

## ***1. Условие задачи***

На плоскости задано множество точек из  $A$  и отрезок  $BC$ . Найти такую точку из  $A$ , чтобы расстояние от нее до отрезка  $BC$  было минимальным

## ***2. Описание алгоритма решения и структур хранения данных***

Для нахождения ближайшей точки будем считать расстояние от каждой точки до прямой по ближайшему пути

Для нахождения ближайшего пути разобьем нахождение пути на 3 условия:

1) Прямая лежит на оси  $x = x$  точки

Тогда считаем расстояние как  $- X1 - X2$

2) Прямая лежит на оси  $y = y$  точки

Тогда считаем расстояние как  $- Y1 - Y2$

3) Прямая не лежит на оси

Тогда считаем расстояние по следующей схеме:

Строим треугольник с 3 точками, где прямая является основанием треугольника, а точка в пространстве вершиной

Находим длины сторон треугольника. Находим полупериметр треугольника

Находим площадь треугольника

Находим высоту треугольника это и будет кратчайший путь от точки до прямой

Перебирая длины сравниваем их с минимальной длиной которая уже была и если нужно переприсваиваем значения и по завершению программы выводим её

Так как мы работаем в пространстве для точного расчета будем использовать числовой тип данных с плавающей точкой - `double`

А также тип `pair` для записи координат точек в 1 переменную. Таким образом используем следующие переменные и типы :

Переменные :

`min_point`, `point`, `cord_line_one`, `cord_line_two` - типа `pair` - для записи 2 координат точки

`flag` - типа `int` - для проверки была ли уже минимальная длина

`min` - типа `double` - для записи длины

Функции :

`double` `length` - функция принимает 2 переменные типа `pair` с координатами точек для расчета длины линии. Возвращает длину линии

`double` `min_lenght_triangle` - функция принимает 3 переменных типа `pair` с координатами точек, 1 координаты - точка из A, 2 и 3 координаты - точки прямой. Функция считает высоту треугольника по алгоритму описанному выше. Возвращает длину высоты.

`double` `min_lenght_x` - функция принимает 3 переменных типа `pair` с координатами точек, 1 координаты - точка из A, 2 и 3 координаты - точки прямой. Функция считающая длину от точки до прямой если они на одной оси x. Возвращает длину от прямой до точки.

`double` `min_lenght_y` - функция принимает 3 переменных типа `pair` с координатами точек, 1 координаты - точка из A, 2 и 3 координаты - точки прямой. Функция считающая длину от точки до прямой если они на одной оси y. Возвращает длину от прямой до точки.

### ***3. Описание входных и выходных данных***

Ввод данных будем осуществлять из текстового файла(`point.txt`), в котором на каждой строке заданы 2 координаты точки (x, y) через пробел.

В основной программе происходит считывание каждой строки текстового файла, занесение значений в соответствующие переменные и работа с ними

После необходимо ввести в консоль координаты точек для прямой и на выход программа даёт координаты точки подходящей под условие

#### 4. Текст программы

```
1. #include <iostream>
2. #include <utility>
3. #include <fstream>
4. #include <string>
5. #include <vector>
6. using namespace std;
7.
8. //Функция возвращает длину линии по формуле нахождения длины линии
   sqrt( (a1-a2)^2 + (b1-b2)^2 )
9. double length(pair <double, double> cord_one, pair <double, double>
   cord_two) {
10.     return sqrt(pow(abs(cord_one.first - cord_two.first), 2) +
   pow(abs(cord_one.second - cord_two.second), 2));
11. }
12.
13. //Функция считающая длины сторон треугольника образованого
   прямой и точкой
14. double min_lenght_triangle(pair <double, double> & point, pair
   <double, double> cord_line_one, pair <double, double> cord_line_two)
   {
15.     double line_1, line_2, line_3;
16.     line_1 = length(point, cord_line_one); // 1 сторона
17.     line_2 = length(point, cord_line_two); // 2 сторона
18.     line_3 = length(cord_line_one, cord_line_two); // основание
   треугольника
19.     double s_pr = (line_1 + line_2 + line_3) / 2; //
   Полупериметр треугольника
20.     double s = sqrt(s_pr * (s_pr - line_1) * (s_pr - line_2) *
   (s_pr - line_3)); // Площадь треугольника
21.     double h = (2 * s) / line_3; // Высота треугольника
22.     return h;
23. }
24.
25. //Функция считающая длину от точки до прямой если они на одной
   оси x
26. double min_lenght_x(pair <double, double> & point, pair
   <double, double> cord_line_one, pair <double, double> cord_line_two)
   {
27.     if (point.first > cord_line_one.first && point.first >
   cord_line_two.first) //Проверяю ниже точка или выше прямой
28.         return abs(max(cord_line_one.first,
   cord_line_two.first) - point.first);
29.     else
30.         return abs(min(cord_line_one.first,
   cord_line_two.first) - point.first);
31. }
32.
```

```

33.      //Функция считающая длину от точки до прямой если они на одной
      оси y
34.      double min_lenght_y(pair <double, double> & point, pair
      <double, double> cord_line_one, pair <double, double> cord_line_two)
      {
35.          if (point.second > cord_line_one.second && point.second >
      cord_line_two.second) // Проверяю правее точка или левее прямой
36.              return abs(max(cord_line_one.second,
      cord_line_two.second) - point.second);
37.          else
38.              return abs(min(cord_line_one.second,
      cord_line_two.second) - point.second);
39.      }
40.
41.      int main() {
42.          setlocale(LC_ALL, "Russian");
43.          pair <double, double> point;
44.          pair <double, double> cord_line_one;
45.          pair <double, double> cord_line_two;
46.
47.          cout << "Введите координаты первой точки" << endl;
48.          cout << "x : "; cin >> cord_line_one.first; //Первая x
      координата линии
49.          cout << "y : "; cin >> cord_line_one.second; //Первая y
      координата линии
50.          cout << "Введите вторую точку" << endl;
51.          cout << "x : "; cin >> cord_line_two.first; //Вторая x
      координата линии
52.          cout << "y : "; cin >> cord_line_two.second; //Вторая y
      координата линии
53.
54.          ifstream file("points.txt");
55.          if (!file.is_open()) { // проверка, успешно ли открыт файл
56.              cout << "Ошибка открытия файла!";
57.              return 1;
58.          }
59.
60.          double min;
61.          int flag = 0;
62.          pair <double, double> min_point;
63.          //Считываем с файла
64.          while (!file.eof() && file >> point.first, file >>
      point.second) {
65.              //Если они на оси x
66.              if (point.first == cord_line_one.first && point.first
      == cord_line_two.first) {
67.                  if (flag == 1 && min > min_lenght_y(point,
      cord_line_one, cord_line_two)) {
68.                      min = min_lenght_y(point, cord_line_one,
      cord_line_two); //Переприсваеваем минимальную длину
69.                      min_point = point; // Переприсваеваем точку
70.                  }
71.                  if (flag == 0) {
72.                      min = min_lenght_y(point, cord_line_one,
      cord_line_two);
73.                      min_point = point;
74.                      flag = 1;

```

```

75.         }
76.     }
77.     //Если они на оси y
78.     if (point.second == cord_line_one.second &&
point.second == cord_line_two.second) {
79.         if (flag == 1 && min > min_lenght_x(point,
cord_line_one, cord_line_two)) {
80.             min = min_lenght_x(point, cord_line_one,
cord_line_two);
81.             min_point = point;
82.         }
83.         if (flag == 0) {
84.             min = min_lenght_x(point, cord_line_one,
cord_line_two);
85.             min_point = point;
86.             flag = 1;
87.         }
88.     }
89.     // Иначе
90.     if (point.first != cord_line_one.first && point.first
!= cord_line_two.first && point.second != cord_line_one.second &&
point.second != cord_line_two.second) {
91.         if (flag == 1 && min > min_lenght_triangle(point,
cord_line_one, cord_line_two)) {
92.             min = min_lenght_triangle(point, cord_line_one,
cord_line_two);
93.             min_point = point;
94.         }
95.         if (flag == 0) {
96.             min = min_lenght_triangle(point, cord_line_one,
cord_line_two);
97.             min_point = point;
98.             flag = 1;
99.         }
100.    }
101.    }
102.    cout << "Ближайшая точка к прямой на координатах (" <<
min_point.first << ", " << min_point.second << ")";
103.    }

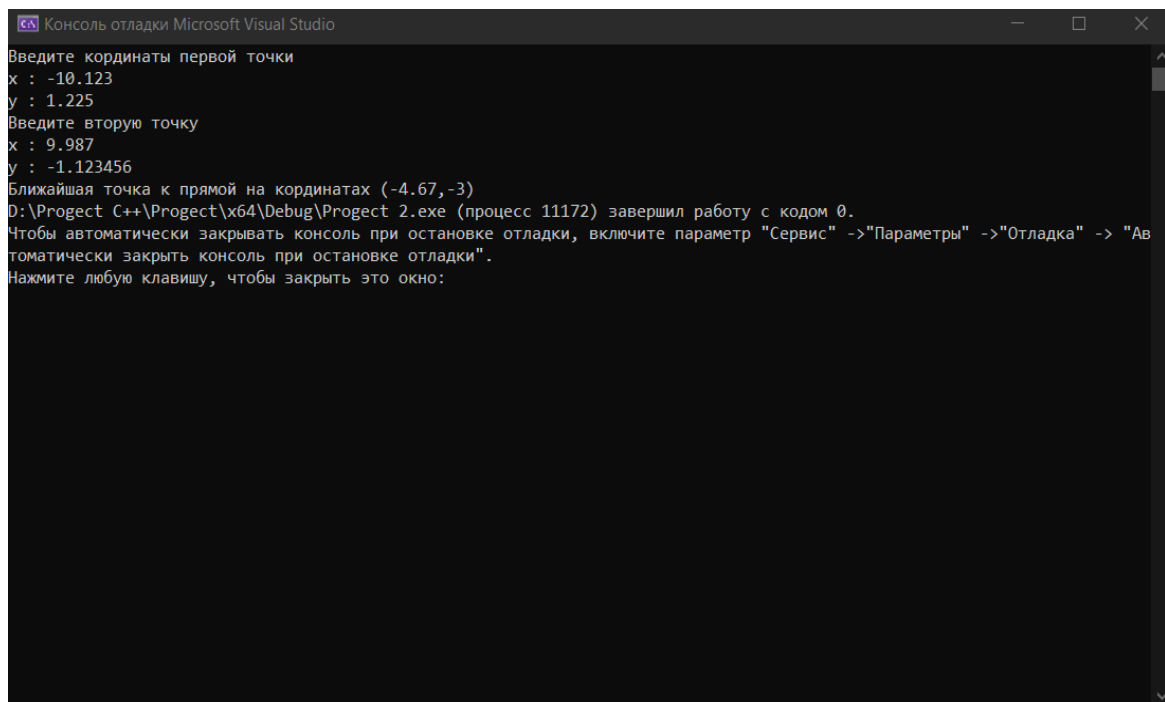
```

## 5. Тестовый запуск программы и вывод результатов работы

В файле point.txt содержатся координаты 5 точек (-7.0 8.0), (-3.0 5.0), (2.05.0), (8.0 -6.0), (-4.67 -3.0).

По запросу программы введем точки для отрезка BC = -10.123, 1.225, 9.987, -1.123456

Координаты точки удовлетворяющей условию - (-4.67 , -3)



```
Консоль отладки Microsoft Visual Studio
Введите координаты первой точки
x : -10.123
y : 1.225
Введите вторую точку
x : 9.987
y : -1.123456
Ближайшая точка к прямой на координатах (-4.67,-3)
D:\Project C++\Project\x64\Debug\Project 2.exe (процесс 11172) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
```

Программа работает корректно.