Отчет к лабораторной работе № 5

Задача:

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.

Алгоритм:

```
1   #include <iostream>
2   #include <cstring>
3   #include <fstream>
4   #include "func.h"
5
6   int main(int argc, char** argv) {
7       if (checkflag(argc, argv) == 0)
8           return 0;
9       int n, i = 0;
10      std::ifstream fin("input.txt");
11      if (argc == 1 || argc == 2 && strcmp(argv[1], "--fromfile"))
12          std::cin >> n;
13      else
14          fin >> n;
15      n = n * 2;
16      double* seg = new double[n];
17      if (argc == 1 || argc == 2 && strcmp(argv[1], "--fromfile"))
18          for (int i = 0; i < n; ++i)
19              std::cin >> seg[i];
        if (argc == 3 || argc == 2 && strcmp(argv[1], "--fromfile")
20  == 0)
21          while (!fin.eof()){
22              fin >> seg[i];
23              ++i;
24          }
25      sortmas(seg, n);
26      if (checkmas(seg, n) == 0)
27        return 0;
28      conjunction(seg, n, argc, argv);
29      delete []seg;
    }
```

Описание алгоритма:

Сначала программа проверяет правильность ввода флагов с помощью функции checkflag(). В случае правильного ввода программа получает исходные значения из файла или с консоли и записывает в массив. Далее происходит сортировка массива по началам отрезков с помощью функции sortmas(), что располагает наиболее вероятные для

объединения отрезки рядом друг с другом. Кроме того, требуется проверить, везде ли начало отрезка меньше конца (в ином случае, это неправильный ввод), что делается с помощью функции checkmas(). Последнее действие – использование функции conjunction(), которая выполняет объединения и выводит их на экран. Более подробно о функция см. далее. Помимо них, в проекте существует Makefile, который собирает проект, а также Google тесты.

Используемые функции:

1) Функция int checkflag()

Содержание функции:

```
1   int checkflag(int argc, char** argv){
2       if (argc > 3){
3           std::cerr << "Wrong quantity of flags. Try again";
4           return 0;
5       }
6       if (argc == 2)
7           if (strcmp(argv[1], "--fromfile") != 0 && strcmp(argv[1], "--
    tofile") != 0){
8               std::cerr << "Wrong flags. Try again\n";
9               return 0;
10          }
11      if (argc == 3){
12          if (strcmp(argv[1], "--fromfile") && strcmp(argv[1], "--tofile")
13          || strcmp(argv[2], "--fromfile") && strcmp(argv[2], "--tofile")
14          || strcmp(argv[1], argv[2]) == 0){
15              std::cerr << "Wrong flags. Try again\n";
16              return 0;
17          }
18      }
19      return 1;
20  }
```

Описание функции:

Функция получает в качестве аргументов целочисленное значение argc, равное количеству аргументов для функции main, основной программы файла main.cpp, и значение типа char** (имя – argv), являющееся массивом массивов типа char и представляющее остальные аргументы для функции main в файле main.cpp. В случае ввода неправильных (неподходящих) для задачи аргументов функция вернет значение 0, в ином случае – 1.

Первый условный оператор проверяет количество введенных аргументов argv, равное значению argc. Минимальное значение данной переменной – 1 (меньше ввести нельзя, аргумент “./lab5”, где lab5 – исполняемый файл, запустит программу, в ином случае запуск просто не произойдет). Так как помимо запускающего аргумента может быть только аргументы “--tofile” и “--fromfile”, то их количество не больше 3. В случае большего количества введенных переменных условный оператор вернет 0, что будет концом функции.

Второй условный оператор и вложенные в него условия проверяют правильность ввода флагов. В случае, если их количество верно (то есть больше 0 и меньше 4), то функция оценивает, нет ли среди них флагов, не являющихся флагами “--tofile” или “--fromfile”. Если такие флаги были найдены, то условный оператор вернет 0, что будет концом функции.

В случае, если все условия были проверены и оказались неверными, функция вернет 1, означающая, что флаги введены верно.

2) Функция void sortmas()

Содержание функции:

```
void sortmas(double* seg, int n){
    for (int i = 0; i < n; i += 2){
        for (int j = 0; j < n - i - 2; j += 2){
            if (seg[j] > seg[j + 2]) {
                std::swap(seg[j], seg[j + 2]);
                std::swap(seg[j + 1], seg[j + 3]);
            }
        }
    }
}
```

Описание функции:

Функция представляет собой сортировку заданного массива seg, размера n, по четным элементам, с их привязкой к следующему элементу. Сортируя четные элементы стандартным пузырьковым методом, следующий за ним нечетный элемент передвигается вместе с ним. Так как функция имеет тип void, она не возвращает никакого значения, однако из-за того, что seg – это массив (то есть seg – указатель на первый элемент массива), то он меняется. Таким образом, в памяти массив будет изменен.

3) Функция int checkmas()

Содержание функции:

```
int checkmas (double* seg, int n){
    for (int i = 0; i < n - 1; i += 2){
        if (seg[i] > seg[i + 1]){
            std::cerr << "Wrong expression. Try again\n";
            return 0;
        }
    }
    return 1;
}
```

Описание функции:

Функция проверяет правильность ввода значений. По условию задачи значения, введенные после первого (который является количеством пар элементов, которые следует ввести далее) – это начала и концы отрезков. В случае, если четный элемент массива, который является первым аргументом функции больше следующего, то функция вернет 0 (так как сначала нужно записать начало отрезка, который должен быть меньше конца). В случае правильности ввода функция вернет 1.

4) Функция void conjunction()

Содержание функции:

```
void conjunction(double* seg, int n, int argc, char**argv){
    std::ofstream fout("output.txt");
    int fn = 0, f = 0;
    for (int i = 1; i < n - 2; i += 2){
        if (seg[i] >= seg [i + 1]){
            seg[i + 1] = seg [i - 1];
            seg [i + 2] = std::max(seg[i], seg[i + 2]);
            seg[i - 1] = seg [i] = 0.001;
```

```
9               f = fn = 1;
10          } else {
11              if (f == 1){
                    if (argc == 3 || argc == 2 && (strcmp(argv[1], "-
12  -tofile") == 0)) {
13                      fout << seg[i - 1] << " " << seg[i] << "\n";
14                      f = 0;
15                  }
16                  else {
17                      std::cout << seg[i - 1] << " " << seg[i] <<
    "\n";
18                      f = 0;
19                  }
20              }
21          }
22      }
23      if (f == 1){
24          if (argc == 3 || argc == 2 && (strcmp(argv[1], "--
    tofile") == 0))
25              fout << seg[n - 2] << " " << seg[n - 1] << "\n";
26          else
27              std::cout << seg[n - 2] << " " << seg[n - 1] << "\n";
28      }
29      if (fn == 0)
30          if (argc == 3 || argc == 2 && (strcmp(argv[1], "--
    tofile") == 0))
31              fout << "NOTHING FOUND\n";
32          else
33              std::cout << "NOTHING FOUND\n";
34  }
```

Описание функции:

Функция получает в качестве аргументов массив дробных чисел, его размер, а также аргументы из функции main файла main.cpp. Аргументы argc и argv нужны для понимания, куда нужно выводить верный ответ – в файл или на экран. Функция выполняет следующее действие: сравнивает два отрезка в отсортированном массиве и, при наличии пересечения у них, соединяет в один (при этом флаги f и fn становятся равными 1). Из массива удаляются (заменяются на 0.001) четыре старых элемента, а на место начала и конца второго отрезка ставятся начало и конец нового. Таким образом, в памяти новый отрезок и остальные будут находиться рядом. В случае, если объединение невозможно, то или выводится последний отрезок (если f == 1, то есть объединение до этого было), или ничего не выводится. В конце, при наличии флага fn == 0 выводится “NOTHING FOUND”, показывая, что объединений не было.