

Інститут спеціального зв'язку та захисту інформації

Національного технічного університету України  
"Київський політехнічний інститут ім. Ігоря Сікорського"

Спеціальна кафедра №5

## **ЛАБОРАТОРНА РОБОТА**

з навчальної дисципліни  
**«Об'єктно-орієнтоване програмування»**

**Тема:** Дослідження об'єктних типів мови C++.

**Виконав:** курсант Жванський Роман

**Перевірив:** доцент Спеціальної кафедри №5  
Куліков В.М.

**Київ 2023**

## **Варіант №3**

### **I. Класи. Протокол класу. Конструктори і деструктори.**

#### **1.1 Завдання**

##### **Завдання 1**

Створити абстрактний тип даних - клас вектор, в якому є покажчик на double, кількість елементів. Визначити конструктор без параметрів, конструктор з параметром, конструктор з двома параметрами. Конструктор без параметрів виділяє місце для одного елемента і ініціалізує його в нуль. Конструктор з одним параметром (розмір вектора) виділяє місце і ініціалізує номером у векторі, конструктор з двома параметрами виділяє місце (перший аргумент) і ініціалізує другим аргументом. Деструктор звільняє пам'ять. Визначити функцію, яка надає елементу вектора деяке значення (параметр за замовчуванням), функцію яка повертає певний елемент вектора. Визначити функцію друку. Визначити функції додавання, множення, віднімання, які здійснюють ці арифметичні операції з даними цього класу і вбудованого double. Визначити методи порівняння: більше, менше або дорівнює. Перевірити роботу цього класу.

##### **Завдання 2**

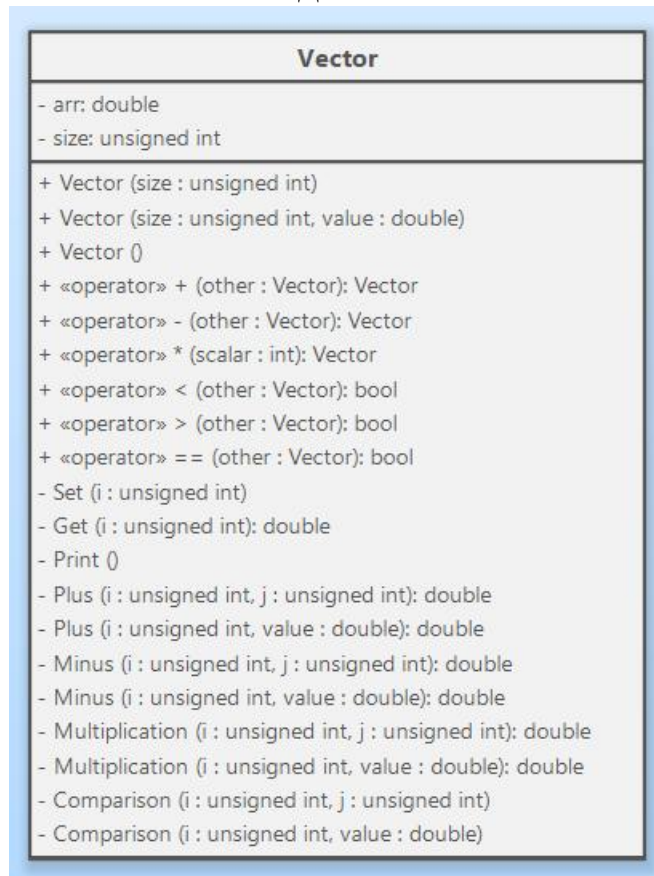
Створити клас матриця. Даний клас містить покажчик на double, розмір рядків і стовпців. Визначити конструктор без параметрів, конструктор з одним параметром і конструктор з двома параметрами, деструктор. Визначити методи доступу для отримання значення елемента (i,j) і адресу цього елемента. Визначити функцію друку. Визначити функції додавання і віднімання (матриці з матрицею), множення матриці на матрицю. Визначити множення матриці на число. Перевірити роботу цього класу.

##### **Завдання 3**

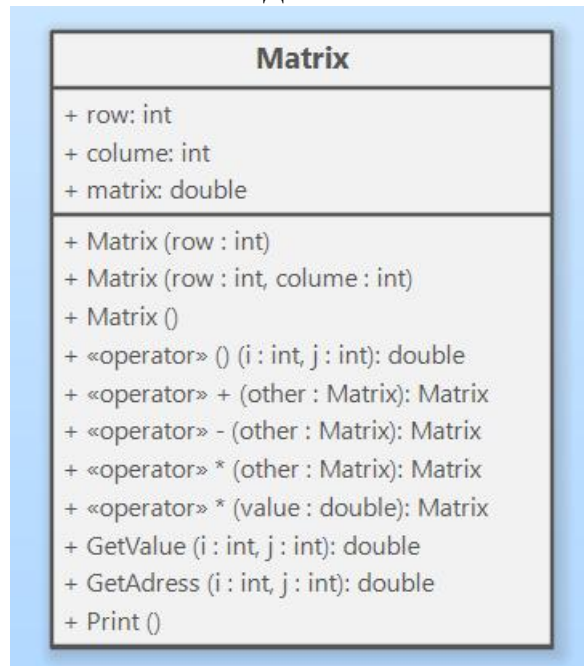
Створити клас типу - прямокутник. Поля - висота і ширина. Функції-методи обчислюють площу, периметр, встановлюють поля і повертають значення. Визначити функцію друку. Перевірити роботу цього класу.

## **1.2 Об'єктні моделі мовою UML (діаграми)**

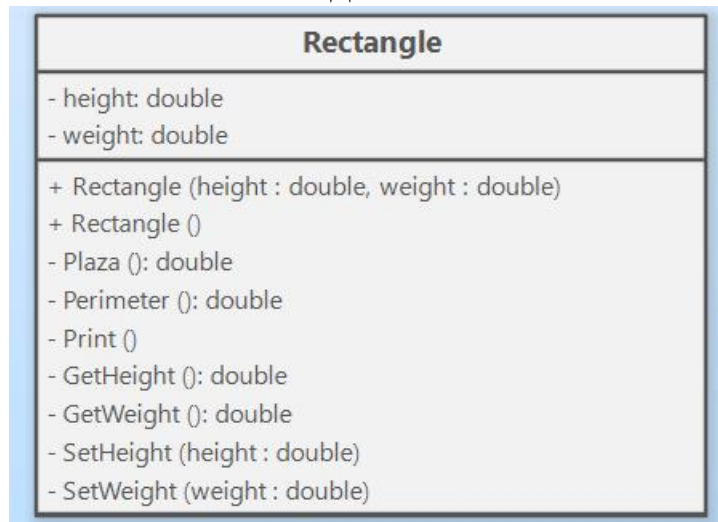
### **Завдання 1**



### **Завдання 2**



### Завдання 3



### 1.3 Тексти програм(з коментарями, що пояснюють відповідність розроблених програм варіанту завдання)

#### Завдання 1

```
#include <iostream>

#define default 10

class Vector
{
public:
    Vector(); // конструктор за замовчуванням
    Vector(unsigned int size); // конструктор з 1 параметром
    Vector(unsigned int size, double value); // конструктор з 2 параметрами
    ~Vector(); // деструктор
    void Set(unsigned int i); // для встановлення значення i-го елемента вектора
    double Get(unsigned int i); // повертає i-тий елемент вектора
    void Print(); // Друк
    double Plus(unsigned int i, unsigned int j); // перший спосіб додавання
    double Plus(unsigned int i, double value); // другий спосіб додавання
    double Minus(unsigned int i, unsigned int j); // перший спосіб
    // віднімання
    double Minus(unsigned int i, double value); // другий спосіб
    // віднімання
    double Multiplication(unsigned int i, unsigned int j); // перший спосіб
    // множення
    double Multiplication(unsigned int i, double value); // другий спосіб
    // множення
```

```

    void Comparison(unsigned int i, unsigned int j);    // перший спосіб
    порівняння

    void Comparison(unsigned int i, double value);    // другий спосіб
    порівняння


    Vector operator+(Vector& other); // перегрузка оператора +
    Vector operator-(Vector& other); // перегрузка оператора -
    Vector operator*(int& scalar);   // перегрузка оператора *
    bool operator<(Vector& other);   // перегрузка оператора <
    bool operator>(Vector& other);   // перегрузка оператора >
    bool operator==(Vector& other);  // перегрузка оператора ==

private:
    double *arr;           // вектор
    unsigned int size;     // тип unsigned бо розмір не може бути від'ємним
};

Vector::Vector()
{
    this->size = 1;
    this->arr = new double[this->size];
    arr[0] = 0;
}

Vector::Vector(unsigned int size)
{
    this->size = size;
    this->arr = new double[this->size];
    for (unsigned int i = 0; i < size; i++)
    {
        arr[i] = i;
    }
}

Vector::Vector(unsigned int size, double value)
{
    this->size = size;
    this->arr = new double[this->size];
    for (unsigned int i = 0; i < size; i++)
    {
        arr[i] = value;
    }
}

```

```

    }
}
Vector::~~Vector()
{
    //delete[] arr;
}
void Vector::Set(unsigned int i)
{
    if (i >= size)
    {
        std::cout << "Errorr.Invalid value!\n";
        return;
    }
    arr[i] = default;
}
double Vector::Get(unsigned int i)
{
    if (i >= size)
    {
        std::cout << "Errorr.Invalid value!\n";
        return -0.0;
    }
    return arr[i];
}
void Vector::Print()
{
    for (unsigned int i = 0; i < size; i++)
    {
        std::cout << "Element [" << i << "] : " << arr[i] << "\n";
    }
}
double Vector::Plus(unsigned int i, unsigned int j)
{
    std::cout << "Plus =" << arr[i] + arr[j];
    return arr[i]+arr[j];
}
double Vector::Plus(unsigned int i, double value)

```

```

{
    std::cout << "Plus =" << arr[i] + value;
    return arr[i]+value;
}

double Vector::Minus(unsigned int i, unsigned int j)
{
    return arr[i]-arr[j];
}

double Vector::Minus(unsigned int i, double value)
{
    return arr[i] - value;
}

double Vector::Multiplication(unsigned int i, unsigned int j)
{
    return arr[i]*arr[j];
}

double Vector::Multiplication(unsigned int i, double value)
{
    return arr[i] * value;
}

void Vector::Comparison(unsigned int i, unsigned int j)
{
    if (i >= size || j >= size || i==j)
    {
        std::cout << "Error.Invalid value \n";
    }
    std::cout << "arr[i] is " << arr[i] << "arr[i] is " << arr[j] << "\n";
    if (arr[i]==arr[j])
    {
        std::cout <<"Elements are equal \n";
    }
    else if (arr[i]>arr[j])
    {
        std::cout << "Elements arr[i] is more than arr[j]\n";
    }
    else
    {

```

```

        std::cout << "Elements arr[j] is more than arr[i]\n";
    }
}

void Vector::Comparison(unsigned int i, double value)
{
    if (i >= size)
    {
        std::cout << "Error.Invalid value \n";
    }
    std::cout << "arr[i] is " << arr[i] << "value is " << value<<"\n";
    if (arr[i] == value)
    {
        std::cout << "Elements are equal \n";
    }
    else if (arr[i] > value)
    {
        std::cout << "Elements arr[i] is more than value\n";
    }
    else
    {
        std::cout << "Elements value is more than arr[i]\n";
    }
}

Vector Vector::operator+(Vector& other)
{
    Vector result(size, 0);
    if (size == other.size) {
        for (int i = 0; i < size; i++) {
            result.arr[i] = arr[i] + other.arr[i];
        }
    }
    return result;
}

Vector Vector::operator-(Vector& other) {
    Vector result(size, 0);
    if (size == other.size) {
        for (int i = 0; i < size; i++) {

```



```

        result.arr[i] = arr[i] - other.arr[i];
    }
}
return result;
}

Vector Vector::operator*(const int& scalar) {
    Vector result(size, 0);
    for (int i = 0; i < size; i++) {
        result.arr[i] = arr[i] * scalar;
    }
    return result;
}

bool Vector::operator<(Vector& other) {
    if (size < other.size) {
        return true;
    }
    else if (size > other.size) {
        return false;
    }
    else {
        for (int i = 0; i < size; i++) {
            if (arr[i] >= other.arr[i]) {
                return false;
            }
        }
        return true;
    }
}

bool Vector::operator>(Vector& other) {
    if (size > other.size) {
        return true;
    }
    else if (size < other.size) {
        return false;
    }
    else {
        for (int i = 0; i < size; i++) {

```

```

        if (arr[i] <= other.arr[i]) {
            return false;
        }
    }
    return true;
}

}

bool Vector::operator==(Vector& other) {
    if (size != other.size) {
        std::cout << "Invalid size\n";
        return false;
    }
    else {
        for (int i = 0; i < size; i++) {
            if (arr[i] != other.arr[i]) {
                std::cout << "Not equal \n";
                return false;
            }
        }
        return true;
    }
}

}

void main()
{
    Vector v3(3, 2.0);
    Vector v2(3);
    v2.Set(2);
    std::cout << "v2[2] = " << v2.Get(2) << "\n";
    std::cout << "v3: \n" ;
    v3.Print();
    std::cout << "v2: \n";

    v2.Print();
    std::cout << "v3 + v2: \n";
    (v3+v2).Print();
    std::cout << "v3 + v2: ";
}

```

```

        v3.Plus(1, 2.0);

        std::cout << "\n";

        std::cout << "v2 < v3 : " << (v2 < v3) << "\n";
    }

```

## Завдання 2

```

#include <iostream>

class Matrix
{
public:
    Matrix(); // конструктор за замовчуванням
    Matrix(int row); // конструктор з 1 параметром
    Matrix(int row, int colume); // конструктор з 2 параметрами
    ~Matrix(); //деструктор
    double GetValue(int row, int colume); // отримання значення матриці
    double *GetAdress(int row, int colume); // отримання адреса
    void Print(); // друк
    double& operator() (int i, int j); //перегрузка оператора ()
    Matrix operator+(Matrix& other); //перегрузка оператора +
    Matrix operator-(Matrix& other); //перегрузка оператора -
    Matrix operator*(Matrix& other); //перегрузка оператора *
    Matrix operator*(double value); //перегрузка оператора * на число
    int row, colume; // рядок, стовпець
    double **matrix; // матриця
};

Matrix::Matrix()
{
    this->row = 1;
    this->colume = 1;
    this->matrix = new double* [row];
    for (int i = 0; i < row; i++)
        this->matrix[i] = new double[colume];
    for (int i = 0; i < row; i++)
        for (int j = 0; j < colume; j++)
            this->matrix[i][j] = 0;
}

Matrix::Matrix(int row)
{

```

```

        this->row = row;
        this->column = 1;
        this->matrix = new double* [row];
        for (int i = 0; i < row; i++)
            this->matrix[i] = new double[column];

        for (int i = 0; i < row; i++)
            for (int j = 0; j < column; j++)
                this->matrix[i][j] = 0;
    }

Matrix::Matrix(int row, int column)
{
    this->row = row;
    this->column = column;
    this->matrix = new double* [row];
    for (int i = 0; i < row; i++)
        this->matrix[i] = new double[column];

    for (int i = 0; i < row; i++)
        for (int j = 0; j < column; j++)
            this->matrix[i][j] = 0;
}

Matrix::~~Matrix()
{
}

double Matrix::GetValue(int i, int j)
{
    return matrix[i][j];
}

double* Matrix::GetAdress(int i, int j)
{
    return &matrix[i][j];
}

void Matrix::Print()
{
    std::cout << "Matrix: \n";
    for (int i = 0; i < row; i++)

```

```

    {
        for (int j = 0; j < colume; j++)
        {
            std::cout << matrix[i][j] << " ";
        }
        std::cout << " \n";
    }
}

double& Matrix::operator() (int i, int j) { // метод отримання значення елемента
    return matrix[i][j];
}

Matrix Matrix::operator+(Matrix& other)
{
    Matrix result(row, colume);
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < colume; j++)
        {
            result(i, j) = matrix[i][j] + other.matrix[i][j];
        }
    }
    result.Print();
    return result;
}

Matrix Matrix::operator-(Matrix& other)
{
    Matrix result(row, colume);
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < colume; j++)
        {
            result(i, j) = matrix[i][j] - other.matrix[i][j];
        }
    }
    result.Print();
    return result;
}

```

```

Matrix Matrix::operator*(Matrix& other)
{
    Matrix result(this->row, other.columne);
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < other.columne; j++)
        {
            double suma = 0.0;
            for (int x = 0; x < columne; x++)
            {
                suma += matrix[i][x] * other.matrix[x][j];
            }
            result(i, j) = suma;
        }
    }
    result.Print();
    return result;
}

Matrix Matrix::operator*(double value)
{
    Matrix result(this->row, this->columne);
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < columne; j++)
        {
            result(i,j) = value * matrix[i][j];
        }
    }
    result.Print();
    return result;
}

int main()
{
    Matrix x(2,2), y(2,2);
    x(0, 0) = 1;
    x(0, 1) = 2;

```

```

x(1, 0) = 3;
x(1, 1) = 4;

y(0, 0) = 1;
y(0, 1) = 2;
y(1, 0) = 3;
y(1, 1) = 4;

std::cout << "x ";
x.Print();
std::cout << "y ";
y.Print();
std::cout << "x * 10 = ";
x * 10.0;
std::cout << "x*y = ";
x* y;
}

```

## Завдання 3

```
#include <iostream>
```

```

class Rectangle
{
public:
    Rectangle(double height, double weight); // конструктор з 2 параметрами
    ~Rectangle(); // конструктор за замовчуванням
    double Plaza(); // площа
    double Perimeter(); // периметр
    void Print(); // друк площі, периметра, ширини, висоти
    double GetHeight(); // отримання висоти
    double GetWeight(); // отримання ширини
    void SetHeight(double height); // задати висоту
    void SetWeight(double weight); // задати ширину
private:
    double height, weight; // висота, ширина
};

Rectangle::Rectangle(double height, double weight)
{

```

```

        this->height = height;
        this->weight = weight;
    }
Rectangle::~Rectangle()
{
}
double Rectangle::Plaza()
{
    return height * weight;
}
double Rectangle::Perimeter()
{
    return 2*(height+weight);
}
void Rectangle::Print()
{
    std::cout << "Height - " << height << " Weight - " << weight << " Perimeter - " << Perimeter() << " Plaza - " << Plaza()<<"\n";
}
double Rectangle::GetHeight()
{
    return height;
}
double Rectangle::GetWeight()
{
    return weight;
}
void Rectangle::SetHeight(double height)
{
    this->height = height;
}
void Rectangle::SetWeight(double weight)
{
    this->weight = weight;
}

```

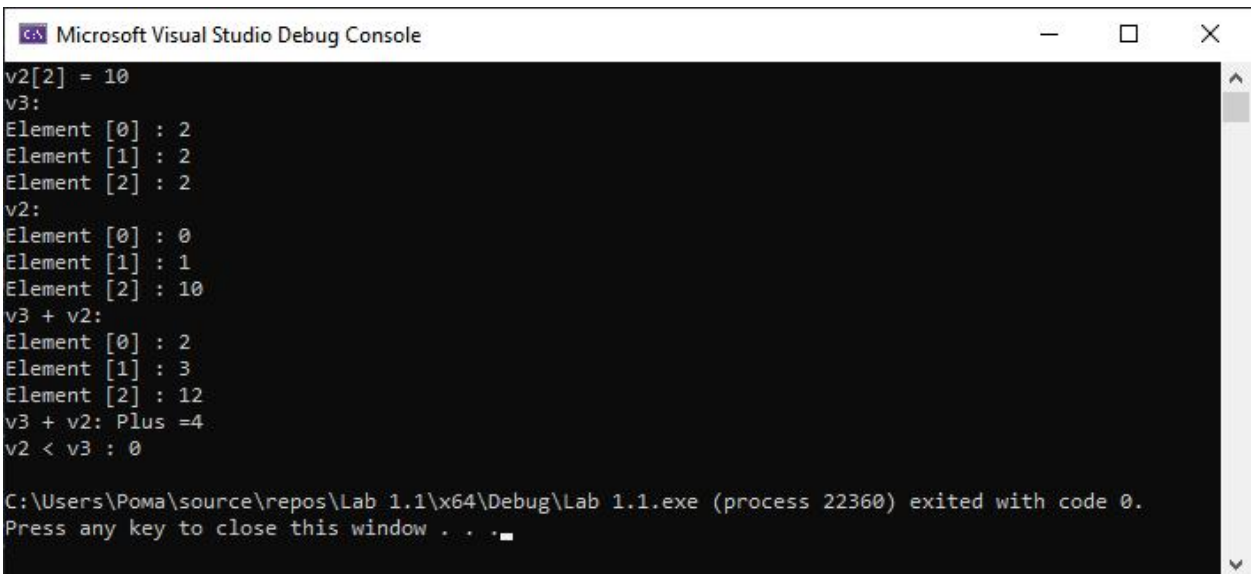


```

int main()
{
    Rectangle x(3,2);
    x.Print();
    x.SetHeight(4);
    x.SetWeight(4);
    x.Print();
}

```

## **1.4 Висновки:** **Завдання 1**



```

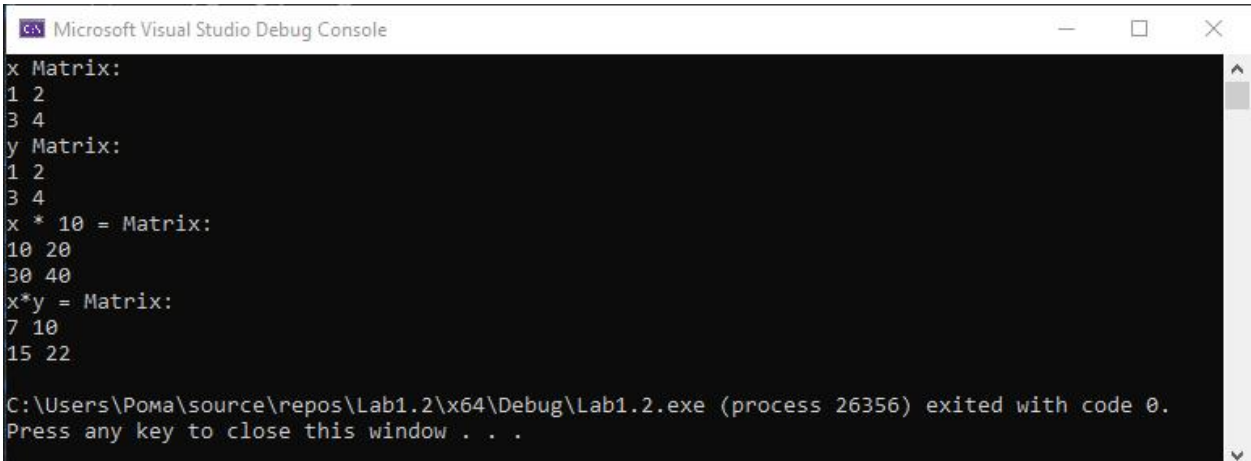
Microsoft Visual Studio Debug Console
v2[2] = 10
v3:
Element [0] : 2
Element [1] : 2
Element [2] : 2
v2:
Element [0] : 0
Element [1] : 1
Element [2] : 10
v3 + v2:
Element [0] : 2
Element [1] : 3
Element [2] : 12
v3 + v2: Plus =4
v2 < v3 : 0

C:\Users\Рома\source\repos\Lab 1.1\x64\Debug\Lab 1.1.exe (process 22360) exited with code 0.
Press any key to close this window . . .

```

Клас працює вірно. Програма виводить вектор на екран, робить заміну 2 елемента на 10(за замовчуванням). Може робити арифметичні операції між двома векторами та їх порівняння. Також може проводити арифметичні дії з окремими елементами двох різних векторів.

## **Завдання 2**



```

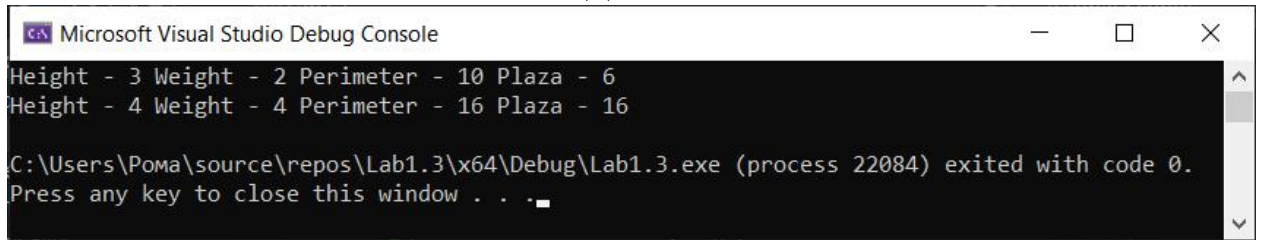
Microsoft Visual Studio Debug Console
x Matrix:
1 2
3 4
y Matrix:
1 2
3 4
x * 10 = Matrix:
10 20
30 40
x*y = Matrix:
7 10
15 22

C:\Users\Рома\source\repos\Lab1.2\x64\Debug\Lab1.2.exe (process 26356) exited with code 0.
Press any key to close this window . . .

```

Клас працює вірно. Програма дозволяє виконувати арифметичні операції між двома матрицями. Також множити матрицю на задане число.

## Завдання 3



```
Microsoft Visual Studio Debug Console
Height - 3 Weight - 2 Perimeter - 10 Plaza - 6
Height - 4 Weight - 4 Perimeter - 16 Plaza - 16
C:\Users\Poma\source\repos\Lab1.3\x64\Debug\Lab1.3.exe (process 22084) exited with code 0.
Press any key to close this window . . .
```

Клас працює вірно. Програма може обчислювати периметр та площа за заданими величинами.

## II. Перетворення типів. Дружні функції.

### Конструктор копіювання.

#### 2.1 Завдання

##### Завдання 1

Створити клас цілих чисел. Клас має конструктор за замовчуванням, конструктор, що перетворює `int` в об'єкт класу. Визначити оператор перетворення об'єкта типу цілих чисел в число типу `int`. Створити клас дійсних чисел та визначити взаємне перетворення з класом цілих чисел. Перевірити роботу цього класу.

##### Завдання 2

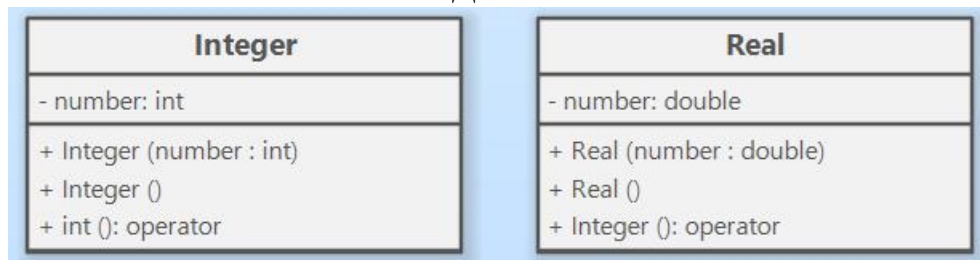
Створити клас цілих чисел `Integer`. Визначити перевантажену функцію, яка повертає максимальне з двох аргументів. Функція не є методом класу цілих чисел. Перевантажені функції мають аргументи типу `int`, `double`, `Integer`. Тіла перевантажених функцій повинні бути однаковими. Перевірити роботу цього класу.

##### Завдання 3

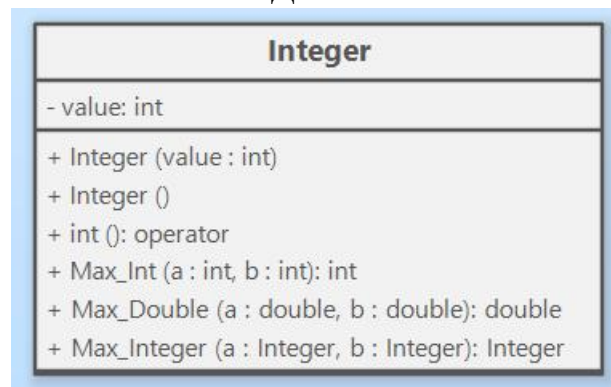
Створити два класи: дійсні (`Float`) і матриця (`float **`). Визначити конструктори - за замовчуванням, з параметром, для класу матриця з двома параметрами, копіювання, деструктори. Визначити функцію множення матриці на дійсне (`Float`), як дружню. Перевірити роботу цього класу.

## 2.2 Об'єктні моделі

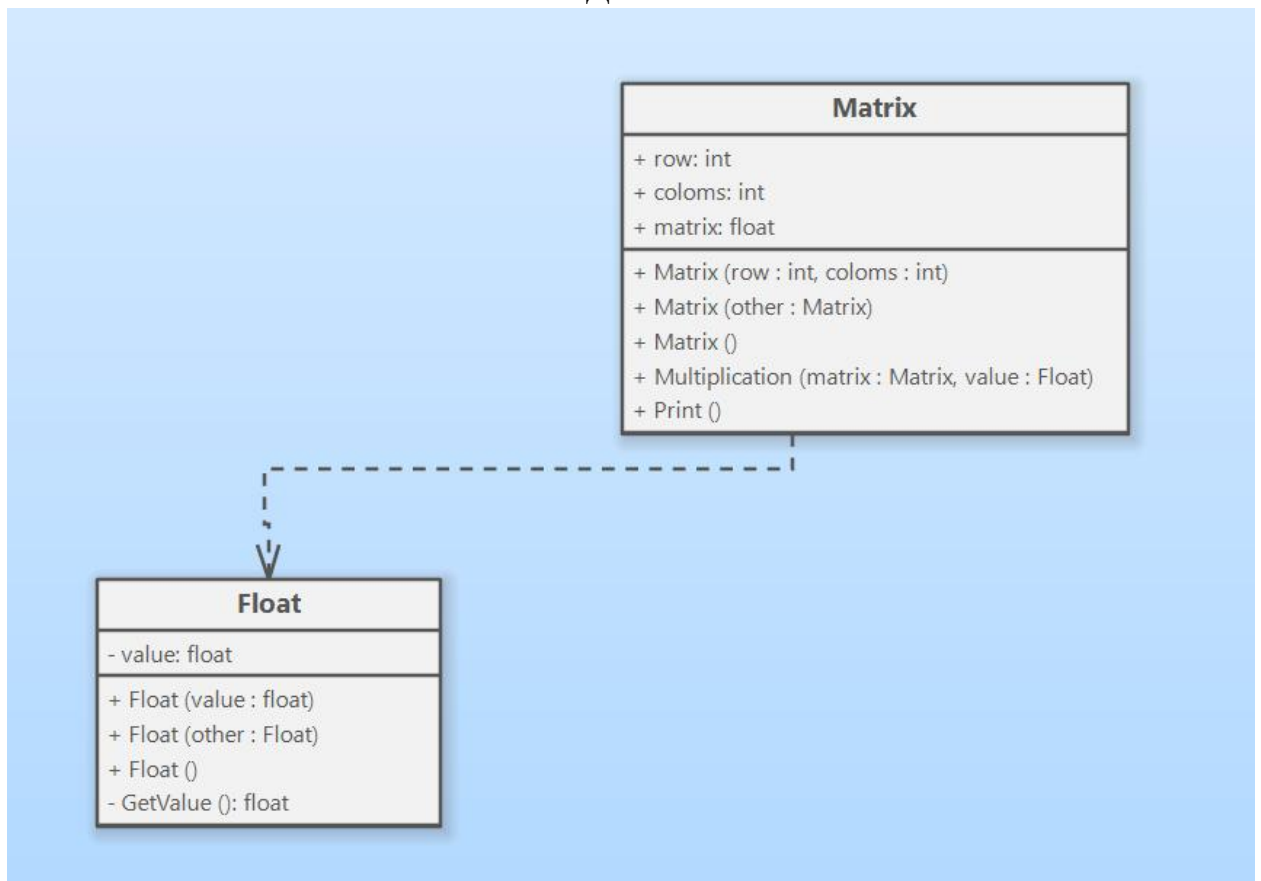
### Завдання 1



### Завдання 2



### Завдання 3



## 2.3 Тексти програм

### Завдання 1

```
#include <iostream>

class Real;
class Integer
{
public:
    Integer();           // конструктор за замовчування
    Integer(int number); // конструктор з 1 параметром
    ~Integer();
    operator int();

private:
    int number;
};

Integer::Integer()
{
    this->number = 0;
}

Integer::Integer(int number)
{
    this->number = number;
}

Integer::~Integer()
{
}

Integer::operator int()
{
    return number;
}

class Real
{
public:
    Real(); // конструктор за замовчування
    Real(double number); // конструктор з 1 параметром
```

```

        ~Real();

        operator Integer();
private:
        double number;
};

Real::Real()
{
    this->number = 0;
}
Real::Real(double number)
{
    this->number = number;
}
Real::~~Real()
{
}
Real::operator Integer()
{
    return Integer((int)number);
}

int main()
{
    int num = 10;
    Integer a = num;
    int num2 = (int)a;
    Real a2(3.14);
    Integer a3 = a2;
    std::cout << "a = " << a << "\n";
    std::cout << "num2 = " << num2 << "\n";
    std::cout << "a3 = " << a3 << "\n";
}

```

## Завдання 2

```
#include <iostream>

class Integer
{
public:

    Integer(int value); // конструктор з 1 параметром
    ~Integer();         // конструктор за замовчування
    operator int();

    friend int Max_Int(int a, int b); // дружня функція для виведення макс числа
    типу int

    friend double Max_Double(double a, double b); // дружня функція для
    виведення макс числа типу double

    friend Integer Max_Integer(Integer& a, Integer &b); // дружня функція для
    виведення макс числа типу Integer
private:
    int value;
};

Integer::Integer(int value)
{
    this->value = value;
}

Integer::~Integer()
{
}

Integer::operator int()
{
    return value;
}

int Max_Int(int a, int b)
{
    return (a > b) ? a : b;
}

double Max_Double(double a, double b)
{
    return (a > b) ? a : b;
}
```

```

}

Integer Max_Integer(Integer& a, Integer& b)
{
    return (a.value > b.value) ? a : b;
}

int main()
{
    Integer a(3), b(1);
    std::cout << "Max Integer - ";
    std::cout << Max_Integer(a, b) << "\n";

    int x = 2, y = 5;
    std::cout << "Max Int - ";
    std::cout << Max_Int(x, y) << "\n";

    double c = 3.12, d = 41.23;
    std::cout << "Max double - ";
    std::cout << Max_Double(c, d) << "\n";
}

```

## Завдання 3

```

#include <iostream>

class Float
{
public:
    Float();           // конструктор за замовчування
    Float(float value); // конструктор з 1 параметром
    Float(Float & other); // конструктор копіювання
    ~Float();
    float GetValue();  // отримання значення

private:
    float value;
};

Float::Float()

```

```

{
    this->value = 0.0;
}
Float::Float(float value)
{
    this->value = value;
}
Float::Float(Float& other)
{
    this->value = other.value;
}
Float::~Float()
{
}
float Float::GetValue()
{
    return value;
}

```

```

class Matrix
{
public:
    Matrix(); // конструктор за замовчування
    Matrix(int row, int coloms); // конструктор з 1 параметром
    Matrix(Matrix & other); // конструктор копіювання
    ~Matrix();
    friend void Multiplication(Matrix & matrix, Float & value); // функція
    множення
    void Print(); // друк
    int row, coloms;
    float** matrix;
};

Matrix::Matrix()
{

```



```

        this->row = 1;
        this->coloms = 1;
        this->matrix = new float* [row];
        for (int i = 0; i < row; i++)
            this->matrix[i] = new float[coloms];

        for (int i = 0; i < row; i++)
            for (int j = 0; j < coloms; j++)
                this->matrix[i][j] = 0;
    }

    Matrix::Matrix(int row, int coloms)
    {
        this->row = row;
        this->coloms = coloms;
        this->matrix = new float* [row];
        for (int i = 0; i < row; i++) {
            this->matrix[i] = new float[coloms];
            for (int j = 0; j < coloms; j++)
                this->matrix[i][j] = 0;
        }
    }

    Matrix::Matrix(Matrix& other)
    {
        this->row = other.row;
        this->coloms = other.coloms;

        matrix = new float* [other.row];
        for (int i = 0; i < other.row; i++) {
            matrix[i] = new float[other.coloms];
            for (int j = 0; j < other.coloms; j++) {
                matrix[i][j] = other.matrix[i][j];
            }
        }
    }

    Matrix::~~Matrix()
    {

```

```

        for (int i = 0; i < row; i++)
        {
            delete[] matrix[i];
        }
        delete[] matrix;
    }

void Matrix::Print()
{
    std::cout << "Matrix:" << std::endl;
    for (int i = 0; i < row; ++i) {
        for (int j = 0; j < coloms; ++j) {
            std::cout << matrix[i][j] << " ";
        }
        std::cout << "\n";
    }
}

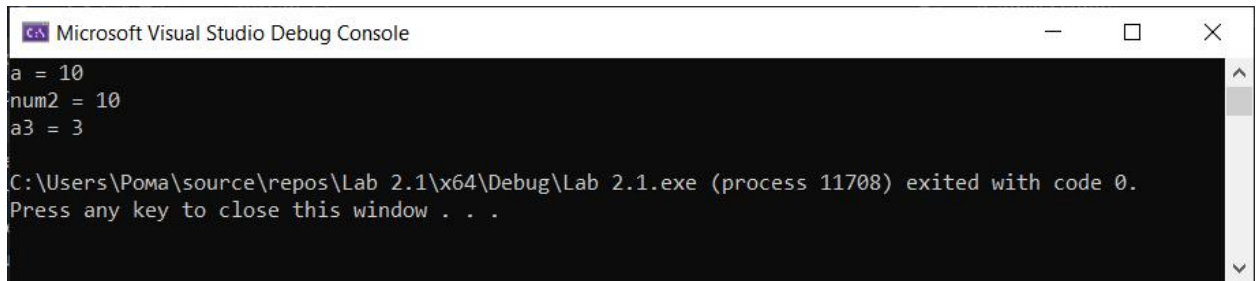
void Multiplication(Matrix& matrix, Float& value)
{
    for (int i = 0; i < matrix.row; ++i)
        for (int j = 0; j < matrix.coloms; ++j)
            matrix.matrix[i][j] *= value.GetValue();
}

int main()
{
    Matrix x(2, 2);
    x.matrix[0][0] = 1;
    x.matrix[0][1] = 2;
    x.matrix[1][0] = 3;
    x.matrix[1][1] = 4;
    x.Print();
    Float z(3);
    std::cout << "matrix * 3 ";
    Multiplication(x, z);
    x.Print();
}

```

## 2.4 Висновок

### Завдання 1



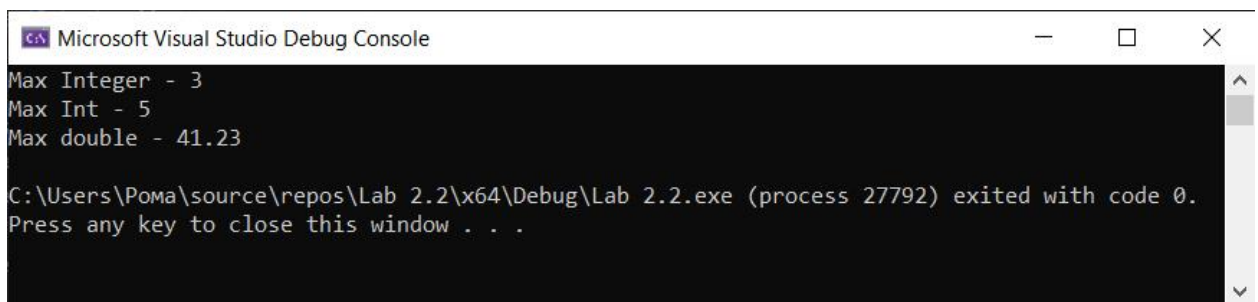
```
Microsoft Visual Studio Debug Console

a = 10
num2 = 10
a3 = 3

C:\Users\Роман\source\repos\Lab 2.1\x64\Debug\Lab 2.1.exe (process 11708) exited with code 0.
Press any key to close this window . . .
```

Клас працює вірно. Програма може перетворювати об'єкти типу цілих чисел число типу `int`. Може перетворювати об'єкт типу `int` в об'єкт класу(`Integer`). Також має взаємне перетворення цілих чисел з класом дійсних(`Real`).

### Завдання 2



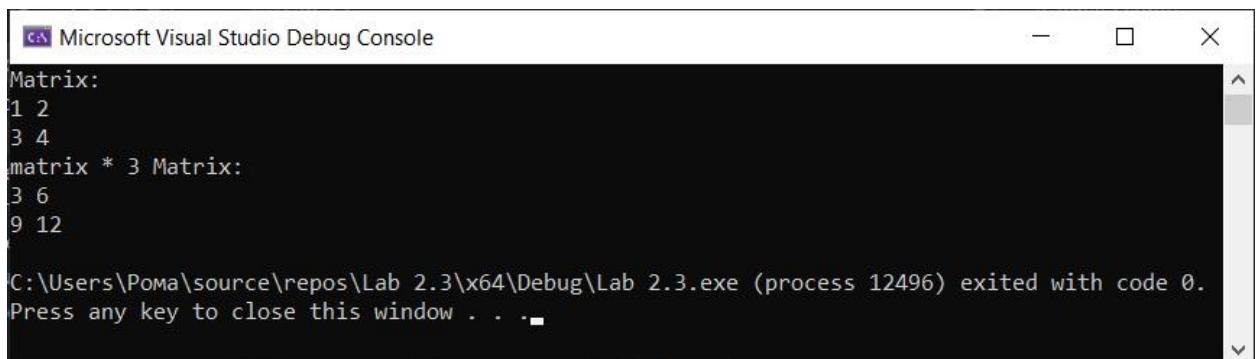
```
Microsoft Visual Studio Debug Console

Max Integer - 3
Max Int - 5
Max double - 41.23

C:\Users\Роман\source\repos\Lab 2.2\x64\Debug\Lab 2.2.exe (process 27792) exited with code 0.
Press any key to close this window . . .
```

Клас працює вірно. Програма може визначати максимальне значення двох аргументів різних типів: `int`, `double`, `Integer`(наш клас).

### Завдання 3



```
Microsoft Visual Studio Debug Console

Matrix:
1 2
3 4
matrix * 3 Matrix:
3 6
9 12

C:\Users\Роман\source\repos\Lab 2.3\x64\Debug\Lab 2.3.exe (process 12496) exited with code 0.
Press any key to close this window . . .
```

Клас працює вірно. Програма може множити матрицю на дійсне число.

### **III. Перевантаження операцій.**

#### **3.1 Завдання**

##### **Завдання 1**

Створити клас дійсних чисел. Визначити оператор ++ (інкремент), як функцію-метод і — (декремент), як дружню функцію. Перевірити роботу цього класу.

##### **Завдання 2**

Створити клас цілих чисел. Визначити оператор – (мінус), як функцію-метод і + (плюс), як дружню функцію. Перевірити роботу цього класу.

##### **Завдання 3**

Створити клас вектор, що містить покажчик на float\*, розмірність вектора. Клас має конструктори за замовчуванням, конструктор з одним і двома параметрами, конструктор копіювання і деструктор. Визначити оператор +, -, \*, - як дружні функції, =, +=, -=, \*=, [] - як функції-методи. Визначити оператори =, +, %, \*, +=, -=, \*= з цілим числом, оператори ++ (інкремент) і – (декремент). Визначити оператори =, +, /, \*, +=, -=, \*= з дійсним числом. Визначити функцію друку. Перевірити роботу цього класу.

##### **Завдання 4**

Створити клас матриця, що містить покажчик на float\*, число рядків і стовпців. Визначити конструктори за замовчуванням, конструктор з одним і з двома параметрами, конструктор копіювання, деструктор. Визначити оператори =, +, -, +=, -=, \*, \*= з об'єктами цього класу, з float і з вектором, визначеним в завданні 3. Визначити оператор []. Перевірити роботу цього класу.

## 3.2 Об'єктні моделі

### Завдання 1

Float
- value: double
+ Float (value : double)
+ Float ()
+ «operator» ++ (): Float
+ «operator» -- (value : Float): Float
+ «operator» << (out : std::ostream, x : Float): std::ostream

### Завдання 2

Integer
- value: int
+ Integer (value : int)
+ Integer ()
+ «operator» - (other : Integer): Integer
+ «operator» + (x : Integer, y : Integer): Integer
+ «operator» << (out : std::ostream, x : Integer): std::ostream

## Завдання 3

Vector
- value: float - size: int
+ Vector (size : int) + Vector (size : int, value : float) + Vector (other : Vector) + Vector () + «operator» [] (i : int): float + «operator» + (x : Vector, y : Vector): Vector + «operator» - (x : Vector, y : Vector): Vector + «operator» * (x : Vector, y : Vector): Vector + «operator» = (other : Vector): Vector + «operator» += (other : Vector): Vector + «operator» -= (other : Vector): Vector + «operator» *= (other : Vector): Vector + «operator» = (x : int): Vector + «operator» + (x : int): Vector + «operator» % (x : int): Vector + «operator» * (x : int): Vector + «operator» += (x : int): Vector + «operator» -= (x : int): Vector + «operator» *= (x : int): Vector + «operator» ++ (): Vector + «operator» -- (): Vector + «operator» = (x : float): Vector + «operator» + (x : float): Vector + «operator» / (x : float): Vector + «operator» * (x : float): Vector + «operator» += (x : float): Vector + «operator» -= (x : float): Vector + «operator» *= (x : float): Vector - Set (i : int, value : float) - Print ()

## Завдання 4

Vector	Matrix
- value: float - size: int	- row: int - coloms: int - matrix: float
+ Vector (size : int) + Vector (size : int, value : float) + Vector (other : Vector) + Vector ()	+ Matrix (row : int) + Matrix (row : int, coloms : int) + Matrix (other : Matrix) + Matrix ()
+ «operator» [] (i : int): float + «operator» [] (i : int): float + «operator» + (x : Vector, y : Vector): Vector + «operator» - (x : Vector, y : Vector): Vector + «operator» * (x : Vector, y : Vector): Vector + «operator» = (other : Vector): Vector + «operator» += (other : Vector): Vector + «operator» -= (other : Vector): Vector + «operator» *= (other : Vector): Vector + «operator» + (x : Vector, y : Matrix): Matrix + «operator» - (x : Vector, y : Matrix): Matrix + «operator» += (y : Matrix, x : Vector): Matrix + «operator» -= (y : Matrix, x : Vector): Matrix + «operator» * (x : Vector, y : Matrix): Matrix + «operator» *= (y : Matrix, x : Vector): Matrix	+ Print () + Set (i : int, matrix : float) + Get () : int + «operator» [] (i : int): float + «operator» [] (i : int, j : int): float + «operator» + (x : Vector, y : Matrix): Matrix + «operator» = (x : Vector): Matrix + «operator» - (x : Vector, y : Matrix): Matrix + «operator» += (y : Matrix, x : Vector): Matrix + «operator» -= (y : Matrix, x : Vector): Matrix + «operator» * (x : Vector, y : Matrix): Matrix + «operator» *= (y : Matrix, x : Vector): Matrix
+ «operator» = (x : int): Vector + «operator» + (x : int): Vector + «operator» % (x : int): Vector + «operator» * (x : int): Vector + «operator» += (x : int): Vector + «operator» -= (x : int): Vector + «operator» *= (x : int): Vector + «operator» ++ () : Vector + «operator» -- () : Vector + «operator» = (x : float): Vector + «operator» + (x : float): Vector + «operator» / (x : float): Vector + «operator» * (x : float): Vector + «operator» += (x : float): Vector + «operator» -= (x : float): Vector + «operator» *= (x : float): Vector - Set (i : int, value : float) - Get () : int - Print () - Geter (i : int): float	

## 3.3 Тексти програм

### Завдання 1

```
#include <iostream>

class Float
{
public:
    Float(); // конструктор за замовчування
    Float(double value); // конструктор з 1 параметром
    ~Float();

    Float& operator++(); // перегрузка оператора ++
    friend Float& operator--(Float& value); // перегрузка оператора --
    friend std::ostream& operator<<(std::ostream& out, Float& x); // перергузка
оператора <<
```

```

private:
    double value;
};
Float::Float()
{
    this->value = 0.0;
}
Float::Float(double value)
{
    this->value = value;
}
Float::~~Float()
{
}
Float& Float::operator++()
{
    ++value;
    return *this;
}
Float& operator--(Float& value)
{
    --value.value;
    return value;
}
std::ostream& operator<<(std::ostream& out, Float& x)
{
    out << x.value;
    return out;
}

int main()
{
    std::cout << "x = " <<x<< "\n";
    std::cout << "--x" << "\n";
    --x;
    std::cout << "x = " << x << "\n";
    std::cout << "++x" << "\n";
}

```



```

    ++x;

    std::cout << "x = " << x << "\n";
}

```

## Завдання 2

```

#include <iostream>

class Integer
{
public:
    Integer();           // конструктор за замовчуванням
    Integer(int value); // конструктор з 1 параметром
    ~Integer();

    Integer operator-(Integer& other); // перегрузка оператора -
    friend Integer operator+(Integer& x, Integer& y); // перегрузка оператора +
    friend std::ostream& operator<<(std::ostream& out, Integer& x); // перегрузка
оператора <<
private:
    int value;
};

Integer::Integer()
{
    this->value = 1;
}

Integer::Integer(int value)
{
    this->value = value;
}

Integer::~~Integer()
{
}

Integer Integer::operator-(Integer& other)
{
    return Integer(this->value - other.value);
}

Integer operator+(Integer& x, Integer& y)
{

```

```

        return Integer(x.value + y.value);
    }

std::ostream& operator<<(std::ostream& out, Integer& x)
{
    out << x.value;
    return out;
}

int main()
{
    Integer x(4), y(24);

    std::cout << "y = " << y << "\n";
    std::cout << "x = " << x << "\n";

    x = x - y;
    std::cout << "x-y = " << x << "\n";
    std::cout << "\n";
    std::cout << "x = " << x << "\n";
    std::cout << "y = " << y << "\n";

    x = x + y;
    std::cout << "x+y = " << x << "\n";
}

```

## Завдання 3

```

#include <iostream>
#include <cmath>

class Vector
{
public:
    Vector(); // конструктор за замовчуванням
    Vector(int size); // конструктор з 1 параметром
    Vector(int size, float* value); // конструктор з 2 параметрами
    Vector(Vector& other); // конструктор копіювання
    ~Vector();

    void Set(int i, float value); // задає i-тому елементу значення
    void Print(); // друк
    float operator[](int i); // перегрузка оператора []
}

```

```

friend Vector operator+(Vector& x, Vector& y); // перегрузка оператора +
friend Vector operator-(Vector& x, Vector& y); // перегрузка оператора -
friend Vector operator*(Vector& x, Vector& y); // перегрузка оператора *
Vector& operator=(Vector& other); // перегрузка оператора =
Vector& operator+=(Vector& other); // перегрузка оператора +=
Vector& operator-=(Vector& other); // перегрузка оператора -=
Vector& operator*=(Vector& other); // перегрузка оператора *=

//Перегрузка для цілих чисел
Vector& operator=(int x);
Vector& operator+(int x);
Vector& operator%(int x);
Vector& operator*(int x);
Vector& operator+=(int x);
Vector& operator-=(int x);
Vector& operator*=(int x);
Vector& operator++();
Vector& operator--();

//Перегрузка для дійсних чисел
Vector& operator=(float x);
Vector& operator+(float x);
Vector& operator/(float x);
Vector& operator*(float x);
Vector& operator+=(float x);
Vector& operator-=(float x);
Vector& operator*=(float x);

private:
    float* value;
    int size;
};

Vector::Vector()
{
    this->size = 0;
    this->value = 0;
}

```

```

}

Vector::Vector(int size)
{
    this->size = size;
    this->value = new float[size];
}

Vector::Vector(int size, float* value)
{
    this->size = size;
    this->value = new float[size];
    for (int i = 0; i < size; i++) {
        this->value[i] = value[i];
    }
}

Vector::Vector(Vector& other)
{
    this->size = other.size;
    this->value = other.value;
}

Vector::~~Vector()
{
}

void Vector::Set(int i, float value)
{
    this->value[i] = value;
}

void Vector::Print()
{
    for (int i = 0; i < size; i++)
    {
        std::cout << "[" << i << "]" << " element - " << value[i] << "\n";
    }
}

float Vector::operator[](int i)
{
    return value[i];
}

```

```

Vector operator+(Vector& x, Vector& y)
{
    Vector result(x.size);
    for (int i = 0; i < x.size; i++)
    {
        result.Set(i, x.value[i] + y.value[i]);
    }
    return result;
}

Vector operator-(Vector& x, Vector& y)
{
    Vector result(x.size);
    for (int i = 0; i < x.size; i++)
    {
        result.Set(i, x.value[i] - y.value[i]);
    }
    return result;
}

Vector operator*(Vector& x, Vector& y)
{
    Vector result(x.size);
    for (int i = 0; i < x.size; i++)
    {
        result.Set(i, x.value[i] * y.value[i]);
    }
    return result;
}

Vector& Vector::operator=(Vector& other)
{
    if (this != &other) {
        delete[] value;
        size = other.size;
        value = new float[size];
        for (int i = 0; i < size; i++) {
            value[i] = other.value[i];
        }
    }
}

```

```

        return *this;
    }

    Vector& Vector::operator+=(Vector& other)
    {
        if (size != other.size) {
            std::cout<<"Errorr.Invalid size";
        }
        for (int i = 0; i < size; i++) {
            value[i] += other.value[i];
        }
        return *this;
    }

    Vector& Vector::operator-=(Vector& other)
    {
        if (size != other.size) {
            std::cout << "Errorr.Invalid size";
        }
        for (int i = 0; i < size; i++) {
            value[i] -= other.value[i];
        }
        return *this;
    }

    Vector& Vector::operator*=(Vector& other)
    {
        if (size != other.size) {
            std::cout << "Errorr.Invalid size";
        }
        for (int i = 0; i < size; i++) {
            value[i] *= other.value[i];
        }
        return *this;
    }

    Vector& Vector::operator=(int x)
    {
        for (int i = 0; i < size; i++) {
            value[i] = x;
        }
    }

```

```

    }

    return *this;
}

Vector& Vector::operator+(int x)
{
    for (int i = 0; i < size; i++) {
        value[i] += x;
    }

    return *this;
}

Vector& Vector::operator%(int x)
{
    for (int i = 0; i < size; i++) {
        value[i] = fmod(value[i], x);
    }

    return *this;
}

Vector& Vector::operator*(int x)
{
    for (int i = 0; i < size; i++) {
        value[i] *= x;
    }

    return *this;
}

Vector& Vector::operator+=(int x)
{
    for (int i = 0; i < size; i++) {
        value[i] += x;
    }

    return *this;
}

Vector& Vector::operator-=(int x)
{
    for (int i = 0; i < size; i++) {
        value[i] -= x;
    }

    return *this;
}

```

```
}  
  
Vector& Vector::operator*=(int x)  
{  
    for (int i = 0; i < size; i++) {  
        value[i] *= x;  
    }  
    return *this;  
}  
  
Vector& Vector::operator++()  
{  
    for (int i = 0; i < size; i++) {  
        value[i] ++;  
    }  
    return *this;  
}  
  
Vector& Vector::operator--()  
{  
    for (int i = 0; i < size; i++) {  
        value[i]--;  
    }  
    return *this;  
}
```

```
  
Vector& Vector::operator=(float x)  
{  
    for (int i = 0; i < size; i++) {  
        value[i] = x;  
    }  
    return *this;  
}  
  
Vector& Vector::operator+=(float x)  
{  
    for (int i = 0; i < size; i++) {  
        value[i] += x;  
    }  
    return *this;  
}
```



```
}  
  
Vector& Vector::operator/(float x)  
{  
    for (int i = 0; i < size; i++) {  
        value[i] /= x;  
    }  
    return *this;  
}  
  
Vector& Vector::operator*(float x)  
{  
    for (int i = 0; i < size; i++) {  
        value[i] *= x;  
    }  
    return *this;  
}  
  
Vector& Vector::operator+=(float x)  
{  
    for (int i = 0; i < size; i++) {  
        value[i] += x;  
    }  
    return *this;  
}  
  
Vector& Vector::operator-=(float x)  
{  
    for (int i = 0; i < size; i++) {  
        value[i] -= x;  
    }  
    return *this;  
}  
  
Vector& Vector::operator*=(float x)  
{  
    for (int i = 0; i < size; i++) {  
        value[i] *= x;  
    }  
    return *this;  
}
```

```

int main()
{
    float a[] = { 1,2,3 };
    float b[] = { 3,2,1 };
    Vector x(3,a), c(3,b);
    std::cout << "Vector x: \n";
    x.Print();
    std::cout << "Vector c: \n";
    c.Print();
    std::cout << "Vector x*c: \n";
    x *= c;
    x.Print();
    std::cout << "Vector x+3: \n";
    x += 3;
    x.Print();
    std::cout << "Vector ++x: \n";
    ++x;
    x.Print();
    std::cout << "Vector --x: \n";
    --x;
    x.Print();
    std::cout << "Vector x* 2.5: \n";
    x * 2.5f;
    x.Print();
}

```

## Завдання 4

```

#include <iostream>
#include <cmath>
class Matrix;
class Vector
{
public:
    Vector(); // конструктор за замовчуванням
    Vector(int size); // конструктор з 1 параметром
    Vector(int size, float* value); // конструктор з 2 параметрами

```

```

Vector(Vector& other);           // конструктор копіювання
~Vector();

void Set(int i, float value);    // задає i-тому елементу значення
int Get();                      // отримання size
void Print();                   //друк
float Geter(int i);             // отримання i-того значення вектора
float operator[](int i);        // перегрузка оператора []
float& operator() (int i);      // перегрузка оператора ()
friend Vector operator+(Vector& x, Vector& y); // перегрузка оператора +
friend Vector operator-(Vector& x, Vector& y); // перегрузка оператора -
friend Vector operator*(Vector& x, Vector& y); // перегрузка оператора *
Vector& operator=(Vector& other); // перегрузка оператора =
Vector& operator+=(Vector& other); // перегрузка оператора +=
Vector& operator--(Vector& other); // перегрузка оператора --
Vector& operator*=(Vector& other); // перегрузка оператора *=

friend Matrix operator+(Vector& x, Matrix& y); // перегрузка оператора +
friend Matrix operator-(Vector& x, Matrix& y); // перегрузка оператора -
friend Matrix operator+=(Matrix& y, Vector& x); // перегрузка оператора +=
friend Matrix operator--(Matrix& y, Vector& x); // перегрузка оператора -=
friend Matrix operator*(Vector& x, Matrix& y); // перегрузка оператора *
friend Matrix operator*=(Matrix& y, Vector& x); // перегрузка оператора *=

// перегрузка операцій з цілими числами
Vector& operator=(int x);
Vector& operator+(int x);
Vector& operator%(int x);
Vector& operator*(int x);
Vector& operator+=(int x);
Vector& operator--(int x);
Vector& operator*=(int x);
Vector& operator++();
Vector& operator--();

// перегрузка операцій з дійсними числами
Vector& operator=(float x);
Vector& operator+(float x);

```

```

    Vector& operator/(float x);
    Vector& operator*(float x);
    Vector& operator+=(float x);
    Vector& operator--(float x);
    Vector& operator*=(float x);

private:
    float* value;
    int size;
};

Vector::Vector()
{
    this->size = 0;
    this->value = 0;
}

Vector::Vector(int size)
{
    this->size = size;
    this->value = new float[size];
}

Vector::Vector(int size, float* value)
{
    this->size = size;
    this->value = new float[size];
    for (int i = 0; i < size; i++) {
        this->value[i] = value[i];
    }
}

Vector::Vector(Vector& other)
{
    this->size = other.size;
    this->value = other.value;
}

Vector::~~Vector()
{
}

```

```

void Vector::Set(int i, float value)
{
    this->value[i] = value;
}

int Vector::Get()
{
    return size;
}

void Vector::Print()
{
    std::cout << "Our vector x: \n";
    for (int i = 0; i < size; i++)
    {
        std::cout << i << " element - " << value[i] << "\n";
    }
}

float Vector::Geter(int i)
{
    return value[i];
}

float Vector::operator[](int i)
{
    return value[i];
}

float& Vector::operator()(int i)
{
    return value[i];
}

Vector operator+(Vector& x, Vector& y)
{
    Vector result(x.size);
    for (int i = 0; i < x.size; i++)
    {
        result.Set(i, x.value[i] + y.value[i]);
    }
    return result;
}

```

```

Vector operator-(Vector& x, Vector& y)
{
    Vector result(x.size);
    for (int i = 0; i < x.size; i++)
    {
        result.Set(i, x.value[i] - y.value[i]);
    }
    return result;
}

```

```

Vector operator*(Vector& x, Vector& y)
{
    Vector result(x.size);
    for (int i = 0; i < x.size; i++)
    {
        result.Set(i, x.value[i] * y.value[i]);
    }
    return result;
}

```

```

Vector& Vector::operator=(Vector& other)
{
    if (this != &other) {
        delete[] value;
        size = other.size;
        value = new float[size];
        for (int i = 0; i < size; i++) {
            value[i] = other.value[i];
        }
    }
    return *this;
}

```

```

Vector& Vector::operator+=(Vector& other)
{
    if (size != other.size) {
        std::cout << "Errorr.Invalid size";
    }
}

```

```

        for (int i = 0; i < size; i++) {
            value[i] += other.value[i];
        }
        return *this;
    }

Vector& Vector::operator--(Vector& other)
{
    if (size != other.size) {
        std::cout << "Errorr.Invalid size";
    }
    for (int i = 0; i < size; i++) {
        value[i] -= other.value[i];
    }
    return *this;
}

Vector& Vector::operator*=(Vector& other)
{
    if (size != other.size) {
        std::cout << "Errorr.Invalid size";
    }
    for (int i = 0; i < size; i++) {
        value[i] *= other.value[i];
    }
    return *this;
}

Vector& Vector::operator=(int x)
{
    for (int i = 0; i < size; i++) {
        value[i] = x;
    }
    return *this;
}

Vector& Vector::operator+(int x)
{
    for (int i = 0; i < size; i++) {
        value[i] += x;
    }
}

```

```

    }

    return *this;
}

Vector& Vector::operator%(int x)
{
    for (int i = 0; i < size; i++) {
        value[i] = fmod(value[i], x);
    }

    return *this;
}

Vector& Vector::operator*(int x)
{
    for (int i = 0; i < size; i++) {
        value[i] *= x;
    }

    return *this;
}

Vector& Vector::operator+=(int x)
{
    for (int i = 0; i < size; i++) {
        value[i] += x;
    }

    return *this;
}

Vector& Vector::operator-=(int x)
{
    for (int i = 0; i < size; i++) {
        value[i] -= x;
    }

    return *this;
}

Vector& Vector::operator*=(int x)
{
    for (int i = 0; i < size; i++) {
        value[i] *= x;
    }

    return *this;
}

```



```

}

Vector& Vector::operator++()
{
    for (int i = 0; i < size; i++) {
        value[i] ++;
    }
    return *this;
}

Vector& Vector::operator--()
{
    for (int i = 0; i < size; i++) {
        value[i]--;
    }
    return *this;
}


Vector& Vector::operator=(float x)
{
    for (int i = 0; i < size; i++) {
        value[i] = x;
    }
    return *this;
}

Vector& Vector::operator+(float x)
{
    for (int i = 0; i < size; i++) {
        value[i] += x;
    }
    return *this;
}

Vector& Vector::operator/(float x)
{
    for (int i = 0; i < size; i++) {
        value[i] /= x;
    }
    return *this;
}

```

```

}

Vector& Vector::operator*(float x)
{
    for (int i = 0; i < size; i++) {
        value[i] *= x;
    }
    return *this;
}

Vector& Vector::operator+=(float x)
{
    for (int i = 0; i < size; i++) {
        value[i] += x;
    }
    return *this;
}

Vector& Vector::operator-=(float x)
{
    for (int i = 0; i < size; i++) {
        value[i] -= x;
    }
    return *this;
}

Vector& Vector::operator*=(float x)
{
    for (int i = 0; i < size; i++) {
        value[i] *= x;
    }
    return *this;
}

}

class Matrix
{
public:
    Matrix();                // конструктор за замовчуванням
    Matrix(int row);        // конструктор з 1 параметром

```

```

Matrix(int row, int coloms); // конструктор з 2 параметром
Matrix(Matrix& other);      // конструктор копіювання
~Matrix();

void Print();              // друк
void Set(int i, float matrix); //задати i-тому елементу значення
int Get();                 //отримуємо row*coloms(тобто розмір)
float operator[](int i);   // перегрузка оператора []
float& operator() (int i, int j); // перегрузка оператора ()
friend Matrix operator+(Vector& x, Matrix& y); // перегрузка оператора +
Matrix& operator=(Vector& x); // перегрузка оператора =
friend Matrix operator-(Vector& x, Matrix& y); // перегрузка оператора -
friend Matrix operator+=(Matrix& y, Vector& x); // перегрузка оператора +=
friend Matrix operator-=(Matrix& y, Vector& x); // перегрузка оператора -=
friend Matrix operator*(Vector& x, Matrix& y); // перегрузка оператора *
friend Matrix operator*=(Matrix& y, Vector& x); // перегрузка оператора *=

private:
    int row, coloms;
    float* matrix;
};

Matrix::Matrix()
{
    this->row = 1;
    this->coloms = 1;
    this->matrix = new float [row];
    this->matrix[0] = 0;
}

Matrix::Matrix(int row)
{
    this->row = row;
    this->coloms = 1;
    this->matrix = new float[row];
}

Matrix::Matrix(int row, int coloms)
{

```

```

        this->row = row;
        this->coloms = coloms;
        this->matrix = new float[row * coloms];
    }
Matrix::Matrix(Matrix& other)
{
    this->row = other.row;
    this->coloms = other.coloms;
    this->matrix = new float[other.row * other.coloms];
    for (int i = 0; i < row * coloms; i++) {
        matrix[i] = other.matrix[i];
    }
}
Matrix::~Matrix()
{
    delete[] matrix;
}
void Matrix::Print()
{
    std::cout << "Our vector y: \n";
    for (int i = 0; i < Get(); i++)
    {
        std::cout << i << " element - " << matrix[i] << "\n";
    }
}
void Matrix::Set(int i, float matrix)
{
    this->matrix[i] = matrix;
}
int Matrix::Get()
{
    return row*coloms;
}
float Matrix::operator[](int i)
{
    return matrix[i];
}

```

```

float& Matrix::operator() (int i, int j) { // метод отримання значення елемента
    return matrix[i*coloms+j];
}

Matrix& Matrix::operator=(Vector& x)
{
    Matrix result(x.Get());
    for (int i = 0; i < x.Get(); i++)
    {
        result.Set(i, x.Geter(i));
        std::cout << "Result - " << result.matrix[i] << "\n";
    }
    return result;
}

Matrix operator+(Vector& x, Matrix& y)
{
    Matrix result(y.Get());
    for (int i = 0; i < y.Get(); i++)
    {
        result.Set(i, x.value[i] + y.matrix[i]);
        std::cout << "Result - " << result.matrix[i] << "\n";
    }
    return result;
}

Matrix operator-(Vector& x, Matrix& y)
{
    Matrix result(y.Get());
    for (int i = 0; i < y.Get(); i++)
    {
        result.Set(i, x.value[i] - y.matrix[i]);
        std::cout << "Result - " << result.matrix[i] << "\n";
    }
    return result;
}

Matrix operator+=(Matrix& y, Vector& x)
{
    if (y.Get() !=x.Get() ) {
        std::cout << "Errorr.Invalid size";
    }
}

```

```

    }

    for (int i = 0; i < y.Get(); i++) {
        y.matrix[i] += x.value[i];
    }

    return y;
}

Matrix operator+=(Matrix& y, Vector& x)
{
    if (y.Get() != x.Get()) {
        std::cout << "Errorr.Invalid size";
    }

    for (int i = 0; i < y.Get(); i++) {
        y.matrix[i] += x.value[i];
    }

    return y;
}

Matrix operator*(Vector& x, Matrix& y)
{
    Matrix result(y.Get());
    for (int i = 0; i < y.Get(); i++)
    {
        result.Set(i, x.value[i] * y.matrix[i]);
        std::cout << "Result - " << result.matrix[i] << "\n";
    }

    return result;
}

Matrix operator*=(Matrix& y, Vector& x)
{
    if (y.Get() != x.Get()) {
        std::cout << "Errorr.Invalid size";
    }

    for (int i = 0; i < y.Get(); i++) {
        y.matrix[i] *= x.value[i];
    }

    return y;
}

```

```

int main()
{
    Matrix y(2, 2);
    y(0, 0) = 1;
    y(0, 1) = 2;
    y(1, 0) = 3;
    y(1, 1) = 4;
    y.Print();
    Vector x(4);
    x(0) = 1;
    x(1) = 2;
    x(2) = 3;
    x(3) = 4;
    x.Print();
    std::cout << "y*x: ";
    y*=x;
    y.Print();
    std::cout << "y=x: \n";
    y=x;
}

```

### 3.4 Висновок

#### Завдання 1



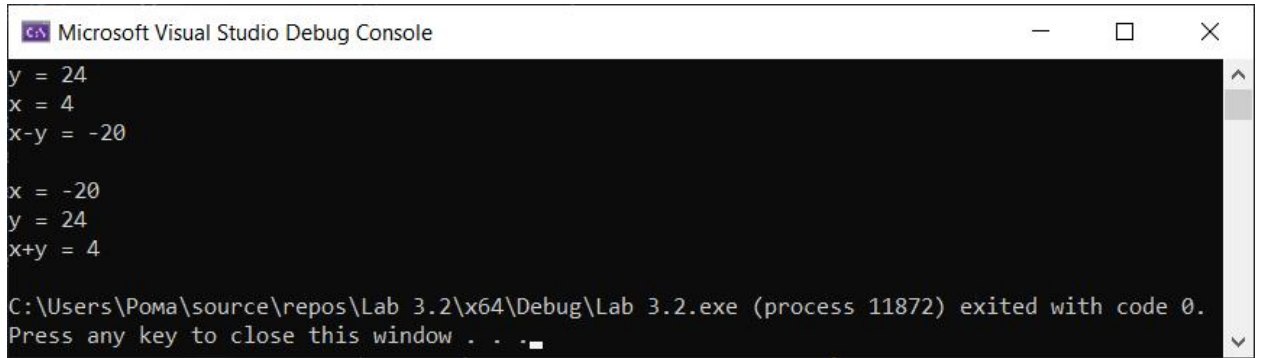
```

Microsoft Visual Studio Debug Console
x = 5
--x
x = 4
++x
x = 5
C:\Users\Роман\source\repos\Lab 3.1\x64\Debug\Lab 3.1.exe (process 18660) exited with code 0.
Press any key to close this window . . .

```

Клас працює вірно. Програма може виконувати унарні операції(інкремент і декремент).

## Завдання 2



```
Microsoft Visual Studio Debug Console

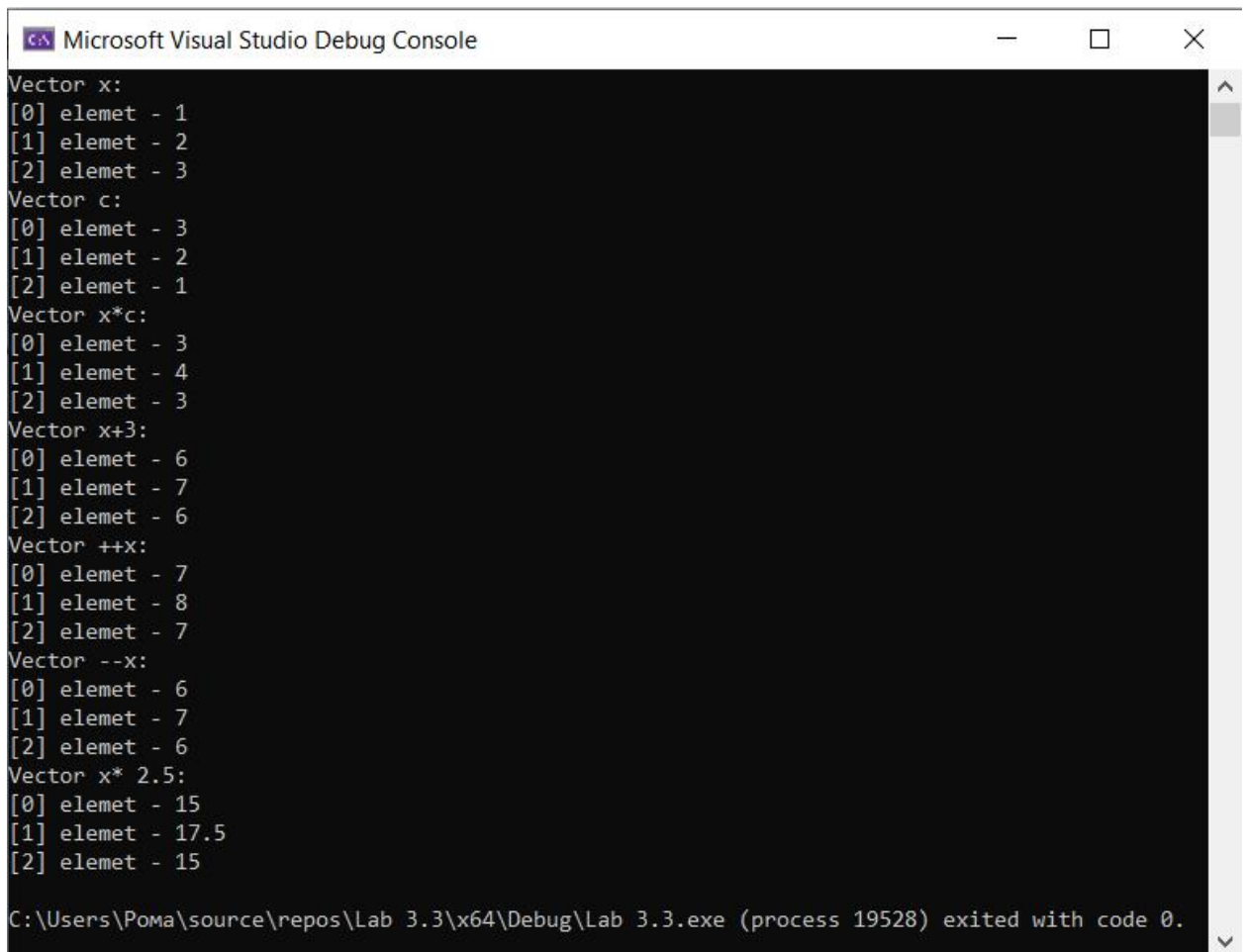
y = 24
x = 4
x-y = -20

x = -20
y = 24
x+y = 4

C:\Users\Poma\source\repos\Lab 3.2\x64\Debug\Lab 3.2.exe (process 11872) exited with code 0.
Press any key to close this window . . .
```

Клас працює вірно. Програма може виконувати бінарні операції(плюс, мінус).

## Завдання 3



```
Microsoft Visual Studio Debug Console

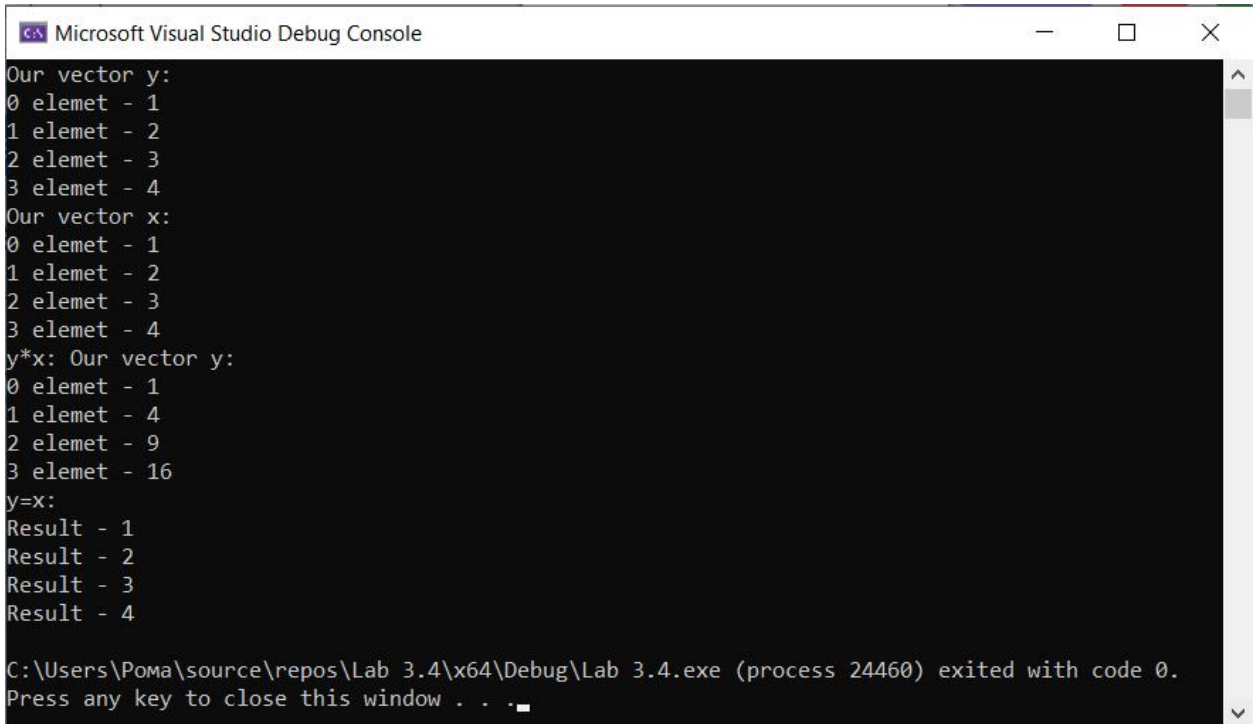
Vector x:
[0] element - 1
[1] element - 2
[2] element - 3
Vector c:
[0] element - 3
[1] element - 2
[2] element - 1
Vector x*c:
[0] element - 3
[1] element - 4
[2] element - 3
Vector x+3:
[0] element - 6
[1] element - 7
[2] element - 6
Vector ++x:
[0] element - 7
[1] element - 8
[2] element - 7
Vector --x:
[0] element - 6
[1] element - 7
[2] element - 6
Vector x* 2.5:
[0] element - 15
[1] element - 17.5
[2] element - 15

C:\Users\Poma\source\repos\Lab 3.3\x64\Debug\Lab 3.3.exe (process 19528) exited with code 0.
```

Клас працює вірно. Програма може виконувати арифметичні операції з цілими та дійсними числами вектора. Також має унарні операції(інкремент і декремент). Дозволяє множити вектор на дійсне число.



## Завдання 4



```
Microsoft Visual Studio Debug Console

Our vector y:
0 element - 1
1 element - 2
2 element - 3
3 element - 4
Our vector x:
0 element - 1
1 element - 2
2 element - 3
3 element - 4
y*x: Our vector y:
0 element - 1
1 element - 4
2 element - 9
3 element - 16
y=x:
Result - 1
Result - 2
Result - 3
Result - 4

C:\Users\Роман\source\repos\Lab 3.4\x64\Debug\Lab 3.4.exe (process 24460) exited with code 0.
Press any key to close this window . . .
```

Класи працюють вірно. Програма має такий самий функціонал як у завданні 3, але додані арифметичні операції між класом Vector і класом Matrix.

### IV. Контрольні запитання

- 1) Клас - абстрактний тип даних, що визначається користувачем і являє собою модель реального об'єкта у вигляді даних та функцій для роботи з ними.
- 2) Конструктор класу – це спеціальний метод-функція класу. Конструктор викликається при створенні об'єкту класу. Як правило, конструктор використовується для виділення пам'яті для об'єкту класу або для початкової ініціалізації внутрішніх даних класу.
- 3) Деструктор – це спеціальний метод, що викликається при видаленні об'єкту. Як правило, деструктор використовується для звільнення пам'яті, динамічно виділеної під внутрішні дані класу.
- 4) Перевантаження функції – це оголошення функції з тим же іменем декілька разів. Таким чином, в деякій області видимості

ім'я “перевантажена” функція оголошується декілька разів. Щоб компілятор міг відрізнати “перевантажені” функції, ці функції повинні відрізнатися між собою списком вхідних параметрів.

- 5) Дружня функція — це функція, яка має доступ до закритих членів класу, наче вона сама є членом цього класу.
- 6) Перевантаження оператору – спосіб оголошення та реалізації оператору таким чином, що він обробляє об'єкти конкретних класів або виконує деякі інші дії. При перевантаженні оператору в класі викликається відповідна операторна функція, яка виконує дії, що стосуються даного класу.