**ЛАБОРАТОРНАЯ РАБОТА № 5**

**«ИССЛЕДОВАНИЕ ПЕРЕГРУЗКИ ОПЕРАТОРОВ»**

**Цель работы**: Исследование назначения и средств создания перегруженных операторов при написании объектно-ориентированных программ.

**Вариант задания**

Для заданного по варианту класса выполнить следующие действия:

- описать конструкторы и деструктор (по необходимости);

- переопределить оператор вывода в поток <<;

- переопределить оператор ввода из потока >>;

- переопределить заданные по варианту операторы;

- предусмотреть обработку ошибок.

Создать два объекта заданного по варианту класса и на их примере продемонстрировать корректную работу всех перегруженных операторов.

Вариант 11

Создать класс координаты Coords (содержит две пары чисел (x1, y1) и (x2, y2)).

Перегрузить операторы:

1) ! как унарный метод класса, проверяющий на совпадение пары координат (x1

на равенство с x2 и аналогично y1 и y2 — одновременно);

2) ++ как унарную дружественную функцию, увеличивающую первую пару координат на случайное число;

3) \* как бинарный метод класса, умножающий координаты на заданное число;

4) != как бинарную дружественную функцию сравнения координат двух объектов.

**2. Код программы на языке C++**

#include <iostream>

using namespace std;

class Coords

{

public:

Coords();

Coords(int x1, int x2, int y1, int y2);

~Coords();

bool operator !();

friend Coords& operator ++(Coords& M);

friend istream& operator >>(istream& in, Coords& M);

friend ostream& operator <<(ostream& out, Coords& M);

friend bool operator !=(const Coords& M1, const Coords& M2);

Coords operator \*(const int mn) {

x1 \*= mn;

x2 \*= mn;

y1 \*= mn;

y2 \*= mn;

return \*this;

}

private:

int x1, x2, y1, y2;

};

Coords::Coords()

{

x1 = x2 = y1 = y2 = 0;

}

Coords::Coords(int x1,int x2,int y1,int y2)

{

this->x1 = x1;

this->x2 = x2;

this->y1 = y1;

this->y2 = y2;

}

Coords::~Coords()

{

}

bool Coords::operator!() {

if (x1 == x2 && y1 == y2) return true;

return false;

}

Coords& operator ++(Coords& M) {

M.x1 += rand();

M.y1 += rand();

return M;

}

bool operator !=(const Coords& M1, const Coords& M2) {

if (M1.x1 == M2.x1 && M1.x2 == M2.x2 && M1.y1 == M2.y1 && M1.y2 == M2.y2) return false;

return true;

}

istream& operator >>(istream& in, Coords& M) {

cout << "Введите x1: ";

while (true) {

if (in >> M.x1) {

break;

}

else {

in.clear();

in.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

cout << "Error. Не число" << endl;

cout << "Введите x1 ";

}

}

cout << "Введите y1: ";

while (true) {

if (in >> M.y1) {

break;

}

else {

in.clear();

in.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

cout << "Error. Не число" << endl;

cout << "Введите y1 ";

}

}

cout << "Введите x2: ";

while (true) {

if (in >> M.y1) {

break;

}

else {

in.clear();

in.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

cout << "Error. Не число" << endl;

cout << "Введите x2 ";

}

}

cout << "Введите y2: ";

while (true) {

if (in >> M.y2) {

break;

}

else {

in.clear();

in.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

cout << "Error. Не число" << endl;

cout << "Введите y2 ";

}

}

return in;

}

ostream& operator <<(ostream& out, Coords& M) {

out << "x1=" << M.x1 << " y1=" << M.y1 << " x2=" << M.x2 << " y2=" << M.y2 << std::endl;

return out;

}

int main()

{

system("chcp 1251");

Coords c1 = Coords(1, 2, 4, 3);

Coords c2 = Coords(2, 2, 2, 2);

cout << "Введите первый объект: " << endl;

cin >> c1;

cout << "Введите второй объект: " << endl;

cin >> c2;

cout << "Первый класс содержит" << endl;

cout << c1;

cout << "Второй класс содержит" << endl;

cout << c2;

cout << "Результат унарной операции ! в первом классе" << endl;

cout << !c1<<endl;

cout << "Результат унарной операции ! в втором классе" << endl;

cout << !c2 << endl;

cout << "Результат унарной дружественной функции ++ в первом классе" << endl;

++c1;

cout << c1;

cout << "Результат бинарного метода класса \* в втором классе" << endl;

c2\* 10;

cout << c2;

cout << "Результат бинарной дружественной функции != двух объектов не равных друг другу" << endl;

cout << (c1 != c2) << endl;

cout << "Результат бинарной дружественной функции != двух объектов равных друг другу" << endl;

Coords c3 = Coords(1, 1, 4, 4); Coords c4 = Coords(1, 1, 4, 4);

cout << (c3 != c4) << endl;

}

**3. Тестирование и отладка**

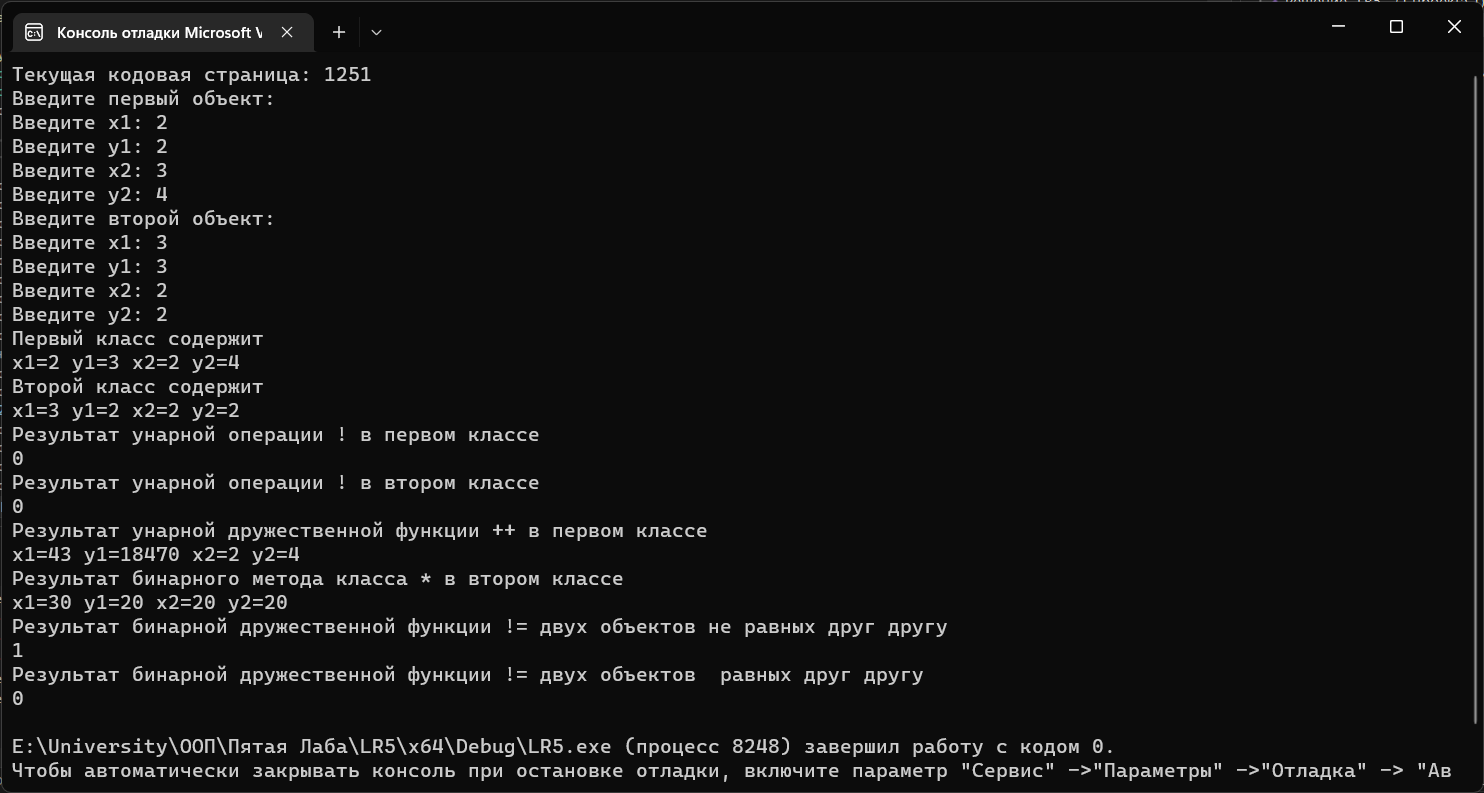
Для тестирования данной программы, используя перегруженный оператор ввода, вводим два объекта Cords. После, был выполнен вывод значений, хранящихся в объектах. После этого были выполнены вызовы перегруженных операторов, результаты полученных изменений так же были выведены на экран. Для тестирования сравнения, были созданы ещё два объекта, схожих между собой.

Рисунок 1 – Результат выполнения программы.

В результате тестирования, видно, что перегруженные операторы выполняют именно те функции, которые были заданы нами.

**Вывод**

При выполнении данной лабораторной работы были получены навыки разработки программ, использующих перегрузки операторов. Полезной особенностью использования перегрузки операторов является повышение читабельности кода, но в случае неправильного использования данной возможности программистом, например, нарушение семантического смысла, код становится более запутанным. Также были повторно закреплены знания создания и работы с конструкторами и деструкторами класса.