МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

ТЕМА: ЛИНЕЙНЫЕ СТРУКТУРЫ ДАННЫХ: СТЕК, ОЧЕРЕДЬ, ДЕК

Студентка гр. 7381 _____ Кушкоева А.О.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2018

Цель работы.

Познакомиться с часто используемыми на практике линейными структурами данных, обеспечивающими доступ к элементам последовательности только через её начало и конец, и способами реализации этих структур, освоить на практике использование стека, очереди и дека для решения задач.

Постановка задачи.

За один просмотр заданного файла F (типа file of Real) и без использования дополнительных файлов вывести элементы файла F в следующем порядке: сначала - все числа, меньшие a, затем - все числа на отрезке $[a, b]$ и наконец - все остальные числа, сохраняя исходный взаимный порядок в каждой из этих групп чисел (a и b задаются пользователем, $a < b$).

Описание алгоритма.

Программа считывает из файла контрольные два числа a и b, затем ряд чисел, которые необходимо сортировать относительно контрольных. Создаются три очереди, реализованные на базе вектора. Каждое число, соответствующее определенному условию сравнения: меньше a, в промежутке от a до b и больше b, заносится в одну из очередей. После окончания обработки данных результат выводится на консоль.

Описание функций и структур данных.

Переменные, используемые в функции main:

- ifilename, ofilename - входной файл.
- fin - входной потоки.
- a, b – граничные значения, относительно которых будет происходить сортировка входных данных.

Создаются три очереди для чисел меньше a, между a и b, больше b. Пока входной файл не закончится, оттуда считываются числа и добавляются в соответствующую очередь

Создается шаблонный класс Queue с шаблонным параметром T (тип хранимых элементов), представляет из себя циклическую очередь на базе динамического массива.

Класс содержит следующие поля:

vsize - длина динамического массива (вектора). qsize - длина очереди. vstart - указатель на начало массива. qstart — указатель на начало очереди. qend - указатель на конец очереди.

Методы класса Queue:

1) Queue(unsigned int new_size) : vsize(size), qsize(0);

Конструктор, принимающий длину массива. Выделяется память под массив, все указатели устанавливаются на начало массива.

2) void push(T value);

Принимает объект типа T, который добавляется в конец очереди по указателю qend, qend сдвигается вправо с учетом цикличности очереди. Если очередь переполнена, то сначала увеличивается в два раза размер массива, а после добавляется элемент.

3) void resize(insigned int new_size);

Принимает новый размер массива. Создается новый массив и копируется в него содержимое старого массива, после чего память под старый массив высвобождается.

4) bool isEmpty();

Возвращает true, если очередь пуста.

5) T pop();

Возвращает первый элемент из очереди по указателю qstart, он сдвигается вправо с учетом цикличности очереди.

6) ~Queue();

Деструктор.

Тестирование программы.

Программа собрана и проверена в операционных системах Xubuntu 18.04 с использованием компилятора g++ и Windows с использованием MinGW. В других ОС и компиляторах тестирование не проводилось.

Таблица 1 – Тестирование программы

Input	Output
a = 110, b = 1500 965 524 123 557 111 125 57 58 63 22 10 111111 12212 5161	57 58 63 22 10 965 524 123 557 111 125 111111 12212 5161
A = 100, b = 200 89 56 20 11 0 4 2 11 211 365 78 966 20 100 6877 6454 444554 21121 1213 21122 23323 6565 45545 1221 484 5445 212121 484848 656 332 21122 23323 6565 45545 1221 484 5445 212121 484848 656 332 250 21122 23323 6565 45545 1221 484 5445 212121 484848 656 332	89 56 20 11 0 4 2 11 78 20 -6877 100 211 365 966 6454 444554 21121 1213 21122 23323 6565 45545 1221 484 5445 212121 4848 48 656 332 21122 23323 6565 45545 1221 484 5445 212121 484848 656 332 250 21122 23323 6565 45545 1221 484 5445 212121 48 4848 656 332

Вывод.

В ходе лабораторной работы были получены основные навыки программирования линейной структуры – очереди – на языке C++. Результатом стала программа, которая использует очередь на базе массива для вывода чисел из файла по заданному условию.

ПРИЛОЖЕНИЕ А

Код головной программы

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include "queue.hpp"
using namespace std;

	int main()
	{
		string name_file;
		int a, b;

		cout << "ввод имени файла" << endl;
		getline(cin, name_file, '\n');
		cout << "Введите a, b (a<b):";
		cin >> a;
		cin >> b;

		ifstream fin(name_file);
		if (!fin) {
			cout << "File not open for reading!\n";
			return 1;
		}

		int begin_size = 5;
		Queue<int> less(begin_size);
		Queue<int> between(begin_size);
		Queue<int> bigger(begin_size);

		int value;
		while (fin >> value) {
			if (value < a) {
				less.push(value);
			}

			if (value >= a && value <= b) {
				between.push(value);
			}

			if (value > b) {
```

```
                bigger.push(value);
            }
        }

        fin.close();

        cout << "-------- Numbers --------" << endl;

        while (!less.isEmpty()) {
            cout << less.pop() << " ";
        }

        while (!between.isEmpty()) {
            cout << between.pop() << " ";
        }

        while (!bigger.isEmpty()) {
            cout << bigger.pop() << " ";
        }

        cout << endl;
        cout << "------------------------" << endl;
    }
```

ПРИЛОЖЕНИЕ Б
ФАЙЛ QUEUE.H

```
template <class T>// исп для описания класса queue как шаблона, чтобы хранить в нем данные любого типа; T-параметр шаблона класса
class Queue { //создание класса
public:
	Queue(unsigned int); //конструктор
	unsigned int get_size();//функция для получения длины очереди
	void push(T); //добавление элемента в очередь
	void resize(unsigned int); //увеличение размера массива(вектора)
	bool isEmpty();//проверка на пустоту очереди
	T pop(); //удаление элемента из очереди
	~Queue(); //деструктор
private:
	unsigned int vsize; //размер вектора
	unsigned int qsize; //размер очереди
	T* vstart; //указатель на начало вектора
	T* qstart; //указатель на начало очереди
	T* qend; // указатель на конец очереди
};

template <class T>
Queue<T>::Queue(unsigned int size) : vsize(size), qsize(0) {
//конструктор
	vstart = new T[size];
	qstart = vstart;
	qend = vstart;
}

template <class T>
unsigned int Queue<T>::get_size() { //получение размера очереди
	return qsize;
}
```

```
template <class T>
void Queue<T>::push(T value) { //value = symbol
	if (qend != qstart || (qsize == 0 && vsize != 0))
		*qend = value;
	else {
		resize(vsize * 2);
		*qend = value;
	}
	if (qend < (vstart + vsize - 1))
		++qend;
	else
		qend = vstart;
	++qsize;
}

template <class T>
void Queue<T>::resize(unsigned int new_size) { // увеличение
вектора,создается новый массив и в него копируется старый массив
	T* new_vect = new T[new_size];
	int i;
	for (i = 0; i < (vsize - (qstart - vstart)); ++i)
		new_vect[i] = qstart[i];
	for (int j = 0; i < vsize; ++i, ++j)
		new_vect[i] = vstart[j];
	qstart = new_vect;
	qend = new_vect + qsize;
	delete vstart;
	vstart = new_vect;
	vsize = new_size;
}

template <class T> //проверка на пустоту очереди
bool Queue<T>::isEmpty() {
```

```
	return qsize == 0;
}

template <class T>
T Queue<T>::pop() { //функция возвращает первый элемент и сдвигает
очередь вправо
	T ret = *qstart;
	if (qstart != (vstart + vsize - 1))
		++qstart;
	else
		qstart = vstart;
	--qsize;
	return ret;
}

template <class T>
Queue<T>::~Queue() { //деструктор
	delete vstart;
}
```