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**

**«Очередь»**

**Выполнил:** студент группы 381706-1

Корнев Никита Алексеевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533799739)

[2. Постановка задачи 4](#_Toc533799740)

[3. Руководство пользователя 5](#_Toc533799741)

[4. Руководство программиста 6](#_Toc533799742)

[4.1. Описание структуры программы 6](#_Toc533799743)

[4.2. Описание структур данных 7](#_Toc533799744)

[4.3 Описание алгоритмов 8](#_Toc533799745)

[5. Заключение 9](#_Toc533799746)

[6. Литература 10](#_Toc533799747)

# Введение

Структура данных (англ. data structure) — программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных в вычислительной технике. Для добавления, поиска, изменения и удаления данных структура данных предоставляет некоторый набор функций, составляющих её интерфейс.

О́чередь — абстрактный тип данных с дисциплиной доступа к элементам «первый пришёл — первый вышел» (FIFO, англ. first in, first out). Добавление элемента (принято обозначать словом enqueue — поставить в очередь) возможно лишь в конец очереди, выборка — только из начала очереди (что принято называть словом dequeue — убрать из очереди), при этом выбранный элемент из очереди удаляется.

Очередь в программировании используется, как и в реальной жизни, когда нужно совершить какие-то действия в порядке их поступления, выполнив их последовательно. Примером может служить организация событий в Windows. Когда пользователь оказывает какое-то действие на приложение, то в приложении не вызывается соответствующая процедура (ведь в этот момент приложение может совершать другие действия), а ему присылается сообщение, содержащее информацию о совершенном действии, это сообщение ставится в очередь, и только когда будут обработаны сообщения, пришедшие ранее, приложение выполнит необходимое действие.

Практически во всех развитых языках программирования реализованы очереди. В CLI для этого предусмотрен класс System.Collections.Queue с методами Enqueue и Dequeue. В STL также присутствует класс queue<>, определённый в заголовочном файле queue. В нём используется та же терминология (push и pop), что и в стеках.

В данной лабораторной работе нам предстоит написать собственную реализацию данной структуры.

# Постановка задачи

1. Написать класс, реализующий очередь.
2. Написать тесты на основе Google Tests для проверки работы класса.

# Руководство пользователя

Чтобы использовать данный класс в своем проекте, необходимо подключить библиотеку «Queue.h». Библиотека позволяет:

* Создавать объекты типа *очередь*:

Queue q1; **конструктор по умолчанию**

Queue q2(10); **конструктор с параметром**

Queue q3 (q2); **конструктор копирования**

* Ставить элементы в *очередь*:

Queue q(5);

q.Put(1);

* Брать элементы из *очереди*:

Queue q(5);

for (int i = 0; i < 5; i++)

q.Put(i);

q.Get();

* Получать размер *очереди*:

q.GetSize();

* Проверять на *пустоту*:

q.IsEmpty();

* Проверятьна *полноту*:

q.IsFull();

# Руководство программиста

## Описание структуры программы

Проект состоит из следующих модулей:

1. Queue

Модуль cодержит файл код «main.cpp», в котором продемонстрирован пример использования структуры.

1. QueueLib

Модуль содержит файл заголовок «Queue.h», описывающий структуру «очередь», а также файл кода «main.cpp». Так как класс шаблонный, main содержит лишь подключение.

1. QueueTest

Модуль содержит файлы кода «test\_main.cpp», «test\_queue.cpp». В последнем реализованы тесты для проверки корректности работы методов данного класса.

## Описание структур данных

#### *Структура TQueue*

Структура **TQueue** реализована в виде класса **TQueue,** наследуемого от класса **TStack.** Поля со спецификатором доступа «protected»:

* + **int start;** *Размер очереди*
  + **int count;** *Конец очереди*

Поля со спецификатором доступа «public»:

* **TQueue();** *Конструктор по умолчанию*
* **TQueue(int size);** *Конструктор с параметром*
* **TQueue(TQueue <T> &A);** *Конструктор копирования*
* **void Put(T A);** *Положить в конец очереди*
* **T Get();** *Взять элемент*
* **bool IsFull();** *Проверка на полноту*
* **bool IsEmpty();** *Проверка на пустоту*

## Описание алгоритмов

Описание некоторых алгоритмов:

* TQueue();

Конструктор по умолчанию, создает очередь нулевого размера.

* TQueue(int size); Конструктор с параметром

Конструктор с параметром, создает очередь размера *size.*

* TQueue(TQueue <T> &A);

Конструктор копирования, создает очередь от уже имеющейся очереди.

* void Put(T A);

Если очередь заполнена – выбрасывает исключение. Присваивает первому (от начала) свободному месту значение *А,* увеличивает значения start и count на 1.

* T Get();

Если очередь пуста – выбрасывает исключение. Возвращает значение первого на очереди элемента. Перемещает значение *top*, характеризующее номер первого занятого места в очереди.

* bool IsFull();

Если count равен размеру очереди, возвращает 1, в противном случае 0.

* bool IsEmpty();

Если count равен 0, возращает 1, в противном случае 0.

# Заключение

В данном лабораторной работе мне удалось:

* Успешно реализовать класс для очереди
* Продемонстрировать пример использования данного класса
* Написать тесты на основе Google Tests для проверки корректной работы данного класса

# Литература

Интернет-источники:

1. <https://ru.wikipedia.org/wiki/Структура_данных>
2. <https://ru.wikipedia.org/wiki/Очередь_(программирование)>

Книги:

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.
2. A.O. Грудзинский. Методы программирования, Издательство Нижегородского госуниверситета, 2006.