**Государственное бюджетное общеобразовательное учреждение "Президентский физико-математический лицей № 239"**

**Нахождение пересечения двух множеств параллельных прямоугольников**

Годовой проект по информатике

Работу выполнила

Ученица 10-1 класса

Смолинская Алиса

1. **Постановка задачи**

На плоскости задано два множества "параллельных" прямоугольников. Найти "пересечение" множеств. То есть, все такие точки плоскости, которые лежат внутри хотя бы одного прямоугольника первого множества, и внутри хотя бы одного прямоугольника второго множества. Отобразить найденное множество "заливкой".

Получить из файла противоположные координаты первого и второго множества прямоугольников. Отобразить на экране пересечение этих множеств "заливкой".

1. **Уточнение исходных и выходных данных**

**2.1. Исходные данные**

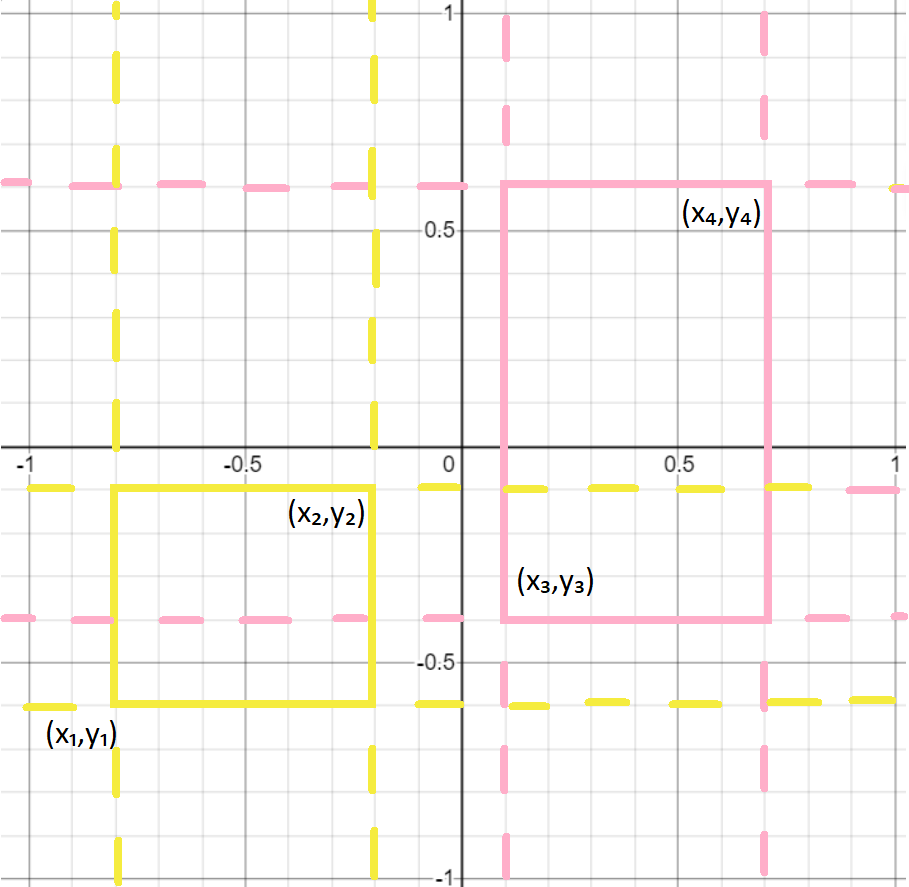
На вход подаются координаты противоположных точек прямоугольников, стороны которых параллельны осям координат и номер множества, в которое они попадают. Их можно задать как вручную или с помощью входного файла, так и случайным образом. Ограничений на объем входного файла нет.

**2.2. Выходные данные**

Необходимо вывести отображение пересечения двух множеств параллельных прямоугольников. Так как ограничений на объем входных данных нет, то ограничений на объем выходных данных тоже.

1. **Математическая модель**

Проверяем, есть ли пересечение по двум осям координат, то есть нет ли такого, что либо x₂ < x₃, или x₄ < x₁, или y₂ < y₃, или y₄ < y₁ (на представленной картинке x₂ < x₃ поэтому пересечение является пустым множеством)

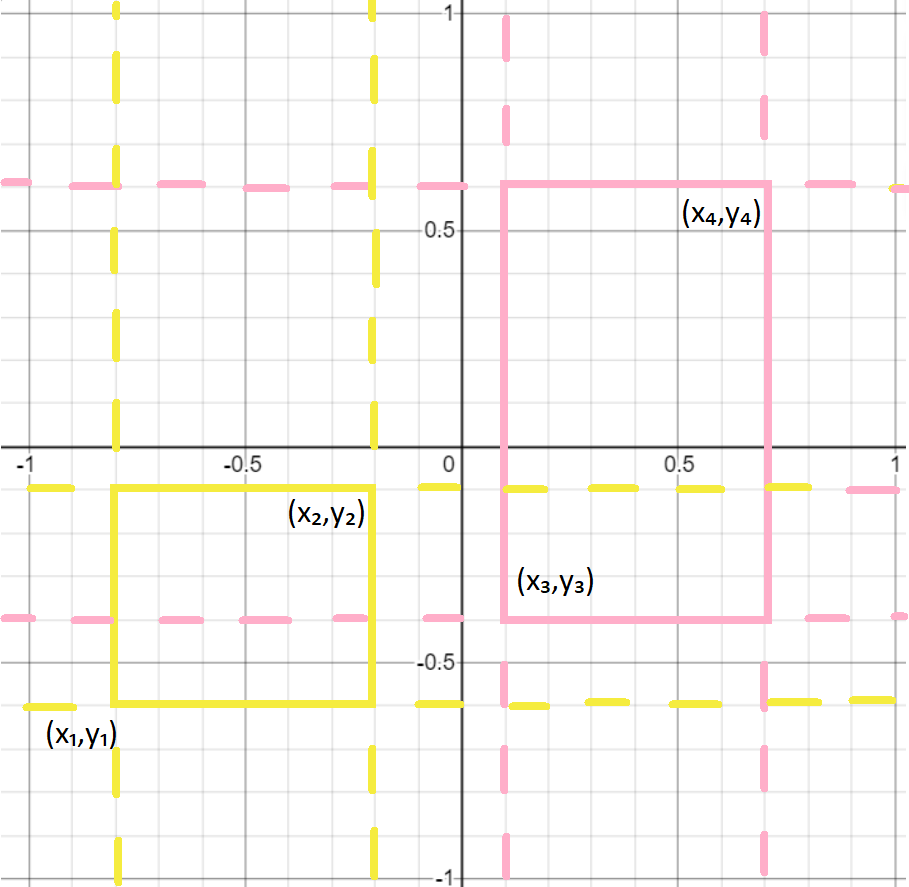
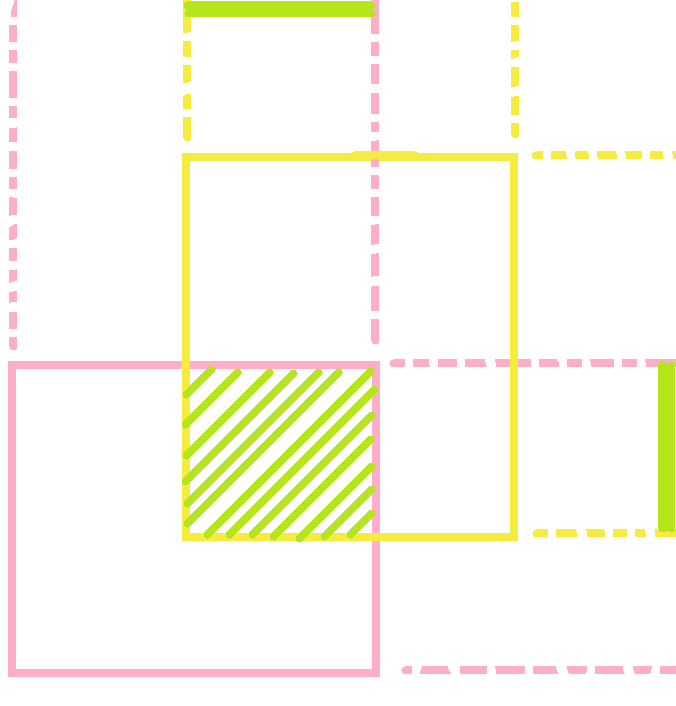


1. **Анализ используемой структуры данных**

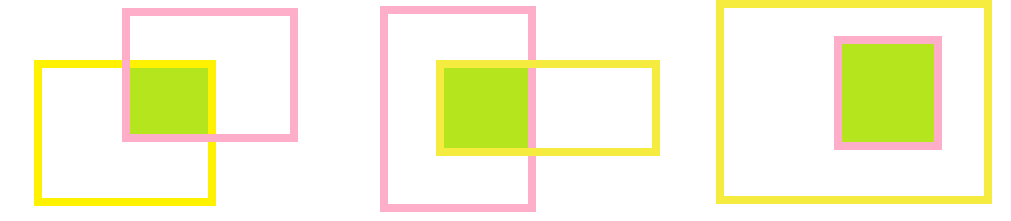
Так как в своем решении я сравниваю координаты каждого прямоугольника из одного множества с каждым прямоугольником из другого множества, то мне необходимо хранить массив с входными данными. Поэтому я создала динамический массив quads, хранящий координаты противоположных точек прямоугольника и номер множества, в которое он входит (тип данных координат - double, тип данных номера множества - int). Выходные данные я храню в динамическом массиве resultQuads, который хранит координаты пересечения прямоугольников (если они есть) , которые являются прямоугольниками (так как пересечение параллельных прямоугольников это либо прямоугольник либо пустое множество) и номер множества ответов (тип данных координат - double, тип данных номера множества - int).

1. **Выбор метода решения (описание алгоритма)**

Перебираем все пары прямоугольников из разных множеств, проверяем, есть ли пересечения по обеим осям координат, иначе общее пересечение является пустым множеством. То есть нет ли такого, что либо x₂ < x₃, или x₄ < x₁, или y₂ < y₃, или y₄ < y₁ (на представленной картинке ниже справа x₂ < x₃ поэтому пересечение является пустым множеством)



Если существует пересечение, то рассмотрим несколько случаев расположения двух прямоугольников относительно друг друга (некоторые из них представлены на картине ниже)



Потом для каждого из возможных случаев выразим координаты пересечения исходных прямоугольников (сравниваем координаты исходных прямоугольников по осям) которое является прямоугольником и добавим в результирующий массив resultQuads. Затем отобразим полученные в результирующем массиве прямоугольники заливкой.

1. **Комментированный листинг**

// перебираем все возможные пары прямоугольников

for (Quad q : quads) {

for (Quad p : quads) {

// будем рассматривать прямоугольники из разных исходных множеств

if (q != p && q.setNumber != p.setNumber) {

// сравниваем координаты прямоугольников по оси абсцисс и ординат и если будет выполняться условие условного оператора, то пересечение прямоугольников есть пустое множество

if((q.c.x < p.a.x || p.c.x < q.a.x) || (q.c.y < p.a.y || p.c.y < q.a.y)){

int k=0;

// иначе пересечение не является пустым множеством

} else {

// задаем координаты противоположных вершин прямоугольника, являющегося пересечением двух рассматриваемых

double x1=10, x2=10, y1=10, y2=10;

// разбираем случаи расположения прямоугольников относительно друг друга по осям абсцисс и ординат и делаем вывод о получившемся пересечении, то есть задаем противоположные координаты прямоугольника, являющегося пересечением двух исходных

if(q.a.x <= p.a.x && q.c.x <= p.c.x){

if(q.a.y <= p.a.y && q.c.y <= p.c.y) {x1 = p.a.x; y1 = p.a.y; x2 = q.c.x ; y2 = q.c.y;}

else if(q.a.y <= p.a.y && q.c.y >= p.c.y) {x1 = p.a.x; y1 = p.a.y; x2 = q.c.x; y2 = p.c.y;}

else if(q.a.y >= p.a.y && q.c.y <= p.c.y) {x1 = p.a.x; y1 = q.a.y; x2 = q.c.x; y2 = q.c.y;}

else if(q.a.y >= p.a.y && q.c.y >= p.c.y) {x1 = p.a.x; y1 = q.a.y; x2 = q.c.x; y2 = p.c.y;}

} else if(q.a.x <= p.a.x && q.c.x >= p.c.x){

if(q.a.y <= p.a.y && q.c.y <= p.c.y) {x1 = p.a.x; y1 = p.a.y; x2 = p.c.x; y2 = q.c.y;}

else if(q.a.y <= p.a.y && q.c.y >= p.c.y) {x1 = p.a.x; y1 = p.a.y; x2 = p.c.x; y2 = p.c.y;}

else if(q.a.y >= p.a.y && q.c.y <= p.c.y) {x1 = p.a.x; y1 = q.a.y; x2 = p.c.x; y2 = q.c.y;}

else if(q.a.y >= p.a.y && q.c.y >= p.c.y) {x1 = p.a.x; y1 = q.a.y; x2 = p.c.x; y2 = p.c.y;}

} else if(q.a.x >= p.a.x && q.c.x <= p.c.x){

if(q.a.y <= p.a.y && q.c.y <= p.c.y) {x1 = q.a.x; y1 = p.a.y; x2 = q.c.x; y2 = q.c.y;}

else if(q.a.y <= p.a.y && q.c.y >= p.c.y) {x1 = q.a.x; y1 = p.a.y; x2 = q.c.x; y2 = p.c.y;}

else if(q.a.y >= p.a.y && q.c.y <= p.c.y) {x1 = q.a.x; y1 = q.a.y; x2 = q.c.x; y2 = q.c.y;}

else if(q.a.y >= p.a.y && q.c.y >= p.c.y) {x1 = q.a.x; y1 = q.a.y; x2 = q.c.x; y2 = p.c.y;}

}else if(q.a.x >= p.a.x && q.c.x >= p.c.x){

if(q.a.y <= p.a.y && q.c.y <= p.c.y) {x1 = q.a.x; y1 = p.a.y; x2 = p.c.x; y2 = q.c.y;}

else if(q.a.y <= p.a.y && q.c.y >= p.c.y) {x1 = q.a.x; y1 = p.a.y; x2 = p.c.x; y2 = p.c.y;}

else if(q.a.y >= p.a.y && q.c.y <= p.c.y) {x1 = q.a.x; y1 = q.a.y; x2 = p.c.x; y2 = q.c.y;}

else if(q.a.y >= p.a.y && q.c.y >= p.c.y) {x1 = q.a.x; y1 = q.a.y; x2 = p.c.x; y2 = p.c.y;}

}

// по получившимся противоположным координатам прямоугольника строим прямоугольник класса Quad с номером множества ответов

Quad quad = new Quad(new Vector2(x1, y1), new Vector2(x2, y2), 3);

// добавляем его в динамический массив ответов

resultQuads.add(quad);

}

}

}

}

1. **Пример работы программы**

**Входные данные:**

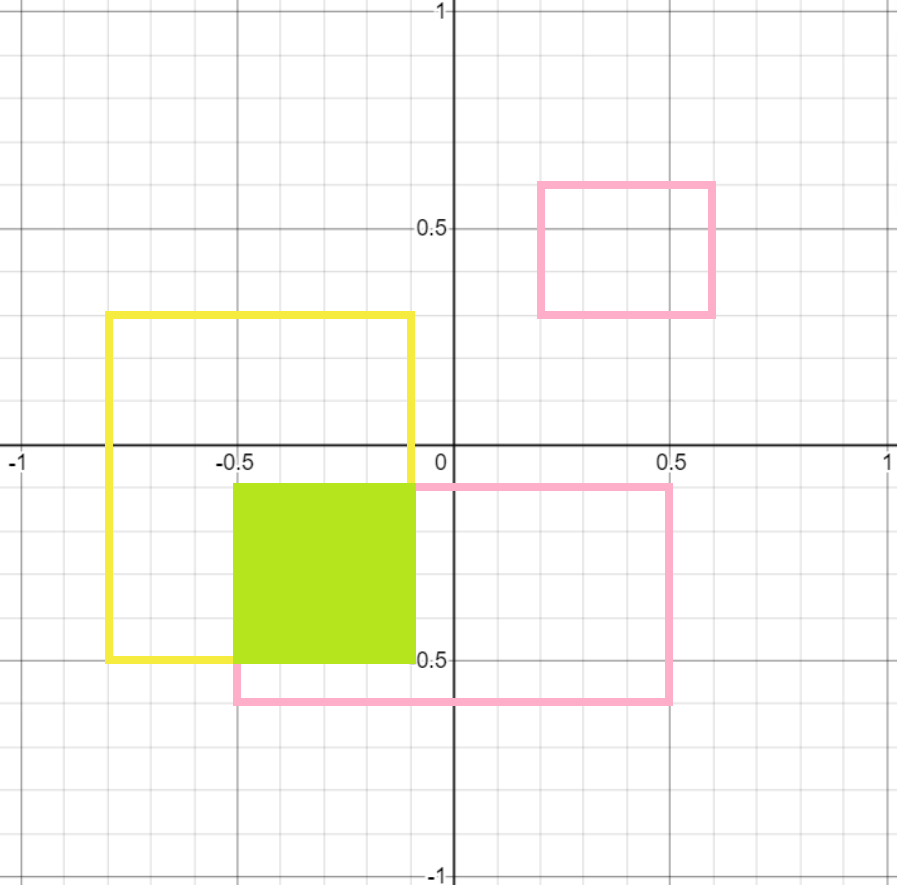
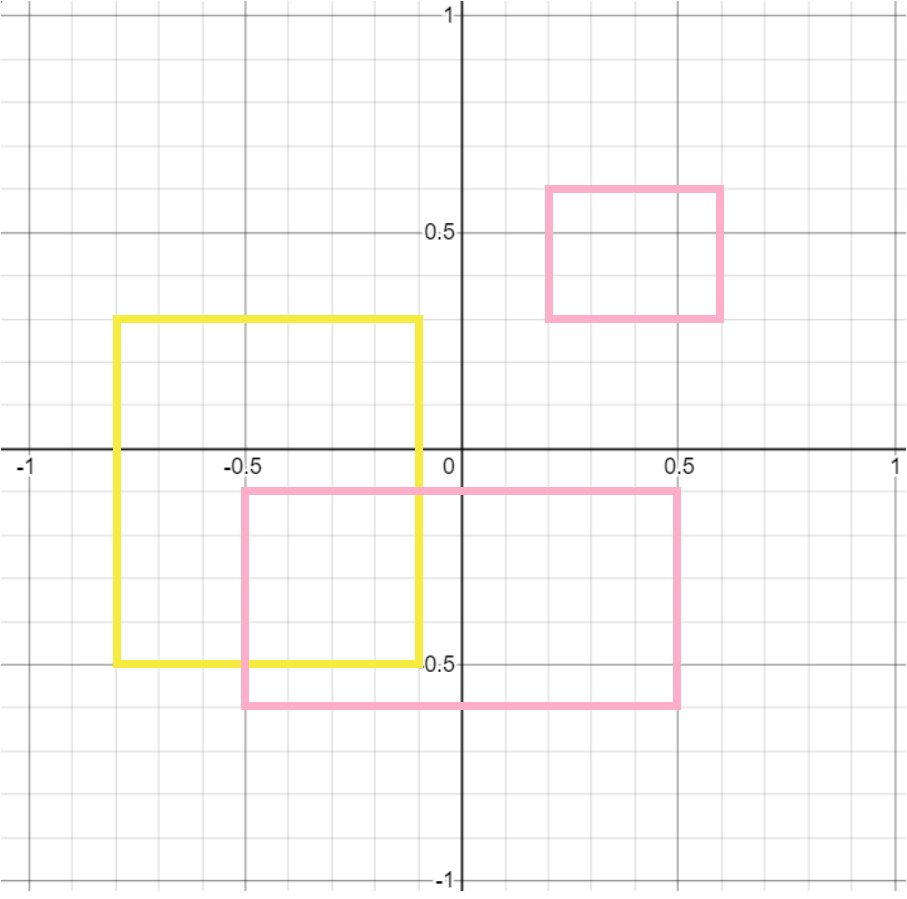
-0.5 -0.1 0.5 -0.6 1

-0.8 -0.5 -0.1 0.3 2

0.6 0.6 0.2 0.3 1

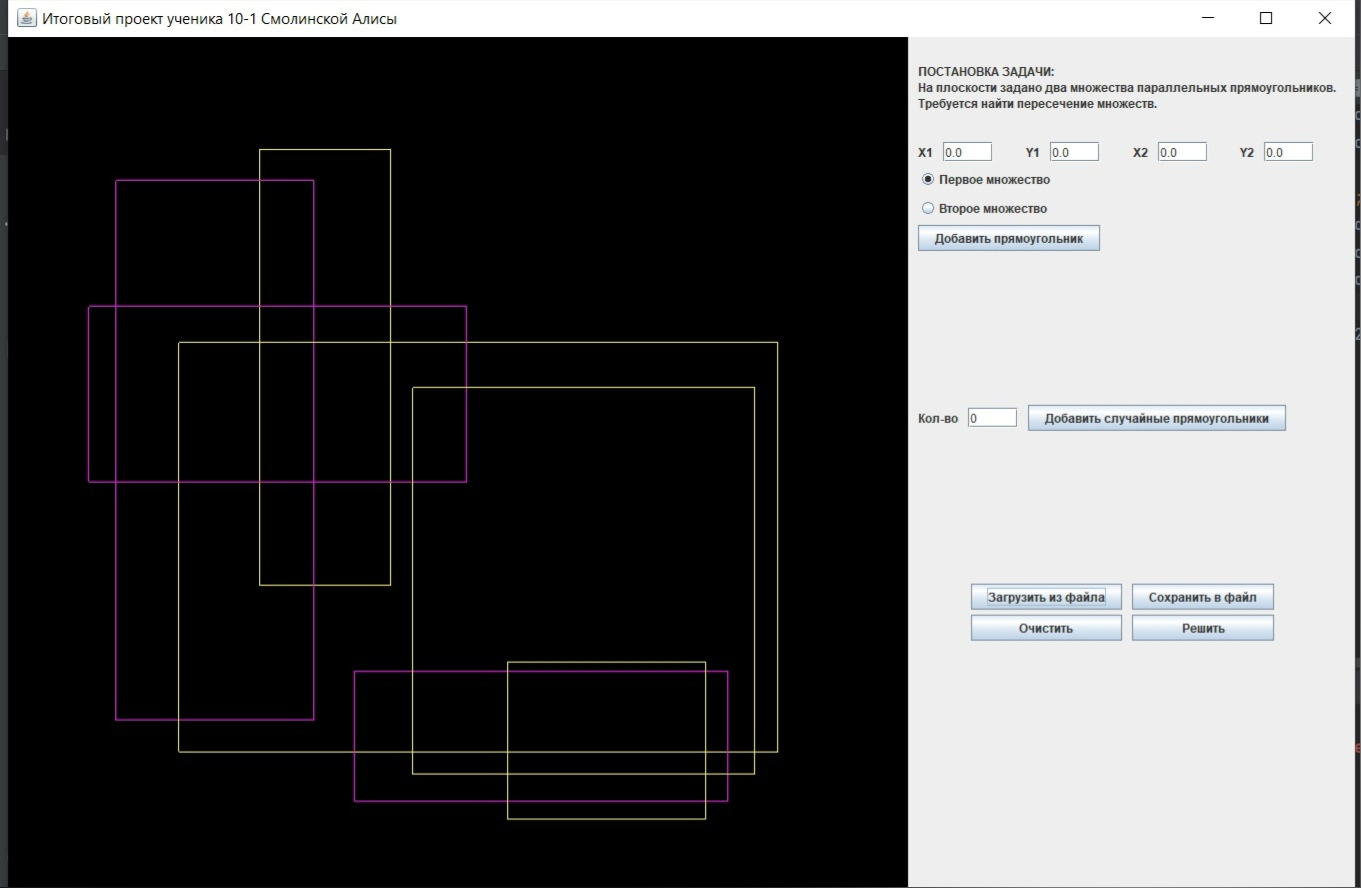
**Выходные данные:**

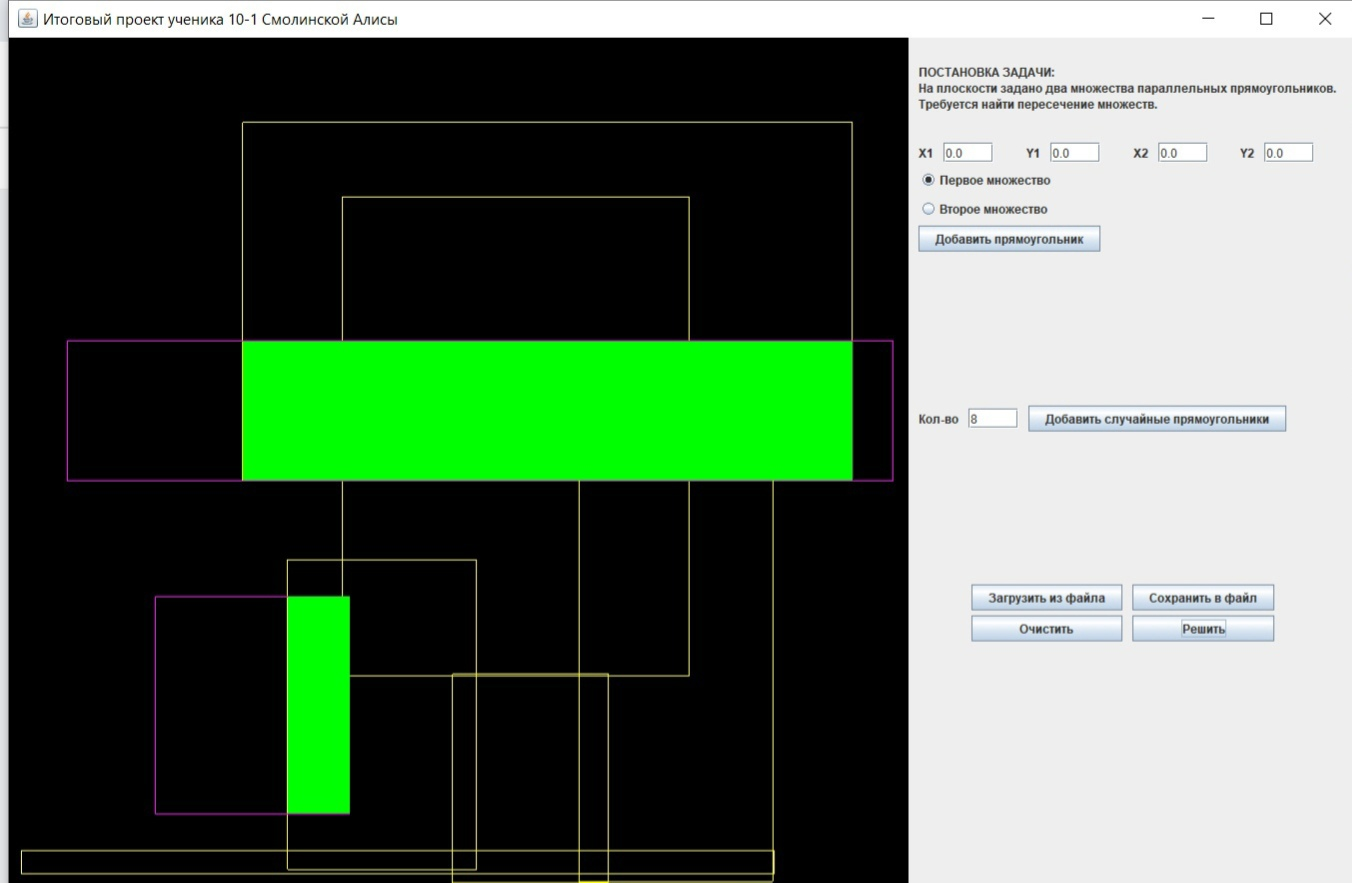
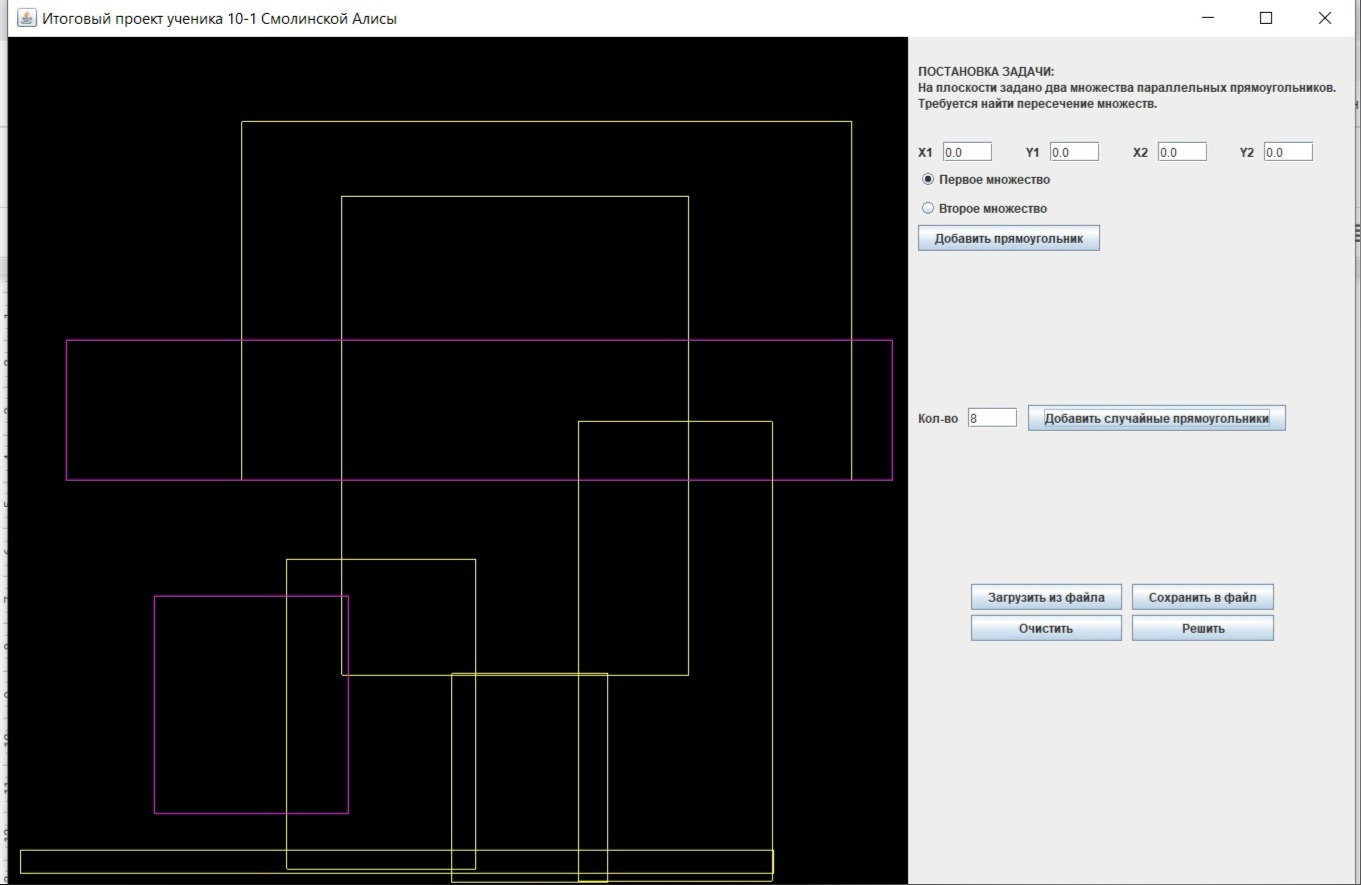
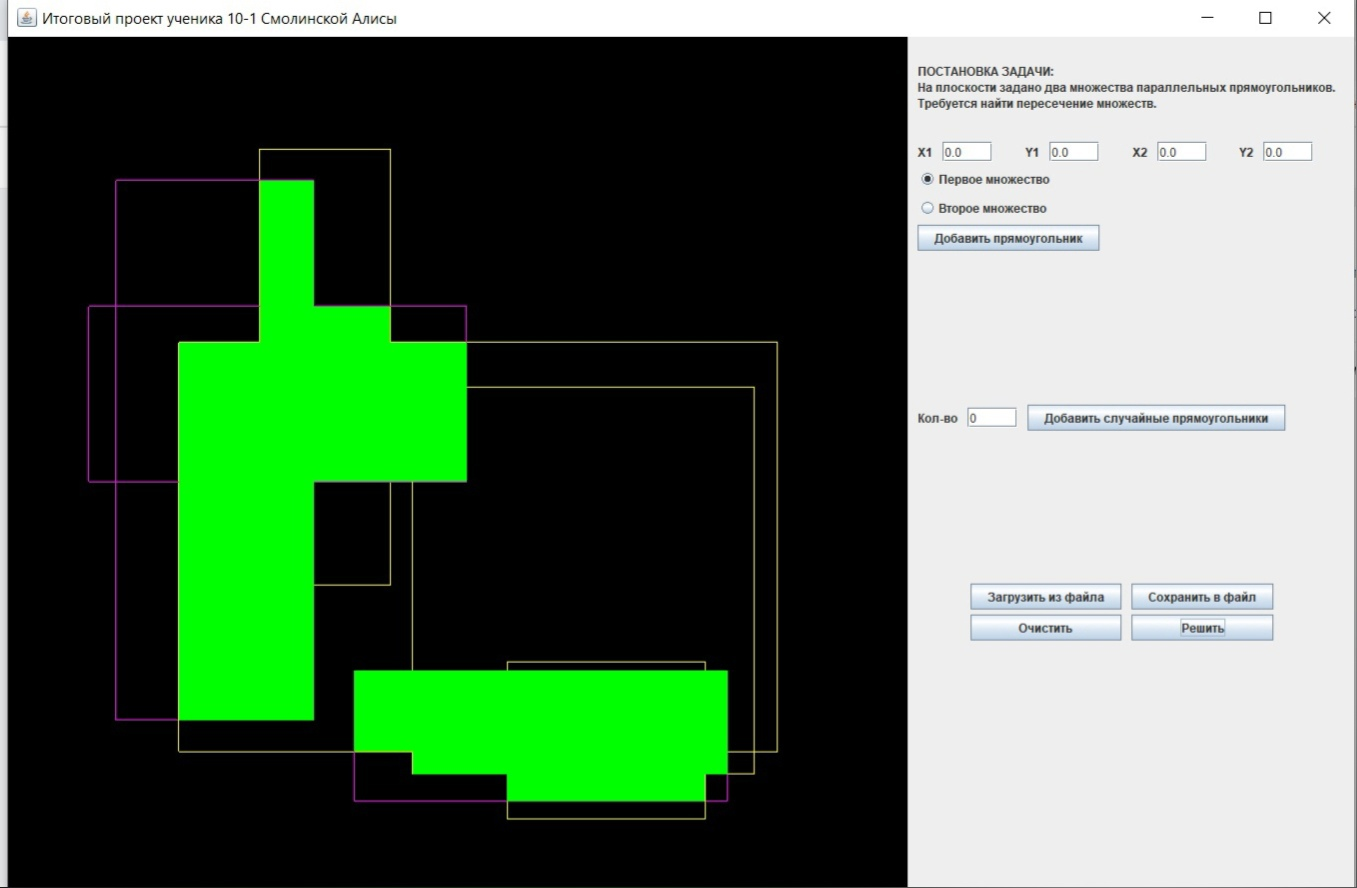
залитая другим цветом область пересечения двух множеств



1. **Анализ правильности решения**

Проверим правильность решения программы в разных случаях расположения друг относительно друга двух прямоугольников





На основании метода решения и приведенных примеров программы выше, можно сделать вывод о том что программа работает верно!