МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ 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**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**Отчет по учебной практике**

**«Структура данных - очередь»**

**Выполнил:** студент группы 381706-1

Голованова Елена Александровна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018.

Оглавление

[1. Введение 3](#_Toc534459553)

[2. Постановка задачи 5](#_Toc534459554)

[3. Руководство пользователя 6](#_Toc534459555)

[4. Руководство программиста 7](#_Toc534459556)

[a. Описание структуры программы 7](#_Toc534459557)

[b. Описание структур данных 8](#_Toc534459558)

[c. Описание алгоритмов 9](#_Toc534459559)

[5. Заключение 11](#_Toc534459560)

[6. Список литературы 12](#_Toc534459561)

# Введение

Эффективное использование памяти является важной задачей программиста. Для решения этой задачи была придумана такая структура как очередь.

Очередь —  [абстрактный тип данных](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) с дисциплиной доступа к элементам «первый пришёл — первый вышел» ([FIFO](https://ru.wikipedia.org/wiki/FIFO), [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *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.

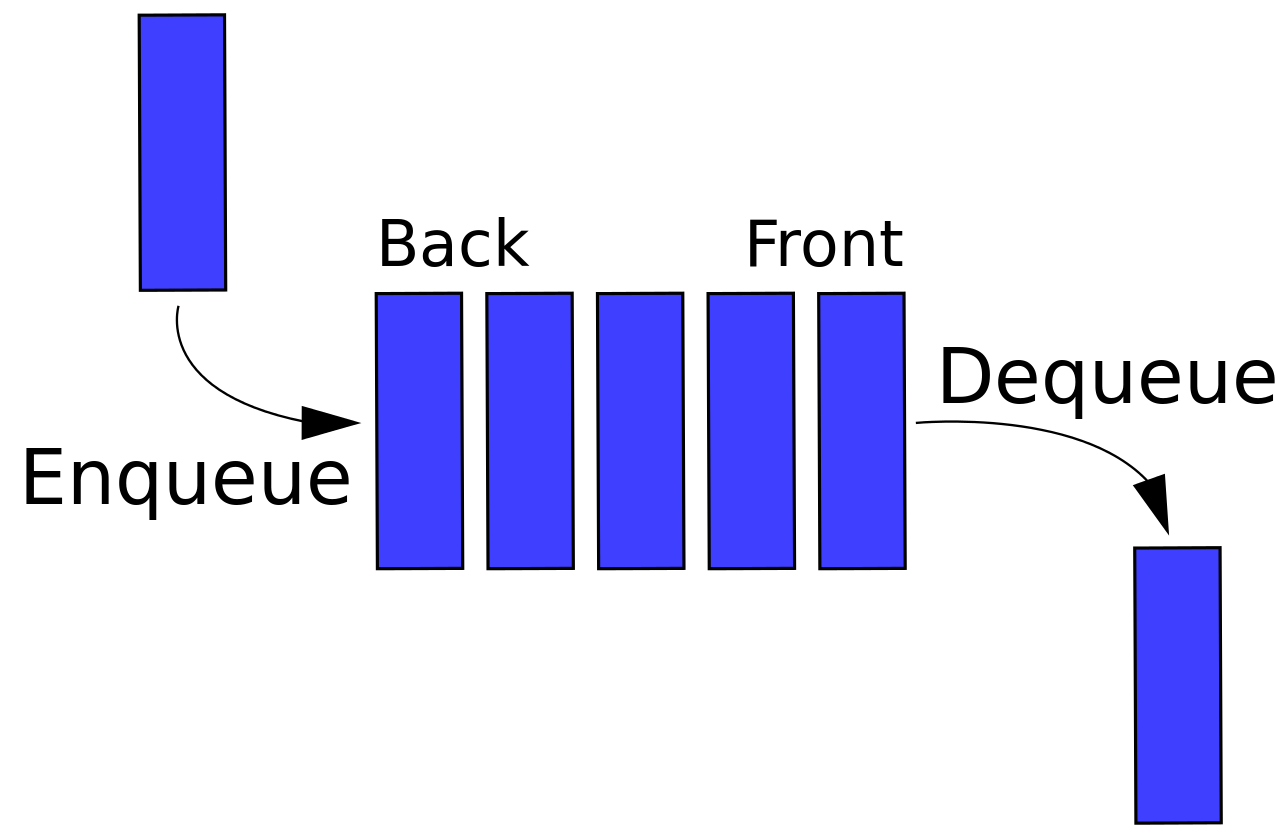


Рис. 1 - Организация очереди.

**Очередь имеет свои преимущества и недостатки.**

**Преимущества:**

* **Простота в разработке.**
* **Экономия памяти.**

**Недостатки:**

* **Количество элементов ограничено размером массива**
* при переполнении очереди требуется перевыделение памяти и копирование всех элементов в новый массив.

Цель данной лабораторной работы – реализовать такую структуру данных как очередь.

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Реализация класса TStack.
2. Реализация класса TQueue.
3. Обеспечение работоспособности тестов и примера использования.
4. Реализация простых тестов на базе Google Test.

# Руководство пользователя

Работа пользователя происходит так: на экране будут проверяться такие функции как добавление и извлечения элементов.

# 4. Руководство программиста

## Описание структуры программы

Для реализации лабораторной работы создается два класса, состоящих из нескольких модулей:

1. Класс TStack:

* Модуль StackLib

Этот модуль содержит реализацию класса стека. Он включает в себя следующие файлы:

* Stack.h – в этом файле определен интерфейс класса стека и реализация таких действий, как конструкторы, деструктор, перегрузка теоретико-множественные операторы (такие как сравнение, присваивание и т.д.), методы проверки ячеек стека на полноту/пустоту, добавления элемента, извлечения элемента из стека.
* Stack.cpp – подключение заголовочного файла Stack.h

2. Класс TQueue:

* Модуль QueueLib

Этот модуль наследуется от StackLib.

* Queue.h – в этом файле определен интерфейс класса очереди и реализация таких действий, как конструктор по умолчанию и с параметром, методы проверки ячеек очереди на полноту/пустоту, положения элемента в конец очереди, извлечения элемента.
* Queue.cpp – подключение заголовочного файла Queue.h.
* Модуль Queue

Пример использования очереди.

* Модуль Queue\_Test

Этот модуль содержит тесты, проверяющие работу каждого методов класса Queue.

* TException

Этот модуль содержит класс исключений.

## Описание структур данных

1. Класс TStack является шаблонным.
   * 1. В зоне protected определены три поля и два метода:

* int Size - размер стека
* int Top - элемент, расположенный на вершине стека
* T\* Mas - элементы стека

2. В зоне public:

* int GetSize() { return Size; } – Возврат размера стека.
* TStack(int n = 0); – Конструктор инициализации.
* TStack(TStack &S); – Конструктор копирования.
* T Get(); – Возврат элемента, расположенного на вершине стека.
* void PrintStack(); – Вывод стека на экран.
* void Put(T A); – Добавление элемента на вершину стека, если стек не полон.
* bool IsFull(); – Проверка на полноту.
* bool IsEmpty(); – Проверка на пустоту. Стек пуст, если в нем нет ни одного элемента, т.е. когда количество элементов равно нулю.
* int operator!=(const TStack& stack) const; – Принимает ссылку на объект класса TStack, выполняет проверку на неравенство.
* int operator==(const TStack& stack) const; – Принимает ссылку на объект класса TStack, выполняет проверку на равенство.
* TStack& operator=(const TStack& stack); – Принимает ссылку на объект класса TStack, приравнивает исходный объект к полученному.

1. Класс Queue является шаблонным.
   * + 1. В зоне protected определены:

* int Start - Начало очереди.
* int num - Кол-во элементов в очереди.

2. В зоне public:

* TQueue(int N = 0); – Конструктор с параметром.
* TQueue(TQueue &Q); – Конструктор копирования.
* T Get(); – Взятие элемента из начала очереди.
* void Put(T A); – Положить элемент в конец очереди.
* bool IsFull(); – Проверка на полноту.
* bool IsEmpty(); – Проверка на пустоту.

## Описание алгоритмов

**Класс TStack.**

В данном разделе будут описаны некоторые (нетривиальные) алгоритмы, используемые для реализации метода класса TStack.

* Put(T A) (На рисунке функция определена как Push)

Добавление элемента в стек происходит так: для начала нужно переместить указатель вершины стека, потом записать элемент в соответствующую позицию динамического массива и увеличить количество элементов. переместить указатель вершины стека, записать элемент в соответствующую позицию динамического массива и увеличить количество элементов.



Рис. 2 – Схема добавления и удаления элемента в стеке.

* T Get() (На рисунке функция определена как Pop)

Если стек не пуст: при удалении элемента из стека необходимо возвратить значение из динамического массива по индексу вершины стека, переместить указатель вершины стека и уменьшить количество элементов.

**Класс TQueue**

* void Put(T A)

Сначала необходимо проверить очередь на полноту. После этого переходим к следующему шагу: присваиваем элементу top пришедшее значение, затем передвигаем вершину. Далее увеличиваем число элементов в очереди на один.

* T Get()

Сначала необходимо проверить очередь на пустоту. Извлекаем элемент из начала очереди. Далее уменьшаем число элементов в очереди на один.

* bool IsEmpty()

Если число элементов в очереди равно нулю, то этот метод возвращает true, иначе – false.

* bool IsFull()

Если число элементов в очереди равно размеру очереди, то этот метод возвращает true, иначе – false.

# 5. Заключение

В результате выполнения лабораторной работы была разработана структура хранения данных – очереди, основанная на наследовании стека.

Получены новые знания и навыки.

Были реализованы тесты для проверки работоспособности вышеперечисленных классов на базе GoogleTest.

# 6. Список литературы

1. Википедия. Статья «Очередь»

[[https://neerc.ifmo.ru/wiki/index.php?title=%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C]](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C%5d)

1. Васильев А.Н. Самоучитель С++ с примерами и задачами. -СПб.: Наука и Техника, 2016. -480с.
2. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.