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 хранения данных: Очередь»**

**Выполнил:** студент группы 381706-1

Соболева Ю.А.

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

аспирант каф. МОСТ ИИТММ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г.

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533027637)

[2. Постановка задачи 4](#_Toc533027638)

[3. Руководство пользователя 5](#_Toc533027639)

[4. Руководство программиста 6](#_Toc533027640)

4.[1. Описание структуры программы 6](#_Toc533027641)

4.[2. Описание структур данных 6](#_Toc533027642)

4.[3. Описание алгоритмов 7](#_Toc533027643)

[5. Заключение 8](#_Toc533027644)

[6. Литература 9](#_Toc533027645)

# **Введение**

**Структура данных** - программная единица, которая определяет метод хранения и обработки различных логически связанных данных в вычислительной технике. Знание структур данных позволяет наиболее компактно и практично расположить данные в памяти компьютера. В данной работе мы рассмотрим такую структуру данных, как **очередь**.

**Очередь –** это структура данных, представляющая собой последовательность элементов, образованная в порядке их поступления. Каждый новый элемент размещается в конце очереди; элемент, стоящий в начале очереди, выбирается из нее первым. В очереди используется принцип доступа к элементам FIFO ( First Input – First Output, "первый пришёл – первый вышел"). В очереди доступны два элемента (две позиции): начало очереди и конец очереди. Поместить элемент можно только в конец очереди, а взять элемент только из ее начала.

**Целью данной лабораторной работы** является разработка структуры данных для хранения очередей, а также освоение таких инструментов разработки программного обеспечения, как система контроля версий Git и фрэймворк для разработки автоматических тестов Google Test.

# **Постановка задачи**

Выполнение работы предполагает решение следующих задач:

1. Реализация класса очереди TQueue, унаследованного от класса TStack.
2. Разработка интерфейса для данных классов.
3. Обеспечение работоспособности примера использования.
4. Реализация нескольких тестов на базе Google Test.

# **Руководство пользователя**

Пример использования стека:

* При запуске программы у пользователя запрашивается количество элементов в очереди.
* Создается очередь заданного размера с элементами типа int.
* Затем пользователю предлагается ввести элементы очереди.
* Очередь выводится.
* Из очереди удаляется элемент, затем она выводится на экран.
* Проверяем очередь на полноту и пустоту.

# **Руководство программиста**

## **Описание структуры программы**

**Проект “Queue” состоит из следующих файлов:**

1. mainQueue.cpp (в нём находится main)

**Проект “QueueLib ” состоит из следующих файлов:**

1. Queue.h (описание класса “TQueue” реализация методов класса “TQueue” и перегрузка операций )
2. QueueLib.cpp

**Проект “ Test” состоит из следующих файлов:**

1. test\_queuelib.cpp (реализация тестов для класса TQueue)

## **Описание структур данных**

***Класс TQueue:***

*Поля:*

* int start; - начало очереди;

*Конструкторы и деструктор:*

* TQueue<T> (int \_size = 0); - конструктор инициализации;
* TQueue<T> (TQueue <T> &A); - конструктор копирования;

*Методы:*

* T Get() ;- забирает элемент из очереди;
* void Put(T а); - добавляет элемент в очередь;
* bool IsFull(); - проверяет полна ли очередь;
* bool IsEmpty(); - проверяет пуста ли очередь;

*Перегруженные операторы:*

Нет перегруженных операторов.

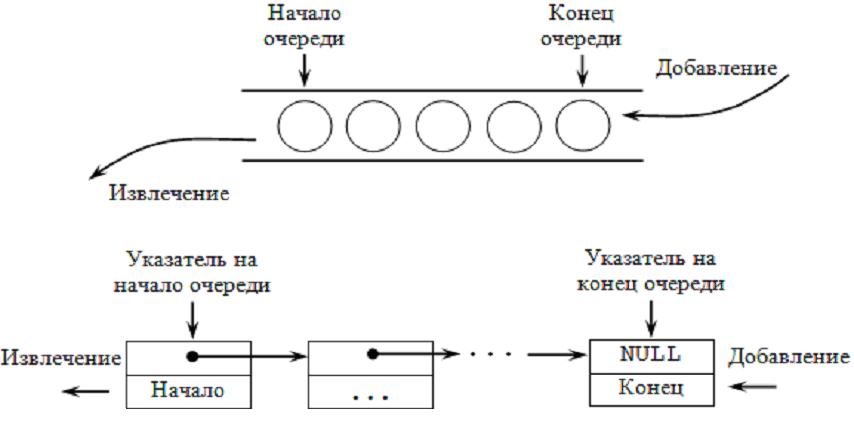
## **Описание алгоритмов**

Метод **Put:**.

При добавлении элемента в очередь обращаемся к элементу массива с индексом *start*, записываем туда полученное значение*.* Значение *start* увеличиваем следующим образом *start = (start + 1) % size*. Увеличиваем число элементов в очереди *count++*.

Метод **Get:**

При удалении элемента из очереди забираем значение элемента массива с индексом top. Значение *top* увеличиваем следующим образом *top = (top + 1) % size.* Уменьшаем число элементов в очереди *count--*.



# **Заключение**

В ходе выполнения лабораторной работы я смогла реализовать такую структуру данных, как очередь. Вместе с ней, согласно заданному интерфейсу, я реализовала класс TQueue. Написание нескольких своих тестов, помогло мне разобраться с системой автоматических тестов Google Test.

В результате проделанной работы у меня получилось

1. Реализовать класс очереди TQueue, унаследованного от класса TStack.

2. Разработать интерфейс для данных классов.

3. Обеспечить работоспособность примера.

4. Реализовать некоторые тесты на базе Google Test.

Таким образом, данная лабораторная работа отвечает всем поставленным задачам. Это означает, что цель работы была достигнута.

# **Литература**

1. Лабораторный практикум. Составители: Барышева И.В., Мееров И.Б., Сысоев А.В., Шестакова Н.В. Под редакцией Гергеля В.П. Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. – 105с.

URL: <http://www.unn.ru/books/met_files/Pract_ADS.pdf>

1. Национальный открытый университет «Интуит». Курс «Основы программирования». Лекция 11: Структуры данных: общее понятие, реализация. Простейшие структуры данных: очередь, стек. Использование стека и обратная польская запись.

URL: <https://www.intuit.ru/studies/courses/2193/67/lecture/1980?page=3>