Отчет.

1) Программа принимает флаги "--tofile" и "--fromfile" как по отдельности, так и одновременно. При считывании данных из файла обязательно должно быть указано имя файла. В иных случаях программа выдаст ошибку некорректного ввода.

```
int main(int argc, char **argv) {
   if (argc < 2 || argc > 5) {
      std::cerr << "Incorrect enter!" << std::endl;
      return -1;
   }
   if (strcmp(argv[1], "--tofile") == 0 && argc < 5) {
         ...

   }
   if (strcmp(argv[1], "--fromfile") == 0 && argc == 3) {
         ...
   }
   if (argc >= 3 && ((strcmp(argv[1], "--fromfile") == 0 && strcmp(argv[2], "--tofile") == 0) ||
            (strcmp(argv[2], "--fromfile") == 0 && strcmp(argv[1], "--tofile") == 0))) {
         ...
      }

   }
```

2) После обработки флагов программа обращается к функциям:

TaskFromFile(если был подан флаг "--fromfile")

```
if (strcmp(argv[1], "--fromfile") == 0 && argc == 3) {
   std::ifstream in(argv[2]);
   TaskFromFile(in);
}
```

TaskToFile(если был подан флаг "--tofile")

```
if (strcmp(argv[1], "--tofile") == 0 && argc < 5) {
   if (argc == 2) {
      std::ofstream out("F");
      TaskToFile(out);
   } else {
      std::ofstream out(argv[2]);
      TaskToFile(out);
   }
}
```

TaskFromToFile(если были поданы оба флага сразу).

```
if (argc >= 3 && ((strcmp(argv[1], "--fromfile") == 0 && strcmp(argv[2], "--tofile") == 0) ||
            (strcmp(argv[2], "--fromfile") == 0 && strcmp(argv[1], "--tofile") == 0))) {
```

```
    if (argc == 3) {
        std::fstream inOut("F");
        TaskFromToFile(inOut);
    } else {
        std::fstream inOut(argv[3]);
        TaskFromToFile(inOut);
    }
}
```

В каждой функции проверяется число отрезков на корректность:

```
char symbol = file.peek();
if (!(symbol >= '1' && symbol <= '9')) {
    std::cerr << "Incorrect enter" << std::endl;
    return -1;
}
```

После чего это значение записывается в count:

```
int count;
file >> count;
```

Каждая из функций создает 2 массива с динамическим выделением памяти

```
double *arrBegin = new double[count + 1];
double *arrEnd = new double[count + 1];
```

и заполняет их так, чтобы в одном из которых хранились начала отрезков, а в другом - их концы. В случае ввода некорректных данных программа завершится с соответствующей ошибкой.

```
int i = 0;
while (i < count && file) {
    file >> a >> b;
    if (a >= b) {
        std::cerr << "Incorrect enter" << std::endl;
        delete[] arrEnd;
        delete[] arrBegin;
        return -1;
    }
    arrBegin[i] = a;
    arrEnd[i] = b;
    ++i;
}
```

3) После заполнения массивов выполняется функция MainFunction(arrBegin, arrEnd, count, k), которая принимает на вход указатели на массивы, количество отрезков и ссылку на флаг(он показывает, выполнялись ли действия над массивом или нет).

```
bool k = false;
MainFunction(arrBegin, arrEnd, count, k);
```

В функции выполняется сортировка отрезков по их начала от меньшего к большему с помощью функции Sort,

```
void Sort(double *array1, double *array2, int count) {
    for (int k = 0; k < count - 1; ++k) {
        for (int j = count - 1; j >= k + 1; --j) {
```

```
            if (array1[j - 1] > array1[j]) {
                Swap((array1 + j - 1), (array1 + j));
                Swap((array2 + j - 1), (array2 + j));
            }
        }
    }
}
```

в которой используется функция Swap, меняющая местами два элемента массива.

```
void Swap(double *a, double *b) {
    double c;
    c = *b;
    *b = *a;
    *a = c;
}
```

После чего выполняется объединениие отрезков начиная со второго: если начало данного лежит внутри предыдущего отрезка, данный отрезок убирается из массивов путем замены на следующий. Если это условие выполняется, перед "удалением" ставится флаг k = true и проверяется лежит ли конец этого отрезка внутри предыдущего. Если нет, то конец предыдущего заменяется на конец данного.

```
for (int l = 1; l < count; ++l) {
    if (arrBegin[l] < arrEnd[l - 1]) {
        k = true;
        if (arrEnd[l] > arrEnd[l - 1]) {
            arrEnd[l - 1] = arrEnd[l];
        }
        for (int m = l; m < count - 1; ++m) {
            arrBegin[m] = arrBegin[m + 1];
            arrEnd[m] = arrEnd[m + 1];
        }
    }
}
```

4) После выполнения данной функции происходит проверка флага: если он false, значит, объединения отрезков не произошло и выведется "NOTHING FOUND".

```
if (!k) {
    std::cerr << "NOTHING FOUND" << std::endl;
    delete[] arrEnd;
    delete[] arrBegin;
    return -2;
}
```

5) В противном случае будут выводиться начала и концы получившихся отрезков со второго, пока начало текущего больше, чем конец предыдущего.

```
int l = 1;
while (arrEnd[l] != arrBegin[l] && arrBegin[l] > arrEnd[l - 1]) {
    std::cout << arrBegin[l] << " " << arrEnd[l] << std::endl;
    ++l;
}
```

6) Начало и конец первого получившегося отрезка выводим до цикла, так как благодаря флагу k уверены, что объединение отрезков произошло(их >= 1). std::cout << arrBegin[0] << " " << arrEnd[0] << std::endl;

7) Далее выделенная память под массивы возвращается и функция завершается, как и вся прорамма!

```
delete[] arrEnd;
delete[] arrBegin;
return 0;
}
```

8) Makefile собирает проект и очищает его от временных файлов.