**Задание**

Составить описание класса прямоугольников со сторонами, параллельными осям координат. Предусмотреть

возможность перемещения прямоугольников на плоскости, изменения размеров, построения наименьшего

прямоугольника, содержащего два заданных прямоугольника, и прямоугольника, являющегося общей частью

(пересечением) двух прямоугольников. Написать программу, демонстрирующую работу с этим классом. Программа

должна содержать меню, позволяющее осуществить проверку всех методов класса.

**Описание основных структур данных и алгоритмов, используемых в работе**

**pdd p, q; // левая нижняя и правая верхняя точки прямоугольника соответственно**

**double a, b; // длины сторон**

**void set\_rect(pdd pp, pdd qq); // Задание прямоугольника**

**Rect(); // Конструктор по умолчанию**

**Rect(pdd pp, pdd qq); // Конструктор с параметрами**

**void print\_points() const; // Вывод точек**

**void change\_sides(double aa, double bb); // Изменение длин сторон**

**void move\_rect(double x, double y); // Перемещение прямоугольника**

**Rect min\_hull(const Rect &obj); // Вычисление мин. прямогольника содержащего 2 заданных прямоугольника**

**Rect intersection(const Rect &obj); // Вычисление прямогольника пересечения 2 заданных прямоугольников**

**Описание интерфейса пользователя программы**

**Интеркактивное меню с полным описанием и сопровождением пользователя в программе.**

**Контрольный пример и результаты тестирования**

Задайте точки прямоугольника 1.

Задайте точки прямоугольника 2.

Введите 1 для изменения сторон прямоугольника.

Введите 2 для изменения координат прямоугольника.

Введите 3 для вычисления мин. прямогольника содержащего 2 заданных прямоугольника.

Введите 4 для вычисления прямогольника пересечения 2 заданных прямоугольников.

Введите 0 для выхода.

0 0 1 1

0.5 0.5 2 2

3

0 0 2 2

4

0.5 0.5 1 1

2

Введите координаты левой нижней точки прямоугольника.

-1 -1

3

-1 -1 2 2

4

0 0 0 0

1

Введите стороны прямоугольника.

3 2

3

-1 -1 2 2

4

0.5 0.5 2 1

0

**Листинг программы**

#include <iostream>

#include <algorithm>

#define pdd pair<double, double>

using namespace std;

/\*  Составить описание класса прямоугольников со сторонами, параллельными осям координат. Предусмотреть

возможность перемещения прямоугольников на плоскости, изменения размеров, построения наименьшего

прямоугольника, содержащего два заданных прямоугольника, и прямоугольника, являющегося общей частью

(пересечением) двух прямоугольников. Написать программу, демонстрирующую работу с этим классом. Программа

должна содержать меню, позволяющее осуществить проверку всех методов класса.    \*/

class Rect

{

private:

    pdd p, q;    // левая нижняя и правая верхняя точки прямоугольника соответственно

    double a, b; // длины сторон

public:

    void set\_rect(pdd pp, pdd qq); // Задание прямоугольника

    Rect(); // Конструктор по умолчанию

    Rect(pdd pp, pdd qq); // Конструктор с параметрами

    void print\_points() const; // Вывод точек

    void change\_sides(double aa, double bb); // Изменение длин сторон

    void move\_rect(double x, double y); // Перемещение прямоугольника

    Rect min\_hull(const Rect &obj); // Вычисление мин. прямогольника содержащего 2 заданных прямоугольника

    Rect intersection(const Rect &obj); // Вычисление прямогольника пересечения 2 заданных прямоугольников

};

void Rect::set\_rect(pdd pp, pdd qq)

{

    if (pp.first < qq.first && pp.second < qq.second)

    {

        p = pp;

        q = qq;

        a = q.first - p.first;

        b = q.second - p.second;

    }

    else

    {

        p = pdd(0, 0);

        q = pdd(1, 1);

        a = 1;

        b = 1;

    }

}

void Rect::print\_points() const{

    cout << p.first << " " << p.second << " " << q.first << " " << q.second << endl;

}

Rect::Rect()

{

    p = q = pdd(0, 0);

    a = b = 0;

}

Rect::Rect(pdd pp, pdd qq)

{

    set\_rect(pp, qq);

}

void Rect::change\_sides(double aa, double bb)

{

    if (aa > 0 && bb > 0)

    {

        a = aa;

        b = bb;

        q = pdd(p.first + aa, p.second + bb);

    }

    else

    {

        a = 1;

        b = 1;

        q = pdd(p.first + aa, p.second + bb);

    }

}

void Rect::move\_rect(double x, double y)

{

    set\_rect(pdd(x, y), pdd(x + a, y + b));

}

Rect Rect::min\_hull(const Rect &obj)

{

    pdd lp = pdd(min(p.first, obj.p.first), min(p.second, obj.p.second));

    pdd rp = pdd(max(q.first, obj.q.first), max(q.second, obj.q.second));

    return Rect(lp, rp);

}

Rect Rect::intersection(const Rect &obj)

{

    if (min\_hull(obj).a < a + obj.a && min\_hull(obj).b < b + obj.b)

    {

        return Rect(pdd(max(p.first, obj.p.first), max(p.second, obj.p.second)), pdd(min(q.first, obj.q.first), min(q.second, obj.q.second)));

    }

    else

    {

        return Rect();

    }

}

int main()

{

    setlocale(LC\_ALL, "Russian");

    pdd a, b;

    cout << "Задайте точки прямоугольника 1.\n";

    cin >> a.first >> a.second >> b.first >> b.second;

    Rect example1 = Rect(a, b);

    cout << "Задайте точки прямоугольника 2.\n";

    cin >> a.first >> a.second >> b.first >> b.second;

    Rect example2 = Rect(a, b);

    cout << "Введите 1 для изменения сторон прямоугольника.\n";

    cout << "Введите 2 для изменения координат прямоугольника.\n";

    cout << "Введите 3 для вычисления мин. прямогольника содержащего 2 заданных прямоугольника.\n";

    cout << "Введите 4 для вычисления прямогольника пересечения 2 заданных прямоугольников.\n";

    cout << "Введите 0 для выхода.\n";

    int n = 0;

    cin >> n;

    double l1, l2;

    Rect tmp = Rect(pdd(0, 0), pdd(1, 1));

    while(n != 0){

        switch (n)

        {

        case 1:

            cout << "Введите стороны прямоугольника.\n";

            cin >> l1 >> l2;

            example1.change\_sides(l1, l2);

            break;

        case 2:

            cout << "Введите координаты левой нижней точки прямоугольника.\n";

            cin >> l1 >> l2;

            example1.move\_rect(l1, l2);

            break;

        case 3:

            tmp = example1.min\_hull(example2);

            tmp.print\_points();

            break;

        case 4:

            tmp = example1.intersection(example2);

            tmp.print\_points();

            break;

        case 0:

            return 0;

        default:

            break;

        }

        cin >> n;

    }

    return 0;

}