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**

оставь надежды всяк сие читающий (карин т.а.)

**Отчет по учебной практике**

**«Структура хранения очереди»**

**Выполнил:**

студент группы 381706­1

Карин Тимофей Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Проверил:**

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc1346832)

[3. Руководство пользователя 5](#_Toc1346833)

[4. Руководство программиста 6](#_Toc1346834)

[4.1. Описание структуры программы 6](#_Toc1346835)

[4.2. Описание структур данных 6](#_Toc1346836)

[4.3. Описание алгоритмов 7](#_Toc1346837)

[5. Заключение 9](#_Toc1346838)

[6. Список литературы 10](#_Toc1346839)

1. Введение

Структура данных — программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных [данных](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_(%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)) в [вычислительной технике](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80). Для добавления, поиска, изменения и удаления данных структура данных предоставляет некоторый набор функций, составляющих её интерфейс.

В данной работе будет рассмотрена такая структура данных как очередь. Она напоминает стек, но в очереди первым извлекается элемент, вставленный первым (FIFO, First-In-First-Out), тогда как в стеке, извлекается элемент, вставленный последним (LIFO). Она работает по тому же принципу, что и очередь в кино: человек, первым вставшим в очередь, первым доберется до кассы и купит билет. Тот, кто встанет в очередь последним, последним купит билет. Очередь — такой же вспомогательный инструмент программиста, как и стек. Кроме того, очереди используются для моделирования реальных ситуаций ожидания: клиентов в банке, вылета самолетов или передачи данных по Интернету.

2. Постановка задачи

Целью данной работы является создание библиотеки, предназначенной для хранения и обработки структуры хранения очереди. Для этого необходимо:

1. Создать класс для работы со стеком;
2. Реализовать методы:
   * конструктор по умолчанию, конструктор копирования, деструктор;
   * добавления элемента в очередь;
   * получения элемента из очереди;
   * проверки на пустоту;
   * проверки на полноту;
3. Обработать исключительные ситуации.

3. Руководство пользователя

Библиотека *QueueLib* предназначена для программистов и не предусматривает работу с пользователями. По этой причине был создан проект *Queue,* содержащий один из вариантов работы с *QueueLib*.

Файл *Queue.cpp,* содержащийся в проекте *Queue,* представляет из себя программу, в которой пользователь может создать свою очередь, состоящую из целочисленных значений, добавлять и извлекать из неё элементы, делать проверки на полноту и пустоту, а также выводить элементы, содержащиеся в очреди на экран.

После запуска программы пользователю необходимо ввести количество элементов в создаваемой очереди.

После создания очереди появится меню действий. Пользователю нужно ввести номер команды:

1. Положить число в очередь;
2. Получить число из очереди;
3. Проверить очередь на полноту;
4. Проверить очередь на пустоту;
5. Вывести элементы очереди на экран;
6. Выйти из программы.

Если в ходе выполнения программы произойдёт исключительная ситуация, на экран будет выведено сообщение об ошибке:

1. Отрицательный размер очереди (происходит, когда пользователь пытается создать очередь с отрицательным количеством элементов);
2. Полная очередь (выбрасывается, когда пользователь пытается положить элемент в уже заполненную очередь);
3. Пустой стек (выбрасывается, когда пользователь пытается забрать элемент из пустой очереди).

Для завершения работы программы необходимо выбрать соответствующий пункт меню.

4. Руководство программиста

4.1. Описание структуры программы

Программа состоит из трёх проектов:

* *QueueLib* – библиотека для работы с очередью. Состоит из файла TQueue.h
* *Queue* – пример использования библиотеки *StackLib*. Состоит из файла Queue;
* *QueueTest* – набор тестов для проверки работоспособности библиотеки *QueueLib.*

Кроме того, в программе используется написанная ранее библиотека *StackLib*, т.к. класс *TQueue* наследуется от класса *TStack*.

Также в программе используется класс исключений *TEсxeption*, содержащийся в файле *Exсeption.h* проекта *Exсeption*.

4.2. Описание структур данных

**Класс *TQueue* (наследуется от TStack со спецификатором доступа public)**

Расположен в файле *Queue.h* проекта *QueueLib*. Является шаблонным, T – шаблонный тип данных, присваиваемый элементам очереди. Класс включает в себя:

Поля со спецификатором доступа protected:

* *int rear* – номер последнего зашедшего в очередь элемента
* *int count* – количество элементов в очереди
* *int size* – (наследуется от класса TStack) максимальное количество элементов в очереди;
* *int top* – (наследуется от класса TStack) номер элемента, добавленного в очередь раньше остальных;
* *T\* mas* – (наследуется от класса TStack) массив элементов очереди;

Поля со спецификатором доступа public:

* *TQueue<T> (int n=0)* – конструктор;
* *TQueue<T> (TQueue<T> &A)* – конструктор копирования;
* *~TQueue<T>()* – деструктор;
* *TQueue& operator=(const TQueue& A) –* оператор присваивания;
* *void Put (T A)* – функция добавления элемента в очередь;
* *T Get()* – функция удаления элемента из очереди;
* *bool IsFull()* – функция проверки очереди на полноту;
* *bool IsEmpty()* – функция проверки очереди на пустоту.

4.3. Описание алгоритмов

* Добавление элемента в очередь

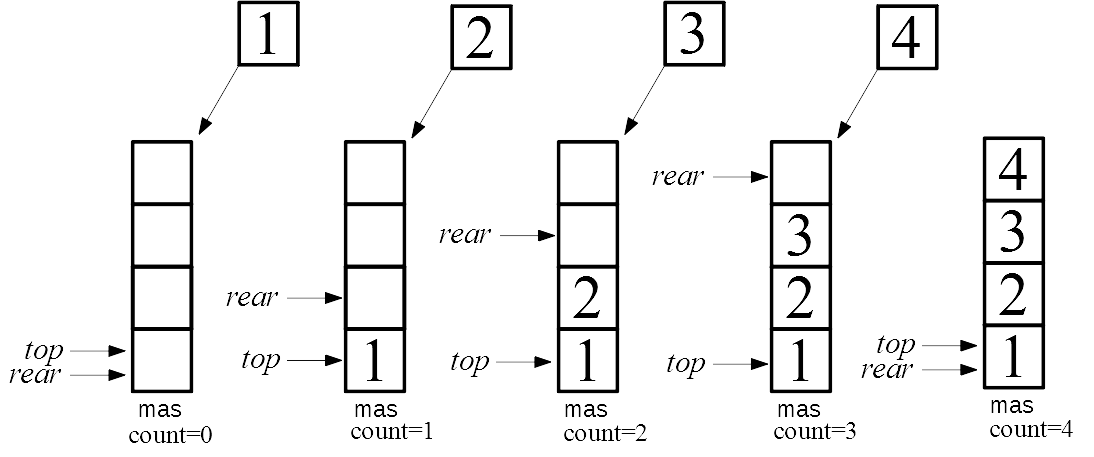
Делаем проверку очереди на полноту. Если места для нового элемента нет, то выбрасываем исключение. Если очередь ещё не полна, то в ячейку массива *mas* под номером *rear* записываем нужный элемент. После этого значение поля *rear* увеличиваем на 1. Если при этом значение *rear* равно размеру массива, то присваиваем ему номер первой ячейки массива (то есть *rear* зациклено). Пример заполнения очереди представлен на рисунке 1.

Рисунок 1. Добавление элементов в очередь

* Удаление элемента из очереди

Проверяем очередь на пустоту. Если очередь пуста, то выбрасываем исключение. Если элементы в очереди есть, то возвращаем значение ячейки массива mas под номером *top*. Затем значение поля *top* увеличиваем на 1. Если при этом значение *top* равно размеру массива, то присваиваем *top* номер первой ячейки массива (то есть поле *top*, как и *rear*, зациклено). Пример удаления элементов из очереди представлен на рисунке 2.

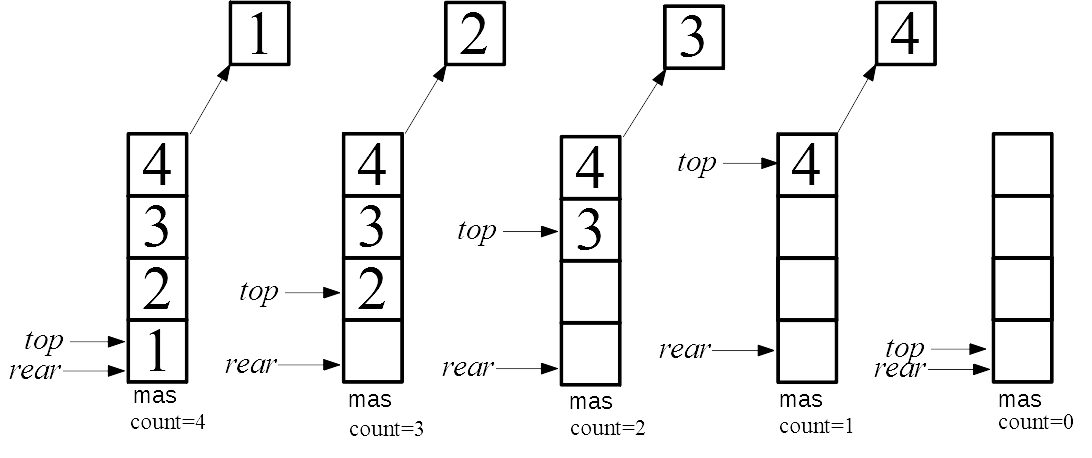


Рисунок 2. Удаление элементов из очереди

5. Заключение

В ходе работы были достигнуты следующие результаты:

* Создан класс *TStack*, предназначенный для работы со стеком;
* Обработаны следующие исключительные ситуации: создание очереди с некорректным размером, переполнение очереди, получение несуществующего элемента;
* Написана программа, демонстрирующая один из вариантов использования очереди;
* Проверены все методы класса *TQueue* с помощью Google Test Framework.

6. Список литературы

1. Ахо Альфред В, Хопкрофт Джон Э и Ульман Джеффри Д Структуры данных и алгоритмы [Книга]. - [б.м.] : Вильямс, 2003.
2. Лафоре Роберт Структуры данных и алгоритмы в Java [Книга]. - СПб : Питер, 2013. - 2 : стр. 704.
3. Павловская Т. А. C/C++ Программирование на языке высокого уровня [Книга]. - СПб : Питер, 2003.
4. Страуструп Бьерн Язык программирования C++ [Книга]. - [б.м.] : Бином, 2004.
5. Структура данных - Википедия [В Интернете] // Wikipedia. - Wikimedia Foundation, Inc, 11 декабрь 2018 г.. - 23 декабрь 2018 г.. - https://ru.wikipedia.org/wiki/Структура\_данных.