**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

Тема: Перегрузка операторов в С++

Студент: Волков

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. **Постановка задачи**

*Вариант 12*

**Разработать класс Rectangle**, представляющий собой прямоугольник со сторонами, параллельными осям координат. Поля – координаты левого нижнего и правого верхнего угла. Требуется реализовать следующие методы: вычисление площади и периметра, перемещения вдоль осей, изменение размеров, сравнение по площади и по периметру. Реализовать метод получения прямоугольника, представляющего общую часть (пересечение) двух прямоугольников. Реализовать метод объединения двух прямоугольников: наименьший прямоугольник, включающего оба заданных прямоугольника.

Необходимо реализовать:

- операцию приведения к типу double, вычисляющую площадь прямоугольника;

- операцию префиксного и постфиксного инкремента, увеличивающую одновременно размеры сторон прямоугольника.

- операции сравнения (больше, меньше, равно);

**Необходимо реализовать пользовательский литерал для работы с константами типа** Rectangle**.**

1. **Описание программы**

Программа написана в двух файлах main.cpp и main.h

В main.h в классе, под названием Rectangle, описаны методы которые умеют выполнять вышеперечисленные функции. В private описаны 2 точки: левый нижний угол прямоугольника и правый верхний - структурой типа *coord*. В этой структуре я храню переменные x и y. Также тут заинкапсулирован метод нахождения разницы, потому что нигде в программе я его не использую.

В public описаны все методы, которые выполняют вышеизложенные функции,в том числе описаны геттеры и сеттеры. За классом находится реализация перегрузки вышеуказанных операторов. Также вне класса описана реализация литерала *\_rec*. Данная реализация работает очень просто: для начала поступленный массив чаров переводится в переменную типа *std::string.* Это сделано для того, чтобы можно было воспользоваться методом данного типа *stod,* который переводит строку в число типа double. Затем в этой же строчке кода используется метод *substr,* который выделяет подстроку из строки. Это делается засчет передачи нужной позиции выделения подстроки, передаваемая как аргументы. Задача же всей этой надстройки - распарсить значения.

Как можно заметить зачем-то сверху к переменной типа string я добавляю запятую. Это сделано для того, чтобы в цикле фор в блок if “зашло” последнее число. Это незначительное действие избавило меня от написания большого количества кода.

В обязательном порядке также сделана “защита от дураков”, которая

завершает программу, когда пользователь в неправильном формате вводит координаты точек.

1. **Наборы и результаты выполнения тестов**
2. **Листинг программы**

***main.cpp***

/\*Волков Матвей Андреевич

\*

\* Разработать класс Rectangle, представляющий собой прямоугольник со сторонами, параллельными осям координат.

\* Поля – координаты левого нижнего и правого верхнего угла. Требуется реализовать следующие методы: вычисление площади

\* и периметра, перемещения вдоль осей, изменение размеров, сравнение по площади и по периметру. Реализовать метод

\* получения прямоугольника, представляющего общую часть (пересечение) двух прямоугольников. Реализовать метод

\* объединения двух прямоугольников: наименьший прямоугольник, включающего оба заданных прямоугольника.

\*

Необходимо реализовать:

- операцию приведения к типу double, вычисляющую площадь прямоугольника;

- операцию префиксного и постфиксного инкремента, увеличивающую одновременно размеры сторон прямоугольника.

- операции сравнения (больше, меньше, равно);

Необходимо реализовать пользовательский литерал для работы с константами типа Rectangle.

\*/

#include "main.h"

int main(){

Rectangle rec1, rec2;

Rectangle lit\_rec = "1,1,2,f"\_rec;

lit\_rec.show\_W\_H();

lit\_rec.show\_perimeter();

lit\_rec = "1,1,2,2"\_rec;

lit\_rec.show\_W\_H();

lit\_rec.show\_perimeter();

rec1.set\_sides(); rec2.set\_sides();

rec1.show\_perim\_differ(rec2);

if(rec1 < rec2){

std::cout << "rec2 have bigger area then rec1" << std::endl;

}

if(rec1 > rec2){

std::cout << "rec1 have bigger area then rec2" << std::endl;

}

if (rec1 == rec2){

std::cout << "rec1 and rec2 have equal area" << std::endl;

}

std::cout << "lit\_rec" << std::endl;

lit\_rec.show\_W\_H();

lit\_rec.show\_perimeter();

lit\_rec++;

lit\_rec.show\_perimeter();

lit\_rec.show\_W\_H();

std::cout << "lit\_rec shifting" << std::endl;

lit\_rec.shift\_x(2);

lit\_rec.show\_coordinates();

lit\_rec.shift\_y(-1);

lit\_rec.show\_coordinates();

std::cout << "\nmerging" << std::endl;

Rectangle rec1\_pl\_lit = lit\_rec.Big\_Rect(rec1);

rec1\_pl\_lit.show\_coordinates();

rec1\_pl\_lit.show\_W\_H();

std::cout << "\nseparation" << std::endl;

Rectangle rec2\_sb\_lit = rec2.Lil\_Rect(lit\_rec);

rec2\_sb\_lit.show\_coordinates();

rec2\_sb\_lit.show\_W\_H();

return 0;

}

***main.h***

#ifndef OOP\_EXERCISE\_02\_MAIN\_H

#define OOP\_EXERCISE\_02\_MAIN\_H

#include <iostream>

#include <cmath>

#include <string>

struct coord{

double x, y;

};

//-------------------start of class--------------------

class Rectangle{

private:

coord p1{}, p2{};

bool is\_seg;

double difference\_perim(Rectangle tmp\_rec) const {

return fabs(tmp\_rec.get\_perimeter() - this->get\_perimeter());

}

void enter\_sides(coord P1, coord P2){

p1.x = P1.x;

p1.y = P1.y;

p2.x = P2.x;

p2.y = P2.y;

if(p1.x > p2.x || p1.y > p2.y){

std::cout << "you entered wrong x and y";

exit(1);

}

if(get\_area() == 0){

is\_seg = true;

}

else{

is\_seg = false;

}

}

//----------end private-------------

public:

Rectangle(){

p1.x = 0;

p1.y = 0;

p2.x = 0;

p2.y = 0;

is\_seg = true;

}

Rectangle(double x\_1,double y\_1,double x\_2,double y\_2) {

p1.x = x\_1;

p1.y = y\_1;

p2.x = x\_2;

p2.y = y\_2;

if(p1.x > p2.x || p1.y > p2.y){

std::cout << "you entered wrong x and y";

exit(1);

}

if(get\_area() == 0){

is\_seg = true;

}

else{

is\_seg = false;

}

}

void set\_sides(){

std::cout << "Enter your points.\n"

"The First one is left-down edge. The Second one is right-up edge.\n" << std::endl;

coord p\_1, p\_2;

std::cin >> p\_1.x >> p\_1.y >> p\_2.x >> p\_2.y;

enter\_sides(p\_1,p\_2);

}

Rectangle(coord p\_1, coord p\_2): p1(p\_1), p2(p\_2){

if(get\_area() == 0){

is\_seg = true;

}

else{

is\_seg = false;

}

if(p1.x > p2.x || p1.y > p2.y){

std::cout << "you entered wrong x and y";

exit(1);

}

}

double get\_perimeter() const {

return 2 \* (p2.x - p1.x + p2.y - p1.y);

}

double get\_perimeter(Rectangle tmp\_rec) const {

return 2 \* (tmp\_rec.p2.x - tmp\_rec.p1.x + tmp\_rec.p2.y - tmp\_rec.p1.y);

}

double get\_area() const {

return (p2.x - p1.x) \* (p2.y - p1.y);

}

double get\_area(Rectangle tmp\_rec) const {

return (tmp\_rec.p2.x - tmp\_rec.p1.x) \* (tmp\_rec.p2.y - tmp\_rec.p1.y);

}

constexpr void shift\_x(double sh\_x){

p1.x += sh\_x;

p2.x += sh\_x;

}

constexpr void shift\_y(double sh\_y){

p1.y += sh\_y;

p2.y += sh\_y;

}

void show\_perim\_differ (Rectangle tmp\_rec) const{

std::cout << "Perimeter difference is " << difference\_perim(tmp\_rec) << std::endl;

}

void show\_W\_H() const {

std::cout << "width is " << this->get\_width() << " height is " << this->get\_height() << std::endl;

}

Rectangle Lil\_Rect(Rectangle tmp\_rec){

Rectangle LilBro;

if((p1.y > tmp\_rec.p2.y) || (tmp\_rec.p1.y > p2.y) ||

(p1.x > tmp\_rec.p2.x) || (tmp\_rec.p1.x < p2.x))

{

return LilBro;

}

LilBro.p1.y = fmax(tmp\_rec.p1.y, p1.y);

LilBro.p2.x = fmin(tmp\_rec.p2.x, p2.x);

LilBro.p1.y = fmax(tmp\_rec.p1.y, p1.y);

LilBro.p2.y = fmin(tmp\_rec.p2.y, p2.y);

LilBro.is\_seg = get\_area() == 0;;

return LilBro;

};

Rectangle Big\_Rect(Rectangle tmp\_rec){

Rectangle BigBro;

BigBro.p2.y = fmax(tmp\_rec.p2.y, p2.y);

BigBro.p1.y = fmin(tmp\_rec.p1.y, p1.y);

BigBro.p1.x = fmin(fmax(p1.x,p2.x),fmin(tmp\_rec.p1.x, tmp\_rec.p2.x));

BigBro.p2.x = fmax(fmin(p1.x,p2.x),fmax(tmp\_rec.p1.x, tmp\_rec.p2.x));

BigBro.is\_seg = get\_area() == 0;

return BigBro;

};

//prefix

Rectangle& operator++(){

p2.x += 1;

p2.y += 1;

return \*this;

}

//postfix

Rectangle operator++(int){

Rectangle result = \*this;

p2.x += 1;

p2.y += 1;

return result;

}

//prefix

Rectangle& operator--(){

if(this->get\_height() - 1 > 0 && this->get\_width() - 1 > 0) {

p2.x -= 1;

p2.y -= 1;

}

else{

std::cout << "I cant" << std::endl;

}

return \*this;

}

//postfix

Rectangle operator--(int){

Rectangle result = \*this;

if(this->get\_height() - 1 > 0 && this->get\_width() - 1 > 0) {

p2.x -= 1;

p2.y -= 1;

}

else{

std::cout << "I cant" << std::endl;

}

return result;

}

bool is\_Seg() const{

return is\_seg;

}

[[nodiscard]] double get\_width()const{

return p2.x - p1.x;

}

[[nodiscard]] double get\_height() const{

return p2.y - p1.y;

}

void show\_coordinates() const {

std::cout << "Point 1 is " << p1.x << ' ' << p1.y << '\n' <<

"Point 2 is " << p2.x << ' ' << p2.y << std::endl;

}

void show\_perimeter() const {

std::cout << "perimeter is " << this->get\_perimeter() << std::endl;

}

void show\_area() const {

std::cout << "area is " << this->get\_area() << std::endl;

}

};

//----------------end of class--------------------------

Rectangle operator"" \_rec(const char \*str, size\_t n){

double xy[4];

std::string s(str);

s += ',';// просто чтобы не писать лишний код

short count = 0; double last = 0;

for (int i = 0; i <= n; ++i) {

if(s[i] == ',' && count < 5){

try {

xy[count] = std::stod(s.substr(last, i - last + 1));

}

catch (const std::invalid\_argument&) {

std::cout << "You entered wrong arguments!\n"

"The obj will be empty" << std::endl;

xy[0] = 0;

xy[1] = 0;

xy[2] = 0;

xy[3] = 0;

}

last = i + 1;

++count;

}

}

return Rectangle(xy[0],xy[1],xy[2],xy[3]);

}

bool operator<(const Rectangle& lhs,const Rectangle& rhs){

return lhs.get\_area() < rhs.get\_area();

}

bool operator>(const Rectangle &lhs, const Rectangle &rhs){

return lhs.get\_area() > rhs.get\_area();

}

bool operator==(const Rectangle &lhs, const Rectangle &rhs){

return lhs.get\_area() == rhs.get\_area();

}

#endif //OOP\_EXERCISE\_02\_MAIN\_H

1. **Вывод**

Благодаря проделанной работе я узнал что такое литералы и как ими пользоваться. Научился пользоваться конструкцией try catch и научился выполнять перегрузки для операторов инкремента и декремента.