

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра ВТ**

**ОТЧЕТ ПО Индивидуальному практическому заданию**  
**по дисциплине «Программирование»**

Студент гр. 9005

\_\_\_\_\_

Самуйлова Е.С

Преподаватель

\_\_\_\_\_

Калмычков В.А.

Санкт-Петербург

2021

## **ИНДИВИДУАЛЬНОЕ ПРАКТИЧЕСКОЕ ЗАДАНИЕ**

### **ТЕМА: СТЕК И ДВУНАПРАВЛЕННАЯ ОЧЕРЕДЬ**

Студент Самуйлова Евгения

Группа 9005

Формулировка задания:

I. Обязательные минимальные требования по загружаемым файлам в систему по адресу `ves.etu.ru` Отдельно загружаются в соответствующие разделы: - результат контрольной работы (для еще не загруженных) в формате pdf или docx, - материалы по индивидуальному практическому заданию: файл с текстом программы (формат текстового файла), файл с исходными данными (формат текстового файла), отчет (формат файла - pdf). Выбор для файлов имени по смыслу, а не длинная последовательность случайных символов. Файлы, которые не соответствуют этим минимальным требованиям, не рассматриваются.

II. Требования по реализации индивидуального практического задания и оформлению отчета  
Общее по заданиям: - используются только **СОБСТВЕННЫЕ** структуры (struct) или классы (class), содержащие: целые данные при вариантах на четные/нечетные значения, вещественные данные для остальных вариантов - ввод осуществляется из файла в указанный вариант внутреннего хранения исходных данных - данные из структуры хранения исходных данных должны быть скопированы в новую структуру хранения результата: в зависимости от варианта формируется единая результирующая структура (указано число 1) или данные по значениям разносятся в разные результирующие структуры (указано число 2) - в отдельных заданиях при создании результирующей структуры (одной или

двух) необходимо, чтобы данные были упорядочены (направление произвольное и выбирается автором), это можно делать либо в процессе включения новых данных, либо итоговой сортировкой после переноса всех данных (опять же на выбор автора) - вывод в файл: контрольный вывод из структуры хранения исходных данных, полученный результат после обработки по заданию из новой структуры хранения результата - НЕ РАЗРЕШАЕТСЯ использование поставляемых с системой разработки библиотек, реализующих частично или полностью изучаемые и указанные к реализации способы представления данных - все должно быть по размещению и обработке данных реализовано с самого начала - все действия должны быть ЯВНО реализованы как ИЗМЕНЕНИЕ внутреннего размещения данных, а не "заглушкой" примитивного вывода в файл в опознанном месте обработки без модификации размещения самих данных

Разделы отчета:

1. Титульный лист
2. Формулировка задания
3. Внешний формат хранения данных (входные и выходные файлы)
4. Внутренний формат хранения данных (индивидуальные их виды по заданию, их графическое представление, изображение их взаимодействия при выполнении индивидуального задания)
5. Таблица с кратким описанием структур/классов, функций, модулей (файлов): имя модуля, имя структуры/класса или функции, назначение, параметры для функции, возвращаемое функцией значение
6. Общее краткое описание алгоритма обработки
7. Текст программы
8. Примеры работы программы (фрагменты исходного и результирующего файлов)
9. Выводы

Общее по отчетам (естественно должны присутствовать все разделы): - должно присутствовать описание формата представления данных во

входных и выходных файлах, а не просто указание их имен - должны присутствовать рисунки, поясняющие внутренний формат хранения данных - должны присутствовать рисунки, поясняющие выполняемые действия над данными по индивидуальному практическому заданию - должна присутствовать таблица описания функций, в том числе и при реализации классов - примеры должны показать реакцию программы на разные входные данные, в частности: при отсутствии подлежащего обработке, при неоднократном присутствии подлежащего обработке, возможные возникающие при обработке ситуации Блок-схемы в этом семестре в отчете НЕ ОФОРМЛЯЮТСЯ.

III. Общие положения по порядку прохождения аттестации: -  
предоставление результатов контрольной работы и зачтенное индивидуальное практическое задание обеспечивают допуск к экзамену

Дата выдачи задания: 11.01.2021

Дата сдачи задания: 17.01.2021

Студент

---

Самуйлов Е.С.

### Внешний формат хранения данных

Формат входного файла:

1 Файл формата .txt

Формат выходного файла:

1 Файл формата .txt

### Внутренний формат хранения данных

Первоначально данные попадают в стек, где сразу же обрезается все, что не входит в указанный диапазон. Затем данные переносятся в двунаправленный список и, после сортировки, печатаются.

### Таблица функций, классов и структур

Имя структуры/ класса/ функции	Назначение	Параметры для функции	Возвращаемое функцией значение
Sumb	структура хранения - двоичный список	-	-
MyStor	с т р у к т у р а х р а н е н и я о б щ и х э л е м е н т о в	-	-
Stack	структура для стека	-	-
NumberWorker	Основной класс программы	-	-
sort	Метод сортировки конечного списка	-	-
putL2	Метод вставки в двунаправленную очередь	флаг flagLast и значение elem	
printer	Метод печати	-	-
reader	метод чтения и заполнения стека	nameOfFile - имя входного файла	-

moveToL2	метод переноса из стека в двунаправленную очередь	-	-
main	метод запуска программы - перебирает все методы в правильном порядке		int - статус завершения программы

## Описание алгоритма

Программа берет два пути, проверяет их корректность, затем один путь передается на считывание и заполнение в стек, при этом проверяются данные на вхождение в диапазон. Далее данные переносятся в двунаправленную очередь, затем они сортируются и методом печати выводятся.

## Текст программы

```

#include <iostream>
#include <fstream>
#include <string>

using namespace std;

struct Stack { //структура для ввода
    Stack *next;
    int elem = 0;
};

struct Sumb //структура вывода - двунаправленный список
{
    Sumb* nexElem; // указатель на следующий элемент
    Sumb* prewElem; //указатель на предыдущее элемент
    int elem = 0; //элемент
};

```

```

struct MyStor {
    string nameOfFile;
    int rangTop = 0;
    int rangLast = 0;
    Sumb *firstElem = NULL;
    Sumb *lastElem = NULL;
    Stack *top = NULL;
}; // структура хранения информации

```

```

MyStor myStor;
class NumberWorker // основной класс
{
public:

```

```

    void sort()
    {
        int finishFlag;
        while(true) // пока не отсортировано
        {
            finishFlag = 0;
            Sumb *elemTwo = new Sumb;
            elemTwo->elem = 0;
            elemTwo->nexElem = NULL;
            elemTwo->prewElem = NULL;

```

```

            Sumb *tempStor = new Sumb;
            tempStor->elem = 0;
            tempStor->prewElem = NULL;
            tempStor->nexElem = NULL;

            int prev = myStor.firstElem->elem;
            Sumb *temp = myStor.firstElem;
            while(true)
            {
                temp = temp->nexElem;
                if (temp == NULL) break;
                if(temp->elem < prev)
                {
                    finishFlag = 1;
                    elemTwo = temp->prewElem; // сохранил второй элемент пары
                    tempStor->nexElem = temp->nexElem; // сохранил ссылку на следующий, за участком смены
                    tempStor->prewElem = elemTwo->prewElem; // сохранил ссылку на участок до смены
                    elemTwo->prewElem = temp; // связал элементы
                    temp->nexElem = elemTwo; // связал элементы
                    elemTwo->nexElem = tempStor->nexElem; // привязал следующий участок
                    if(tempStor->prewElem == NULL) {
                        myStor.firstElem = temp; // если первый
                        temp->prewElem = NULL;
                    }
                    else {
                        temp->prewElem = tempStor->prewElem; // привязал предыдущий участок

```

```

        temp->prevElem->nexElem = temp; // привязал предыдущий участок
    }
    if(tempStor->nexElem == NULL) {
        myStor.lastElem = elemTwo; // если последний
        break;
    }
    else{
        tempStor->nexElem->prevElem = elemTwo; // привязал следующий участок
    }

    prev = elemTwo->elem;
}
else prev = temp->elem;
}
if(finishFlag == 0)
    break;
}
}; // метод сортировки

void putL2(int elem, int flagLast)
{
    if(myStor.firstElem == NULL) // самый маленький
    {
        myStor.firstElem = new Sumb;
        myStor.firstElem->elem = elem;
        myStor.firstElem->prevElem = NULL;

```

```

    }

    else
    {
        Sumb *temp = myStor.firstElem;
        while(true)
        {
            if(temp->nexElem == NULL)
                break;
            else temp = temp->nexElem;
        }

        if(flagLast == 1)
        {
            myStor.lastElem = new Sumb;
            myStor.lastElem->prevElem = temp;
            temp->nexElem = myStor.lastElem;
            myStor.lastElem->nexElem = NULL;
            myStor.lastElem->elem = elem;
        } else {
            Sumb *newElem = new Sumb;
            newElem->elem = elem;
            newElem->nexElem = NULL;
            newElem->prevElem = temp;
            temp->nexElem = newElem;
        }
    }
}

```



```

    }
}; // метод вставки нового элемента в двунаправленную очередь

void stackPut(int info)
{
    if(info > (myStor.rangLast - 1) && info < (myStor.rangTop + 1)) //Проверка диапазона - все что вне отсекается
    {
        Stack *newElem = new Stack;
        newElem->next = myStor.top;
        newElem->elem = info;
        myStor.top = newElem;
    }
}; // метод вставки нового элемента в стек готов

void moveToL2()
{
    int flagFinish = 0;
    while(true)
    {
        Stack *temp = myStor.top->next;
        if(myStor.top->next == NULL)
        {
            putL2(myStor.top->elem, flagLast: 1); //обработка
            flagFinish = 1;
        } else putL2(myStor.top->elem, flagLast: 0); //обработка
        myStor.top->elem = NULL;
        delete myStor.top;
        myStor.top = temp;
        if (flagFinish == 1) break; // что бы не делать лишний шаг цикла
    }
}; //перенос данных из стека в двунаправленную очередь готово

int intMaker(int rangeTemp, char elem) //готов
{
    int rangeTempNew = rangeTemp;
    if (elem >= '0' && elem <= '9')
    {
        rangeTempNew *= 10;
        rangeTempNew += (elem - '0');
    }
    else if (elem == ' ' || elem == '\n') exceptionHelper( flag: 1);
    else
    {

```

```

    cout << "sorry, you write not a number";
    exit( _Code: 0);
}
return rangeTempNew;
} // вспомогательный метод для корректного перебора int

void rangParser(string range) // парсим и заполняем данные границ
{
    int i = 0;
    int rangeTemp = 0;
    for(i; range[i]!=' '; i++)
    {
        rangeTemp = intMaker(rangeTemp,range[i]);
    }
    i++;
    myStor.rangTop = rangeTemp;
    rangeTemp = 0;
    for(i; i<range.length(); i++)
    {
        rangeTemp = intMaker(rangeTemp,range[i]);
    }
    myStor.rangLast = rangeTemp;
    if (myStor.rangTop <= myStor.rangLast)
    {
        exceptionHelper( flag: 1);
    }
};

void exceptionHelper(int flag)
{
    if(flag == 0)
    {
        cout << "Somphing error in your path to file. Pleas, write all again.";
        exit( _Code: 0);
    }
    else
    {
        cout << "sorry, you write wrong boundaries. Pleas, check the specification again";
        exit( _Code: 0);
    }
}

void printer()
{
    ofstream out; // поток для записи
    out.open(myStor.nameOfFile, ios::app); // открываем файл для записи
    if (out.is_open()) {
        Sumb *temp = myStor.firstElem;
        while (true)

```

```

        out << temp->elem;
        out << '\n';
        temp->elem = NULL;
        if (temp->nexElem == NULL)
        {
            temp->nexElem = nullptr;
            break;
        }
        temp = temp->nexElem;
        temp->prevElem->nexElem = nullptr;
        temp->prevElem = nullptr;
    }
}

out.close();
}; //финальный вывод отсортированного 12 списка

void reader(string nameOfFile) // чтение списка
{
    ifstream in;
    in.open(nameOfFile);
    string line;
    if (in.is_open())
    {
        getline(&in, &line);
        rangParser(line);
        while (getline(&in, &line))
        {
            {
                stackPut(info: stoi(line));
            }
        }
        in.close();
    }
};

int pathChecker(string nameOfFile)
{
    ifstream file;
    int result = 0;
    file.open(nameOfFile);
    file.seekg(0, ios::end);
    if (!(file.is_open())) {
        NumberWorker().exceptionHelper(flag: 0);
    }
    if (file.tellg() == 0) result = 1;
    file.close();
    return result;
};
};

```

```

int main() {
    string nameOfFile;
    cout << "Hello!\n"
        "This program can will sort in ascending order and crop numbers by somphing range.\n"
        "On the start you need take path to input file (with inform) and path to output file (where program put result).\n"
        "On file input you need in first line put the range by pattern 'MAX MIN'(example 290 56),\n"
        "and on the other line data (new element on new line!)\n"
        "Example of file:\n"
        "290 56\n"
        "23\n"
        "56\n"
        "13\n"
        "100\n"
        "200\n"
        "150\n"
        "Please, take path on file input\n";
    cin >> nameOfFile; // where i take text
    if (NumberWorker().pathChecker(nameOfFile) == 1) NumberWorker().exceptionHelper( flag: 0);
    cout << "Please, write path to output file\n";
    cin >> myStor.nameOfFile; // where i write answer
    NumberWorker().pathChecker(myStor.nameOfFile);
    NumberWorker().reader(nameOfFile); // считываю стек
    NumberWorker().moveToL2(); // переношу данные в двунаправленный список
    NumberWorker().sort(); // сортирую полученный список

    return 0;
}

```

## Примеры работы программы

Больше тестов содержится в папке “тесты”.

1) проверка с корректными путями и входящими данными

входные данные

test0 – Блокнот

Файл Правка Формат

290 40

23

45

200

30

66

20

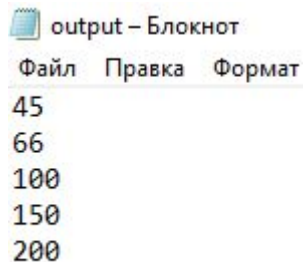
150

100

## Приветствие и введенные пути

```
C:\tmp\prog_2_1_17\cmake-build-debug\prog_2_1_17.exe
Hello!
This program can will sort in ascending order and crop numbers by somphing range.
On the start you need take path to input file (with inform) and path to output file (where program put result).
On file input you need in first line put the range by pattern 'MAX MIN'(example 290 56),
and on the other line data (new element on new line!)
Example of file:
290 56
23
56
13
100
200
150
Please, take path on file input
C:\tmp\test0.txt
Please, write path to output file
C:\tmp\output.txt
```

## Вывод программы

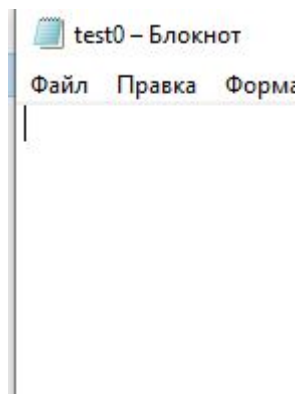


```
output - Блокнот
Файл  Правка  Формат
45
66
100
150
200
```

## 2) Некорректные пути

```
C:\tmp\prog_2_1_17\cmake-build-debug\prog_2_1_17.exe
Hello!
This program can will sort in ascending order and crop numbers by somphing range.
On the start you need take path to input file (with inform) and path to output file (where program put result).
On file input you need in first line put the range by pattern 'MAX MIN'(example 290 56),
and on the other line data (new element on new line!)
Example of file:
290 56
23
56
13
100
200
150
Please, take path on file input
C:\tmp\nbnnbn
Somphing error in your path to file. Pleas, write all again.
```

## 3) пустой файл ввода



## ВЫВОД программы

```
C:\tmp\prog_2_1_17\cmake-build-debug\prog_2_1_17.exe
Hello!
This program can will sort in ascending order and crop numbers by somphing range.
On the start you need take path to input file (with inform) and path to output file (where program put result).
On file input you need in first line put the range by pattern 'MAX MIN'(example 290 56),
and on the other line data (new element on new line!)
Example of file:
290 56
23
56
13
100
200
150
Please, take path on file input
C:\tmp\test0.txt
Somphing error in your path to file. Pleas, write all again.
Process finished with exit code 0
```

## Выводы

В ходе работы я научился работать со стеком и двунаправленной очередью, делать вставку и удаление элементов.