# ЛАБОРАТОРНАЯ РАБОТА №5

# ИССЛЕДОВАНИЕ ПЕРЕГРУЗКИ ОПЕРАТОРОВ

Цель работы

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

Задания

1. В ходе самостоятельной подготовки изучить основы работы с перегруженными операторами;
2. Разработать программу на языке С++ согласно варианту 11 методических указаний (Рисунок 1). В программе должны быть описаны конструкторы и деструктор заданного класса, переопределены операторы ввода и вывода в поток, а также операторы, заданные по варианту. Создать два объекта заданного класса и на их примере продемонстрировать корректную работу всех перегруженных операторов;

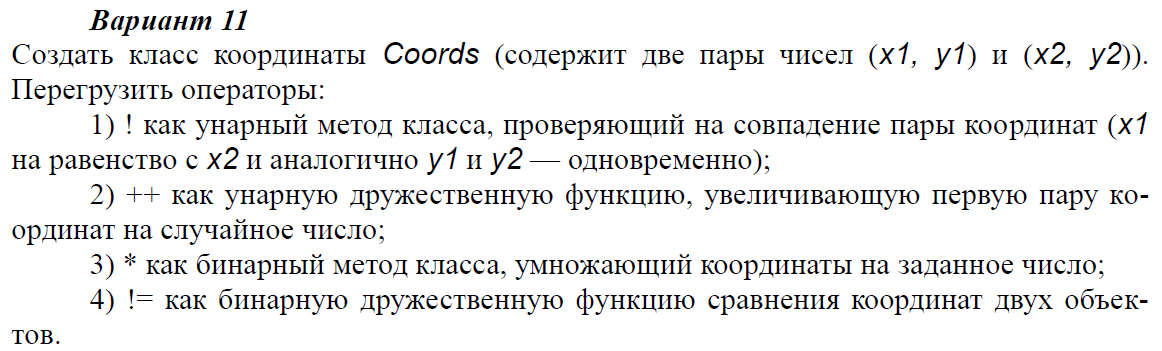


Рисунок 1 – Вариант задания

1. Разработать тестовые примеры и выполнить тестирование программы;
2. Получить результаты работы программы и исследовать её свойства для различных режимов работы, сформулировать выводы;

Текст программы

#include <iostream>

using namespace std;

// Класс координат двух точек

class Coords

{

private:

int x1, y1; // точка 1

int x2, y2; // точка 2

public:

Coords(int c1 = 0, int c2 = 0, int v1 = 0, int v2 = 0) // конструктор

{

x1 = c1;

y1 = c2;

x2 = v1;

y2 = v2;

}

bool operator !() // сравнение двух точек

{

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

return true;

}

Coords& operator \*(int i) // умножение координат на число

{

x1 \*= i;

y1 \*= i;

x2 \*= i;

y2 \*= i;

return \*this;

}

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

friend bool operator !=(Coords& obj1, Coords& obj2);

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

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

};

istream& operator>>(istream& in, Coords& obj) // перегрузка потока ввода

{

in >> obj.x1;

in >> obj.y1;

in >> obj.x2;

in >> obj.y2;

return in;

}

ostream& operator<<(ostream& out, Coords& obj) // перегрузка потока вывода

{

out << '(' << obj.x1;

out << ", " << obj.y1;

out << "), (" << obj.x2;

out << ", " << obj.y2;

out << ")" << endl;

return out;

}

Coords& operator++(Coords& obj) // координаты точки 1 + случ. число

{

int i = rand() % 10 + 1;

obj.x1 += i;

obj.y1 += i;

return obj;

}

bool operator !=(Coords& obj1, Coords& obj2) // сравнение с другой парой точек

{

if ((obj1.x1 == obj2.x1) && (obj1.y1 == obj2.y1) && (obj1.x2 == obj2.x2) && (obj1.y2 == obj2.y2)) return false;

return true;

}

int main()

{

Coords a, b;

cin >> a;

cout << "A: " << a;

cout << "!A : " << !a << endl;

cout << "++A : " << ++a;

cout << "A\*(-1) : " << a \* -1;

cin >> b;

cout << "A != B : " << (a != b);

return 0;

}

Тестовые примеры

Программа была протестирована для значений, при которых перегруженные логические операторы возвращают значение true. В результате программа верно выполнила все операции (Рисунок 2).

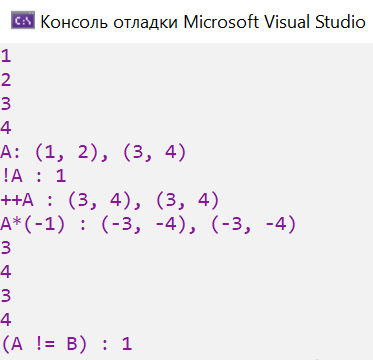


Рисунок 2 – Тестирование программы

Затем похожее тестирование было выполнено при таких значениях, чтобы логические операторы возвращали false. Программа также сработала верно (Рисунок 3).

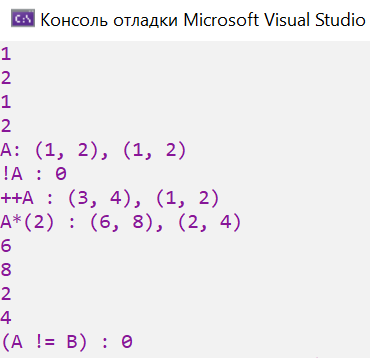


Рисунок 3 – Тестирование программы

Вывод

В ходе работы был изучен механизм перегрузки операторов при объектно-ориентированном программировании на языке C++. Была написана программа, содержащая перегрузку унарных и бинарных операторов в качестве методов и дружественных функций класса.