Лабораторная работа № 5

В данной работе я использовал класс написанный лично мной (как обычно), а также некоторые функции. Во-первых я использовал fstream и его некоторые методы для работы с файлами, такими как ifstream и ofstream, также я использовал два метода вывода из std – cin и cout.

Подробнее про мой класс

Я назвал его start. В качестве приватных переменных я создал массив дробных чисел на 10000 чисел(макс возможное кол-во чисел на ввод) и переменную size, которая отвечает за “размер” массива, также указывает на конец заполненного массива(изначально равен 0). Далее перейдём к публичным вещам. Во-первых инициализатор класса. Далее у меня есть метод addToEnd – он просто записывает передаваемое значение в конец массива (в ячейку с индексом size) и увеличивает size на 1. ReplaceNum – просто заменяет число с указанным индексом на новое число. insertNum вставляет число на указанную позицию и сдвигает все числа справа на одну ячейку вправо. SizeM – возвращает размер массива. returnNum -возвращает число с определённым индексом. deleteNum – удаляет число под указанным индексом и сдвигает все элементы справа на один индекс влево

Подробнее про суть программы

Моя программа решает всё достаточно очевидно. Я рассмотрел, что два отрезка могут находиться в 6 позициях относительно друг друга. Первый отрезок полностью левее правого, первый отрезок левее правого, но они пересекаются, первый отрезок входит во второй, второй отрезок левее первого, второй отрезок левее первого, но они пересекаются, второй отрезок полностью входит в первый. Именно эти 6 случаев я и обрабатываю. В классе я храню отрезки отсортированными по левой координате от меньшей к большей. Сначала, когда нам вводят первые координаты, я просто сразу записываю их в класс. Далее идут проверки, сначала проверяю, что второй отрезок “меньше” первого в классе, если да, то делаю insertNum на первое место. Иначе иду далее. Далее я обрабатываю возможность просто добавить отрезок в массив, если он не пересекается. Далее, если отрезок пересекается(проверка идёт введённых координат и координат из одного отрезка), то я проверяю каким образом два отрезка пересекаются и обрабатываю это. Ну и если ни одно из условий не сработало, то просто добавляю координаты в конец массива. Однако, как я уже заметил, я

проверяю на пересечение только два отрезка и может оказаться так, что отрезки пересекаются, но я их не отработал, для этого мне требуется ещё один цикл, где я обрабатываю ещё раз получившийся массив, где я сразу выполняю объединение отрезков, и удаление ненужных и далее вывод.

Различия между файлами sol* только в формате ввода и вывода данных, у sol1 ввод и вывод через консоль, sol3 – либо вывод, либо ввод через файл, sol5 – и ввод и вывод через файл.

В мейкфайле у меня три функции реализовано- 1)скомпилировать весь проект, 2)удалить объектный файл, 3)удалить объектный файл и все txt файлы.

КОД

Makefile

```
all:
	g++ main.cpp sol1.cpp sol3i.cpp sol3o.cpp sol5.cpp -o m
clean:
	rm -rf *.o m
distclean:
	rm -rf *.o m
	rm *.txt
```

main.cpp

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

using namespace std;

int main(int argc, char *argv[])
{
   ifstream fin;
   ofstream fout;
```

```
  int tmp;


  if(argc == 1){
   cout << "OK\n";
   sol1();
  }else if((argc == 3)||(argc == 2)){
     if(strcmp(argv[1], "--fromfile") == 0){
       cout << "OK3 from\n";
       sol3i(argv);
     }else if(strcmp(argv[1], "--tofile") == 0){
       cout << "OK3 to\n";
       sol3o(argc, argv);
     }else{
       cout << "Incorect flag\n";
       return 0;
     }
  }else if((argc == 5)||(argc == 4)){
    bool flag = true;
      if((strcmp(argv[1], "--fromfile") == 0)||(strcmp(argv[2], "--fromfile") == 0)||(strcmp(argv[3], "--
fromfile") == 0)){
        if(strcmp(argv[1], "--fromfile") == 0){
          cout << "OK5 from1\n";
          flag = true;
        }else{
          cout << "OK5 from3\n";
          flag = false;
        }
      }else{
        cout << "Incorrect flag\n";
        return 0;
      }
      if((strcmp(argv[1], "--tofile") == 0)||(strcmp(argv[3], "--tofile") == 0)){
        if(strcmp(argv[1], "--tofile") == 0){
          cout << "OK5 to1\n";
```

```
        }else{
          cout << "OK5 to3\n";
        }
       }else{
        cout << "Incorrect flag\n";
        return 0;
       }
       sol5(argc, argv, flag);
   }
   return 0;
}
```

Func.h

```
void sol1();
void sol3i(char *argv[]);
void sol3o(int argc, char *argv[]);
void sol5(int argc, char *argv[], bool f);
```

Sol1.cpp

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

using std::cin;
using std::cout;
//using namespace std;

class start{
   private:
      int size = 0;
      float num[10000];
   public:
```

```
start(){}

void addToEnd(float tmp){
        num[size] = tmp;
        ++size;
    }

void replaceNum(int i, float tmp){
        num[i] = tmp;
    }

void insertNum(int i, float numb){
    float tmp1 = num[i];
    float tmp2;
    num[i] = numb;
    for(int j = i; j < size; ++j){
        tmp2 = num[j+1];
        num[j+1] = tmp1;
        tmp1 = tmp2;
        }
    ++size;
    }

int sizeM(){
    return size;
    }

float returnNum(int i){
    return num[i];
```

```
        }

    void deleteNum(int i){
        --size;
        for(int j = i; j < size; ++j)
            num[j] = num[j+1];
        }
  };

void sol1(){
  int amogus;
  cin >> amogus;
  if (amogus == 0){
    cout << "Nothing found\n";
  }else{
   start first;
   start second;
   double a, b;
   for(int i = 0; i < amogus; ++i){
      cin >> a >> b;

      if(first.sizeM() == 0){
         first.addToEnd(a);
         second.addToEnd(b);
         //cout << "Done addToEnd first time\n";
      }else
         {
         bool flag = false;
         for(int i = 0; i < first.sizeM(); ++i){
```

```
        if (b < first.returnNum(i)){
            first.insertNum(i, a);
            second.insertNum(i, b);
            //cout << "Done insertNum less than first\n";
            flag = true;
            break;
          }else if ((a < first.returnNum(i))&&(b > first.returnNum(i))){
              first.replaceNum(i,a);
              //cout << "Done replaceNum begin less\n";
              if(b > second.returnNum(i)){
                 second.replaceNum(i, b);
                 //cout << "Done replaceNum end bigger\n";
                 }
              flag = true;
              break;
            }
        }
      if (!flag){
        first.addToEnd(a);
        second.addToEnd(b);
        //cout << "Done addToEnd biggest\n";
        }
      flag = false;
      }
    }


for (int i = 0; i < first.sizeM(); ++i){
```

```
    if ((first.returnNum(i) <= first.returnNum(i+1))&&(second.returnNum(i) > first.returnNum(i+1))){
        if(second.returnNum(i) >= second.returnNum(i+1)){
            first.deleteNum(i+1);
            second.deleteNum(i+1);
            //cout << "Done deleteNum next in previous\n";
        }else{
            first.deleteNum(i+1);
            second.replaceNum(i, second.returnNum(i+1));
            second.deleteNum(i+1);
            //cout << "Done deleteNum and replaceNum previous + next\n";
            }
    }
  }

  for(int i = 0; i < first.sizeM(); ++i)
    cout << first.returnNum(i) << " " << second.returnNum(i) << '\n';
  cout << "finished\n";
}
}
```

Sol3i.cpp

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

using std::cin;
using std::cout;
//using namespace std;
```

```
class start{
   private:
      int size = 0;
      float num[10000];
   public:
      start(){}

      void addToEnd(float tmp){
            num[size] = tmp;
            ++size;
         }

      void replaceNum(int i, float tmp){
            num[i] = tmp;
         }

      void insertNum(int i, float numb){
         float tmp1 = num[i];
         float tmp2;
         num[i] = numb;
         for(int j = i; j < size; ++j){
            tmp2 = num[j+1];
            num[j+1] = tmp1;
            tmp1 = tmp2;
            }
         ++size;
         }

      int sizeM(){
```

```
        return size;
        }

    float returnNum(int i){
        return num[i];
        }

    void deleteNum(int i){
        --size;
        for(int j = i; j < size; ++j)
            num[j] = num[j+1];
        }
  };

void sol3i(char *argv[]){
  std::ifstream fin;
  fin.open(argv[2]);
  int amogus;
  fin >> amogus;
  if (amogus == 0){
    cout << "Nothing found\n";
  }else{
   start first;
   start second;
   double a, b;
   for(int i = 0; i < amogus; ++i){
      fin >> a >> b;

      if(first.sizeM() == 0){
```

```
    first.addToEnd(a);
    second.addToEnd(b);
    //cout << "Done addToEnd first time\n";
  }else
    {
    bool flag = false;
    for(int i = 0; i < first.sizeM(); ++i){
      if (b < first.returnNum(i)){
            first.insertNum(i, a);
            second.insertNum(i, b);
            //cout << "Done insertNum less than first\n";
            flag = true;
            break;
        }else if ((a < first.returnNum(i))&&(b > first.returnNum(i))){
              first.replaceNum(i,a);
              //cout << "Done replaceNum begin less\n";
              if(b > second.returnNum(i)){
                second.replaceNum(i, b);
                //cout << "Done replaceNum end bigger\n";
                }
              flag = true;
              break;
            }
      }
    if (!flag){
      first.addToEnd(a);
      second.addToEnd(b);
      //cout << "Done addToEnd biggest\n";
      }
```

```
        flag = false;
        }
      }


  for (int i = 0; i < first.sizeM(); ++i){
     if ((first.returnNum(i) <= first.returnNum(i+1))&&(second.returnNum(i) > first.returnNum(i+1))){
        if(second.returnNum(i) >= second.returnNum(i+1)){
           first.deleteNum(i+1);
           second.deleteNum(i+1);
           //cout << "Done deleteNum next in previous\n";
        }else{
           first.deleteNum(i+1);
           second.replaceNum(i, second.returnNum(i+1));
           second.deleteNum(i+1);
           //cout << "Done deleteNum and replaceNum previous + next\n";
           }
     }
  }

  for(int i = 0; i < first.sizeM(); ++i)
     cout << first.returnNum(i) << " " << second.returnNum(i) << '\n';
  cout << "finished\n";
  fin.close();
 }
}
```

Sol3o.cpp

```
#include <iostream>
```

```
#include <fstream>
#include "func.h"

using std::cin;
using std::cout;
//using namespace std;

class start{
   private:
      int size = 0;
      float num[10000];
   public:
      start(){}

      void addToEnd(float tmp){
            num[size] = tmp;
            ++size;
         }

      void replaceNum(int i, float tmp){
            num[i] = tmp;
         }

      void insertNum(int i, float numb){
         float tmp1 = num[i];
         float tmp2;
         num[i] = numb;
         for(int j = i; j < size; ++j){
            tmp2 = num[j+1];
```

```
            num[j+1] = tmp1;
            tmp1 = tmp2;
            }
        ++size;
        }

    int sizeM(){
        return size;
        }

    float returnNum(int i){
        return num[i];
        }

    void deleteNum(int i){
        --size;
        for(int j = i; j < size; ++j)
            num[j] = num[j+1];
        }
  };

void sol3o(int argc, char *argv[]){
  std::ofstream fout;
  if(argc == 3)
    fout.open(argv[2]);
  else fout.open("output.txt");
  int amogus;
  cin >> amogus;
  if (amogus == 0){
```

```
 fout << "Nothing found\n";
}else{
 start first;
 start second;
 double a, b;
 for(int i = 0; i < amogus; ++i){
    cin >> a >> b;

    if(first.sizeM() == 0){
       first.addToEnd(a);
       second.addToEnd(b);
       //cout << "Done addToEnd first time\n";
    }else
       {
       bool flag = false;
       for(int i = 0; i < first.sizeM(); ++i){
          if (b < first.returnNum(i)){
                first.insertNum(i, a);
                second.insertNum(i, b);
                //cout << "Done insertNum less than first\n";
                flag = true;
                break;
             }else if ((a < first.returnNum(i))&&(b > first.returnNum(i))){
                   first.replaceNum(i,a);
                   //cout << "Done replaceNum begin less\n";
                   if(b > second.returnNum(i)){
                      second.replaceNum(i, b);
                      //cout << "Done replaceNum end bigger\n";
                      }
```

```
                flag = true;
                break;
            }
        }
        if (!flag){
            first.addToEnd(a);
            second.addToEnd(b);
            //cout << "Done addToEnd biggest\n";
            }
        flag = false;
        }
    }


  for (int i = 0; i < first.sizeM(); ++i){
     if ((first.returnNum(i) <= first.returnNum(i+1))&&(second.returnNum(i) > first.returnNum(i+1))){
        if(second.returnNum(i) >= second.returnNum(i+1)){
            first.deleteNum(i+1);
            second.deleteNum(i+1);
            //cout << "Done deleteNum next in previous\n";
        }else{
            first.deleteNum(i+1);
            second.replaceNum(i, second.returnNum(i+1));
            second.deleteNum(i+1);
            //cout << "Done deleteNum and replaceNum previous + next\n";
            }
     }
  }
```

```
  for(int i = 0; i < first.sizeM(); ++i)
     fout << first.returnNum(i) << " " << second.returnNum(i) << '\n';
  cout << "finished\n";
  fout.close();
 }
}
```

Sol5.cpp

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

using std::cin;
using std::cout;
//using namespace std;

class start{
   private:
      int size = 0;
      float num[10000];
   public:
      start(){}

      void addToEnd(float tmp){
            num[size] = tmp;
            ++size;
         }

      void replaceNum(int i, float tmp){
```

```
        num[i] = tmp;
    }

    void insertNum(int i, float numb){
        float tmp1 = num[i];
        float tmp2;
        num[i] = numb;
        for(int j = i; j < size; ++j){
            tmp2 = num[j+1];
            num[j+1] = tmp1;
            tmp1 = tmp2;
            }
        ++size;
        }

    int sizeM(){
        return size;
        }

    float returnNum(int i){
        return num[i];
        }

    void deleteNum(int i){
        --size;
        for(int j = i; j < size; ++j)
            num[j] = num[j+1];
        }
};
```

```
void sol5(int argc, char *argv[], bool f){
  std::ifstream fin;
  std::ofstream fout;
  if(f){
    fin.open(argv[2]);
      if(argc == 5)
        fout.open(argv[4]);
      else fout.open("output.txt");
  }else{
    if(argc == 5){
      fin.open(argv[4]);
      fout.open(argv[2]);
    }else{
      fin.open(argv[3]);
      fout.open("output.txt");
    }
  }
  int amogus;
  fin >> amogus;
  if (amogus == 0){
    fout << "Nothing found\n";
  }else{
   start first;
   start second;
   double a, b;
   for(int i = 0; i < amogus; ++i){
       fin >> a >> b;
```

```
if(first.sizeM() == 0){
   first.addToEnd(a);
   second.addToEnd(b);
   //cout << "Done addToEnd first time\n";
}else
   {
   bool flag = false;
   for(int i = 0; i < first.sizeM(); ++i){
      if (b < first.returnNum(i)){
            first.insertNum(i, a);
            second.insertNum(i, b);
            //cout << "Done insertNum less than first\n";
            flag = true;
            break;
         }else if ((a < first.returnNum(i))&&(b > first.returnNum(i))){
               first.replaceNum(i,a);
               //cout << "Done replaceNum begin less\n";
               if(b > second.returnNum(i)){
                  second.replaceNum(i, b);
                  //cout << "Done replaceNum end bigger\n";
                  }
               flag = true;
               break;
            }
      }
   if (!flag){
      first.addToEnd(a);
      second.addToEnd(b);
      //cout << "Done addToEnd biggest\n";
```

```
            }
          flag = false;
          }
      }


  for (int i = 0; i < first.sizeM(); ++i){
     if ((first.returnNum(i) <= first.returnNum(i+1))&&(second.returnNum(i) > first.returnNum(i+1))){
          if(second.returnNum(i) >= second.returnNum(i+1)){
             first.deleteNum(i+1);
             second.deleteNum(i+1);
             //cout << "Done deleteNum next in previous\n";
          }else{
             first.deleteNum(i+1);
             second.replaceNum(i, second.returnNum(i+1));
             second.deleteNum(i+1);
             //cout << "Done deleteNum and replaceNum previous + next\n";
             }
     }
  }

  for(int i = 0; i < first.sizeM(); ++i)
     fout << first.returnNum(i) << " " << second.returnNum(i) << '\n';
  cout << "finished\n";
  fin.close();
  fout.close();
 }
}
```