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 ЛАБОРАТОРНОЙ РАБОТЕ**

**«Структура хранения данных: Очередь»**

**Выполнил:** студент группы 381706-2

Мышкин Андрей Александрович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Руководитель:**

ассистент кафедры МОСТ ИИТММ,

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018

**Содержание**

1. Введение………………………………………………………………………...3

2. Постановка задачи……………………………………………………...………5

3. Руководство пользователя…………………………………………...………...6

4. Руководство программиста…………………………………………...………..8

4.1. Описание структур данных…..……………………………………….8

4.2. Описание алгоритмов………………………………………………….8

4.3. Описание структуры программы….………………………..………...9

6. Заключение…………………………………………………………………….10

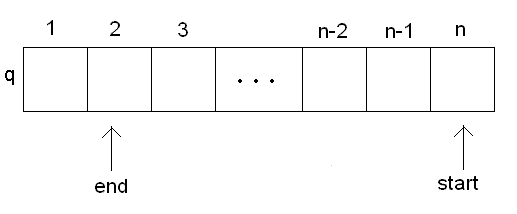
7. Литература……………….……………………………………………………11

**Введение**

Очередь — 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.

**Способы реализации очереди**

**Массив**



(1)

Первый способ представляет очередь в виде массива и двух целочисленных переменных start и end.

Обычно start указывает на голову очереди, end — на элемент, который заполнится, когда в очередь войдёт новый элемент. При добавлении элемента в очередь в q[end] записывается новый элемент очереди, а end уменьшается на единицу. Если значение end становится меньше 1, то мы как бы циклически обходим массив, и значение переменной становится равным n. Извлечение элемента из очереди производится аналогично: после извлечения элемента q[start] из очереди переменная start уменьшается на 1. С такими алгоритмами одна ячейка из n всегда будет незанятой (так как очередь с n элементами невозможно отличить от пустой), что компенсируется простотой алгоритмов.

Преимущества данного метода: возможна незначительная экономия памяти по сравнению со вторым способом; проще в разработке.

Недостатки: максимальное количество элементов в очереди ограничено размером массива. При его переполнении требуется перевыделение памяти и копирование всех элементов в новый массив.

**Связный список**

Второй способ основан на работе с динамической памятью. Очередь представляется в качестве линейного списка, в котором добавление/удаление элементов идет строго с соответствующих его концов.

Преимущества данного метода: размер очереди ограничен лишь объёмом памяти.

Недостатки: сложнее в разработке; требуется больше памяти; при работе с такой очередью память сильнее фрагментируется; работа с очередью несколько медленне.

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.

**Постановка задачи**

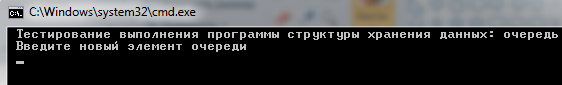
Основная задача – это реализовать структуру хранения данных такой, как класс очередь. Класс очередь будет наследником от класса стек, то есть класс стек базовый.

В классе очередь должны быть реализованы следующие операции:

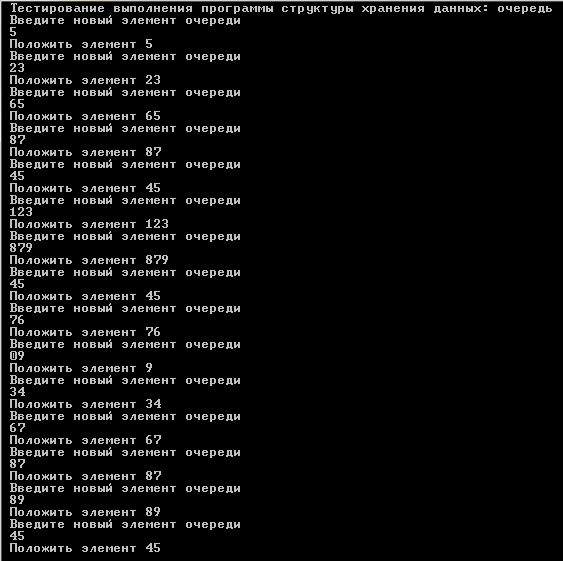
* Проверка пустоты очереди
* Проверка полноты очереди
* Добавление нового элемента
* Удаление элемента

**Руководство пользователя**

С помощью данной программы пользователь может проверить сам, как работает такая структура хранения данных, как очередь. При открытии программа сообщает о проведении тестирования на работу очереди. После чего пользователю предоставляется возможность заполнить данную очередь своими значениями. Для упрощения процесса тестирование проходит на целочисленных данных.

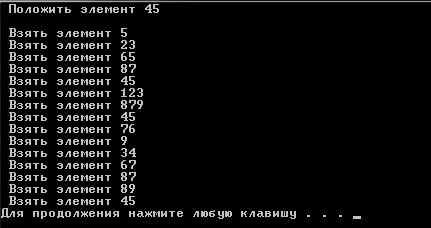


(2)



(3)

После заполнения очереди элементами программа автоматически производит удаление элементов из этой очереди, при котором происходит одновременное чтение головных элементов очереди по всем правилам.



(4)

После всего этого на консоли полностью представлена поэтапная работа очереди.

**Руководство программиста**

**Описание структур данных.**

Так как класс очередь - есть класс, унаследованный от класса стек, следовательно, и структура класса очередь унаследована от структуры класса стек. То есть все поля, и методы класса стек имеет класс очередь.

Поля класса:

Унаследованные

ValType \* mem – память для элементов в стеке

int t – индекс последнего занятого в mem

int memSize – размер всей памяти, выделенной под стек

Свои

int I1 – индекс первого элемента очереди

Методы класса:

Унаследованные

int IsEmpty() – метод класса, отвечающий за проверку пустоты

int GetSize() – метод класса, выдающий размер памяти, занимаемой стеком

int GetCount() - метод класса, выходом которого является количество элементов в стеке

int IsFull(); - метод класса, отвечающий за проверку и контроль перезаполнения

void Put(const int Val) – метод класса, отвечающий за добавление нового элемента в стек

Свои

int GetNextIndex(int index) – метод класса, который позволяет получить следующий индекс от index

ValType Get() – метод класса, который берет элемент из очереди при этом делая его удаление

**Описание алгоритмов.**

ValType TQueue<ValType>::Get()

ValType temp;

if (this->IsEmpty() == 1)

throw TExeption(DataEmpty);

else

temp = this->mem[I1];

I1 = GetNextIndex(I1);

this->t--;

**Описание структуры программы.**

Queue – модуль для тестирования выполнения программы структуры хранения данных: очередь, главный файл которого main.cpp

QueueLib – модуль реализации класса очередь В него входит заголовочный файл - TQueue.h

QueueTest – тест класса очередь, используя Google C++ Testing Framework, основной файл queue-text.cpp

**Заключение**

В ходе проведение данной лабораторной работы была создана и протестирована такая структура хранения данных как очередь. В классе очередь были реализованы такие операции как: проверка полноты и переполнения очереди, добавление нового элемента, удаление элемента очереди. Сам очередь была реализована при помощи наследования от класса стек.

Также были освоены инструменты разработки программного обеспечения, как система контроля версий Git и фрэймворк для разработки автоматических тестов Google Test.

**Литература**

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015
2. https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C\_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)