МІНІСТЕРТСВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра АПОТ

Звіт

з практичного заняття №2

з дисципліни «Об’єктно-орієнтоване програмування»

# на тему: «Перевантаження ОПЕРАЦІЙ»

Варіант №12

Виконав: Перевірила:

ст. групи КИУКИу-21-1

Сургай Р. В.

Малькова І. А.

Харків 2021

МЕТА РОБОТИ: Засвоєння поняття перевантаження операцій, придбання навичок перевантаження операторів: унарних, бінарних, привласнення та операцій з потоками.

ЗАВДАННЯ:

1. Продумати, які стандартні операції можуть бути визначені для класу, що розроблен в ході практичного заняття №1.

2. Продумати інтерпретацію цих операцій для розроблюваного класу.

3.Поліпшити інтерфейс класу, доповнивши його функціями, що перевантажують ті чи інші операції. Частина перевантажувальних функцій реалізувати як члени-функції, інші - як дружні функції.

4. Для організації доступу і ініціалізації даних класу перевантажити операції "[ ]" і / або "( )". Дані операції можна не перевантажувати для стеків, черг, кілець.

5. Перевантажити операції "==" і "! =" для порівняння об'єктів на рівність і на нерівність. Для сутностей, на множині яких визначено відношення порядку, перевантажити операції ">" і "<" або "> =" і "<=" .

6. Перевантажити операції “>>” і “ <<” для введення-виведення.

ІНТЕРФЕЙС:

#include <iostream>

#include <string>

using namespace std;

// Класс прямоугольник

class Rectangle {

private:

double lenght;

double width;

public:

string name;

double sq;

double per;

Rectangle();

Rectangle(double, double, string);

void setRectangle(double, double, string);

void getRectangle();

double perimetr();

double square();

void print();

// Перегрузка Сложения

Rectangle operator+ (const Rectangle & other) {

Rectangle rez;

rez.lenght = this->lenght + other.lenght;

rez.width = this->width + other.width;

return rez;

}

// Перегрузка вывода

friend ostream& operator<< (ostream & out, const Rectangle & v) {

return out << "Lenght = " << v.lenght <<

"\nWidth = " << v.width << endl;

}

// Перегрузка сравнения БОЛЬШЕ

bool operator> (const Rectangle & other) {

if ((this->lenght \* this->width) > (other.lenght \* other.width)) {

return true;

}

else {

return false;

}

}

// Перегрузка сравнения МЕНЬШЕ

bool operator< (const Rectangle & other) {

if ((this->lenght \* this->width) < (other.lenght \* other.width)) {

return true;

}

else {

return false;

}

}

};

РЕАЛІЗАЦІЯ:

// Конструктор по умолчанию

Rectangle::Rectangle() {

lenght = 1;

width = 1;

name = "None";

}

// Конструктор для ввода длины, ширины и имени прямоугольника

Rectangle::Rectangle(double l, double w, string n) {

lenght = l;

width = w;

name = n;

}

// Функция сет

void Rectangle::setRectangle(double l, double w, string n) {

lenght = (l < 0) ? (l \* (-1)) : l;

width = (w < 0) ? (w \* (-1)) : w;

name = n;

}

// Метод вычисления периметра

double Rectangle::perimetr() {

per = 2 \* (lenght + width);

return per;

}

// Метод вычисления площади

double Rectangle::square() {

sq = lenght \* width;

return sq;

}

// Метод вывода информации

void Rectangle::print() {

cout << "Длина прямоугольника " << name << ": " << lenght << endl;

cout << "Ширина прямоугольника " << name << ": " << width << endl;

cout << "Периметр прямоугольника " << name << " равен: " << per << endl;

cout << "Площадь прямоугольника " << name << " равна: " << sq << "\n \n";

}

// Функция сравнения периметров двух прямоугольников

double maxSquare(Rectangle a, Rectangle b) {

// Перегружаем операторы сравнения

if (a < b) {

cout << "У прямоугольника " << b.name << " площадь БОЛЬШЕ, чем у прямоугольника " << a.name << endl;

}

if(a > b) {

cout << "У прямоугольника " << a.name << " площадь БОЛЬШЕ, чем у прямоугольника " << b.name << endl;

}

else cout << "Площадь прямоугольников: " << a.name << " и " << b.name << " - РАВНА!" << endl;

return 0;

}

ФУНКЦІЯ MAIN():

int main() {

setlocale(LC\_ALL, "rus"); // Русификатор

//Rectangle r1;

//r1.setRectangle(11.4, 5.6, "ABCD");

Rectangle r1 = Rectangle(11.4, 5.6, "ABCD");

r1.perimetr();

r1.square();

Rectangle r2 = Rectangle(12.1, 3.8, "XYZQ");

r2.perimetr();

r2.square();

r1.print();

r2.print();

double maxQ = maxSquare(r1, r2);

// Тестирование перегрузок

cout << endl << "--- Перегрузки ---" << endl;

Rectangle r3;

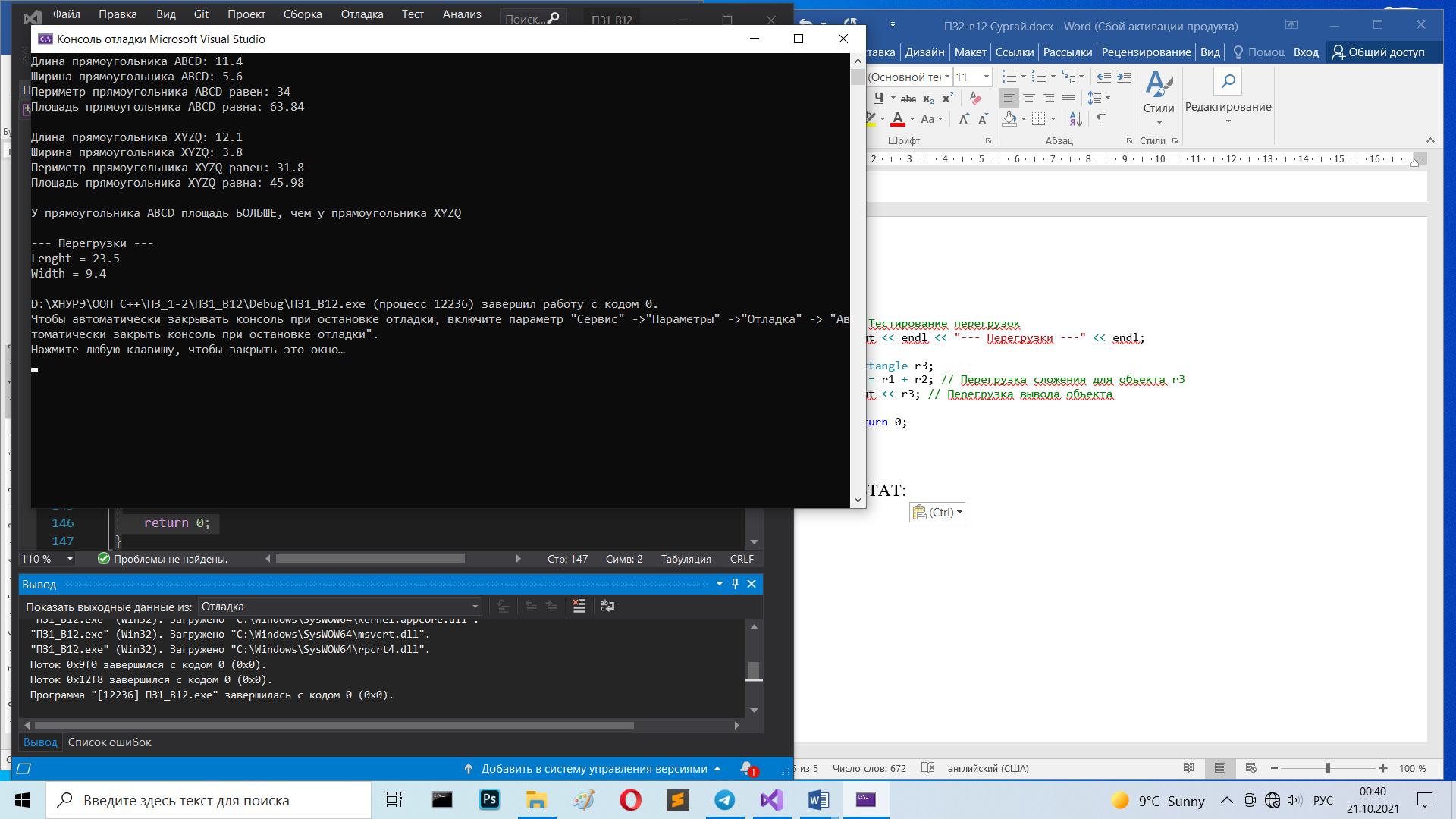
r3 = r1 + r2; // Перегрузка сложения для объекта r3

cout << r3; // Перегрузка вывода объекта

return 0;

}

РЕЗУЛЬТАТ:



ВИСНОВКИ: Практична робота №2 базується на практичній роботі №1. Були перевантажені оператори: «+» (складання), «<<» (виведення), «<» (менше), «>» (більше)