Министерство образования и науки РФ

Севастопольский государственный университет

Кафедра информатики и управления в технических системах

ОТЧЕТ

по лабораторной работе №6

ИССЛЕДОВАНИЕ ПЕРЕГРУЗКИ ОПЕРАТОРОВ

по дисциплине «Объектно-ориентированное программирование»

Выполнил:

Студент группы ИВТ/б 22-о

Черняев Н.Г.

Проверил:

Тимофеев И.С.

г. Севастополь 2018

**Цель работы:**

Исследование назначения и средств создания перегруженных операторов при написании объектно-ориентированных программ.

**Вариант №10**

Создать класс матрица целых чисел.

- описать конструкторы и деструктор (по необходимости);

- переопределить оператор вывода в поток <<;

- переопределить оператор ввода из потока >>;

Перегрузить операторы:

1) -- как унарный метод класса, меняющий знак для отрицательных элементов матрицы;

2) ++ как унарную дружественную функцию, заменяющую нулевые элементы матрицы на максимальный элемент;

3) + как бинарный метод класса, суммирующий поэлементно две матрицы;

4) >= как бинарную дружественную функцию, поэлементно сравнивающую матрицы между собой

**Текст программы:**

#include "stdafx.h"

#include <iostream>

#include <cstring>

#include <sstream>

#include <math.h>

using namespace std;

class MatrixInt {

private:

int x;

int y;

int m[20][20];

public:

MatrixInt() {

int i, j;

x = y = 2;

for (i = 0; i < x; i++) {

for (j = 0; j < y; j++) {

m[i][j] = 0;

}

}

};

MatrixInt(int x, int y) {

int i, j;

for (i = 0; i < x; i++) {

for (j = 0; j < y; j++) {

m[i][j] = 0;

}

}

};

const MatrixInt& operator--() {

int i, j;

for (i = 0; i < x; i++) {

for (j = 0; j < y; j++) {

m[i][j] = abs(m[i][j]);

}

}

return \*this;

};

const MatrixInt& operator+(MatrixInt& q) {

int i, j;

MatrixInt z;

for (i = 0; i < this->x; i++) {

for (j = 0; j < this->y; j++) {

z.m[i][j] = this->m[i][j] + q.m[i][j];

}

}

return z;

};

friend const ostream& operator<<(ostream& out, const MatrixInt& q);

friend const istream& operator>>(istream& in, MatrixInt& q);

friend const MatrixInt& operator++(MatrixInt& q);

friend const bool& operator>=(MatrixInt& q1, MatrixInt& q2);

};

const ostream& operator<<(ostream& out, const MatrixInt& q) {

int i, j;

for (i = 0; i < q.x; i++) {

for (j = 0; j < q.y; j++) {

out << q.m[i][j] << " ";

}

out << "\n";

}

return out;

}

const istream& operator>>(istream& in, MatrixInt& q) {

int i, j;

for (i = 0; i < q.x; i++) {

cout << "Enter " << i + 1 << " line" << endl;

for (j = 0; j < q.y; j++) {

cout << "Enter " << j + 1 << " element " << i + 1 << " stroki" << endl;

in >> q.m[i][j];

}

}

return in;

}

const MatrixInt& operator++(MatrixInt& q) {

int i, j, max;

max = q.m[0][0];

for (i = 0; i < q.x; i++) {

for (j = 0; j < q.y; j++) {

if (q.m[i][j] > max) {

max = q.m[i][j];

}

}

}

for (i = 0; i < q.x; i++) {

for (j = 0; j < q.y; j++) {

if (q.m[i][j] == 0) {

q.m[i][j] = max;

}

}

}

return q;

};

const bool& operator>=(MatrixInt& q1, MatrixInt& q2) {

int i, j;

int k1 = 0;

int k2 = 0;

for (i = 0; i < q1.x; i++) {

for (j = 0; j < q1.y; j++) {

if (q1.m[i][j] >= q2.m[i][j]) {

k1++;

}

else {

k2++;

}

}

}

if (k1 >= k2) {

return true;

}

else {

return false;

}

};

int main() {

MatrixInt ob1, ob2, ob3;

cout << "1st Matr:" << endl;

cin >> ob1;

cout << "2nd Matr:" << endl;

cin >> ob2;

cout << "Matrix 1:" << endl << ob1;

cout << "Matrix 2:" << endl << ob2;

--ob1;

--ob2;

cout << "--Matrix 1:" << endl << ob1;

cout << "--Matrix 2:" << endl << ob2;

++ob1;

++ob2;

cout << "++Matrix 1:" << endl << ob1;

cout << "++Matrix 2:" << endl << ob2;

ob3 = ob1 + ob2;

cout << "Sum Matr1+Matr2:" << endl << ob3;

cout << "Comparision Matr1 and Matr2:" << endl;

if (ob1 >= ob2) {

cout << "Matr1>=Matr2" << endl;

}

else {

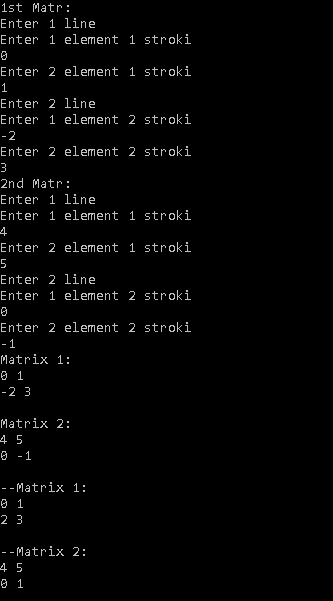
cout << "Matr1<Matr2" << endl;

};

return 0;

}

**Тестовые примеры:**



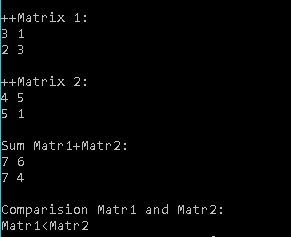


рис.1 – пример работы программы

**Вывод:**

В ходе данной лабораторной работы были исследованы назначения и средства создания перегруженных операторов при написании объектно-ориентированных программ.