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

Голубева А. С.

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

аспирант каф. МОСТ ИИТММ

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

Нижний Новгород

2018.

Содержание

[1.Введение 3](#_Toc532519773)

[2.Постановка задачи 4](#_Toc532519774)

[3.Руководство пользователя 5](#_Toc532519775)

[4.Руководство программиста 6](#_Toc532519776)

[4.1Описание структуры программы 6](#_Toc532519777)

[4.2Описание структур данных 6](#_Toc532519778)

[4.3Описание алгоритмов 7](#_Toc532519779)

[5.Заключение 9](#_Toc532519780)

[6.Литература 10](#_Toc532519781)

# **Введение**

В рамках данной лабораторной работы требуется реализовать 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.

# **Постановка задачи**

Выполнение работы предполагает решение следующих задач:

1. Реализация класса TQueue, на основе класса TStack;
2. Реализация класса TException для обработки исключений;
3. Реализация тестов и обеспечение их работоспособности;
4. Пример использования и обеспечение его работоспособности.

**Очередь**

Для работы с очередью предлагается реализовать следующие операции:

* Метод Put – добавить элемент в очередь;
* Метод Get – удалить элемент из очереди;
* Метод IsEmpty – проверить очередь на пустоту;
* Метод IsFull – проверить очередь на полноту.

# **Руководство пользователя**

Пример использования очереди:

* 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.
* Демонстрируется работа на экран конструктора копирования.
* Из очереди удаляется элемент, затем она выводится на экран.

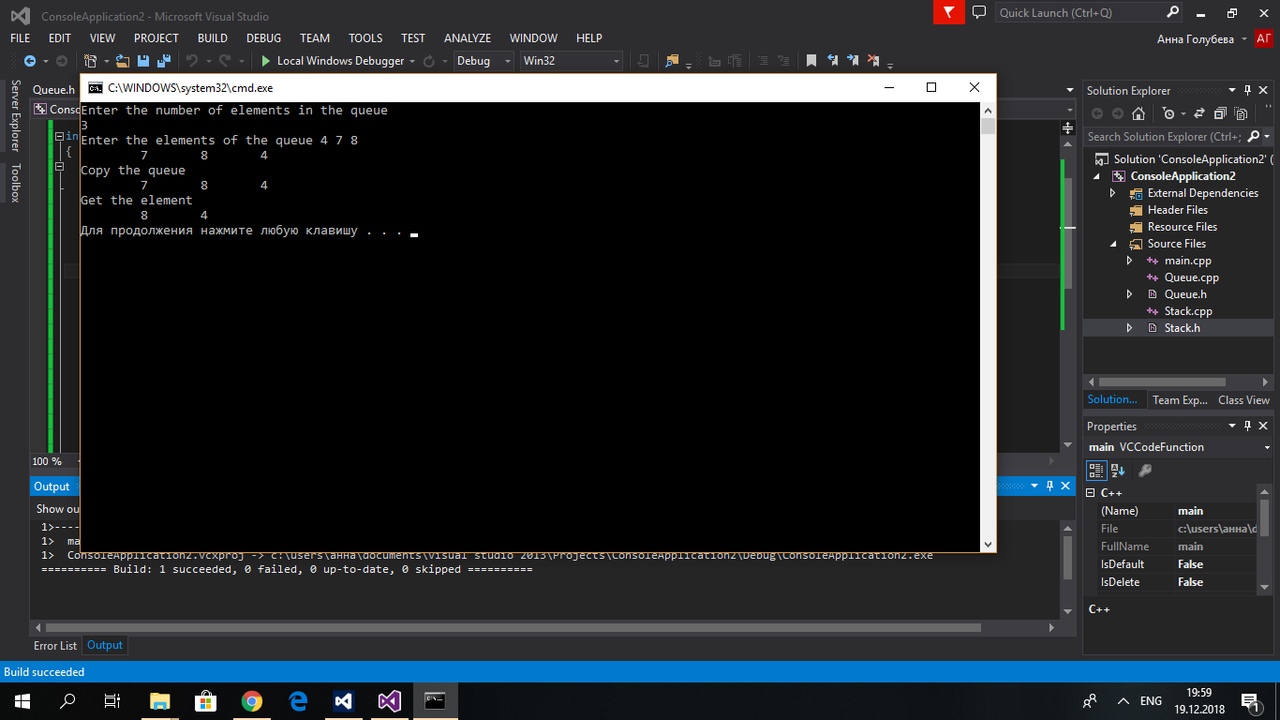


Рисунок 1.Пример работы очереди

# **4. Руководство программиста**

## **Описание структуры программы**

Программа состоит из следующих модулей:

* Модуль Queue - содержит файл main.cpp, в котором реализован пример использования очереди.
* Модуль QueueLib –содержит файл Queue.h, в котором описан интерфейс и реализация шаблонного класса *TQueue*.
* Модуль QueueTest – содержит тесты для класса TQueue.
* Модуль Exception – содержит файл Exception.h, в котором реализован класс исключений TException.

## **Описание структур данных**

**Класс TException**

**Private:**

Поле string message – переменная, которая хранит сообщение об ошибке в виде строки.

**Public:**

TException(string \_message)– конструктор;

void Print() – вывод сообщения об ошибке на экран.

**Класс TQueue**

Класс TQueue является шаблонным. Также класс TQueue наследуется от ранее разработанного класса TStack.

**Наследуются из класса TStack:**

int size – размер очереди.

int top – индекс ячейки в массиве, в которой расположен элемент на выход.

T\* mas – указатель на область памяти, выделенную под хранение элементов очереди.

**Protected:**

int start – индекс ячейки в массиве, в которую можно добавить элемент.

int count – количество элементов в очереди.

**Public:**

TQueue (int n = 0) – конструктор.

TQueue (TQueue <T> &q) – конструктор копирования.

virtual ~ TQueue() – деструктор.

void Put(T a) – добавить элемент в очередь.

T Get() – удалить элемент из очереди.

bool IsFull() – проверка очереди на полноту.

bool IsEmpty() – проверка очереди на пустоту.

void PrintQueue() – печать очереди на экран.

## **Описание алгоритмов**

Стоит заметить, что для хранения очереди используется не просто массив, а кольцевой буфер:

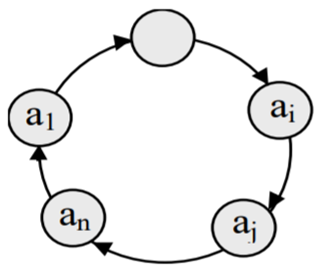


Рисунок 2. Кольцевой буфер

Рассмотрим работу методов Put и Get.

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++*.

**Удаление элемента из очереди.**

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--*.

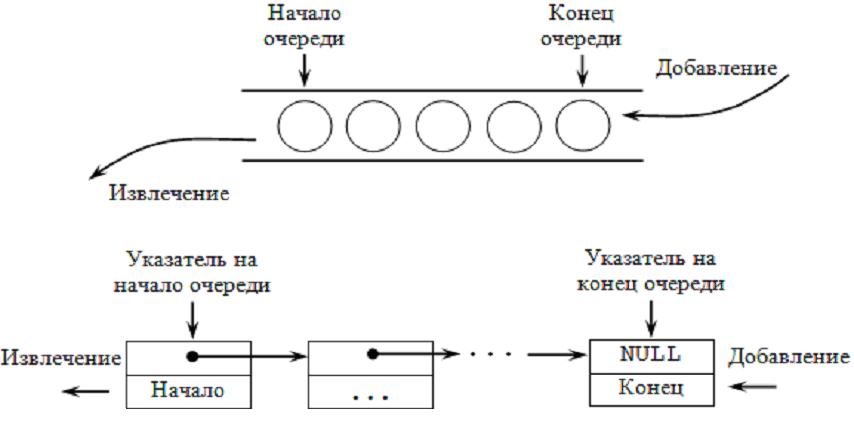


Рисунок 3. Добавление и извлечение элемента из очереди

# **Заключение**

В результате лабораторной работы разработана и реализована структура данных Очередь, также реализованы операции для работы с очередью. Успешно выполнены тесты, проверяющие работоспособность методов класса TQueue. Приведен пример, демонстрирующий работу с очередью.

# **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

(Дата обращения: 19.12.2018)

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

(Дата обращения: 19.12.2018)