Experiment3-董皓彧

环境：

```
g++.exe (x86_64-win32-seh-rev1, Built by MinGW-Builds project) 13.2.0
Visual Stdio Code 1.86.2
```

作业仓库地址：
https://github.com/FHYQ-Dong/Tsinghua-Program-Design-Assignments-2/tree/main/Experiment3

必做题

Experiment3-1

题目：

```
•先声明一个点类Point，成员为其坐标x，y，并设计构造函数（可赋初值也可以不赋初值）、复制构造函数、析构函数（打印信息，表示其被调用），设置新值函数Set，打印成员坐标值函数Print
•再声明一个矩形类Rectangle，其成员为矩形的左下角和右上角两个Point对象，并设计Rectangle构造函数（分别由x1，y1，x2，y2坐标值构造，或由p1，p2两个点对象构造，可赋初值也可以不赋初值）、复制构造函数、析构函数（打印信息，表示其被调用），设置新值函数Set，打印成员值函数Print，计算矩形面积函数Area，以及其它你认为对访问此类对象有用的成员函数。并用此Rectangle类定义对象，调用所有成员函数
```

输入格式：

```
略
```

输出格式：

```
打印出的函数调用信息
```

代码：

```
#include <iostream>

class Point {
    private:
        double x, y;

    public:
        Point(): x(0), y(0) {
            Print();
            std::cout << "Point Default Constructor called" << std::endl;
        }
        Point(double xx, double yy): x(xx), y(yy) {
            Print();
            std::cout << "Point Constructor called" << std::endl;
        }
```

```
        Point(const Point &px): x(px.x), y(px.y) {
            Print();
            std::cout << "Point Copy Constructor called" << std::endl;
        }
        ~Point() {
            Print();
            std::cout << "Point Destructor called" << std::endl;
        }
        void set(double x, double y) {
            this->x = x;
            this->y = y;
        }
        void set(const Point &p) {
            x = p.x;
            y = p.y;
        }
        void setX(double x) {
            this->x = x;
        }
        void setY(double y) {
            this->y = y;
        }
        double getX() const {
            return x;
        }
        double getY() const {
            return y;
        }
        void Print() const {
            std::cout << "(" << x << ", " << y << ") ";
        }
        void move(double dx, double dy) {
            x += dx;
            y += dy;
        }
        Point &operator=(const Point &p) {
            x = p.x;
            y = p.y;
            return *this;
        }
        bool operator==(const Point &p) const {
            return x == p.x && y == p.y;
        }
        bool operator!=(const Point &p) const {
            return x != p.x || y != p.y;
        }
};

class Rectangle {
    private:
        Point left_bottom, right_top;

    public:
        Rectangle(): left_bottom(), right_top() {
            Print();
```

```
            std::cout << "Rectangle Default Constructor called" << std::endl;
        }
        Rectangle(const Point &p1, const Point &p2): left_bottom(p1),
right_top(p2) {
            Print();
            std::cout << "Rectangle Constructor called" << std::endl;
        }
        Rectangle(double x1, double y1, double x2, double y2): left_bottom(x1,
y1), right_top(x2, y2) {
            Print();
            std::cout << "Rectangle Constructor called" << std::endl;
        }
        Rectangle(const Rectangle &r): left_bottom(r.left_bottom),
right_top(r.right_top) {
            Print();
            std::cout << "Rectangle Copy Constructor called" << std::endl;
        }
        ~Rectangle() {
            Print();
            std::cout << "Rectangle Destructor called" << std::endl;
        }
        void set(const Point &p1, const Point &p2) {
            left_bottom.set(p1);
            right_top.set(p2);
        }
        void set(double x1, double y1, double x2, double y2) {
            left_bottom.set(x1, y1);
            right_top.set(x2, y2);
        }
        void setLeftBottom(const Point &p) {
            left_bottom.set(p);
        }
        void setRightTop(const Point &p) {
            right_top.set(p);
        }
        Point getLeftBottom() const {
            return left_bottom;
        }
        Point getRightTop() const {
            return right_top;
        }
        void Print() const {
            std::cout << "Left Bottom: ";
            left_bottom.Print();
            std::cout << "Right Top: ";
            right_top.Print();
        }
        double Area() const {
            return (right_top.getX() - left_bottom.getX()) * (right_top.getY() -
left_bottom.getY());
        }
        double Perimeter() const {
            return 2 * (right_top.getX() - left_bottom.getX() + right_top.getY()
- left_bottom.getY());
        }
```

```
        bool isSquare() const {
            return (right_top.getX() - left_bottom.getX()) == (right_top.getY()
- left_bottom.getY());
        }
        Point Center() const {
            return Point((left_bottom.getX() + right_top.getX()) / 2,
(left_bottom.getY() + right_top.getY()) / 2);
        }
        Rectangle &operator=(const Rectangle &r) {
            left_bottom = r.left_bottom;
            right_top = r.right_top;
            return *this;
        }
        bool operator==(const Rectangle &r) const {
            return left_bottom.getX() == r.left_bottom.getX() &&
left_bottom.getY() == r.left_bottom.getY() && right_top.getX() ==
r.right_top.getX() && right_top.getY() == r.right_top.getY();
        }
        bool operator!=(const Rectangle &r) const {
            return left_bottom.getX() != r.left_bottom.getX() ||
left_bottom.getY() != r.left_bottom.getY() || right_top.getX() !=
r.right_top.getX() || right_top.getY() != r.right_top.getY();
        }
        void move(double dx, double dy) {
            left_bottom.move(dx, dy);
            right_top.move(dx, dy);
        }
};

inline void test() {
    Point p1(1, 1), p2(2, 2);
    std::cout << ">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>" <<
std::endl;

    Rectangle r1(p1, p2);
    r1.Print(); std::cout << std::endl; // Left Bottom: (1, 1) Right Top: (2, 2)
    std::cout << "<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<" <<
std::endl;

    std::cout << "Area: " << r1.Area() << std::endl; // Area: 1
    std::cout << "Perimeter: " << r1.Perimeter() << std::endl; // Perimeter: 4
    std::cout << "Is Square: " << r1.isSquare() << std::endl; // Is Square: 1
    Point center1 = r1.Center();
    std::cout << "Center: (" << center1.getX() << ", " << center1.getY() << ")"
<< std::endl; // Center: (1.5 1.5)
    std::cout << ">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>" <<
std::endl;

    r1.move(1, 1);
    r1.Print(); std::cout << std::endl; // Left Bottom: (2, 2) Right Top: (3, 3)
    Point center2 = r1.Center();
    std::cout << "Center: (" << center2.getX() << ", " << center2.getY() << ")"
<< std::endl; // Center: (2.5 2.5)
    std::cout << "<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<" <<
std::endl;
```

```cpp
    Rectangle r2(r1);
    r2.Print(); std::cout << std::endl; // Left Bottom: (2, 2) Right Top: (3, 3)
    std::cout << ">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>" << std::endl;

    Rectangle r3;
    r3 = r2;
    r3.Print(); std::cout << std::endl; // Left Bottom: (2, 2) Right Top: (3, 3)
    std::cout << "<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<" << std::endl;
}

int main() {
    test();
    return 0;
}
```

输入1：

```
```

输出1：

```
(1, 1) Point Constructor called
(2, 2) Point Constructor called
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
(1, 1) Point Copy Constructor called
(2, 2) Point Copy Constructor called
Left Bottom: (1, 1) Right Top: (2, 2) Rectangle Constructor called
Left Bottom: (1, 1) Right Top: (2, 2)
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Area: 1
Perimeter: 4
Is Square: 1
(1.5, 1.5) Point Constructor called
Center: (1.5, 1.5)
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Left Bottom: (2, 2) Right Top: (3, 3)
(2.5, 2.5) Point Constructor called
Center: (2.5, 2.5)
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
(2, 2) Point Copy Constructor called
(3, 3) Point Copy Constructor called
Left Bottom: (2, 2) Right Top: (3, 3) Rectangle Copy Constructor called
Left Bottom: (2, 2) Right Top: (3, 3)
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
(0, 0) Point Default Constructor called
(0, 0) Point Default Constructor called
Left Bottom: (0, 0) Right Top: (0, 0) Rectangle Default Constructor called
Left Bottom: (2, 2) Right Top: (3, 3)
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Left Bottom: (2, 2) Right Top: (3, 3) Rectangle Destructor called
(3, 3) Point Destructor called
```

```
(2, 2) Point Destructor called
Left Bottom: (2, 2) Right Top: (3, 3) Rectangle Destructor called
(3, 3) Point Destructor called
(2, 2) Point Destructor called
(2.5, 2.5) Point Destructor called
(1.5, 1.5) Point Destructor called
Left Bottom: (2, 2) Right Top: (3, 3) Rectangle Destructor called
(3, 3) Point Destructor called
(2, 2) Point Destructor called
(2, 2) Point Destructor called
(1, 1) Point Destructor called
```