**5 ЛАБОРАТОРНАЯ РАБОТА №5**

**«ИССЛЕДОВАНИЕ ПЕРЕГРУЗКИ ОПЕРАТОРОВ»**

**5.1 Цель работы**

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

**5.2 Вариант задания – 8**

Разработать программу на языке C++, которая должна содержать:

- описание заданного класса с переопределенными для него операторами;

- два объекта полученного класса;

- демонстрацию работы унарных операторов на одном объекте;

- демонстрацию работы бинарных операторов над созданными объектами.

Ввод и вывод данных должен осуществляться перегруженными операторами работы с потоками.

Требуется создать класс прямоугольник Rectangle (хранит стороны a и b). Перегрузить операторы:

! как унарный метод класса, проверяющий, является ли прямоугольник квадратом;

- как унарную дружественную функцию определения разности между длинами сторон;

== как бинарный метод класса, сравнивающий два прямоугольника на равенство площадей;

+ как бинарную дружественную функцию нахождения общей площади фигур.

**5.3 Ход работы**

5.3.1 В программе были создан класс rectangle полями которого являютя длина и ширина прямоугольника. Созданы конструкторы и деструктор. Осуществлены перегрузки операторов заданых по варианту, а так же осуществлена перегрузка ввода вывода данных. В функции main проведены действия с двумя объектами этого класса, продемонстрирована работа перегрузок функций.

5.3.2 Написана программа на С++ согласно вышеописанного алгоритма.

#include <iostream>

#include <windows.h>

//===================================================================================

// Класс - прямоугольник

class rectangle {

// Длина - a; ширина - b;

int a, b;

public:

//конструктор по умолчанию

rectangle() { a = 15; b = 10; };

//конструктор с параметрами

rectangle(int \_a, int \_b) { a = \_a; b = \_b; };

//деструктор

~rectangle() = default;

// ! унарный метод класса, проверяющий, является ли прямоугольник квадратом

// true - квадрат; false - прямоугольник

bool operator!() { return ((a == b) ? true : false); };

// == бинарный метод класса, сравнивающий два прямоугольника на равенство площадей

// true - равные; false - разные

bool operator==(rectangle ob2) { return (((a\*b) == (ob2.a\*ob2.b)) ? true : false); };

// - унарная дружественная функция определения разности между длинами сторон

friend int operator-(rectangle obj);

// + бинарная дружественную функция нахождения общей площади фигур

friend long operator+(rectangle obj1, rectangle obj2);

// функция перегрузки оператора вывода в поток >>

friend std::ostream & operator<< (std::ostream &out, const rectangle \*obj);

// функция перегрузки оператора ввода из потока >>

friend std::istream & operator>> (std::istream &in, rectangle \*obj);

};

//===================================================================================

// Функция перегрузки оператора вывода в поток

std::ostream & operator<<(std::ostream &out, const rectangle \*obj) {

out << "Длина = " << obj->a << "; ширина = " << obj->b << std::endl << std::endl;

return out;

}

// Функция перегрузки оператора ввода из потока

std::istream & operator>>(std::istream &in, rectangle \*obj) {

std::cout << "Введите длину прямоугольнка: ";

in >> obj->a;

std::cout << "Введите ширину прямоугольнка: ";

in >> obj->b;

std::cout << std::endl;

return in;

}

// - унарная дружественная функция определения разности между длинами сторон

int operator-(rectangle obj) {

return abs(obj.a-obj.b);

}

// + бинарная дружественная функция нахождения общей площади фигур

long operator+(rectangle obj1, rectangle obj2) {

return (obj1.a\*obj1.b + obj2.a\*obj2.b);

}

//===================================================================================

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("color B0");

rectangle \*obj1 = new rectangle();

rectangle \*obj2 = new rectangle();

char ch = '0';

while(1) {

std::cin >> obj1 >> obj2;

std::cout << obj1 << obj2;

//=============================================================================

std::cout << "Унарный метод класса, проверяющий, является ли прямоугольник квадратом" << std::endl;

if (!(\*obj1)) {

std::cout << "Первый прямоугольник является квадратом" << std::endl;

}

else {

std::cout << "Первый прямоугольник не является квадратом" << std::endl;

}

if (obj2->operator!()) {

std::cout << "Второй прямоугольник является квадратом" << std::endl;

}

else {

std::cout << "Второй прямоугольник не является квадратом" << std::endl;

}

std::cout << std::endl;

//=============================================================================

std::cout << "Бинарный метод класса, сравнивающий два прямоугольника на равенство площадей" << std::endl;

// if (\*obj1 == \*obj2)

if (obj1->operator==(\*obj2)) {

std::cout << "площади прямоугольников равны" << std::endl;

}

else {

std::cout << "площади прямоугольников не равны" << std::endl;

}

std::cout << std::endl;

//=============================================================================

std::cout << "Унарная дружественная функция определения разности между длинами сторон" << std::endl;

std::cout << "Разность между длинами строк первого прямоугольника = " << -\*obj1 << std::endl;

std::cout << "Разность между длинами строк второго прямоугольника = " << operator-(\*obj2) << std::endl;

std::cout << std::endl;

//=============================================================================

std::cout << "Бинарная дружественная функция нахождения общей площади фигур" << std::endl;

// \*obj1+\*obj2

std::cout << "Общая площадь прямоугольников = " << operator+(\*obj1, \*obj2) << std::endl;

std::cout << std::endl;

//=============================================================================

std::cout << "Выйти? (y/n) >> ";

std::cin >> ch;

if (ch == 'y') break;

system("cls");

}

delete obj1;

delete obj2;

return 0;

}

5.3.3 Выполнена отладка программы.

Результаты тестирования отображены на рисунке 5.1. На изображении изображено как вводятся данные, а именно длина и ширина двух прямоугольников с помощью перегрузки ввода данных, затем выводятся введенные значения с помощью перегрузки вывода данных. Затем выполняются различные действия с объектами класса и результат выполнения выводится на экран. В конце спрашивается осуществить ли действия ещё раз или выйти. На рисунке 5.2 отображены аналогичные действия, но с другими данными в качестве длины и ширины прямоугольников.

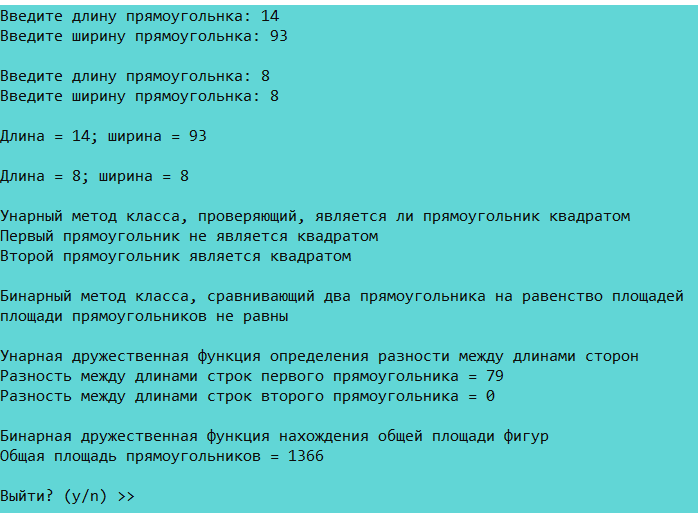


Рисунок 5.1 – Выполнение программы при первых значениях прямоугольника

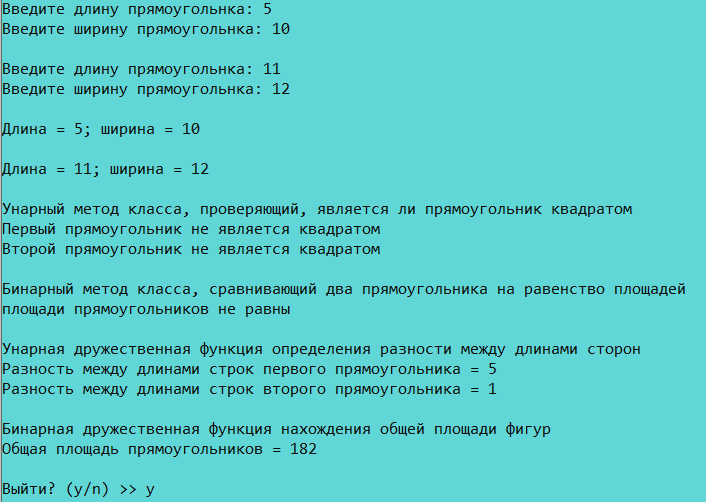


Рисунок 5.2 – Выполнение программы при вторых значениях прямоугольника

Результаты тестирования полностью соответствуют ожиданиям.

**Выводы**

В ходе выполнения данной лабораторной работы были получены навыки разработки программ, использующих перегруженные операторы. Были закреплены навыки разработки и отладки программ, использующих дружественные функции. Полученные во время разработки навыки помогут разрабатывать более сложные программы с использованием классов и объектов, более эффективные по времени выполнения алгоритмы.