# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**ЗВІТ**

**Про виконання лабораторної роботи №2**

**«Перевизначення операторів. Використання дружніх функцій»**

**з дисципліни «Об’єктно-орієнтоване програмування»**

Виконав:

слухач групи ПЗС-11

Гринчук Тарас

Прийняла:

доц. Кортєєва Т.О.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2014

**Тема роботи**: Перевизначення операторів. Використання дружніх функцій.

## 1. Завдання

Написати програму алгоритмічною мовою С++ згідно із завданням, отриманим відвикладача за табл. 1. Реалізувати заданий класовий тип із заданими методами, використовуючи механізм перевантаження функцій та операцій, а також значення параметрів за замовчуванням. Написати фрагмент програми з використанням класового типу та заданих методів.

**Варіант 5.**

**Класовий тип:** Динамічний двомірний масив **\*\*p**

**Методи класу:** Конструктор повинен дозволяти створювати об'єкти з ініціалізацією і без неї. Визначити конструктор копій. Перевантажити операції -, =, виводу <<, вводу >> для об'єктів класу, а також дружні операції <тип>\*, +=.

## 2. Блок-схеми алгоритмів функцій програми



Рис. 2.1. Перезавантаження оператора «=» Рис. 2.2. Унарний «-»

Блок-схема перезавантаження функції віднімання двох матриць ***const Matrix operator-(const Matrix& left, const Matrix& right)****,* зображена на рис. 2.3. На рис. 2.4. зображена функція множення двох матриць ***const Matrix operator\*(const Matrix& left, const Matrix& right).***



Рис. 2.3. Бінарний "-" (дві матриці) Рис. 2.4. Множення елементів двох матриць

Блок-схема перезавантаження функції віднімання матриці та числа ***const Matrix operator-(const Matrix& left, int right)****,* зображена на рис. 2.5. На рис. 2.6. зображена функція множення матриці на число ***const Matrix operator\*(const Matrix& left, int right).***



Рис. 2.5. Бінарний "-" (матриця - число) Рис. 2.6. Множення матриці на число

Блок-схема перезавантаження оператора «+=» для двох матриць ***Matrix& operator+=(Matrix& left, const Matrix& right)****,* зображена на рис. 2.7. На рис. 2.8. зображена функція перезавантаження оператора «+=» для матриці та числа ***Matrix& operator+=(Matrix& left, int right).***



Рис. 2.7. Оператор матриця += матриця Рис. 2.8. Оператор матриця += число

Блок-схема перезавантаження оператора виводу матриці на екран «<<» ***ostream& operator<<(ostream& output, const Matrix& matr)****,* зображена на рис. 2.9. На рис. 2.10. зображена функція перезавантаження оператора вводу матриці з клавіатури «>>» ***istream& operator>>(istream& input, const Matrix& matr).***



Рис. 2.9. Оператор виводу матриці «<<» Рис. 2.10. Оператор вводу матриці «>>»

**3. Текст програми на мові програмування С++**

#include <iostream>

#include <conio.h>

#include <math.h>

using namespace std;

class Matrix {

public:

int\* data;

int size;

//конструктор з ініціалізацією початкових данних

//першим числом повиинно бути розмір матриці зі знаком мінус

//в протилежному випадку заповнення масиву початковими значеннями

//не відбудеться

Matrix(int matSize, ...) {

size = matSize;

if(size<0) size = -size;

data = new int[size\*size];

if(matSize < 0) {

int \*p = &matSize;

for(int i = 0; i < size\*size; i++) {

p++;

\*(data+i) = \*(p);

}

}

}

//конструктор без параметрів

//створюється масив 2\*2

Matrix() {

size = 2;

data = new int[4];

}

//конструктор копій

Matrix(const Matrix &p) {

size = p.size;

data = new int[size\*size];

int\* k = p.data;

for(int i = 0; i < size\*size; i++)

\*(data+i) = \*(k+i);

}

//деструктор

~Matrix() {

delete[] data;

}

//перезавантаження оператора []

int\* operator[](int row) {

return data+row\*size;

}

//перезавантаження оператора =

Matrix& operator=(const Matrix& right) {

//перевірка на самоприсвоєння

if (this == &right) {

return \*this;

}

size = right.size;

int\* k = right.data;

for(int i = 0; i < size\*size; i++)

\*(data+i) = \*(k+i);

return \*this;

}

//Дружні функції

//унарний "-"

friend const Matrix operator-(const Matrix& right);

//бінарний "-" (матриця3 = матриця1 - матриця2)

friend const Matrix operator-(const Matrix& left, const Matrix& right);

//бінарний "-" (матриця3 = матриця1 - число)

friend const Matrix operator-(const Matrix& left, int right);

//матриця1 += матриця2;

friend Matrix& operator+=(Matrix& left, const Matrix& right);

//матриця1 += число;

friend Matrix& operator+=(Matrix& left, int right);

//"\*" (матриця3 = матриця1 \* матриця2)

friend const Matrix operator\*(const Matrix& left, const Matrix& right);

//"\*" (матриця2 = матриця1 \* число)

friend const Matrix operator\*(const Matrix& left, int right);

//перезавантаження виводу масиву на екран

friend ostream& operator<<(ostream&, const Matrix&);

//перезавантаження вводу масиву з клавіатури

friend istream& operator>>(istream&, const Matrix&);

};

//унарний "-"

const Matrix operator-(const Matrix& right) {

Matrix left(right.size);

int\* k = right.data;

for(int i = 0; i < right.size\*right.size; i++)

\*(left.data+i) = -\*(k+i);

return left;

}

//бінарний "-" (матриця3 = матриця1 - матриця2)

const Matrix operator-(const Matrix& left, const Matrix& right) {

Matrix result(left.size);

int\* k = left.data;

int\* m = right.data;

for(int i = 0; i < left.size\*left.size; i++)

\*(result.data+i) = \*(k+i) - \*(m+i);

return result;

}

//бінарний "-" (матриця3 = матриця1 - число)

const Matrix operator-(const Matrix& left, int right) {

Matrix result(left.size);

int\* k = left.data;

for(int i = 0; i < left.size\*left.size; i++)

\*(result.data+i) = \*(k+i) - right;

return result;

}

//"\*" (матриця3 = матриця1 \* матриця2)

const Matrix operator\*(const Matrix& left, const Matrix& right) {

Matrix result(left.size);

int\* k = left.data;

int\* m = right.data;

for(int i = 0; i < left.size\*left.size; i++)

\*(result.data+i) = (\*(k+i)) \* (\*(m+i));

return result;

}

//"\*" (матриця2 = матриця1 \* число)

const Matrix operator\*(const Matrix& left, int right) {

Matrix result(left.size);

int\* k = left.data;

for(int i = 0; i < left.size\*left.size; i++)

\*(result.data+i) = (\*(k+i)) \* right;

return result;

}

//матриця1 += матриця2;

Matrix& operator+=(Matrix& left, const Matrix& right) {

int\* m = right.data;

for(int i = 0; i < left.size\*left.size; i++)

\*(left.data+i) += \*(m+i);

return left;

}

//матриця1 += число;

Matrix& operator+=(Matrix& left, int right) {

for(int i = 0; i < left.size\*left.size; i++)

\*(left.data+i) += right;

return left;

}

//перезавантаження виводу масиву на екран

ostream& operator<<(ostream& output, const Matrix& matr) {

int\* m = matr.data;

for(int i = 0, k = 0; i < matr.size; i++) {

for(int j = 0; j < matr.size; j++, k++) {

output<<(\*(m+k))<<"\t";

}

output<<"\n";

}

return output;

}

//перезавантаження вводу масиву з клавіатури

istream& operator>>(istream& input, const Matrix& matr) {

int\* m = matr.data;

for(int i = 0, k = 0; i < matr.size; i++)

for(int j = 0; j < matr.size; j++, k++) {

cout<<"["<<i+1<<"]["<<j+1<<"]: ";

input>>(\*(m+k));

}

return input;

}

void main() {

const int N = 2;

//з початковою ініціалізацією

Matrix a(-2,1,2,3,4);

cout<<"a = {1,2,3,4}:\n"<<a;

//без параметрів - по замовчуванню 2\*2

Matrix b;

cout<<"\nInput b:\n";

cin>>b;

cout<<"\nb:\n"<<b;

//з вказаним розміром, без ініціалізації

Matrix c(N);

c = -b;

cout<<"\nc = -b\n"<<c;

//копіюванням

Matrix d(c);

cout<<"\nMatrix d(c)\n"<<d;

d = b - 5;

cout<<"\nd = b - 5\n"<<d;

b = d - a;

cout<<"\nb = d - a\n"<<b;

c = a \* 2;

cout<<"\nc = a \* 2\n"<<c;

a = b \* c;

cout<<"\na = b \* c\n"<<a;

a += 3;

cout<<"\na += 3\n"<<a;

a += c;

cout<<"\na += c\n"<<a;

\_getch();

}

**4. Результат виконання програми**

Запустимо програму на виконання (рис. 4.1). Введемо значення елементів вхідної матриці. Отримаємо результати виконання операцій над матрицями на дисплеї.

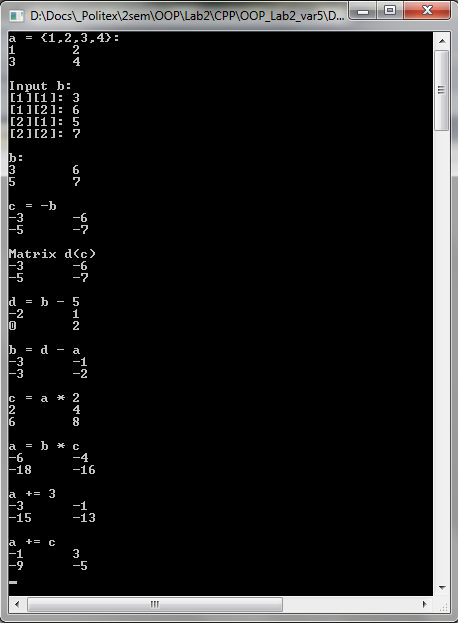


Рис. 4.1. Результат виконання програми

## ВИСНОВКИ

На даній лабораторній роботі я навчився створювати класи, перезавантажувати оператори та функції класів, реалізовувати конструктори класів з параметрами та без. Працювати з дружніми функціями класу.