МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**Отчет по учебной практике**

**«Структура данных: Очередь»**

**Выполнил:** студент группы 381706-1

Нечаева Екатерина Владимировна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc532752145)

[2. Постановка задачи 4](#_Toc532752146)

[3. Руководство пользователя 5](#_Toc532752147)

[4. Руководство программиста 6](#_Toc532752148)

[4.1 Описание структуры программы 6](#_Toc532752149)

[4.2 Описание структур данных 6](#_Toc532752150)

[4.3 Описание алгоритмов 7](#_Toc532752151)

[5. Заключение 8](#_Toc532752152)

[6. Литература 9](#_Toc532752153)

# Введение

В данной работе будет рассмотрена структура данных – очередь.

Очередь характеризуется таким порядком обработки значений, при котором вставка новых элементов производится в конец очереди, а извлечение – из начала.

Очередь в программировании используется, как и в реальной жизни, когда нужно совершить какие-то действия в порядке их поступления, выполнив их последовательно.

Существует несколько способов реализации очереди:

* с помощью одномерного массива;(используется в данной работе)
* с помощью связанного списка;
* с помощью класса объектно-ориентированного программирования.

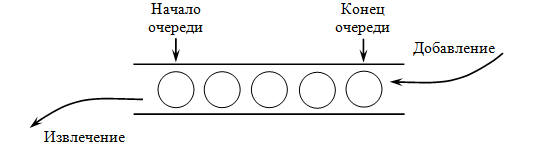


Рисунок 1. Схема работы очереди.

# Постановка задачи

В рамках данной лабораторной работы нужно реализовать структуру данных – очередь.

Хранить данную структуру нужно с помощью массива. Данный способ позволяет организовать и впоследствии обрабатывать очередь, имеющую фиксированный размер.

Вместо «мигрирующей» очереди, наиболее приемлемо реализовать очередь на базе циклического массива. Таким образом, очередь становится круговой. Вместо того, чтобы отслеживать только «конец» , мы также должны следить за началом очереди. Когда она подходит к концу массива, очередь возвращается на его начало.

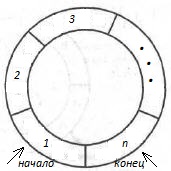


Рисунок 1. Круговая очередь.

Для работы с очередью будут реализованы операции:

* добавления элемента в очередь
* извлечения элемента из очереди с последующим удалением
* проверка очереди на полноту/пустоту
* печать очереди на консоль

# Руководство пользователя

В начале запуска программы пользователю предлагается ввести размер очереди. После этого есть два варианта: либо заполнить данные вручную, либо случайными числами. Выбрав один из вариантов, на консоль выводится полученная очередь. Затем программой извлекается первый элемент и снова выводится уже новая очередь. После этого пользователем вводится новое число, которое записывается в конец очереди и выводится на экран.

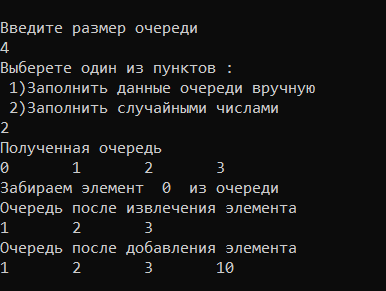


Рисунок 1. Пример использования очереди и ее методов.

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль queue. Содержит пример использования очереди(*mainqueue.cpp*)*.*
* Модуль libqueue – статическая библиотека. Содержит заголовочный файл *queueh*, в котором реализован шаблонный класс TQueue.
* Модуль queueTest. Содержит 5 тестов, описанных в файле *queueTest.cpp*.
* Модуль libthrows – библиотека для исключений. Содержит заголовочный файл *throws.h*, в котором он и реализован.

## Описание структур данных

#### Класс TException – класс исключений.

Содержит одно защищенное поле *string str,* переменная строкового типа, которая хранит сообщение об исключении.

Содержит публичные методы

* TException(std::string \_str); //конструктор-инициализатор
* void Print(); //вывод данного исключения на консоль

#### Класс TQueue – очередь.

**Поля, которые наследуются от стека:**

* int size; //размер стека
* int top; //вершина стека
* T\* mas; //массив элементов

**Поля класса queue:**

* int count; //кол-во элементов в очереди
* int start; //начало очереди

**Публичные методы:**

TQueue(int \_size = 5); //конструктор-инициализатор

void Put(const T &A); //добавить элемент в очередь

T Get(); //вывести элемент с последующим удалением

void Print(); //печать очереди на консоль

bool IsEmpty(); //проверка на пустоту

bool IsFull(); //проверка на полноту

## Описание алгоритмов

**Добавление элемента в очередь.**

При добавлении элемента в очередь обращаемся к переменной *top*, увеличиваем

значение на 1 и в массив с данным индексом добавляем элемент. Чтобы значение *top* не вышло за границы массива используется формула при проверке *top > size* (размер очереди) и если это выполняется следует формула *top = (top + 1) % size*. Также увеличивается *count* на 1.

**Удаление элемента из очереди.**

При удалении элемента из очереди обращаемся к переменной *start* и забираем значение по этому индексу. Значение start увеличиваем на 1*.* Чтобы значение *start* не вышло за границы массива используется формула при проверке *start > size* (размер очереди) и если это выполняется следует формула *start = (start + 1) % size*. Уменьшаем значение *count* на 1.

# Заключение

В данной лабораторной работе были рассмотрены понятие очереди и реализация класса Queue, то есть круговая очередь на основе массива.

Также были реализованы операции добавления, удаления элемента из очереди и проверки на полноту и пустоту.

Программа прошла все созданные тесты на базе Google Test.

# Литература

1. Википедия: свободная электронная энциклопедия: на русском языке [Электронный ресурс]

https://ru.wikipedia.org/wiki/Очередь\_(программирование)

1. Очередь [Электронный ресурс]

https://prog-cpp.ru/data-queue/

1. Структура данных: очередь [Электронный ресурс]

<http://cppstudio.com/post/5159/>

1. Очередь (структура данных) [Электронный ресурс]

<http://kvodo.ru/queue.html>