**Государственное бюджетное общеобразовательное учреждение "Президентский физикоматематический лицей № 239"**

**«Нахождение прямой с отрезком максимальной длины**

**внутри «параллельного» эллипса»**

**Годовой проект по информатике.**

Автор: Кармалитова Екатерина, 10-7 класс.

**Постановка задачи:**

« На плоскости задано множество точек, и "параллельный" эллипс. Множество точек образует все возможные прямые, которые могут быть построены парами точек множества. Найти такую прямую (и такие две точки, через которые она проходит), что эта прямая пересекает указанный эллипс, и при этом длина отрезка прямой, находящейся внутри эллипса, максимальна. »

Для полного понимания условия задачи нужно, в первую очередь, ввести понятие «параллельный эллипс». Итак, это эллипс, который задается "параллельным" прямоугольником, в который вписан, который задается двумя точками — противоположными вершинами. Стороны параллельны осям координат. В самой же задаче требуется задать множество точек – вручную или рандомно, с эллипсом все то же. Затем перебрать все точки попарно каждая с каждой, образуя различные прямые, которые будут (или не будут) пересекать эллипс. В процессе перебора необходимо оценивать длину отрезка прямой внутри данной фигуры, запоминая максимальную. В качестве решения выделить найденную прямую, отрезок и две подходящие точки.

**Уточнение исходных и выходных данных:**

Исходные данные: Во-первых, это множество точек, где каждая точка задается с помощью двух координат (тип данных – double). Количество точек не ограничено, начинается от двух. Есть возможность задать все множество вручную, но, так как, на мой взгляд, это не удобно, то присутствует способ задания сразу определенного количества точек рандомным способом.

Во-вторых, «параллельный» эллипс. Его я задаю единственным верным способом – через координаты двух противоположенных вершин прямоугольника (тип данных – double), в который можно вписать эллипс. Единственное требование – заданные две точки не могут иметь одинаковых координат ни по Х, ни по Y – в противном случае они не будут являться противоположенными вершинами прямоугольника. Также можно задать рандомный эллипс – тогда программа сама выберет координаты двух точек вершин случайным образом.

Выходные данные: На выходе программа выдает две подходящие точки множества, которое было задано, и прямую, которая проходит через них – способ задания прямой по двум точкам. На этой же прямой другим цветом выделяется отрезок внутри эллипса, но дополнительных условий ни на что это не накладывает.

**Математическая модель:**

1. Центр эллипса и его полуоси.

Для того, чтобы в дальнейшем определять длину отрезка прямой внутри эллипса, нам нужны координаты его центра и длины полуосей.

X1 = (AX + BX) / 2;

Y1 = (AY + BY) / 2;

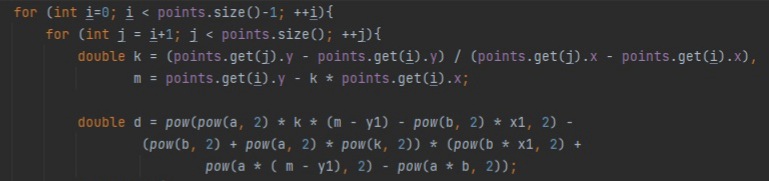
a = (AX – BX) / 2;

b = (AY – BY) / 2;

Где за AX и AY, BX и BY мы принимаем координаты точек, с помощью которых был задан «параллельный» эллипс, a – “ширина”, b – “высота”.

2. Вычисление коэффициентов прямых и дискриминанта для проверки на пересечение прямой и эллипса.

Вот, как это выглядит в виде кода:

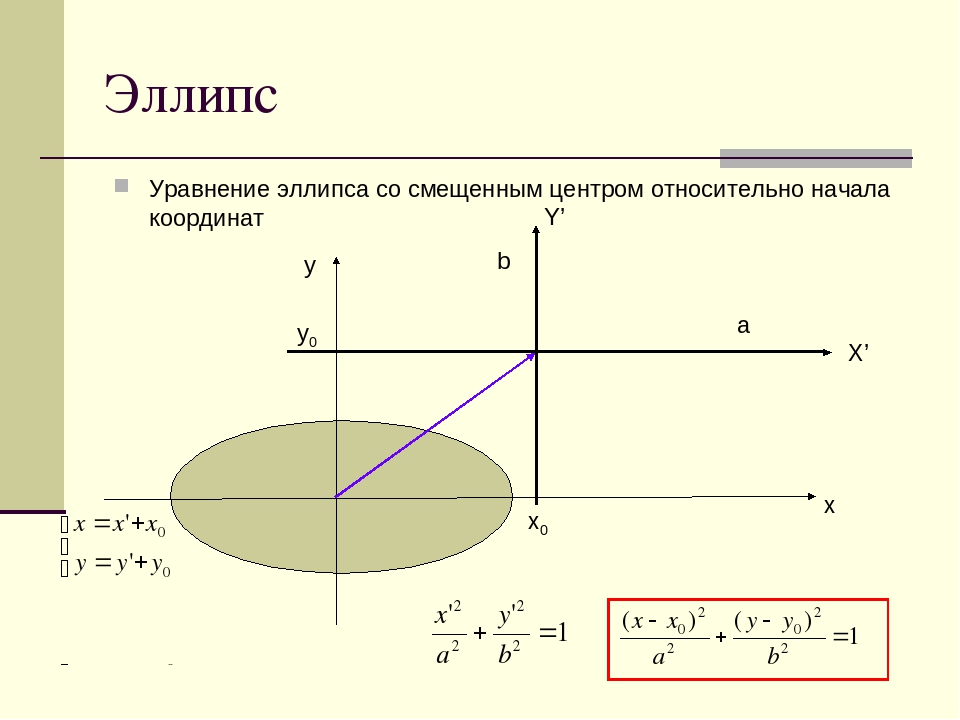


Пусть у нас есть две точки множества: i (x2;y2), j (x3;y3). Тогда взяв за уравнение прямой y = kx + m, находим коэффициенты:

k = (y3 – y2) / (x3 – x2);

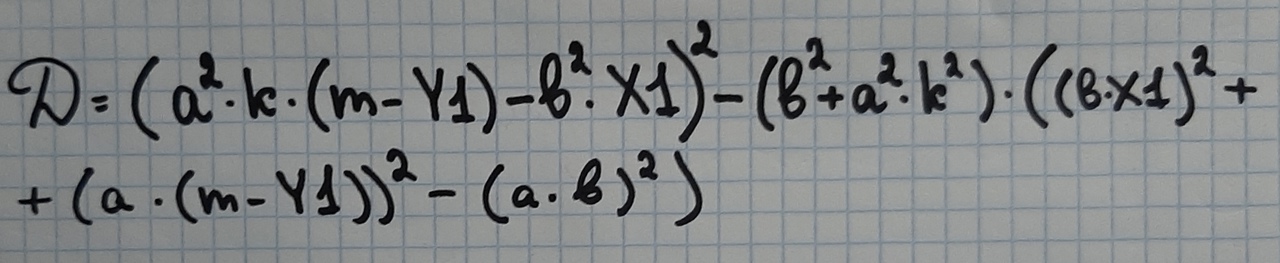
m = y2 – k \* x2;

Далее мы подставляем полученные данные в уравнение эллипса:



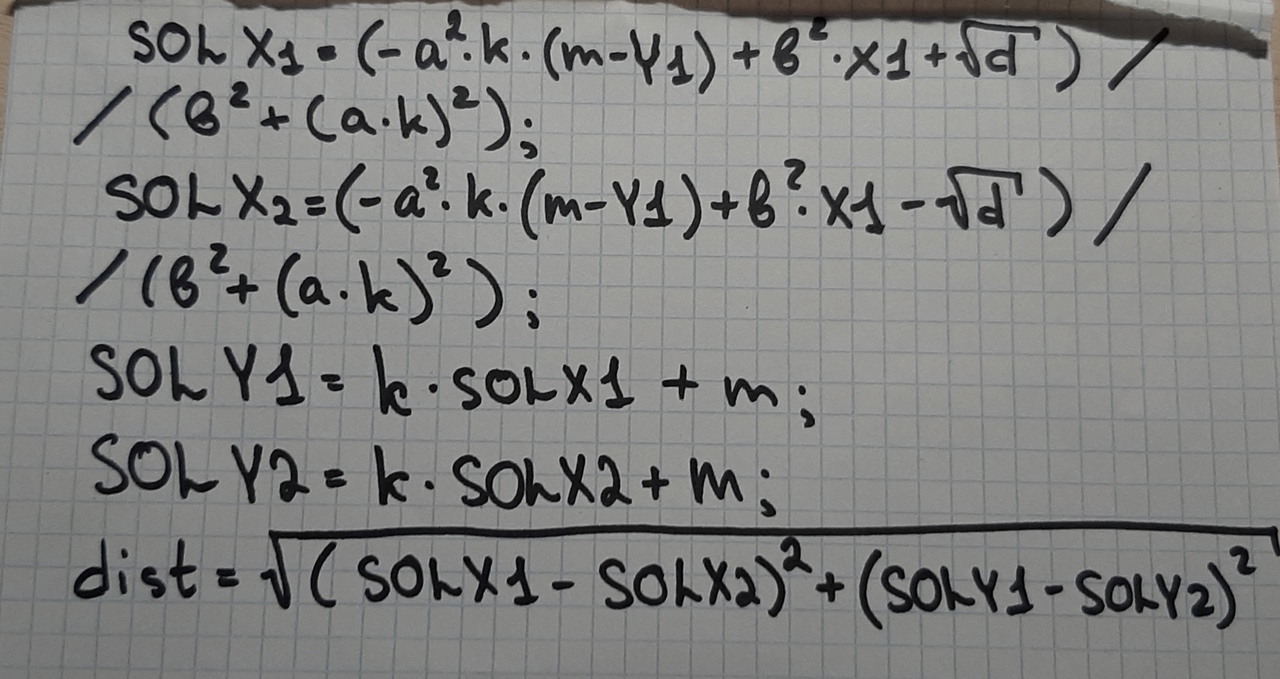
Где на место «y» подставим уравнение получившейся прямой, а на место x0 и y0, найденные раньше X1 и Y1. Получится равенство с одной неизвестной – «x».

Формула дискриминанта будет иметь вид:



Обязательное условие – дискриминант строго больше 0, иначе прямая будет либо просто касаться эллипса в одной точке, либо вообще не пересечет его.

3. Нахождение точек пересечения прямой с эллипсом и длины отрезка внутри эллипса.



Где SOLX1 и SOLY1 – координаты одной точки пересечения, SOLX2 и SOLY2 – координаты второй точки пересечения, а dist – искомая длина отрезка.

**Анализ используемой структуры данных:**

Мне не нужно хранить множество точек. Для этого я создаю динамический массив ArrayList. “На лету” их хранить не получится, потому что в процессе решения мы неоднократно обращаемся к этому множеству. В процессе решения я также храню значение dist – длина отрезка внутри эллипса. Его хранение необходимо, так как, во-первых, это переменное значение, а, во-вторых, является частью ответа. Помимо всего прочего, я храню изначально заданные эллипс, ведь все решение построено на взаимодействии с ним, и координаты двух результативных точек – эти значения тоже меняются по мере работы программы до тех пор, пока не будет достигнут результат, искомый в задаче.

Для того, чтобы иметь возможность нарисовать отрезок внутри эллипса другим цветом, приходится хранить и значения точек, в которых найденная прямая пересекает фигуру. И того, имеется массив из четырех точек – две точки пересечения и две решения.

**Выбор метода решения:**

Первым делом я задаю эллипс и множество точек. Как это сделать я уже описала выше. Теперь, когда исходные данные имеются, я нахожу длины полуосей эллипса и координаты его центра. Далее я начинаю перебирать точки через цикл for. Для каждой пары вычисляются коэффициенты прямой и дискриминант, чтобы проверить на пересечение с эллипсом. Если d оказывается больше 0, то переходим к подсчету длины отрезка прямой внутри эллипса. Сравнивая полученное значение с максимальным, которое изначально принято за ноль, мы либо запоминаем его, как новое максимальное – в этом случае также запоминаются координаты точек, образующих прямую, и точек пересечения с эллипсом; либо не запоминаем.

Когда все пары точек оказываются перебранными, программа рисует найденную прямую, выделяет точки, являющиеся решением, и выделяет другим цветом отрезок внутри фигуры – фактически рисует новую прямую, ограниченную двумя точками, над найденной.

**Комментированный листинг:**

public Point[] solve() { // Начало решения. Point[] maxpoints = new Point[4]; //Массив из 4 точек, которые мы будем хранить. То есть две точки пересечения с эллипсом и две, которые образуют искомую прямую. double max = 0.0d; //Обнуляем максимальную длину отрезка. double a = (ellips.posAX-ellips.posBX) / 2,  b = (ellips.posAY-ellips.posBY) / 2; x1 = (ellips.posAX+ellips.posBX) / 2, y1 = (ellips.posAY+ellips.posBY) / 2;

//Задаем длины полуосей эллипса и координаты его центра. for (int i=0; i < points.size()-1; ++i){ for (int j = i+1; j < points.size(); ++j){

//Перебираем пары точек. double k = (points.get(j).y - points.get(i).y) / (points.get(j).x - points.get(i).x), m = points.get(i).y - k \* points.get(i).x;//Вычисляем коэффициенты прямой через две точки. double d = *pow*(*pow*(a, 2) \* k \* (m - y1) - *pow*(b, 2) \* x1, 2) - (*pow*(b, 2) + *pow*(a, 2) \* *pow*(k, 2)) \* (*pow*(b \* x1, 2) + *pow*(a \* ( m - y1), 2) - *pow*(a \* b, 2));

//Ищем дискриминант для проверки на пересечение. if (d > 0){ //Ставим условие на то, чтобы прямая пересекала эллипс в двух точках. double solX1 = (- *pow*(a, 2) \* k \* (m - y1) + *pow*(b, 2) \* x1 + *sqrt*(d)) / (*pow*(b, 2) + *pow*(a \* k, 2)), solX2 = (- *pow*(a, 2) \* k \* (m - y1) + *pow*(b, 2) \* x1 - *sqrt*(d)) / (*pow*(b, 2) + *pow*(a \* k, 2)), solY1 = k \* solX1 + m, solY2 = k \* solX2 + m, dist = *sqrt*(*pow*(solX1 - solX2, 2) + *pow*(solY1 - solY2, 2));

//Находим две точки пересечения прямой с фигурой и длину отрезка внутри эллипса. if (dist > max){ //Сравниваем найденную длину с максимальной. max = dist;  maxpoints[0] = new Point(solX1, solY1); maxpoints[1] = new Point(solX2, solY2); maxpoints[2] = points.get(i); maxpoints[3] = points.get(j);

//Объявляем новую максимальную длину и запоминаем точки из множества и точки пересечения. } } } } return maxpoints; //Получаем точки решения и пересечения на выход.}

**Пример работы программы:**

Входные данные:

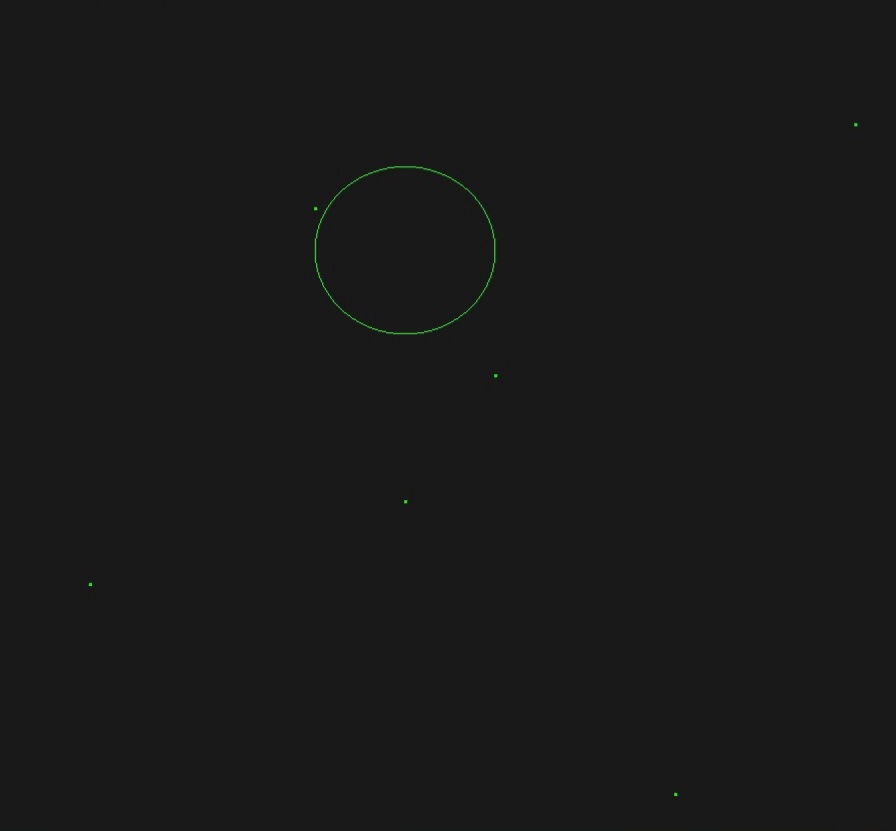
Точки:

(0.1; 0.1), (-0.1; -0.2), (0.9; 0.7),

(-0.3; 0.5), (-0.8; -0.4), (0.5; -0.9).

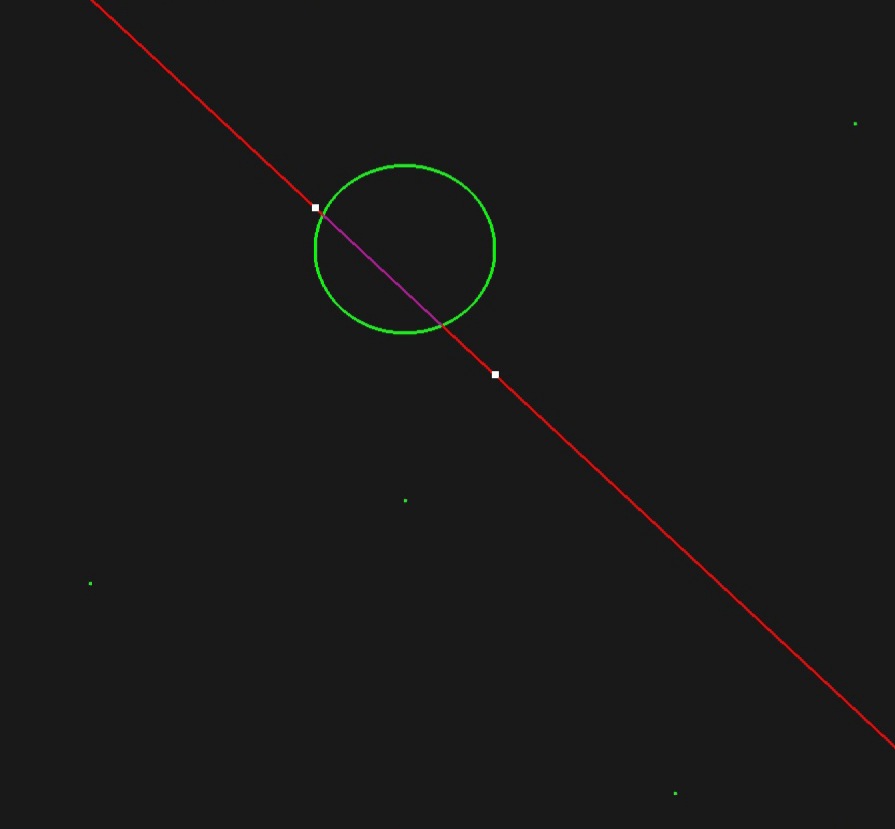
Эллипс:

(0.1; 0.6), (-0.3; 0.2).



Выходные данные:

Все выходные данные нарисованы.

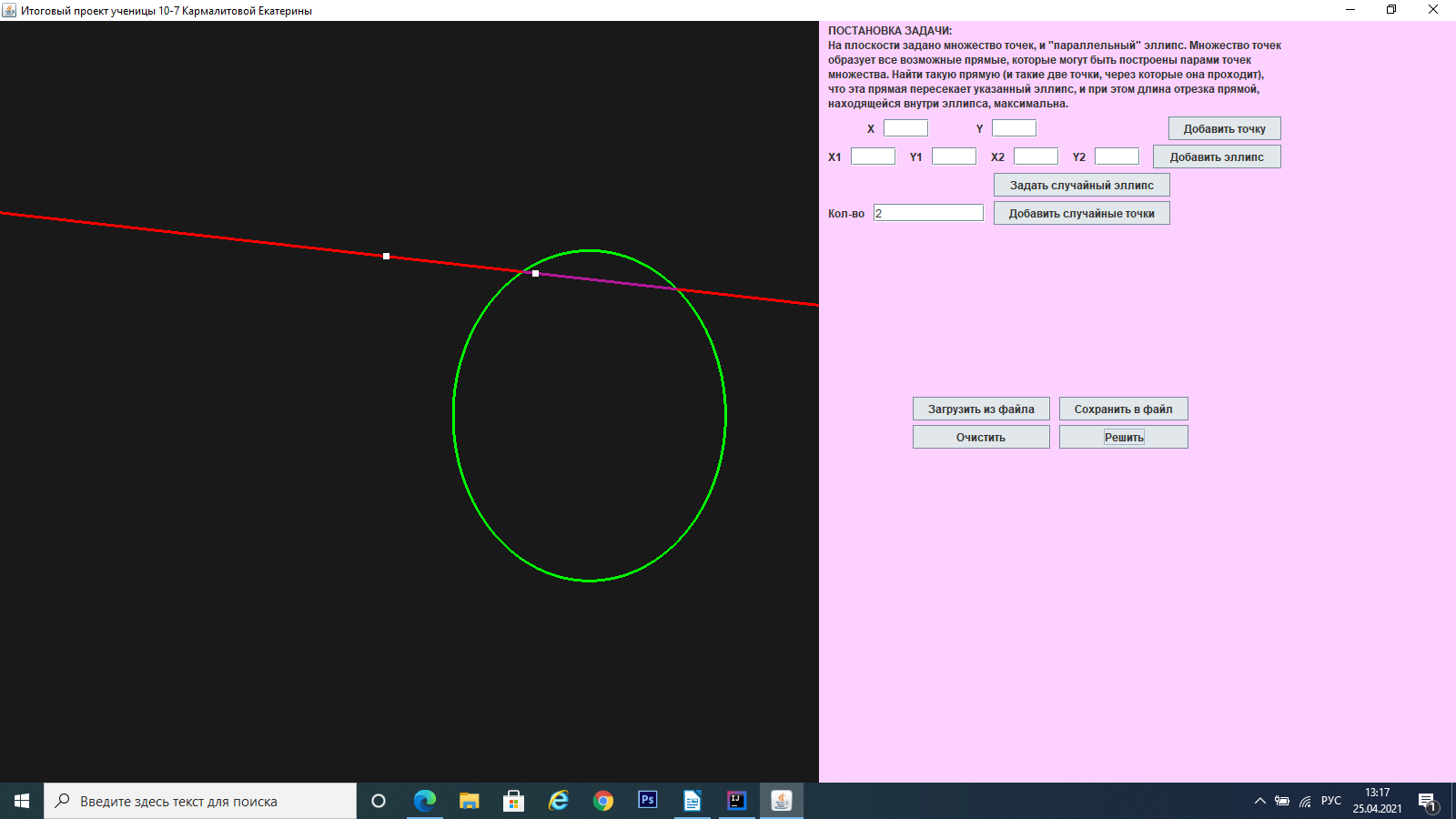


**Анализ правильности решения:**

Программа, очевидно, работает правильно. Решение весьма примитивно и практически не содержит в себе моменты, где можно допустить ошибку.

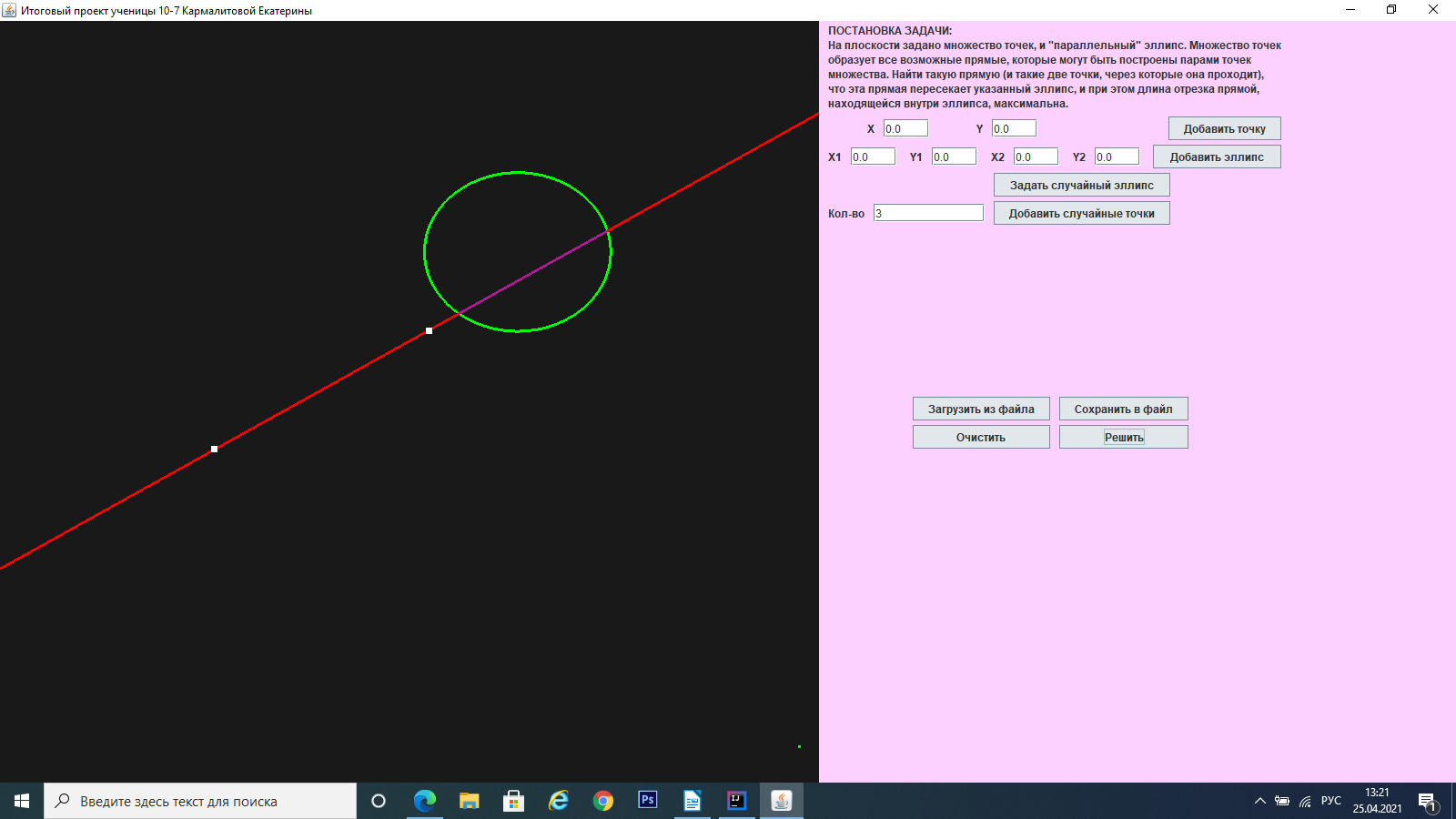
Конечно, на мой взгляд, можно было бы сделать так, что при отсутствии решения программа говорила бы об этом, а не просто ничего не рисовала, но по условию задачи такого требования не стояло, так что я оставила за собой право ничего не выводить в таком случае.

Пример задачи, которая решена очевидно верно:

****

Как мы видим, множество точек изначально содержит только две тоски, причем прямая через которые точно пересекает эллипс. Значит и решение, очевидно, одно. Что и показано тут.

Еще один пример очевидно верного решения:



Тут уже в множестве три точки, но сразу видно, что только одна прямая из трех возможных пересекает эллипс. Она и нарисована.