**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

Тема: Перегрузка операторов в С++

Студент: Базаргармаев Нима

Группа: 80-201

Преподаватель: Чернышов Л.Н.

Дата: 01.11.2019

Оценка:

Москва, 2019

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 состоит из трех файлов: 2.cpp, Rectangle.h, Rectangle.cpp. В 2.cpp содержится основной интерфейс программы (см. листинг кода). В Rectangle.h и Rectangle.cpp содержатся необходимые объявления и определения функций для класса Rectangle. Программа ожидает интерактивный ввод до тех пор, пока вы не нажмете 0. Реализованы методы подсчета периметра и площади, перемещения вдоль координатных осей и остальные методы, необходимые по условию задания. Так же перегружены базовые операции сравнения и пользовательский литерал “”\_rectangle.

1. Набор testcases

1

1 1 2 2

2

3

4

0

2

3

4

2

4

1

5

6

5

1

6

0 0 3 3

7

0 0 3 3

8

1 1 3 5

1

1 1 2 2

9

0 0 1 1

1

1 1 3 4

10

2 2 5 5

1. Результаты выполнения тестов.

MENU :

0. Exit program

1. Create rectangle

2. Perimeter

3. Area

4. Move rectangle by y

5. Move rectangle by x

6. Resize rectangle

7. Compare by perimeter

8. Compare by area

9. Get intersection of rectangles

10. Get minimal rectangle with intersection

11. Print menu

Enter the number of action: 1

Input the coordinates of rectangle

1 1 2 2

(1, 1), (2, 2)

Enter the number of action: 2

Perimeter of rectangle: 4

Enter the number of action: 3

Area of rectangle: 1

Enter the number of action: 4

Input the coordinates: 0

(1, 0), (2, 1)

Enter the number of action: 2

Perimeter of rectangle: 4

Enter the number of action: 3

Area of rectangle: 1

Enter the number of action: 4

Input the coordinates: 2

(1, 2), (2, 3)

Enter the number of action: 4

Input the coordinates: 1

(1, 1), (2, 2)

Enter the number of action: 5

Input the coordinates: 6

(7, 1), (8, 2)

Enter the number of action: 5

Input the coordinates: 1

(1, 1), (2, 2)

Enter the number of action: 6

Input lower and upper coordinates: 0 0 3 3

(0, 0), (3, 3)

Enter the number of action: 7

Input the values of second rectangle: 0 0 3 3

First rect perimeter is bigger

Enter the number of action: 8

Input the values of second rectangle: 1 1 3 5

First rect area is bigger

Enter the number of action: 1

Input the coordinates of rectangle

1 1 2 2

(1, 1), (2, 2)

Enter the number of action: 9

Input the values of second rectangle: 0 0 1 1

(0, 0), (2, 2)

Enter the number of action: 1

Input the coordinates of rectangle

1 1 3 4

(1, 1), (3, 4)

Enter the number of action: 10

Input the values of second rectangle: 2 2 5 5

(2, 2), (3, 4)

Enter the number of action: 0

1. Листинг программы

**2.cpp:**

/\*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 "Rectangle.h"

#include <iostream>

using namespace std;

int main() {

Rectangle rect;

int option = 11;

bool flag = true;

while (flag) {

switch (option) {

case 0:

flag = false;

break;

case 1: {

Coordinate lower, upper;

cout << "Input the coordinates of rectangle" << endl;

cin >> lower.x >> lower.y >> upper.x >> upper.y;

rect.set\_lower(lower);

rect.set\_upper(upper);

rect.Print();

break;

}

case 2:

cout << "Perimeter of rectangle: " << rect.perimeter() << endl;

break;

case 3:

cout << "Area of rectangle: " << rect.area() << endl;

break;

case 4: {

cout << "Input the coordinates: ";

Coordinate y;

cin >> y.y;

rect.setY(y);

rect.Print();

break;

}

case 5: {

cout << "Input the coordinates: ";

Coordinate x;

cin >> x.x;

rect.setX(x);

rect.Print();

break;

}

case 6: {

cout << "Input lower and upper coordinates: ";

Coordinate lower, upper;

cin >> lower.x >> lower.y >> upper.x >> upper.y;

rect.setSize(lower, upper);

rect.Print();

break;

}

case 7: {

Rectangle tmp;

Coordinate lower, upper;

cout << "Input the values of second rectangle: ";

cin >> lower.x >> lower.y >> upper.x >> upper.y;

tmp.set\_lower(lower);

tmp.set\_upper(upper);

if (rect.comparePerimeter(rect, tmp) == true) {

cout << "Second rect perimeter is bigger" << endl;

} else {

cout << "First rect perimeter is bigger" << endl;

}

break;

}

case 8: {

Rectangle tmp;

Coordinate lower, upper;

cout << "Input the values of second rectangle: ";

cin >> lower.x >> lower.y >> upper.x >> upper.y;

tmp.set\_lower(lower);

tmp.set\_upper(upper);

if (rect.compareArea(rect, tmp) == true) {

cout << "Second rect area is bigger" << endl;

} else {

cout << "First rect area is bigger" << endl;

}

break;

}

case 9: {

Rectangle tmp;

Coordinate lower, upper;

cout << "Input the values of second rectangle: ";

cin >> lower.x >> lower.y >> upper.x >> upper.y;

tmp.set\_lower(lower);

tmp.set\_upper(upper);

rect = rect.intersection(rect, tmp);

rect.Print();

break;

}

case 10: {

Rectangle tmp;

Coordinate lower, upper;

cout << "Input the values of second rectangle: ";

cin >> lower.x >> lower.y >> upper.x >> upper.y;

tmp.set\_lower(lower);

tmp.set\_upper(upper);

rect = rect.getMinIntersection(rect, tmp);

rect.Print();

break;

}

case 11:

cout << "MENU :" << endl <<

"0. Exit program" << endl <<

"1. Create rectangle" << endl <<

"2. Perimeter" << endl <<

"3. Area" << endl <<

"4. Move rectangle by y" << endl <<

"5. Move rectangle by x" << endl <<

"6. Resize rectangle" << endl <<

"7. Compare by perimeter" << endl <<

"8. Compare by area" << endl <<

"9. Get intersection of rectangles" << endl <<

"10. Get minimal rectangle with intersection" << endl <<

"11. Print menu" << endl;

break;

}

if (flag) {

cout << "Enter the number of action: ";

cin >> option;

}

}

return 0;

}

**Rectangle.h:**

#pragma once

#include <iostream>

#include <iomanip>

#include <cstdint>

struct Coordinate { // координаты угловых точек

double x, y;

};

class Rectangle {

public:

void Print() const; // печать

void set\_lower(const Coordinate& coord); // задаем координаты

void set\_upper(const Coordinate& coord);

double perimeter() const; // периметр

double area() const; // площадь

void setY(const Coordinate& coord); // перемещение вдоль оси оУ

void setX(const Coordinate& coord); // перемещение вдоль оси оХ

void setSize(const Coordinate& coord1, const Coordinate& coord2); // изменить размер прямоугольника

bool comparePerimeter(const Rectangle& lhs, const Rectangle& rhs) const; // сравнение по периметру

bool compareArea(const Rectangle& lhs, const Rectangle& rhs) const; // сравнение по площади

Rectangle intersection(const Rectangle& lhs, const Rectangle& rhs) const; // пересечение прямоугольников

Rectangle getMinIntersection(const Rectangle& lhs, const Rectangle& rhs) const; // наименьший из объединения

friend Rectangle operator "" \_rectangle(const char\* str, size\_t size); // пользовательский литерал

private:

Coordinate lower\_left\_point, upper\_right\_point;

};

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

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

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

**Rectangle.cpp:**

#include "Rectangle.h"

#include <exception>

#include <stdexcept>

#define SET\_POINT(field) \

if (coord.field <= lower\_left\_point.field) { \

upper\_right\_point.field -= lower\_left\_point.field - coord.field; \

lower\_left\_point.field = coord.field; \

} else if (coord.field >= lower\_left\_point.field \

&& coord.field <= upper\_right\_point.field) { \

upper\_right\_point.field += coord.field; \

lower\_left\_point.field += coord.field; \

} else if (coord.field >= upper\_right\_point.field) { \

lower\_left\_point.field += coord.field; \

upper\_right\_point.field += coord.field; \

}

void Rectangle::set\_lower(const Coordinate& coord) {

lower\_left\_point = coord;

}

void Rectangle::set\_upper(const Coordinate& coord) {

upper\_right\_point = coord;

}

void Rectangle::Print() const {

std::cout << "(" << lower\_left\_point.x << ", " << lower\_left\_point.y <<

"), (" << upper\_right\_point.x << ", " << upper\_right\_point.y << ")" << std::endl;

}

double Rectangle::perimeter() const {

return 2 \* ((upper\_right\_point.x - lower\_left\_point.x)

+ (upper\_right\_point.y - lower\_left\_point.y));

}

double Rectangle::area() const {

return (upper\_right\_point.x - lower\_left\_point.x)

\* (upper\_right\_point.y - lower\_left\_point.y);

}

void Rectangle::setY(const Coordinate& coord) {

SET\_POINT(y)

}

void Rectangle::setX(const Coordinate& coord) {

SET\_POINT(x)

}

void Rectangle::setSize(const Coordinate& coord1, const Coordinate& coord2) {

lower\_left\_point = coord1;

upper\_right\_point = coord2;

}

bool Rectangle::comparePerimeter(const Rectangle& lhs, const Rectangle& rhs) const {

return lhs.perimeter() < rhs.perimeter();

}

bool Rectangle::compareArea(const Rectangle& lhs, const Rectangle& rhs) const {

return lhs.area() < rhs.area();

}

Rectangle Rectangle::intersection(const Rectangle& lhs, const Rectangle& rhs) const {

Rectangle rect;

if (lhs.lower\_left\_point.x < rhs.lower\_left\_point.x) {

rect.lower\_left\_point = lhs.lower\_left\_point;

rect.upper\_right\_point = rhs.upper\_right\_point;

} else if (lhs.lower\_left\_point.x > rhs.lower\_left\_point.x &&

lhs.upper\_right\_point.x < rhs.upper\_right\_point.x) {

rect.lower\_left\_point = lhs.lower\_left\_point;

rect.upper\_right\_point = rhs.upper\_right\_point;

} else if (rhs.lower\_left\_point.x < lhs.lower\_left\_point.x) {

rect.lower\_left\_point = rhs.lower\_left\_point;

rect.upper\_right\_point = lhs.upper\_right\_point;

}

return rect;

}

Rectangle Rectangle::getMinIntersection(const Rectangle& lhs, const Rectangle& rhs) const {

Rectangle rect;

if ((lhs.lower\_left\_point.x <= rhs.lower\_left\_point.x) &&

(lhs.lower\_left\_point.y <= rhs.lower\_left\_point.y)) {

rect.lower\_left\_point = rhs.lower\_left\_point;

rect.upper\_right\_point = lhs.upper\_right\_point;

} else if ((lhs.lower\_left\_point.x <= rhs.lower\_left\_point.x) &&

(lhs.lower\_left\_point.y >= rhs.lower\_left\_point.y)) {

rect.lower\_left\_point.x = rhs.lower\_left\_point.x;

rect.lower\_left\_point.y = lhs.lower\_left\_point.y;

rect.upper\_right\_point.x = lhs.upper\_right\_point.x ;

rect.upper\_right\_point.y = rhs.upper\_right\_point.y;

} else if ((lhs.lower\_left\_point.x >= rhs.lower\_left\_point.x) &&

(lhs.lower\_left\_point.y >= rhs.lower\_left\_point.y)) {

rect.lower\_left\_point = lhs.lower\_left\_point;

rect.upper\_right\_point = rhs.upper\_right\_point;

} else if ((rhs.lower\_left\_point.x <= lhs.lower\_left\_point.x) &&

(rhs.lower\_left\_point.y >= lhs.lower\_left\_point.y)) {

rect.lower\_left\_point.x = lhs.lower\_left\_point.x;

rect.lower\_left\_point.y = rhs.lower\_left\_point.y;

rect.upper\_right\_point.x = rhs.upper\_right\_point.x;

rect.upper\_right\_point.y = lhs.lower\_left\_point.y;

}

return rect;

}

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

return lhs.area() < rhs.area();

}

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

return lhs.area() > rhs.area();

}

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

return lhs.area() == rhs.area();

}

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

Rectangle rect;

std::string s(str, size);

double l\_x, l\_y, u\_x, u\_y;

std::istringstream is(s);

is.ignore(1);

is >> l\_x;

is.ignore(2);

is >> l\_y;

is.ignore(4);

is >> u\_x;

is.ignore(1);

is >> u\_y;

rect.lower\_left\_point.x = l\_x;

rect.lower\_left\_point.y = l\_y;

rect.upper\_right\_point.x = u\_x;

rect.upper\_right\_point.y = u\_y;

return rect;

}

**Вывод:**

Научился создавать пользовательские литералы, а также научился перегружать стандартные арифметические операторы и оператор стандартного ввода <<.

## Список литературы

1. Перегрузка операторов в C++ [Электронный ресурс]  
   URL: <https://habr.com/ru/post/132014/>
2. Пользовательские литералы в C++11 [Электронный ресурс]  
   URL: <https://habr.com/ru/post/140357/>