На вход программа получает флаги.
Если ключевых слов не введено, то программа получает данные из консоли и выводит их тоже в консоль.
Если введен флаг --fromfile, то после него обязательно должен следовать файл, из которого получаются данные, ответ выводится в консоль.
Если введен флаг --tofile (после него не обязательно указывать имя файл, потому что программа сама может задать файл), то данные вводятся в консоль и полученный ответ будет в файле.
Так же может быть введено оба этих флага, тогда программа будет брать данные из файла и выводить их в другой файл.

Общий алгоритм программы:

Создаются переменные целочисленного типа, n отвечает за то, что будет введено в консоли сначала (то есть количество отрезков), счётчик i нужен для того, чтобы записать все координаты отрезков в массивы, счётчик kol нужен для того, чтобы знать, сколько отрезков в итоге подходит.

Переменные типа double ch1 и ch2 нужны для того, чтобы вводить начало и конец отрезка.
Я создала 3 массива типа double, в массив begin будут в дальнейшем записываться все начала отрезков, в массив end будут записываться все концы отрезков, массив otv нужен для того, чтобы туда поместить в итоге все отрезки, которые будут нам подходить.

int m = n;	в данной строчке я задаю ещё одну целочисленную переменную, чтобы создать статический массив, и в дальнейшем записать все введенные отрезки.

begin[i] = ch1;
end[i] = ch2;
++i;
	в данной части кода я добавляю элементы в массивы и увеличиваю счетчик, чтобы каждый следующий отрезок был записан на своем новом месте. Последние элементы массива я приравниваю к нулю, для того чтобы числа были одинаковыми и не какими-то бесконечно большими.

Функция swap отвечает за сортировку массивов по возрастанию. Сортировка пузырьком осуществляется по элементам массива begin. Элементы массива end сортируются в том же порядке, что и массива begin.

for (int h = 0; h < n+1; ++h) {
  if (begin[h] == 0 && end[h] == 0) {
  del(begin, end, n, h);
  }
}
	Здесь я удаляю пару элементов, которая равна 0, 0, чтобы она не мешала дальнейшему выполнению действий.

Удаление осуществляется при помощи функции del. Функция получает массивы, которые должны подвергнуться изменениям, и номер элемента, который должен удалиться. Удаление происходит при помощи сдвига последующих элементов на текущий.

```
void del(double *first, double *second, int n, int k){
  for (int h = k; h < n; ++h) {
    first[h] = first[h+1];
    second[h] = second[h+1];
  }
```

Далее, начиная с первой пары, я сравниваю конец текущего отрезка и начало следующего.
Если конец меньше, чем начало, то переходим к следующей паре.
Если конец больше начала, и координаты текущего отрезка не равны, и координаты следующего отрезка не равны, то проверяются концы второго отрезка.
Если координата конца второго элемента меньше, чем координата конца первого элемента, то первый отрезок записывается в массив с итоговым результатом otv при помощи функции res, а второй отрезок просто удаляется при помощи функции del.
Если же наоборот координаты второго конца больше, чем координаты первого конца, то концу первого отрезка присваивается значение конца второго отрезка, пара координат первого отрезка записывается в массив с ответами при помощи функции res, и удаляются координаты второго отрезка при помощи функции del.
Если пара была записана в массив с ответами, то счётчик kol увеличивается на 2, так как было добавлено 2 элемента. После каждой проверки программа переходит к следующей паре чисел.

```
void res(double *first, double *second, double *otv, int k, int i) {
   otv[k] = first[i];
   otv[k+1] = second[i];
}
```

функция записывает в массив otv подходящие пары элементов (отрезки).

Если счётчик kol равен 0, значит в нем ничего нет, поэтому выводится NOTHING FOUND. Если наоборот, то будут выводиться пары чисел, пока kol не будет равен 0.

Работа с файлами:

Если введен флаг --fromfile:
Переменная inp будет обращением к файлу, из которого берутся данные. Если файл закрыт, программа завершает свою работу. Если всё хорошо, то далее данные будут браться из этого файла. `inp >> n;`

Если введен флаг --tofile:
Переменная f будет обращением к файлу, в который записывается ответ. Запись ответа:
`f << otv[z] << " " << otv[z + 1] << "\n";`

Если оба этих флага, то объединяются 2 верхних условия для этих флагов.