МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ
УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра програмної інженерії та інтелектуальних технологій управління

Звіт з лабораторної роботи № 1
з дисципліни «Основи програмування (частина 2)»

Виконав:
ст. гр. КН-221в Є.Р. Шулюпов

Перевірив:
доцент. каф. ПІТУ А.А. Пашнєв

Харків 2022

ТЕМА: СТВОРЕННЯ ТА ВИКОРИСТАННЯ КЛАСІВ

1. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

1.1 Клас для представлення простого дробу

Створити клас для представлення простого дробу. Реалізувати конструктори, функцію скорочення дробу, а також перевантажити операції +, -, *, /, введення та виведення. Здійснити демонстрацію можливостей класу в функції main().

1.2 Клас для представлення двовимірного масиву

Розробити клас для представлення двовимірного масиву (матриці) цілих чисел довільних розмірів. Створити конструктори та деструктор, перевантажити операції додавання, віднімання і множення (згідно з правилами роботи з матрицями), звертання за індексом, введення з потоку та виведення в потік. Створити власні класи винятків та генерувати відповідні об'єкти-винятки, якщо неможливо виконати ту чи іншу операцію.

Створити окрему функцію, яка отримує посилання на матрицю і виконує над масивом дії, вказані в таблиці. Функція не повинна бути методом класу або дружньою функцією.

27 варіант - Усі від'ємні елементи з непарними значеннями повинні бути збільшені в два рази.

У функції main() здійснити тестування всіх можливостей класу з перехопленням можливих винятків, а також розв'язати індивідуальну задачу.

1.3 Підрахунок суми введених значень

Створити клас з одним закритим елементом даних цілого типу, геттером і конструктором з одним параметром. В цьому ж класі створити закрите статичне поле, яке зберігає суму цілих елементів даних всіх раніше створених об'єктів. Під час кожного виклику конструктора до статичного поля повинно додаватися нове значення. Статична публічна функція цього ж класу повинна повертати цю суму.

У функції main() створити декілька об'єктів і вивести отриману суму.

2. ОСНОВНА ЧАСТИНА

2.1 Основні теоретичні положення

Для створення користувацьких типів застосовують класи, також можна використовувати інші мовні конструкції – переліки, структури й об'єднання.

Перелік – набір констант.

Структуру об’єднують значення різних типів.

Об'єднання – це структура, у якій всі елементи розташовані за однією адресою.

Клас (class) – це структурований тип даних, набір елементів даних різних типів і функцій для роботи з цими даними.

Для реалізації інкапсуляції (приховування данних) мова C++ надає рівні доступу до елементів класу: public (публічні), protected (захищені) та private (закриті).

У класів є спеціальні функції-елементи: конструктори та деструктори.

Конструктори призначені для ініціалізації даних об'єктів.

Деструктори призначені для знищення даних об’єктів.

Клас може бути описаний у глобальній області видимості, а також у класовій та локальній областях.

Функції або класи, оголошені як друзі класу, мають доступ до його закритих та захищених елементів.

2.2 Описання розробленого за стосунку

Завдання 1.1

```
#include <iostream>
#include <fstream>
using namespace std;

class fraction {

private:
     int n;
     int d;
public:

    fraction(int n = 1, int d = 1) {
        this->n = n;
        this->d = d;
    }

    void simpl() {
        if (n != 0) {
            if (d != 0) {
                int nEmpty = n;
                int dEmpty = d;
                int remainder = dEmpty % nEmpty;
                while (remainder != 0) {
                    dEmpty = nEmpty;
                    nEmpty = remainder;
                    remainder = dEmpty % nEmpty;
                }
                int gcd = nEmpty;
                if (gcd != 0) {
                    if (gcd < 0) {
                        if (n < 0 && d < 0 || n > 0 && d < 0) {
                            n /= gcd; d /= gcd;
                        }
                        else if (n < 0 && d > 0) {
                            n /= -gcd; d /= -gcd;
                        }
                    }
                    else {
                            if (n < 0 && d > 0 || n > 0 && d > 0) {
                                n /= gcd; d /= gcd;
                            }
                            else if (n > 0 && d < 0) {
                                n /= -gcd; d /= -gcd;
                            }
                    }

                    }
                }
            }
        }


    friend fraction operator+ (fraction f1,  fraction f2) {
        fraction temp;
        if ((f1.d == 0) || (f2.d == 0)) {
            temp.n = f1.n * f2.d + f1.d * f2.n;
            temp.d = 0;
        }
        else
```

```
        {
            temp.n = f1.n * f2.d + f1.d * f2.n;
            temp.d = f1.d * f2.d;
            temp.simpl();
        }
        return temp;
    }


    friend fraction operator- (fraction f1,  fraction f2) {
        fraction temp;
        if ((f1.d == 0) || (f2.d == 0)) {
            temp.n = f1.n * f2.d + f1.d * f2.n;
            temp.d = 0;
        }
        else {
            temp.n = f1.n * f2.d - f1.d * f2.n;
            temp.d = f1.d * f2.d;
            temp.simpl();
        }
        return temp;
    }


    friend fraction operator* (fraction f1, fraction f2) {
        fraction temp;
        if ((f1.d == 0) || (f2.d == 0)) {
            temp.n = f1.n * f2.d + f1.d * f2.n;
            temp.d = 0;
        } else
        {
        temp.n = f1.n * f2.n;
        temp.d = f1.d * f2.d;
        temp.simpl();
        }
        return temp;
    }


    friend fraction operator/ (fraction f1,  fraction f2)
    {
        fraction temp;
        if ((f1.d == 0) || (f2.d == 0)) {
            temp.n = f1.n * f2.d + f1.d * f2.n;
            temp.d = 0;
        }
        else {
            temp.n = f1.n * f2.d;
            temp.d = f1.d * f2.n;
            temp.simpl();
        }
        return temp;
    }


    friend istream& operator>>(istream&, fraction& frctn);
    friend ostream& operator<<(ostream& out, const fraction& frctn);

};

// input and output
ostream& operator<<(ostream& out, const fraction& frctn) {
    if (frctn.d == 0) {
        out << "zero division error" << "\n";
    }
```

```
    else if (frctn.n == 0) {
        out << 0 << "\n";
    }
    else if (frctn.d == 1){
        out << frctn.n << "\n";
    }
    else {
        out << frctn.n << "/" << frctn.d << "\n";
    }
    return out;
}

istream& operator>>(istream& in, fraction& number)
{
    in >> number.n >> number.d;
    return in;
}

int main()
{

    setlocale(LC_ALL, "Ukrainian");
    fraction first;
    fraction second;

    cout << "введіть дані першого дробу: " << "\n";
    cin >> first;
    first.simpl();
    cout << "перший дріб: " << first << "\n";

    cout << "введіть дані другого дробу: " << "\n";
    cin >> second;
    second.simpl();
    cout << "другий дріб: " << second << "\n";


    cout << "сума: " << first + second << "\n";
    cout << "різниця: " << first - second << "\n";
    cout << "множення: " << first * second << "\n";
    cout << "ділення: " << first / second << "\n";

    system("pause");
    return 0;
       }
```

Приклад працездатності програми до завдання 1.1 (Рисунок 2.2.1)

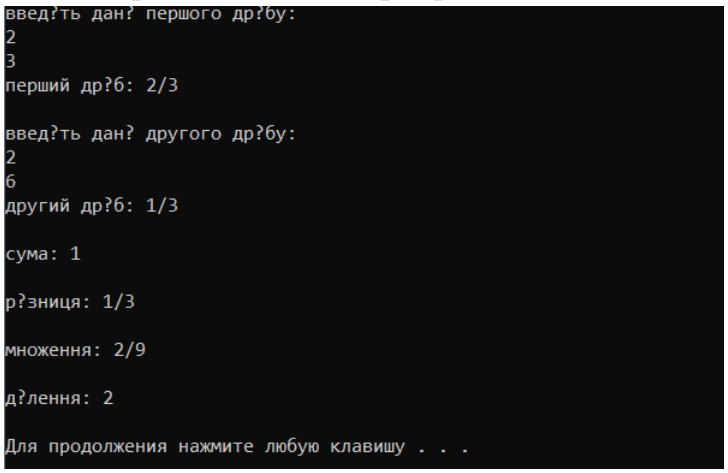

Рисунок 2.2.1 – Приклад працездатності програми до першого завдання

Завдання 1.2

```
#include <fstream>
#include <iostream>
using namespace std;


class matrix
{
private:
    int Row, Col;
    int** Value;
public:


    class badData //класс проверки ввода
    {
        int bad_value;
    public:
        badData(int value) : bad_value(value) {}
        badData() {}
    };


    class imposibleAction  //класс невозможных действий
    {
        int bad_data;
    public:
        imposibleAction(int value) : bad_data(value) {}
        imposibleAction() {}
    };




    matrix(int, int);
    matrix(const matrix&);


    int GetRow();
    int GetCol();

    int& operator()(int, int);
    const int& operator()(int, int) const;
    int* operator[] (int index) {
        return this->Value[index];
    }

    friend istream& operator>>(istream& istr, matrix& m);
    friend ostream& operator<<(ostream& ostr, const matrix& m);
    friend matrix operator+(matrix& m1, matrix& m2);
    friend matrix operator-(matrix& m1, matrix& m2);
    friend matrix operator*(matrix& m1, matrix& m2);
    friend matrix operator*(matrix& m1, int& number);

    ~matrix();

};



matrix::matrix(int row, int col)
```

```
{
    Row = row;
    Col = col;
    Value = new int* [row];
    for (int i = 0; i < row; i++) Value[i] = new int[col];

}

matrix::matrix(const matrix& m) //копирующий конструктор
    :Row(m.Row), Col(m.Col)
{
    Value = new int* [Row];
    for (int i = 0; i < Row; i++)  Value[i] = new int[Col];
    for (int i = 0; i < Row; i++)
    {
        for (int j = 0; j < Col; j++)
            Value[i][j] = m.Value[i][j];
    }
}

int matrix::GetRow()
{
    return Row;
}

int matrix::GetCol()
{
    return Col;
}

istream& operator>>(istream& istr, matrix& m)
{
    for (int i = 0; i < m.GetRow(); i++)
    {
        for (int j = 0; j < m.GetCol(); j++)
        {
            istr >> m.Value[i][j];
        }
    }
    return istr;
}

ostream& operator<<(ostream& ostr, const matrix& m)
{
    for (int i = 0; i < m.Row; i++)
    {
        for (int j = 0; j < m.Col; j++)
        {
            ostr << m.Value[i][j] << "\t";
        }
        ostr << "\n";
    }
    return ostr;
}

matrix operator+(matrix& m1, matrix& m2)
{
    matrix temp(m1.GetRow(), m1.GetCol());
    for (int i = 0; i < m1.GetRow(); i++)
    {
        for (int j = 0; j < m1.GetCol(); j++)
        {
            temp(i, j) = m1(i, j) + m2(i, j);
        }
    }
```

```
    return temp;

}

matrix operator-(matrix& m1, matrix& m2)
{
    matrix temp(m1.GetRow(), m1.GetCol());
    for (int i = 0; i < m1.GetRow(); i++)
    {
        for (int j = 0; j < m1.GetCol(); j++)
        {
            temp(i, j) = m1(i, j) - m2(i, j);
        }
    }
    return temp;
}

matrix operator*(matrix& m1, matrix& m2) {
    matrix temp(m1.GetRow(), m2.GetCol());
    for (int i = 0; i < m1.GetRow(); i++)
    {
        for (int j = 0; j < m2.GetCol(); j++) {
            temp(i, j) = 0;
            for (int cell = 0; cell < m2.GetRow(); cell++)
            {
                temp(i, j) += m1(i, cell) * m2(cell, j);
            }
        }
    }
    return temp;
}


matrix operator*(matrix& m1, int& number) {
    matrix temp(m1.GetRow(), m1.GetCol());
    for (int i = 0; i < m1.GetRow(); i++)
    {
        for (int j = 0; j < m1.GetCol(); j++)
        {
            temp(i, j) = m1(i, j) * number;
        }
    }
    return temp;
}

int& matrix::operator()(int row, int col) {
    return Value[row][col];
}

const int& matrix::operator()(int row, int col) const {
    return Value[row][col];
}

matrix::~matrix()
{
    for (int i = 0; i < Row; i++)
        delete[] Value[i];
    delete[] Value;
}

matrix var27(matrix& m1)
{
    matrix temp(m1.GetRow(), m1.GetCol());
    for (int i = 0; i < m1.GetRow(); i++)
    {
```

```
        for (int j = 0; j < m1.GetCol(); j++)
        {
            if ((m1(i, j) < 0) && (m1(i, j) % 2 != 0))
            {
                temp(i, j) = m1(i, j) * 3;
            }
            else {
                temp(i, j) = m1(i, j);

            }
        }
    }
    return(temp);
}

int main()
{
    setlocale(LC_ALL, "Ukrainian");
    int m, n, q, w, mult, func; bool swtchr = true;
    try {
        cout << "введіть кількість рядків та стовбців матриці A \n"; cin >> q >> w;


        if ((q <= 0) or (w <= 0))
            throw matrix::badData();
        matrix a(q, w);


        cout << "введіть кількість рядків та стовбців матриці B \n"; cin >> m >> n;

        if ((m <= 0) or (n <= 0))
            throw matrix::badData();
        matrix b(m, n);

        cout << "заповніть матрицю A:\n";  cin >> a;
        cout << "заповніть матрицю B:\n";  cin >> b;


        while (swtchr)
        {
            cout << "Перша матрица: \n" << a << endl << "Друга матрица: \n" << b << "\n \n \n";


            cout << "індивідуальне завдання 27ого варіанту(Усі від'ємні елементи з непарними
значеннями повинні бути збільшені в два рази): \n \n \n";
            cout << "матриця A \n" << var27(a);
            cout << "матриця B \n" << var27(b) << "\n \n \n";

            cout << "Що потрібно знайти?: \n" << "1. сумма \n" << "2. різниця \n" << "3. добуток
матриць \n" << "4.добуток матриць на число\n"
                << "5. взяття по індексу\n" << "6. кінець програми\n";
            cin >> func;
            cout << endl;
            switch (func)
            {
            case 1:
                if (a.GetRow() == b.GetRow() and a.GetCol() == b.GetCol())
                {
                    cout << a + b << endl;
                }
                else
                {
                    cout << "Не можна знайти суму матриць із різним розміром \n";
                    throw matrix::imposibleAction();
                }
```

```
                break;
            case 2:
                if (a.GetRow() == b.GetRow() and a.GetCol() == b.GetCol()) {
                    cout << a - b << endl;
                }
                else
                {
                    cout << "Не можна знайти різницю матриць із різним розміром \n";
                    throw matrix::imposibleAction();
                }
                break;
            case 3:
                if (a.GetCol() == b.GetRow())
                {
                    cout << a * b << endl;
                }
                else
                {
                    cout << "Кількість стовбців першої матриці повинна має рівнятися кількості
рядків другої \n";
                    throw matrix::imposibleAction();
                }
                break;
            case 4:
                cout << "введіть множник:\n";
                cin >> mult;
                cout << "множена матриця A: \n" << a * mult << "\n";
                cout << "множена матриця B: \n" << b * mult << "\n";
                break;
            case 5:
                cout << "введіть шукаємі індекси:\n";
                int z, x;
                cin >> z >> x;
                if ((z <= 0) or (x <= 0)) {
                    throw matrix::badData();
                }
                else
                {
                    if ((z > q) or (x > w)) {
                        throw matrix::badData();
                    }
                    else {
                        cout << "з матриці A: \n" << a(z - 1, x - 1) << "\n";
                    }
                    if ((z > m) or (x > n)) {
                        throw matrix::badData();
                    }
                    else {
                        cout << "з матриці B: \n" << b(z - 1, x - 1) << "\n";
                    }
                }
                break;
            case 6:
                return 0;
            default:
                throw matrix::badData();
            }
            cout << "Продовжити? \n 0.Ні \n 1.так \n";
            cin >> swtchr;
        }



    }
    catch (matrix::badData& data) {
```

```
        cout << "помилка вводу даних \n";
    }
    catch (matrix::imposibleAction& data) {
        cout << "помилка математичної операції \n";
    }

    system("pause");
    return 0;
};
```

Приклад працездатності програми до завдання 1.2 (Рисуноки 2.2.2-2.2.3)

```
введ?ть к?льк?сть рядк?в та стовбц?в матриц? A
2
2
введ?ть к?льк?сть рядк?в та стовбц?в матриц? B
2
2
заповн?ть матрицю A:
-1
-2
-3
-4
заповн?ть матрицю B:
-2
-4
-6
-8
Перша матриця:
-1      -2
-3      -4

Друга матриця:
-2      -4
-6      -8
```

Рисунок 2.2.2 – Приклад працездатності програми до другого завдання

```
Що потр?бно знайти?:
1. сумма
2. р?зниця
3. добуток матриць
4.добуток матриць на число
5. взяття по ?ндексу
6. к?нець програми
3

14      20
30      44

Продовжити?
 0.Н?
 1.так
0
?ндив?дуальне завдання 27ого вар?анту:
матриця A
-3      -2
-9      -4
матриця B
-2      -4
-6      -8
Для продолжения нажмите любую клавишу . . .

C:\Users\esulu\Desktop\FOLDER_HXZE\1 курс\labs_op\2
```

Рисунок 2.2.3 – Приклад працездатності програми до другого завдання

Завдання 1.3

```
#include <iostream>
```

```
using namespace std;
class summa {
public:
    summa(int val)
    : term(val) {
        sum += term;
    }

    static int GetSum() {
        return sum;
    }

private:
    static int sum;
    int term;
};

int summa::sum = 0;

int main()
{
    int a = 8;
    summa one(a);
    cout << "+" << a << " = " << one.GetSum() << "\n";
    a = 15;
    summa two(a);
    cout << "+" << a << " = " << two.GetSum() << "\n";
    a = 46;
    summa three(46);
    cout << "+" << a << " = " << three.GetSum() << "\n";

    system("pause");
    return 0;
}
```

Приклад працездатності програми до завдання 1.3 (Рисунок 2.2.4)

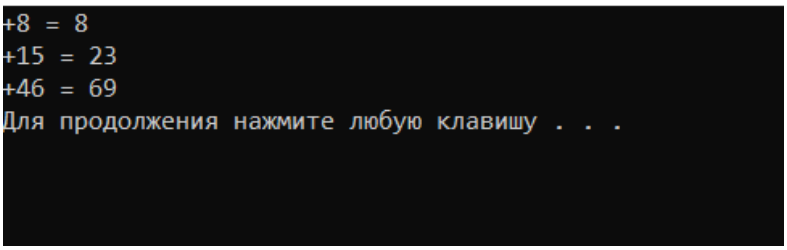

Рисунок 2.2.4 – Приклад працездатності програми до другого завдання

ВИСНОВКИ

У ході лабораторної роботи, я ознайомився з класами в C++ та отримав навички зі їх створюванням та елементарною роботою з ними. Для виконання завдань я відтворив класи для представлення простого дробу, двовимірного масиву та клас для підрахунку суми введених значень.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1 Іванов Л. В. Основи програмування (частина 2) Лабораторна робота 1 – Створення та використання класів C++, Методичні вказівки; http://www.iwanoff.inf.ua/programming_2_ua/LabTraining01.html (дата звернення до джерела: 29.04.22)