ЛАБОРАТОРНАЯ РАБОТА №3 ПО ЯЗЫКАМ ПРОГРАММИРОВАНИЯ

ПОГОРЕЛОВ ИЛЬЯ

Условие:

Дано натуральное число строк 0 < T <= 10000, в каждой из которых заданы 2 вещественных числа с точностью до второго знака после запятой - это отрезки на числовой прямой. В выводе запишите отрезки в том же формате, которые являются объединениями заданных отрезков. Учитывайте, что отрезки в выводе не должны иметь пересечений и должны идти по порядку. В случае отсутствия выведите сообщение "NOTHING FOUND". Код программы должен содержать ввод с консоли флага --tofile и --fromfile. Первый говорит нам, что ввод будет задан в заданный пользователем в командной строке файл (в случае отсутствия файла создать его), а второй говорит о считывании из файла заданного пользователем. Предусмотреть возможность исполнения обоих флагов. Учесть возможность некорректного ввода и обработать исключюючения. На дополнительный балл (на 10 баллов) добавьте тесты, по возможности Google-Tests.

Решение:

Идея моего решения заключается в занесении всех границ отрезков в один массив. Этот массив нужно отсортировать, таким образом первым элементом массива станет нижняя граница одного из отрезков. Далее я создаю переменную flag типа int. Изначальное значение flag=0. Далее встречая нижнюю границу отрезка, будем прибавлять 1 к значению flag и -1 в случае, когда элемент массива является закрывающей границей отрезка. Таким образом, когда flag вновь примет значение 0, значит либо отрезок закончился, либо до следующего начала отрезков пересечений нет. Таким образом, значение элемента массива, на котором flag стал равен 0 будет закрывающей границей отрезка пересечения, а минимальное значение элемента массива, при flag = 1 будет открывающей границей отрезка объединения.

Реализация:

```
int
read_cin()
{
                std::cin >> l;// считываем количество отрезков
                if ((l*10)%10!=0 || l <= 1){
                    std::cout<<" Wrong number of args";
                    exit(1);
                    return 1;
                    // если число дробное или неположительное, то выдаем ошибку ввода
                }
                else{
                    start = new double[l];
                    end = new double[l];
                    //заполняем массив strat начальными координатами отрезка, а массив end
            конечными

                    for (int i = 0; i < l; ++i){
                        std::cin >> start[i] >> end[i];
                        if (start[i] > end[i]){
                            std::cout<< "Wrong numbers";
                            exit(1);
                        считываем координаты отрезка, если начальная координата больше
            конечной, то выдаем ошибку ввода
                        }
                    }
                    data = new double[2 * l];
                    for (int i = 0; i < l; i++) {
                        data[i] = start[i];
                    }
                    for (int i = l; i < 2 * l; i++) {
                        data[i] = end[i - l];
                    }
                    // заполняем массив всеми координатами
                    return 0;

                }
            }
```

Данная функция обеспечивает ввод данных с консоли.

```
int
read_file()
{
                    std::ifstream file("fromfile.txt");
                    if (!file.is_open()) {
                        std::cout << "Error" << std::endl;
                        return 1;
                        // в случае, файл нельзя открыть выдаем ошибку
                    }

                    file >> l; // считываем количество отрезков
                    if ((l*10)%10!=0 || l <= 1){
                        std::cout<<" Wrong number of args";
                        exit(1);
                        return 1;
                        // аналогичная проверка корректности ввода

                    }
                    else{

                        start = new double[l];
                        end = new double[l];

                        for (int i = 0; i < l; ++i) {
                            file >> start[i] >> end[i];
                            if (start[i] > end[i]){
                                std::cout<< "Wrong numbers";
                                exit(1);
                                // также аналогичная проверка корректности ввода
                            }
                        }
                        data = new double[2 * l];
                        for (int i = 0; i < l; i++) {
                            data[i] = start[i];
                        }
                        for (int i = l; i < 2 * l; i++) {
                            data[i] = end[i - l];
                        }
                        // заполнение массива всеми координатами отрезков
                        return 0;
                    }
                }
```

Данная функция обеспечивает чтение данных из файла

```
int
poisk(double*
M, int r,
double zn)
                {
                    int i;
                    for (i = 0;i < r;i++)
                    {
                        if (M[i] == zn) return i;
                    }
                    return -1;
                }
```

Функция поиска возвращает индекс элемента, если он есть в массиве. Далее эта функция понадобится для проверки, какой координатой является точка (началом или концом отрезка)

```
double
func(int
to_file)
{
                double temp;
                for (int j = 0; j < 2 * l; j++) {
                    for (int i = 1; i < 2 * l; i++) {
                        if (data[i] < data[i - 1]) {
                            temp = data[i];
                            data[i] = data[i - 1];
                            data[i - 1] = temp;
                        }
                    }
                }
                // сортируем массив по возрастанию
                int counter = 0; // счетчик отрезков объединения
                double top = -1; // закрывающая граница
                double low = 100000; // открывающая граница
                int flag = 0; // флаг, принцип работы которого описан выше
                for (int i = 0; i < 2 * l; i++) {
                    if (poisk(start, l, data[i]) != -1 && (poisk(end, l, data[i]) != -1)) {
                        flag += 0;
                    }
                    // если два отрезка имеют одинаковую координату начала и конца, например
            [1,2];[2,3], то не нужно менять флаг, так как объединением будет отрезок от
            первой координаты начала то последней конца
                    if (poisk(start, l, data[i]) != -1 && (poisk(end, l, data[i]) == -1)) {
                        flag += 1;
                    }
                    else if ((poisk(end, l, data[i]) != -1) && (poisk(start, l, data[i]) ==
            -1)) {
                        flag -= 1;
                    }
                    if (flag == 1){
                        low = std::min(low,data[i]);
                        // присваиваем нижней границе значение координаты отрезка
                    }
                    if (flag == 0) {
                        top = std::max(top, data[i]);
                        if (to_file == 0) {
                        // флаг to_file обозначает вариант выводы ответа в файл, если флаг =
            1, в консоль если флаг = 0
                            std::cout << low << " " << top << std::endl;
                        }
                        else {
                            std::ofstream out;
                            out.open("tofile.txt", std::ios_base::app);
                            out << low << " " << top << '\n';
```

```
                out.close();
            }
            counter += 1;
            low = 10000000;
            top = -1;
            // обновляем нижнюю и верхнюю границы отрезков и добавляем 1 в
счетчик отрезков
        }


    }
    delete[] start;
    delete[] end;
    delete[] data;
    if (counter == 1) {
        std::cout << "Nothing was found";
        // если программа вывела столько же отрезков, сколько и получила на
вход, то заданные отрезки не имеют пересечений
        return 1;
    }
    return 0;
}
```

Описанная выше функция выполняет алгоритм поиска отрезков объединений

```
#include
<iostream>
            #include <cstring>
            #include "func.h"


            int main(int argc, char**argv)
            {
                std::cout << "Программа читает только из файла fromfile.txt и записывает
            только в файл tofile.txt, если имеются соответствующие флаги! :)\n";
                if (argc == 1) {
                    read_cin();
                    func(0);
                    exit(0);
                }
                else {
                    if (argc == 2) {
                        if (strcmp(argv[1], "--tofile") == 0) {
                            read_cin();
                            func(1);
                            exit(0);
                        }
                        else if (strcmp(argv[1], "--fromfile") == 0) {
                            read_file();
                            func(0);
                            exit(0);
                        }
                    }
                    else if (argc == 3) {
                        if (strcmp(argv[1], "--tofile") == 0 && strcmp(argv[2], "--
            fromfile") == 0 ||
                            strcmp(argv[2], "--tofile") == 0 && strcmp(argv[1], "--
            fromfile") == 0) {
                            if (read_file())
                                exit(1);
                            func(1);
                            exit(0);
                        }
                        else exit(1);
                    }
                }

            }
```

В main я считываю флаги и по ним обеспечиваю вывод в консоль, либо в файл