
Main Page
Classes
Files
Search
File List
Here is a list of all documented files with brief descriptions:
[detail level 1 2 3]
CLionProjects
den123
Functions.cpp файл с описанием классов и функций
Functions.h Заголовочный файл с описанием классов и функций
main.cpp MAIN FILE
Main Page
Classes
Files
Search
Public Attributes | List of all members
Obj Class Reference
создаем класс, который содержит в себе два числа с плавающей точкой: начало и конец отрезка More...
#include <Functions.h>
Public Attributes
double start
координата начала
double end
координата конца
Detailed Description
создаем класс, который содержит в себе два числа с плавающей точкой: начало и конец отрезка
Данный класс имеет только координаты начала и конца
The documentation for this class was generated from the following file:
/Users/semyonzotov/CLionProjects/den123/Functions.h
Main Page
Classes
Files
Search
CLionProjects
den123
Functions
Functions.cpp File Reference
файл с описанием классов и функций More...
#include "Functions.h"
Functions
int operator< (const Obj &obj1, const Obj &obj2)
void swapp (Obj &a, Obj &b)
double max (double &a, double &b)
Detailed Description
файл с описанием классов и функций
Данный файл содержит в себе определения основных классов, используемых в программе
swapp()
void swapp (Obj & a,
Obj & b
)
void swapp(Obj& a, Obj& b){
Obj k;
k = a;
a = b;
b = k;
}
функция которая меняет два элемента(аналог std::swap)

Function Documentation

◆ max()

```
double max ( double &  a,
             double &  b
           )
```

```
double max(double& a, double& b){
 if (a >= b){
     return a;
 }
 else{
     return b;
 }
}
```

функция которая берет максимум из двух чисел(аналог std::max)

◆ operator<()

```
int operator< ( const Obj &  obj1,
                const Obj &  obj2
              )
```

```
int operator<(const Obj& obj1, const Obj& obj2){
 return obj1.start < obj2.start;
}
```

перегрузка оператора меньше, чтобы сравнивать объекты типа **Obj** по координате начала

Main Page | Classes | Files | Search

CLionProjects | den123

Classes | Functions

Functions.h File Reference

Заголовочный файл с описанием классов и функций More...

Go to the source code of this file.

Classes

class	**Obj**
	создаем класс, который содержит в себе два числа с плавающей точкой: начало и конец отрезка More...

Functions

int	**operator<** (const **Obj** &obj1, const **Obj** &obj2)
void	**swapp** (**Obj** &a, **Obj** &b)
double	**max** (double &a, double &b)

Detailed Description

Заголовочный файл с описанием классов и функций

Данный файл содержит в себе определения основных классов и функций, используемых в программе

Function Documentation

◆ max()

```
double max ( double &  a,
             double &  b
           )
```

создаем функцию, которая выбирает большее из двух числе с плавающей точкой

Parameters

a,b сравниваемые значения

Returns

nothing

```
double max(double& a, double& b){
 if (a >= b){
     return a;
 }
 else{
     return b;
 }
}
```

функция которая берет максимум из двух чисел(аналог std::max)

◆ operator<()

```
int operator< ( const Obj &  obj1,
                const Obj &  obj2
              )
```

создаем перегрзку оператора "меньше", как отдельную функцию

Parameters

Obj1,Obj2 сравниваемые значения

Returns

True,если первое меньше, иначе False

```
int operator<(const Obj& obj1, const Obj& obj2){
    return obj1.start < obj2.start;
}
```

перегрузка оператора меньше, чтобы сраванивать объекты типа **Obj** по координате начала

◆ swapp()

```
void swapp ( Obj &  a,
             Obj &  b
           )
```

создаем функци, которая меняет два элемента, типа **Obj**, местами

Parameters

Obj1,Obj2 сравниваемые значения

Returns

nothing

```
void swapp(Obj& a, Obj& b){
    Obj k;
    k = a;
    a = b;
    b = k;
}
```

функция которая меняет два элемента(аналог std::swap)

Main Page | Classes | Files | Search

CLionProjects | den123

Functions

main.cpp File Reference

MAIN FILE. More...

```
#include <iostream>
#include <fstream>
#include "Functions.h"
```

Functions

int main (int argc, char **argv)

Detailed Description

MAIN FILE.

если нет флага

```
if (argc == 1) {
    std::cerr << "Enter the '--tofile' or '--fromfile'";
    return 0;
}
```

если флагов больше двух или первый флаг не равен,и "tofile", "--fromfile"

```
if ((strcmp(argv[1], "--tofile") == 1 && strcmp(argv[1], "--fromfile") == 1) || argc > 3) {
    std::cerr << "Wrong command, try again";
    return 0;
}
```

если флаг один и равен "--tofile"

```
if ((strcmp(argv[1], "--tofile") == 0) && argc == 2) {
    std::ofstream output;//создаем объект класса ofstream
    output.open("output1.txt");//связываем объект с файлом
    if (output.is_open()) {
        int N;//количество отрезков
        std::cin >> N;
        if (N < 1 || N > 10000) {//проверка на то,что количество отрезков больше 0 и меньше 10001
            std::cerr << "Numbers of rows > 0 and <= 10000";
        }
        Obj obj;//создаем переменную типа Obj
        Obj A[N];//создаем массив A, который хранит набор значений типа Obj
        for (int i = 0; i < N; i++) {
            std::cin >> obj.start;
            std::cin >> obj.end;
            if (obj.start > obj.end) {//проверка на то, что конец отрезка не может быть перед его началом
                std::cerr << "The end can't be before the start";
            }
            A[i] = obj;//добавляем значения в массив A
        }
        //сортировка массива методом пузырька по началу отрезка
        for (int i = 0; i + 1 < N; i++) {
            for (int j = 0; j + 1 < N - i; j++) {
                if (A[j + 1] < A[j]) {
                    swapp(A[j], A[j + 1]);
                }
            }
        }
```

```
        //в отсортированном массиве находим пересечения отрезков
        double l = 0, r = 0;//создаем перменные которые хранят в своих значениях координаты объеденения
        int cnt = 0;
        for (int i = 0; i + 1 < N; i++) {
            if (A[i].end >= A[i + 1].start &&
                cnt == 0) {//если до этого не было объеденения и конец первого >= начала второго
                l = A[i].start;//присваиваем значению l начало первого отрезка
                r = max(A[i].end, A[i + 1].end);//присаиваем r максимальный конец из двух отрезков
                cnt += 1;//в счетчик записываем, что было объеденение
            } else if (r >= A[i + 1].start &&
                       cnt != 0) {//если до этого было хотя бы одно объеденение и конец первого >= начала второго
                r = max(r, A[i + 1].end);//присаиваем r максимальный конец из двух отрезков, l не меняем
                cnt += 1;//в счетчик записываем, что было объеденение
            } else {//конец первого < начала второго и было объеденение, то выводим ответ
                if (l != 0 && r != 0) {
                    output << l << " " << r << std::endl;
                }
                cnt = 0;//зануляем счетчик
            }
        }
        if (cnt >= 1) {//если счетчик больше 1, то осталось объеденение, которое мы не вывели
            output << l << " " << r;
        }
        if (l == 0 && r == 0) {//если объеденений не было, выводим "NOTHING FOUND"
            output << "NOTHING FOUND";
        }
    }
}
```

если флаг один и равен "--fromfile"

```
if ((strcmp(argv[1], "--fromfile") == 0) && argc == 2) {
    std::ifstream fin;
    fin.open("input.txt");
    int N;
    fin >> N;
    if (N < 1 || N > 10000) {
        std::cerr << "Numbers of rows > 0 and <= 10000";
    }
    Obj obj;
    Obj A[N];
    for (int i = 0; i < N; i++) {
        fin >> obj.start;
        fin >> obj.end;
        if (obj.start > obj.end) {
            std::cerr << "The end can't be before the start";
        }
        A[i] = obj;
    }

    for (int i = 0; i + 1 < N; i++) {
        for (int j = 0; j + 1 < N - i; j++) {
            if (A[j + 1] < A[j]) {
                swapp(A[j], A[j + 1]);
            }
        }
    }

    double l = 0, r = 0;
    int cnt = 0;
    for (int i = 0; i + 1 < N; i++) {
        if (A[i].end >= A[i + 1].start && cnt == 0) {
            l = A[i].start;
            r = max(A[i].end, A[i + 1].end);
            cnt += 1;
        } else if (r >= A[i + 1].start && cnt != 0) {
            r = max(r, A[i + 1].end);
            cnt += 1;
        } else {
            if (l != 0 && r != 0) {
                std::cout << l << " " << r << std::endl;
            }
            cnt = 0;
        }
    }
    if (cnt >= 1) {
        std::cout << l << " " << r;
    }
    if (l == 0 && r == 0) {
        std::cout << "NOTHING FOUND";
    }
    fin.close();
}
```

если флага два и первый из них "--tofile", а второй "--fromfile"

```
if (argc == 3) {
    if (strcmp("--tofile", argv[1]) == 0) {
        if (strcmp("--fromfile", argv[2]) == 0) {
            std::ifstream kin;
            kin.open("input.txt");
            std::ofstream output;
            output.open("output1.txt");
            if (output.is_open()) {
                int N;
                kin >> N;
                if (N < 1 || N > 10000) {
                    std::cerr << "Numbers of rows > 0 and <= 10000";
                }
                Obj obj;
                Obj A[N];
                for (int i = 0; i < N; i++) {
                    kin >> obj.start;
                    kin >> obj.end;
                    if (obj.start > obj.end) {
                        std::cerr << "The end can't be before the start";
                    }

                    A[i] = obj;
                }

                for (int i = 0; i + 1 < N; i++) {
                    for (int j = 0; j + 1 < N - i; j++) {
                        if (A[j + 1] < A[j]) {
                            swapp(A[j], A[j + 1]);
                        }
                    }
                }

                double l = 0, r = 0;
                int cnt = 0;
                for (int i = 0; i + 1 < N; i++) {
                    if (A[i].end >= A[i + 1].start && cnt == 0) {
                        l = A[i].start;
                        r = max(A[i].end, A[i + 1].end);
                        cnt += 1;
                    } else if (r >= A[i + 1].start && cnt != 0) {
                        r = max(r, A[i + 1].end);
                        cnt += 1;
                    } else {
                        if (l != 0 && r != 0) {
                            output << l << " " << r << std::endl;
                        }
                        cnt = 0;
                    }
                }
                if (cnt >= 1) {
                    output << l << " " << r;
                }
                if (l == 0 && r == 0) {
                    output << "NOTHING FOUND";
                }
                kin.close();
            }
        } else {
            std::cerr << "Wrong command, try again";
            return 0;
        }
    } else {
        std::cerr << "Wrong command, try again";
        return 0;
    }
}
```