МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Отчет по лабораторной работе**

на тему:

**«Структура данных очередь»**

**Выполнил:** студент группы 3822Б1ПР2

Наумов Богдан Александрович

Нижний Новгород

2023

Оглавление

[1. Введение 3](#_Toc153711256)

[2. Постановка задачи 4](#_Toc153711257)

[3. Руководство пользователя 5](#_Toc153711258)

[4. Руководство программиста 6](#_Toc153711259)

[4.1.Описание структуры программы 6](#_Toc153711260)

[4.2.Описание структур данных 6](#_Toc153711261)

[4.3.Описание алгоритмов 7](#_Toc153711262)

[5. Эксперименты 8](#_Toc153711263)

[6. Заключение 9](#_Toc153711264)

[7. Приложение 10](#_Toc153711265)

## Введение

Очередь - это набор элементов, управляемых по принципу "первый пришел - первый ушел" (FIFO). То есть элементы добавляются в конец очереди и удаляются из ее начала. Очереди широко используются в программировании для моделирования различных процессов, таких как обработка запросов и управление задачами.

Очереди могут быть реализованы с помощью массивов или связанных списков. Основные операции, которые можно выполнять с очередью, включают добавление элементов в конец (enqueue), извлечение элементов из начала (dequeue) и извлечение элементов из начала без их удаления (peek).

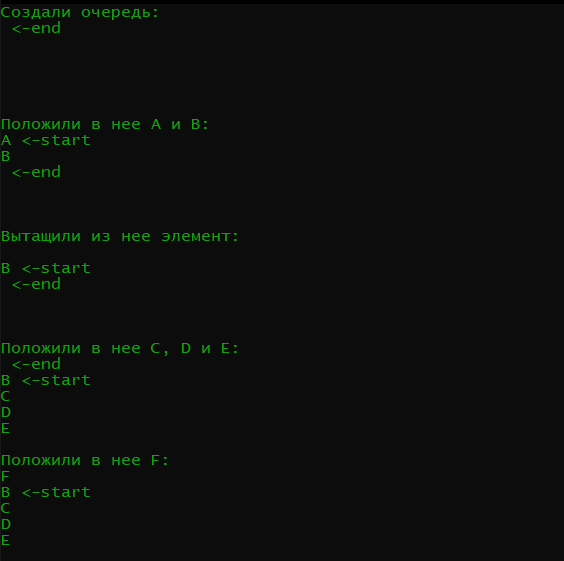
## Постановка задачи

Чтобы выполнить работу нужно решить следующие задачи:

1. Реализовать шаблонный класс TQueue.
2. Написать пример программы, которая будет демонстрировать работу класса TQueue.
3. Написать набор тестов с использованием GoogleC++ TestingFramework для проверки работы класса TQueue.

## Руководство пользователя

При запуске программы, демонстрирующей работу класса TQueue, она выводит на экран следующие примеры:



*Рис.1 (результат работы программы)*

## 4. Руководство программиста

## 4.1.Описание структуры программы

Программа состоит из следующих модулей:

* Модуль **include**. Статическая библиотека. Включает в себя заголовочный файл TQueue.h, в котором описаны методы с реализацией класса TQueue.
* Модуль **test\_TQueue**. Набор тестов для реализованного класса. Включает в себя файл **test\_TQueue.cpp**. Реализованы они с помощью использования фреймворка **Google Test**.
* Модуль **samples** - пример использования классов. Включает в себя файл с реализацией **sample\_TQueue.cpp**.

## 4.2.Описание структур данных

Класс **TQueue**:

Поля:

int size – размер очереди

int start – индекс начала очереди

int end – индекс конца очереди

int count – количество элементов в очереди

T\* pMem – массив для хранения элементов очереди

Конструкторы и деструктор:

TQueue(int \_size = 1)

TQueue(TQueue<T>& q)

~TQueue()

Методы:

void Push(T a) – поместить элемент в очередь

T Get() – вытащить элемент из очереди

bool IsFull() – проверка на заполненность

bool IsEmpty() – проверка на пустоту

Перегрузки:

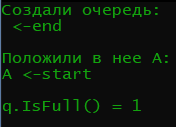
friend ostream& operator<<(ostream& ostr, const TQueue& q) - вывод

## 4.3.Описание алгоритмов

В очереди элементы удаляются в порядке, в котором они были добавлены, применяя принцип "первым вошел — первым вышел" (англ. first-in, first-out — FIFO). Очередь имеет начало (start) и конец (end), и когда элемент помещается в очередь, он занимает место в ее конце. Элемент, который выводится из очереди, всегда находится в ее начале.

## 5. Эксперименты

Очередь размера 1:



## 6. Заключение

В данной лабораторной работе все задачи были успешно выполнены. Была создана структура данных - очередь. Работа на практике и изучение устройства автоматических тестов помогли мне лучше понять ее принципы. Это позволяет убедиться в правильной работе реализованного класса.

## 7. Приложение

TQueue.h

#include <iostream>

using namespace std;

template <class T>

class TQueue

{

protected:

int size;

int start;

int end;

int count;

T\* pMem;

public:

TQueue(int \_size = 1) : size(\_size), start(-1), end(0), count(0), pMem(new T[size]) {

if (size <= 0)

throw -1;

for (int i = 0; i < size; i++)

pMem[i] = NULL;

}

TQueue(TQueue<T>& q) {

this->size = q.size;

this->pMem = new T[size];

this->start = q.start;

this->end = q.end;

for (int i = 0; i < size; i++)

this->pMem[i] = q.pMem[i];

}

~TQueue() {

if (this->pMem != nullptr)

delete[] this->pMem;

}

void Push(T a) {

if (IsFull())

throw "Cannot push elements, queue is full";

if (start < 0) {

start = 0;

}

count++;

pMem[end] = a;

if (end == size - 1) end = 0;

else end++;

}

T Get() {

if (IsEmpty())

throw "Cannot get elements, queue is empty";

T res = pMem[start];

pMem[start] = NULL;

count--;

if (start == size - 1) start = 0;

else start++;

return res;

}

bool IsFull() {

return count == size;

}

bool IsEmpty() {

return count == 0;

}

friend ostream& operator<<(ostream& ostr, const TQueue& q) {

for (int i = 0; i < q.size; i++) {

if (q.start == i)

ostr << q.pMem[i] << " <-start" << endl;

else if (q.end == i)

ostr << q.pMem[i] << " <-end" << endl;

else

ostr << q.pMem[i] << endl;

}

return ostr;

}

};

sample\_TQueue.cpp

#include "TQueue.h"

#include <locale.h>

using namespace std;

int main() {

setlocale(LC\_ALL, "Rus");

TQueue<char> q(5);

cout << "Создали очередь:" << endl << q << endl;

q.Push('A');

q.Push('B');

cout << "Положили в нее А и В:" << endl << q << endl;

q.Get();

cout << "Вытащили из нее элемент:" << endl << q << endl;

q.Push('C');

q.Push('D');

q.Push('E');

cout << "Положили в нее C, D и E:" << endl << q << endl;

q.Push('F');

cout << "Положили в нее F:" << endl << q << endl;

}