Report for Assignment 3

Name: Kenan Duan

ID: 1929926

1. Problem statement

In this exercise, the main part is to input some points with their x and y into a txt file. Then, according to the requirement, a class and corresponding function should be created so that these points can be read, check, delete and calculate by the users. Also, another class and corresponding function are aimed to output the parameters of the best fitting line, the distance from the point to the line, best point, worst point and the standard error of each distance.

2. Analysis

a) class FileOp:

Variable:

1) int num

This variable should be public so that all the function in the class are able to access.

Function:

1) void input()

This function allows users input from the keyboard and store the data into the txt file. One line will only store only one point so that it will be easy for the read operation.

2) void show()

This function is able to show all the point in the file and it will be labeled with the number of each point

3) void del()

This function asks the users to input the point from the keyboard. The input point will be deleted from the file.

4) void check()

This function asks the user to input the desired point from the keyboard. It will check whether the input is in the file. If it does, this function will print out the sequence number of the point; otherwise, it will print out this point is not in the file.

5) void checkAndDel()

The method of this function is similar to the fourth function: void check(). The only difference is that this function will delete the point from the file if it indeed in the file.

b) class LineFitting:

Variable:

1) double a, b, a1, a2, b1, b2, d1, d2

These are the parameters when calculating the best fitting line equation.

2) int length = -1

This integer is the number of points in the txt file. Since there is no inheritance between this class and class FileOp, the number of points should be get at very beginning. However, the while loop for .eof() function will do one more loop than the number of points. Therefore, the initialization of this integer will be -1.

3) double mean = 0

This variable will record the mean of the distance.

Function:

a) void getPri()

This function will get all the parameters according to the equation.

b) void showEqu()

After getting a and b from the last function, this will simply print out the equation in the form of ax+by+c=0.

c) void calDis()

This function will calculate all the distance of each point to the best fitting line. It also has the ability to print out a certain distance of a certain point that user input.

d) void bestPoint()

All the distance of each point will be stored in a double array. Then by using a loop to get the minimal distance and print out the position.

e) void worstPoint()

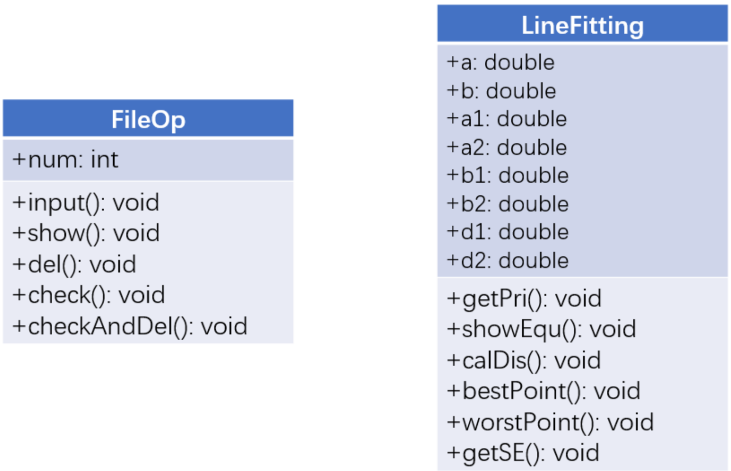
Same way as the bestPoint(), but to get the maximal distance.

f) void getSE()

Get all the distance and calculate the standard error according to the formula.

3. Design

3.1 ER diagram



3.2 Specific design

a) **class FileOp:**

1) void input()

Since "cin" cannot recognize space, the value of x and y are input separately. They will be stored in the file and separate by the space, which allows the following operation become easier.

2) void show()

Create a fstream and use ">>" to output the point from the file.

3) void delete()

Due to the same reason, the value of x and y of the point that user want to delete will be separately input. After the input, the input point will be converted into a string temp1. Read the txt file by getline() function and also convert it into a string temp2. Use a for loop to get each line in the txt file and compare temp1 and temp2. If they are not the same, the content in the original file will be transferred into another temporary txt file; otherwise, it will not. After the whole loop, the content in the temp txt file will transferred in to the original file and the temporary file will be deleted.

4) void check()

It is as the same principle as the delete operation. Convert the input point into a string variable temp1. And use .getline() to get each point from the file. Add a counter into the loop to show the position of the point in the file. If it is in the file, then it will output the position; otherwise, it will print out "The input point is not in the file.".

5) void checkAndDel()

This function is just a combination of check() and delete(). It will check the input point whether it is in the file and delete it if it is.

b) **class LineFitting:**

1) void getPri()

First, get the number of point in the file so that this number can be used in the loop. Then create two arrays to store the values of x and y separately. After that, according to the formula gain the parameters.

2) void showEqu()

This function is to output the equation with the parameters a, b and c. However, when the best fitting line is calculated, it is in the form of y = ax + b. Therefore, the coefficient of y will always be -1.

3) void calDis()

Create an array to store each distance generated by each point. This function will also ask user to input the point number to check the certain distance for a certain point.

4) void bestPoint()

Create an array to store each distance generated by each point. Use a for loop to find put which point generate a minimal distance and do the count. The count variable will be the position of the point. Print out the position and the distance of the best point.

5) void worstPoint()

Create an array to store each distance generated by each point. Use a for loop to find put which point generate a maximal distance and do the count. The count variable will be the position of the point. Print out the position and the distance of the worst point.

6) void getSE()

Create an array to store each distance generated by each point. Get the sum of all the distance in order to calculate the mean of distance. Calculate the standard error of distance for all the points according to the formula.

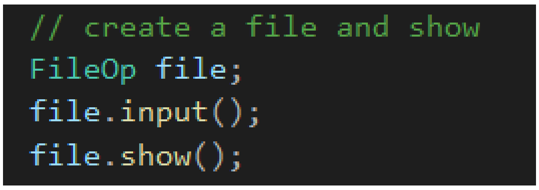
4. Implementation

File name: "main.cpp" in the file "199926".

5. Testing

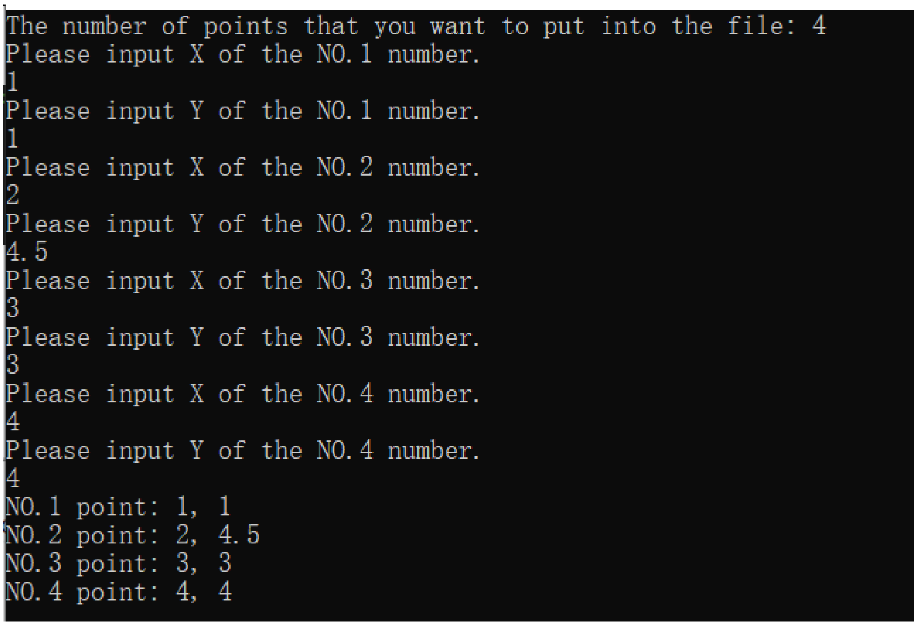
1) Create a text file using the “FileOp” class with some points

Source code:



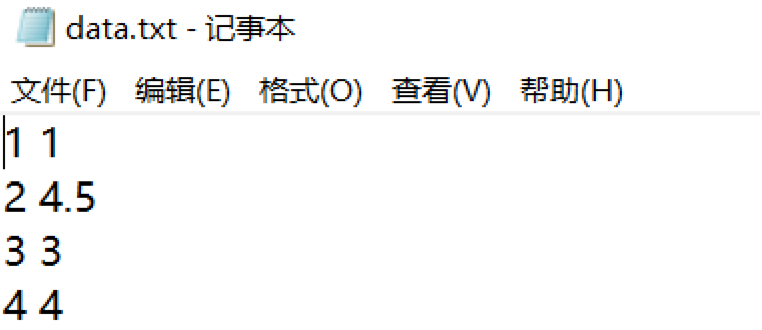
```
// create a file and show
FileOp file;
file.input();
file.show();
```

The result:



```
The number of points that you want to put into the file: 4
Please input X of the NO.1 number.
1
Please input Y of the NO.1 number.
1
Please input X of the NO.2 number.
2
Please input Y of the NO.2 number.
4.5
Please input X of the NO.3 number.
3
Please input Y of the NO.3 number.
3
Please input X of the NO.4 number.
4
Please input Y of the NO.4 number.
4
NO.1 point: 1, 1
NO.2 point: 2, 4.5
NO.3 point: 3, 3
NO.4 point: 4, 4
```

data.txt file:

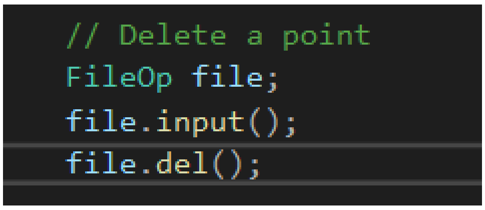


```
1 1
2 4.5
3 3
4 4
```

The result indicates that it can allow users to input points from the keyboard and save these points into a text file and read points from the text file.

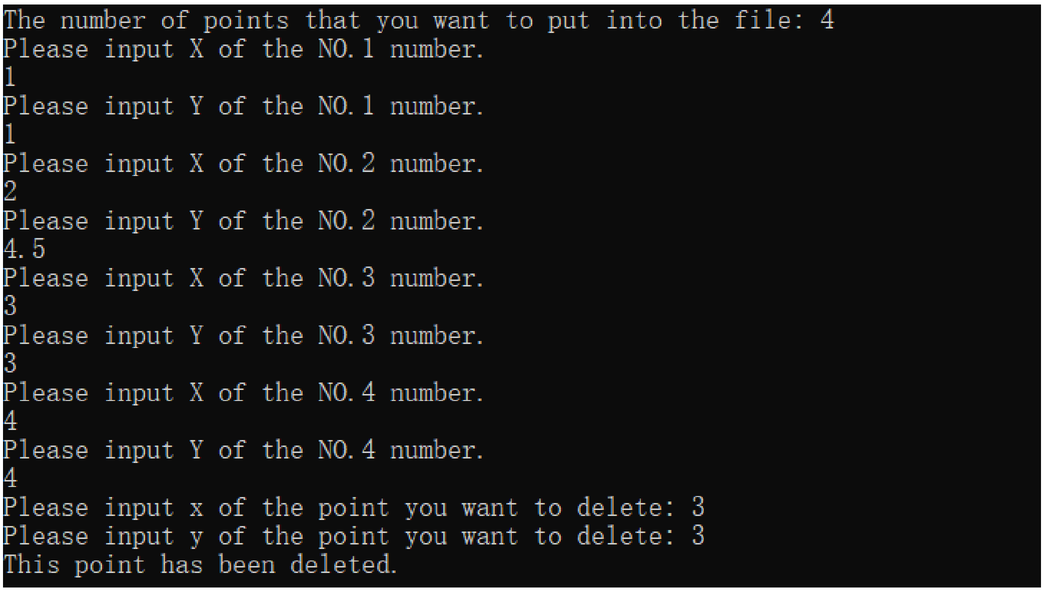
2) Delete a point from the text file

Source code:



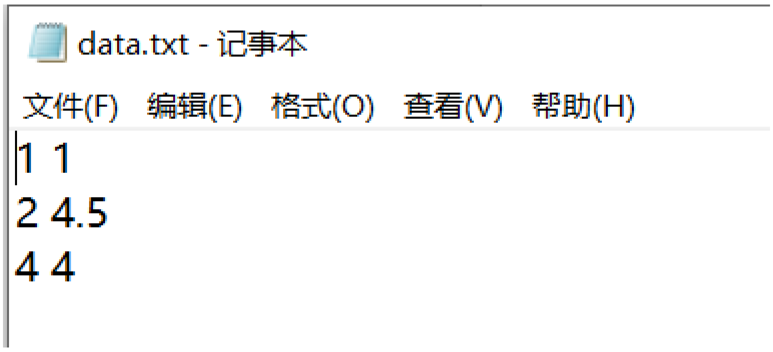
```
    // Delete a point
    FileOp file;
    file.input();
    file.del();
```

The result:



```
The number of points that you want to put into the file: 4
Please input X of the NO.1 number.
1
Please input Y of the NO.1 number.
1
Please input X of the NO.2 number.
2
Please input Y of the NO.2 number.
4.5
Please input X of the NO.3 number.
3
Please input Y of the NO.3 number.
3
Please input X of the NO.4 number.
4
Please input Y of the NO.4 number.
4
Please input x of the point you want to delete: 3
Please input y of the point you want to delete: 3
This point has been deleted.
```

data.txt file:

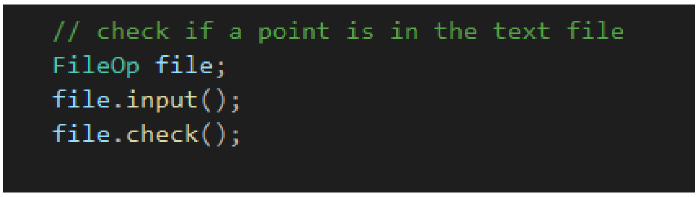


```
1 1
2 4.5
4 4
```

The result indicates that delete a point from the text file.

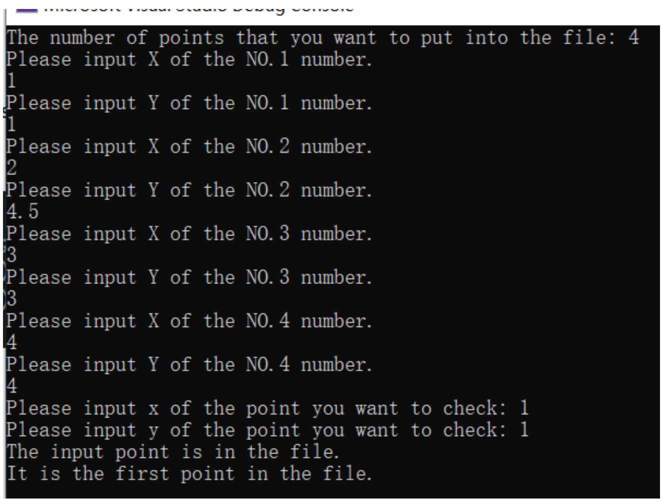
3) check if a point is in the text file

source code:

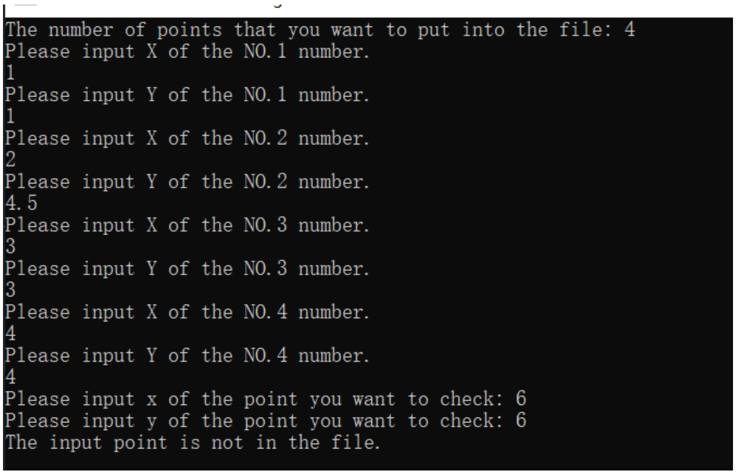


```
// check if a point is in the text file
FileOp file;
file.input();
file.check();
```

The result:

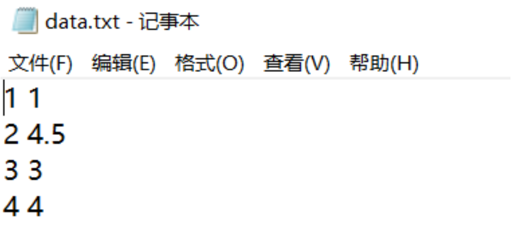


```
The number of points that you want to put into the file: 4
Please input X of the NO.1 number.
1
Please input Y of the NO.1 number.
1
Please input X of the NO.2 number.
2
Please input Y of the NO.2 number.
4.5
Please input X of the NO.3 number.
3
Please input Y of the NO.3 number.
3
Please input X of the NO.4 number.
4
Please input Y of the NO.4 number.
4
Please input x of the point you want to check: 1
Please input y of the point you want to check: 1
The input point is in the file.
It is the first point in the file.
```



```
The number of points that you want to put into the file: 4
Please input X of the NO.1 number.
1
Please input Y of the NO.1 number.
1
Please input X of the NO.2 number.
2
Please input Y of the NO.2 number.
4.5
Please input X of the NO.3 number.
3
Please input Y of the NO.3 number.
3
Please input X of the NO.4 number.
4
Please input Y of the NO.4 number.
4
Please input x of the point you want to check: 6
Please input y of the point you want to check: 6
The input point is not in the file.
```

data.txt file:

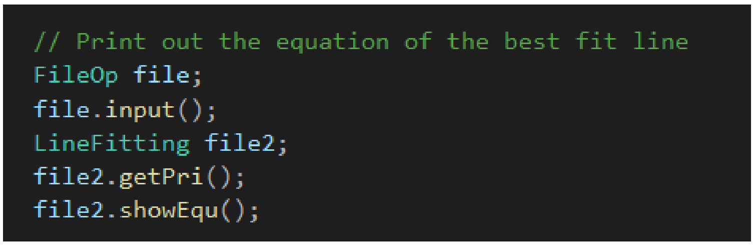


```
1 1
2 4.5
3 3
4 4
```

The result indicates that it can check if a point is in the text file.

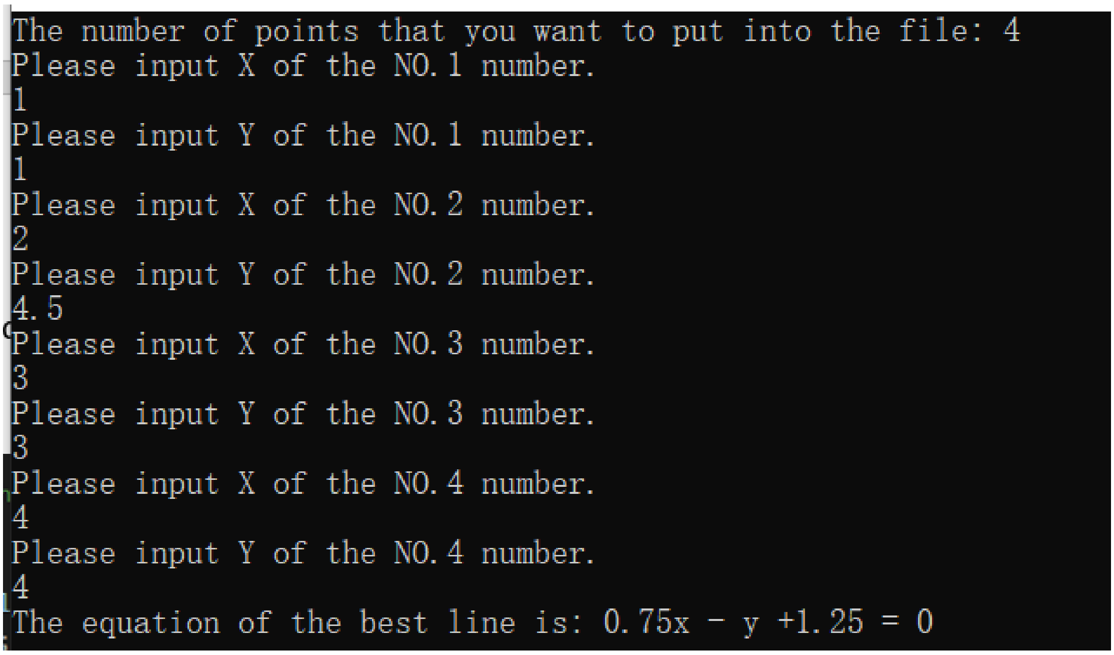
4) Print out the equation of the best fit line

Source code:



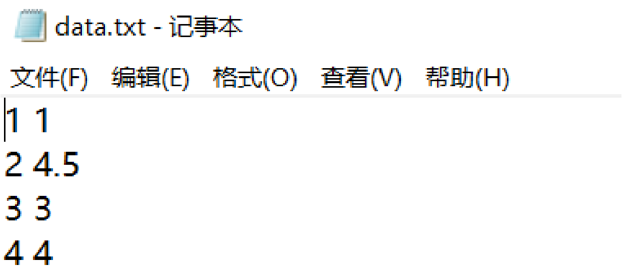
```
// Print out the equation of the best fit line
FileOp file;
file.input();
LineFitting file2;
file2.getPri();
file2.showEqu();
```

The result:



```
The number of points that you want to put into the file: 4
Please input X of the NO.1 number.
1
Please input Y of the NO.1 number.
1
Please input X of the NO.2 number.
2
Please input Y of the NO.2 number.
4.5
Please input X of the NO.3 number.
3
Please input Y of the NO.3 number.
3
Please input X of the NO.4 number.
4
Please input Y of the NO.4 number.
4
The equation of the best line is: 0.75x - y +1.25 = 0
```

data.txt file:

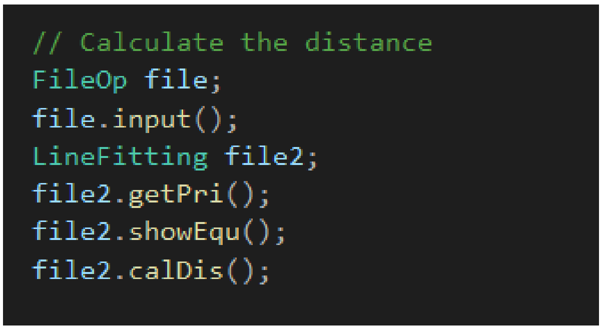


```
1 1
2 4.5
3 3
4 4
```

The result indicates that it can show the best fitting line equation.

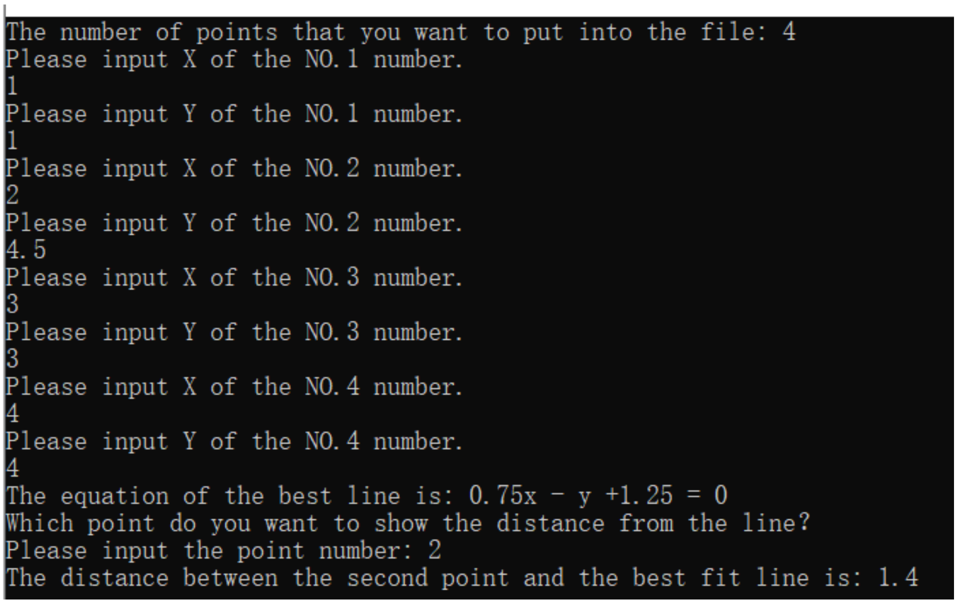
5) Calculate the distance between each point in your file and the best fit line

Source code:



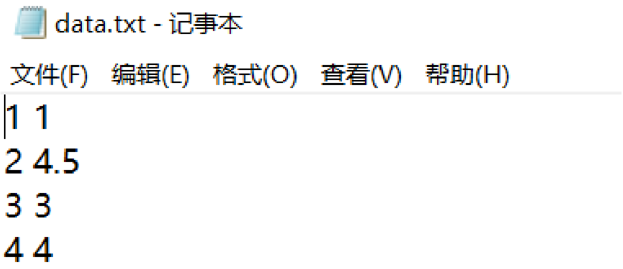
```
// Calculate the distance
FileOp file;
file.input();
LineFitting file2;
file2.getPri();
file2.showEqu();
file2.calDis();
```

The result:



```
The number of points that you want to put into the file: 4
Please input X of the NO.1 number.
1
Please input Y of the NO.1 number.
1
Please input X of the NO.2 number.
2
Please input Y of the NO.2 number.
4.5
Please input X of the NO.3 number.
3
Please input Y of the NO.3 number.
3
Please input X of the NO.4 number.
4
Please input Y of the NO.4 number.
4
The equation of the best line is: 0.75x - y +1.25 = 0
Which point do you want to show the distance from the line?
Please input the point number: 2
The distance between the second point and the best fit line is: 1.4
```

data.txt file:

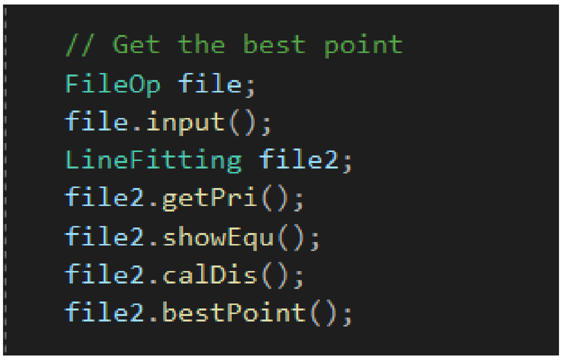


```
1 1
2 4.5
3 3
4 4
```

The result indicates that it can calculate the distance between each point in file and the best fit line.

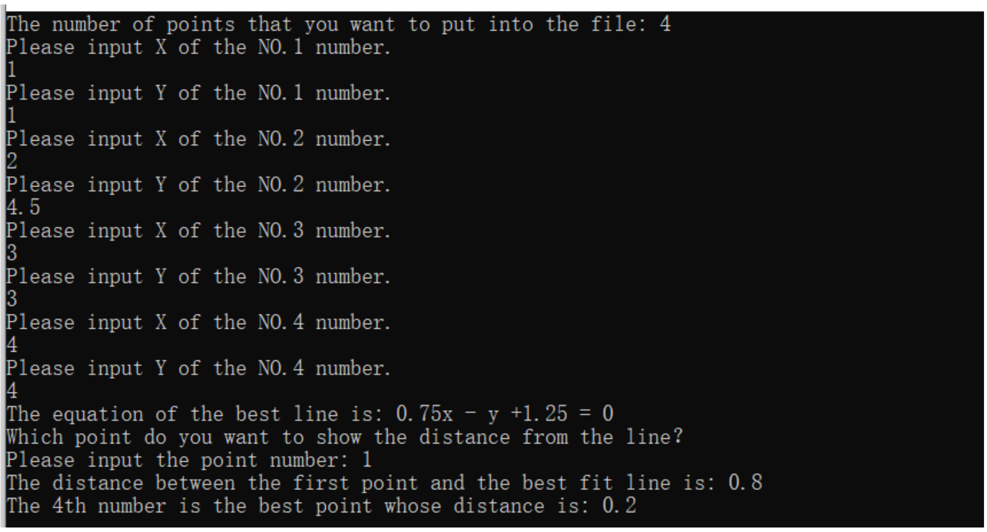
6) Get the best point

Source code:



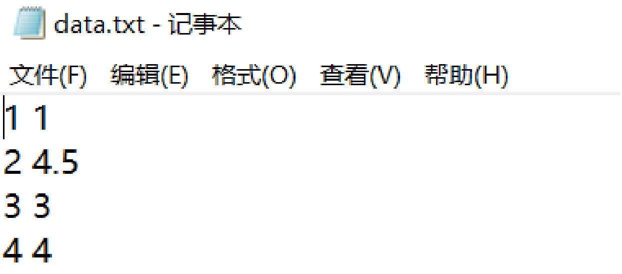
```
// Get the best point
FileOp file;
file.input();
LineFitting file2;
file2.getPri();
file2.showEqu();
file2.calDis();
file2.bestPoint();
```

The result:



```
The number of points that you want to put into the file: 4
Please input X of the NO.1 number.
1
Please input Y of the NO.1 number.
1
Please input X of the NO.2 number.
2
Please input Y of the NO.2 number.
4.5
Please input X of the NO.3 number.
3
Please input Y of the NO.3 number.
3
Please input X of the NO.4 number.
4
Please input Y of the NO.4 number.
4
The equation of the best line is: 0.75x - y +1.25 = 0
Which point do you want to show the distance from the line?
Please input the point number: 1
The distance between the first point and the best fit line is: 0.8
The 4th number is the best point whose distance is: 0.2
```

data.txt file:

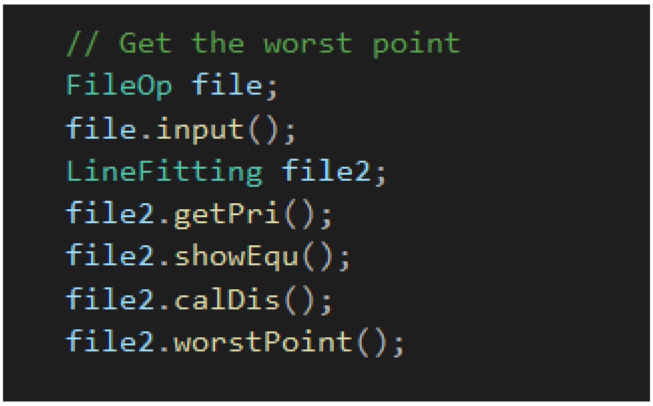


```
1 1
2 4.5
3 3
4 4
```

The result indicates that it can get the best point.

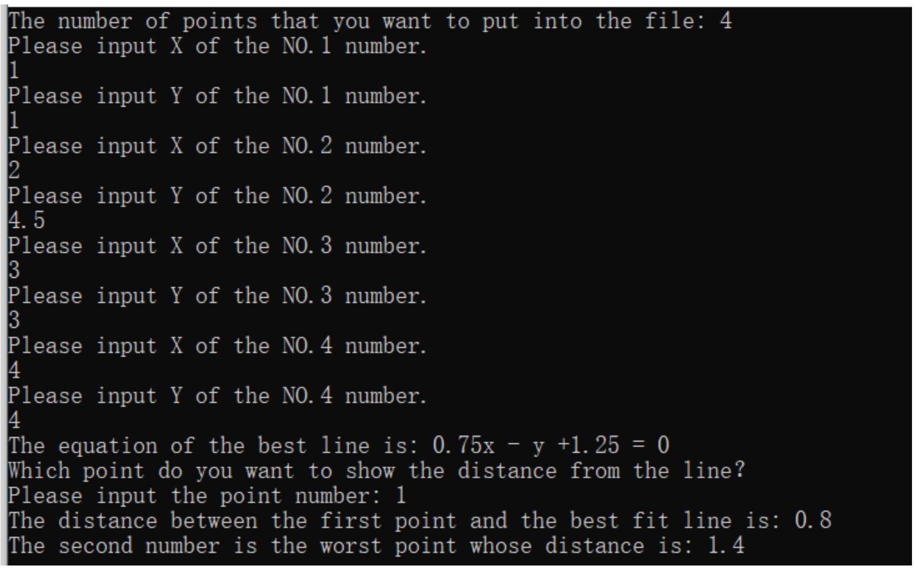
7) Get the worst point

Source code:



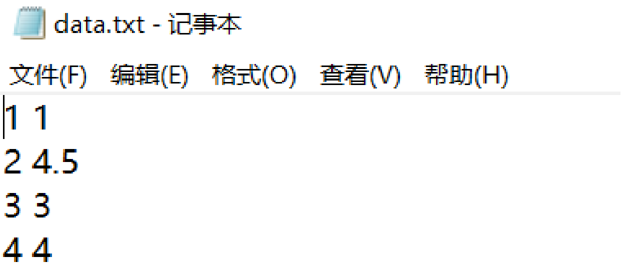
```
    // Get the worst point
    FileOp file;
    file.input();
    LineFitting file2;
    file2.getPri();
    file2.showEqu();
    file2.calDis();
    file2.worstPoint();
```

The result:



```
The number of points that you want to put into the file: 4
Please input X of the NO.1 number.
1
Please input Y of the NO.1 number.
1
Please input X of the NO.2 number.
2
Please input Y of the NO.2 number.
4.5
Please input X of the NO.3 number.
3
Please input Y of the NO.3 number.
3
Please input X of the NO.4 number.
4
Please input Y of the NO.4 number.
4
The equation of the best line is: 0.75x - y +1.25 = 0
Which point do you want to show the distance from the line?
Please input the point number: 1
The distance between the first point and the best fit line is: 0.8
The second number is the worst point whose distance is: 1.4
```

data.txt file:

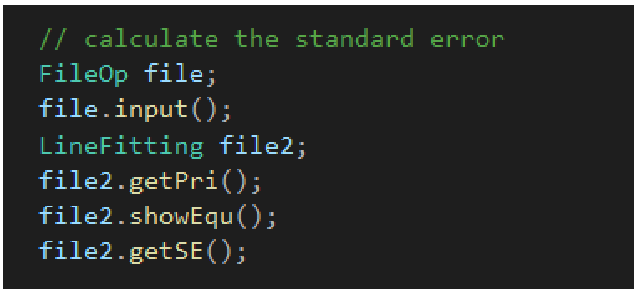


```
1 1
2 4.5
3 3
4 4
```

The result indicates that it can get the worst point.

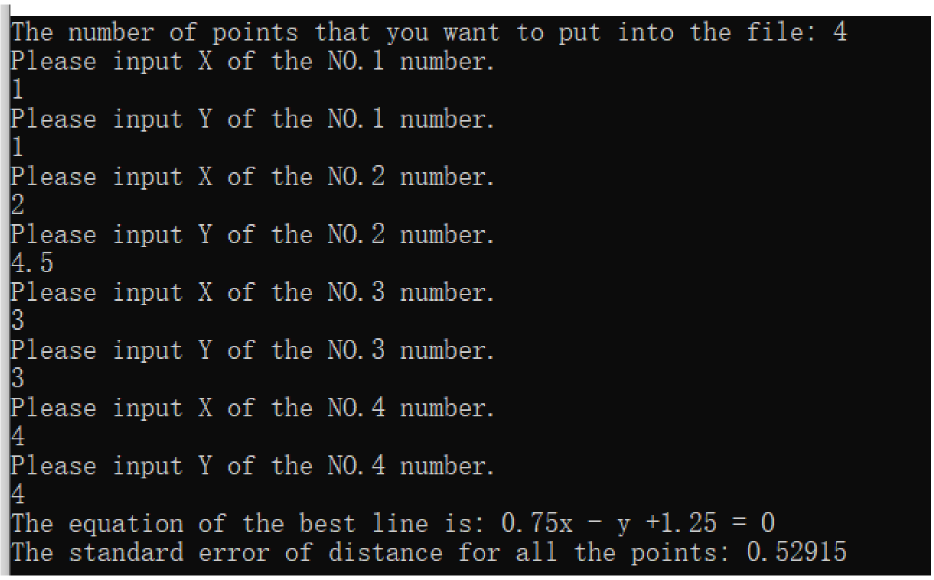
8) calculate the standard error of distance for all the points

source code:



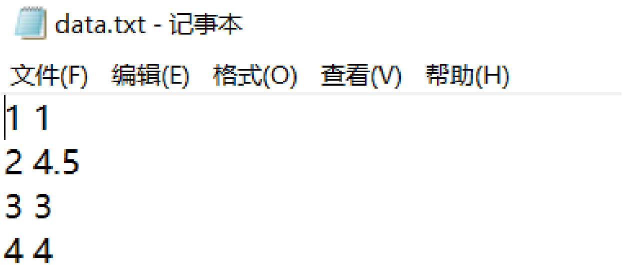
```
// calculate the standard error
FileOp file;
file.input();
LineFitting file2;
file2.getPri();
file2.showEqu();
file2.getSE();
```

The result:



```
The number of points that you want to put into the file: 4
Please input X of the NO.1 number.
1
Please input Y of the NO.1 number.
1
Please input X of the NO.2 number.
2
Please input Y of the NO.2 number.
4.5
Please input X of the NO.3 number.
3
Please input Y of the NO.3 number.
3
Please input X of the NO.4 number.
4
Please input Y of the NO.4 number.
4
The equation of the best line is: 0.75x - y +1.25 = 0
The standard error of distance for all the points: 0.52915
```

data.txt file:

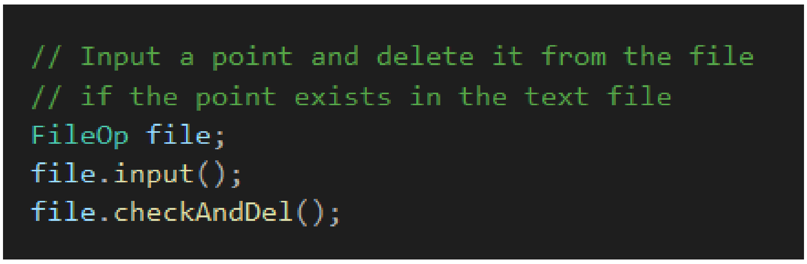


```
1 1
2 4.5
3 3
4 4
```

The result indicates that it can get the standard error of distance for all the points.

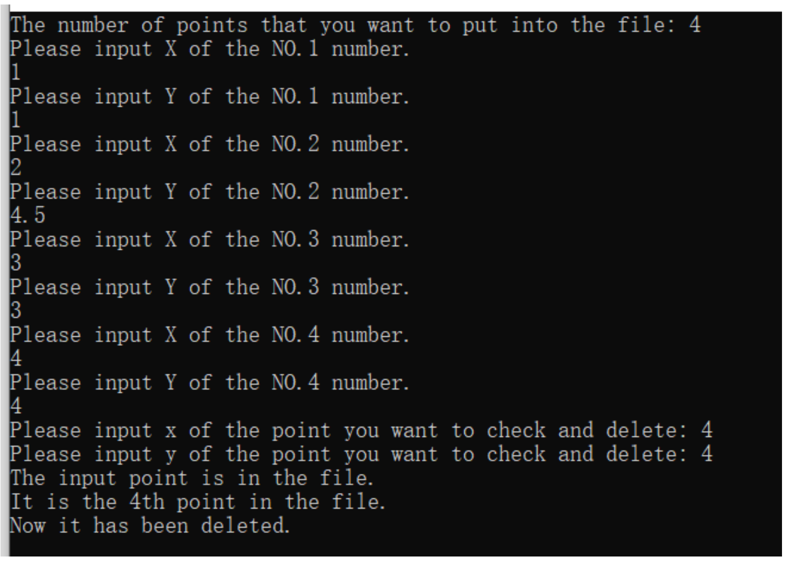
9) Input a point and delete it from the file if the point exists in the text file

Source code:



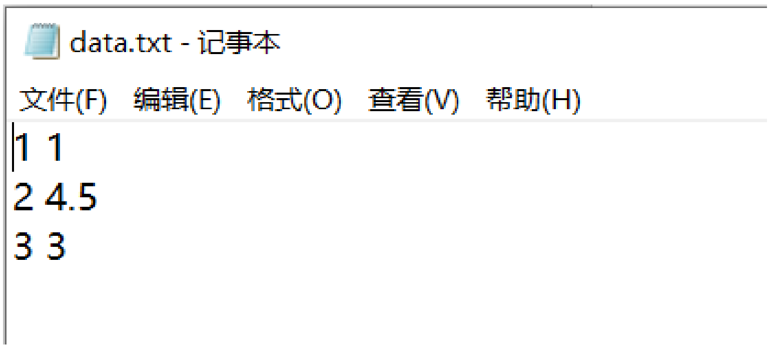
```
// Input a point and delete it from the file
// if the point exists in the text file
FileOp file;
file.input();
file.checkAndDel();
```

The result:



```
The number of points that you want to put into the file: 4
Please input X of the NO.1 number.
1
Please input Y of the NO.1 number.
1
Please input X of the NO.2 number.
2
Please input Y of the NO.2 number.
4.5
Please input X of the NO.3 number.
3
Please input Y of the NO.3 number.
3
Please input X of the NO.4 number.
4
Please input Y of the NO.4 number.
4
Please input x of the point you want to check and delete: 4
Please input y of the point you want to check and delete: 4
The input point is in the file.
It is the 4th point in the file.
Now it has been deleted.
```

Data.txt file:



```
1 1
2 4.5
3 3
```

The result indicates that it can input a point and delete it from the file if the point exists in the text file.