudent;
      TADOConnection *ADOConnection1;
      TADOQuery *ADOQuery1;
      TDataSource *DataSource1;
      TEdit *EditName;
      TEdit *EditBirth;
      TEdit *EditYear;
      TEdit *EditGrade;
      TButton *ButtonAddGrade;
      TEdit *EditSearchYear;
      TADOQuery *ADOQuery2;
      TButton *ButtonFindBadGrades;
      TButton *ButtonFindBySearchYear;
      TLabel *Label1;
      TLabel *Label2;
      TLabel *Label3;
      TLabel *Label4;
      TLabel *Label5;
      TGroupBox *GroupBox1;
      void __fastcall FormCreate(TObject *Sender);
      void __fastcall ButtonAddStudentClick(TObject *Sender);
      void __fastcall ButtonAddGradeClick(TObject *Sender);
      void __fastcall ButtonFindBadGradesClick(TObject *Sender);
      void __fastcall ButtonFindBySearchYearClick(TObject *Sender);
private:    // User declarations
public:           // User declarations
```

```
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

8.6 Результат виконання програми наведено на рисунках 8.2—8.7

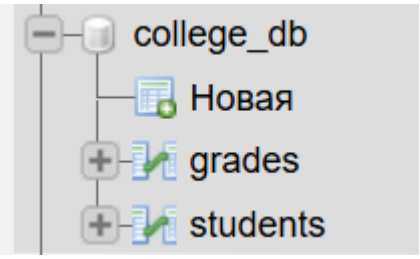


Рисунок 8.2 — Додавання нового запису

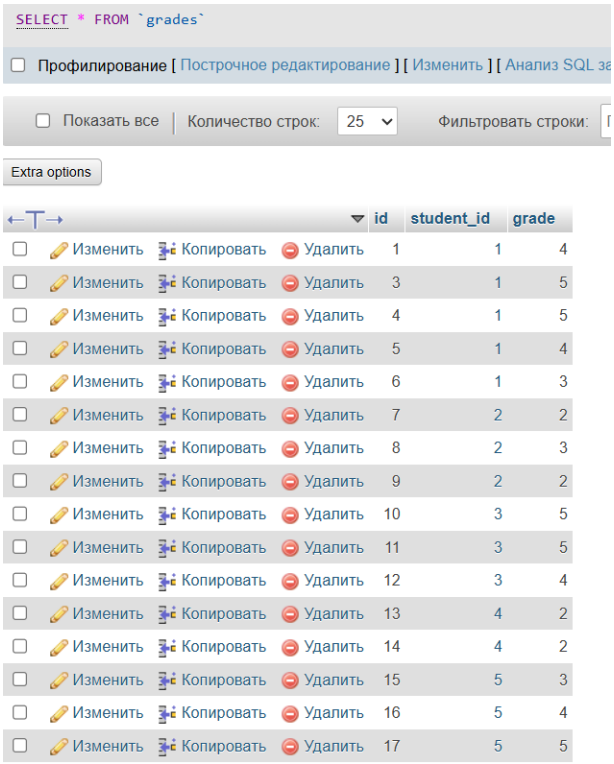


Рисунок 8.3 — Додавання нового запису

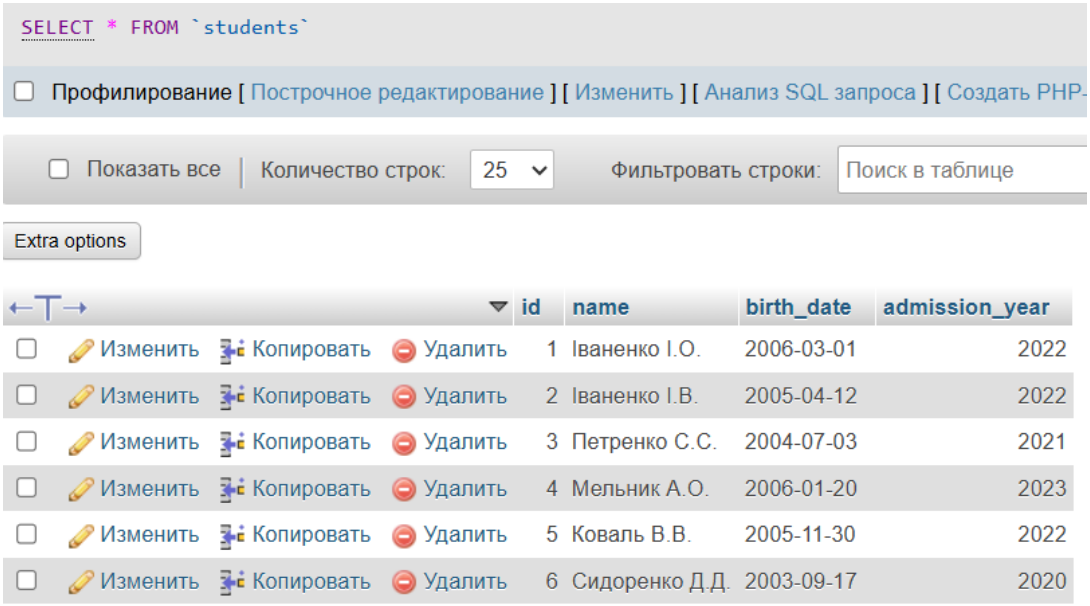


Рисунок 8.4 — Додавання нового запису

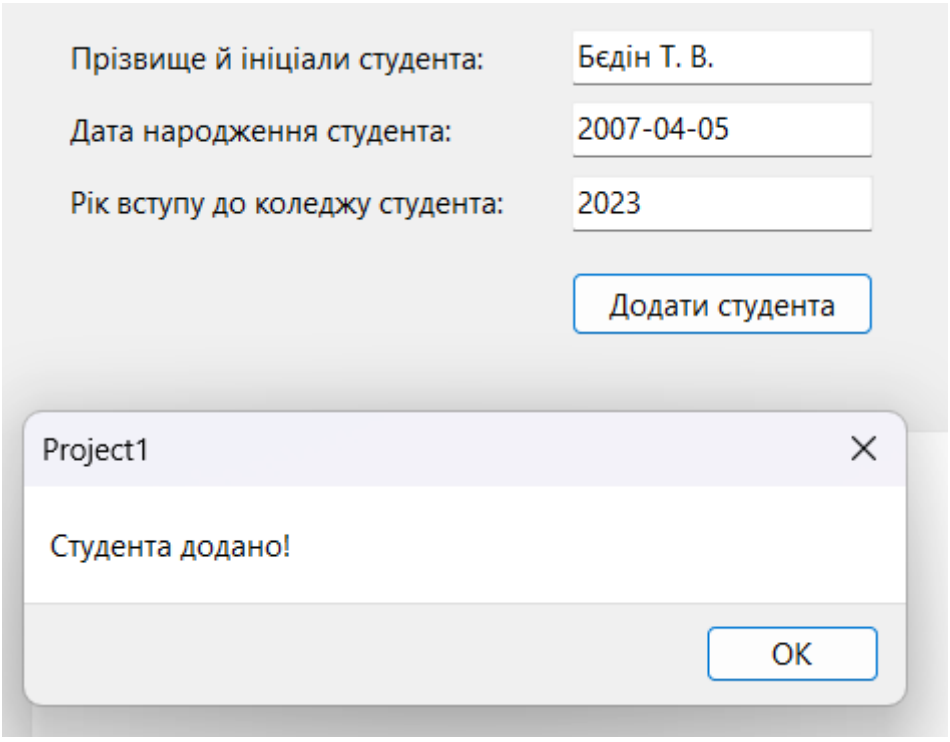


Рисунок 8.5 — Додавання нового запису

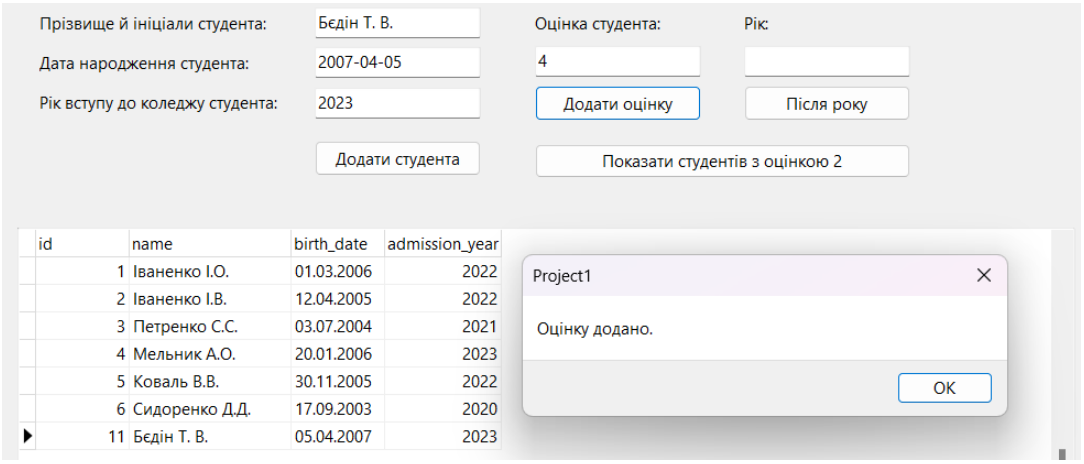


Рисунок 8.6 — Додавання оцінки до вибраного студента в таблиці.

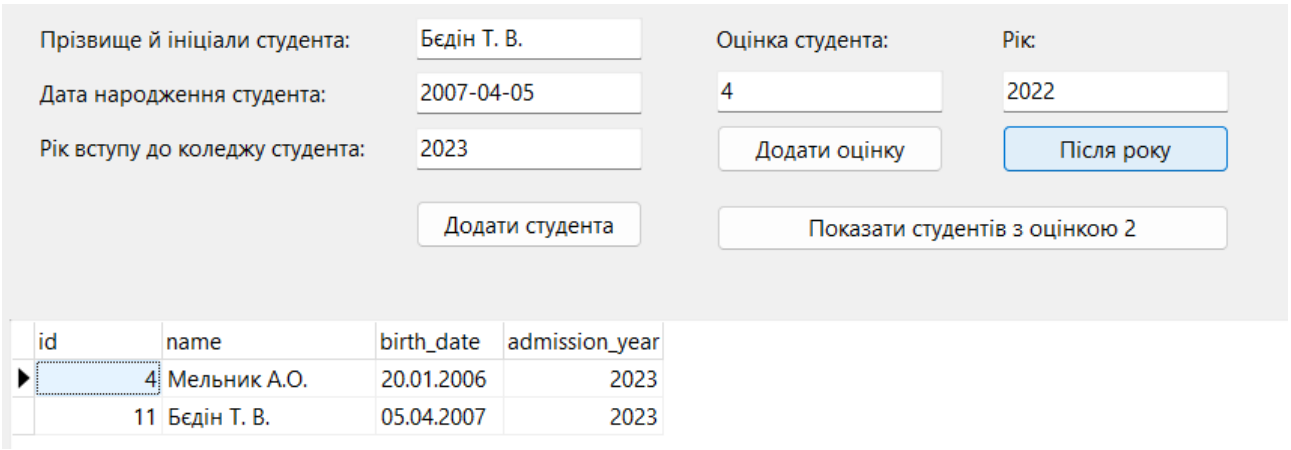


Рисунок 8.7 — Фільтр за роком

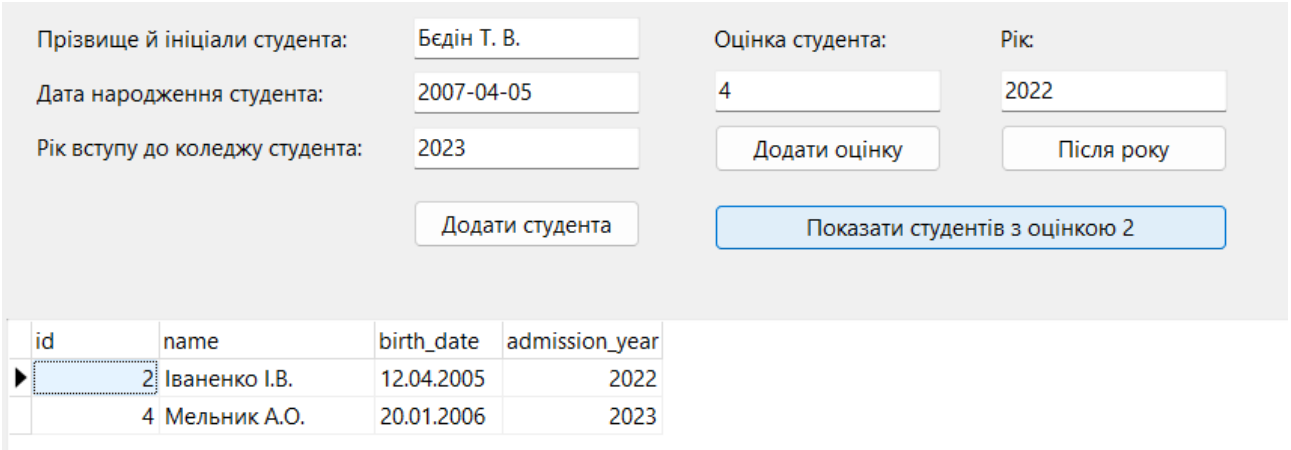


Рисунок 8.8 — Студенти з оцінкою 2

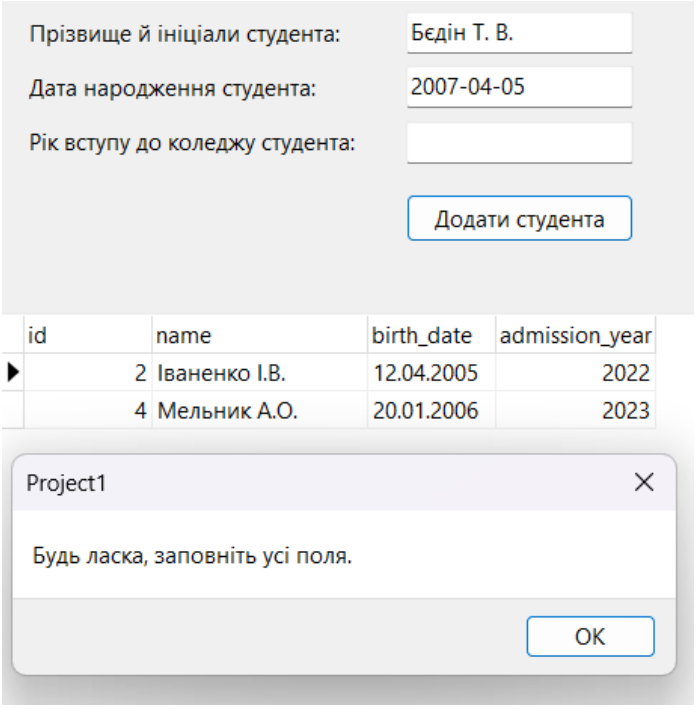


Рисунок 8.9 — Перевірка на порожні поля введення

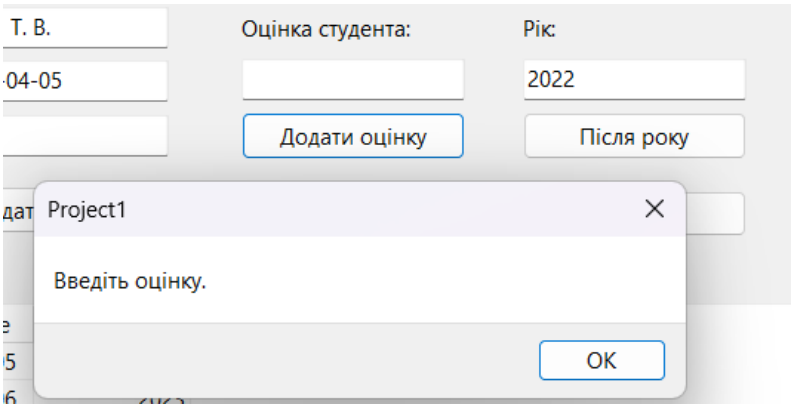


Рисунок 8.10 — Перевірка на порожні поля введення

Висновки

Закріплено навички з проєктування структури бази даних, ERD, реалізації додавання даних в таблиці БД та пошуку за запитом, використання компонентів-списків та обробки повідомлень.

	Вик.	Бєдін Т. В.			НП.ПЗ.221.03.ЗВ	Арк.
	Пер.	Гапоненко Н.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		