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
Митягина Дарья Сергеевна
_____ Подпись

Научный руководитель:
ассистент каф. МОСТ ИИТММ
_____ Лебедев И.Г

Нижний Новгород
2018.

Оглавление

1. Введение

В данной лабораторной работе будет реализована такая структура как очередь.

Рис. 1 – Принцип работы очереди

Удобно хранить данные в следующем виде:

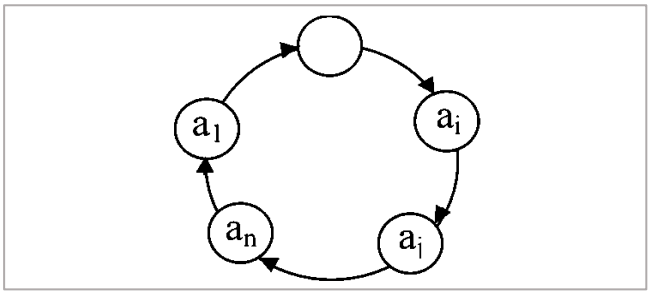

Рис. 2 – Кольцевой буфер

Структура хранения, получаемая из вектора расширением отношения следования парой $p(a_n,a_1)$, называется циклическим или кольцевым буфером.
Стандартный набор операций (часто у разных авторов он не идентичен), выполняемых над очередями, совпадает с тем, что используется при обработке стеков.

Очередью (англ. – queue) называется структура данных, из которой удаляется первым тот элемент, который был первым в очередь добавлен. То есть очередь в программировании соответствует «бытовому» понятию очереди. Очередь также называют структурой типа FIFO (first in, first out — первым пришел, первым ушел).

Цель данной работы: реализация очереди.

2. Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Реализация методов шаблонного класса TStack согласно заданному интерфейсу.
2. Реализация методов шаблонного класса TQueue согласно заданному интерфейсу.
3. Обеспечение работоспособности тестов и примера использования.
4. Реализация заготовок тестов, покрывающих все методы класса TQueue.

3. Руководство пользователя

После запуска программы пользователь увидит следующее:

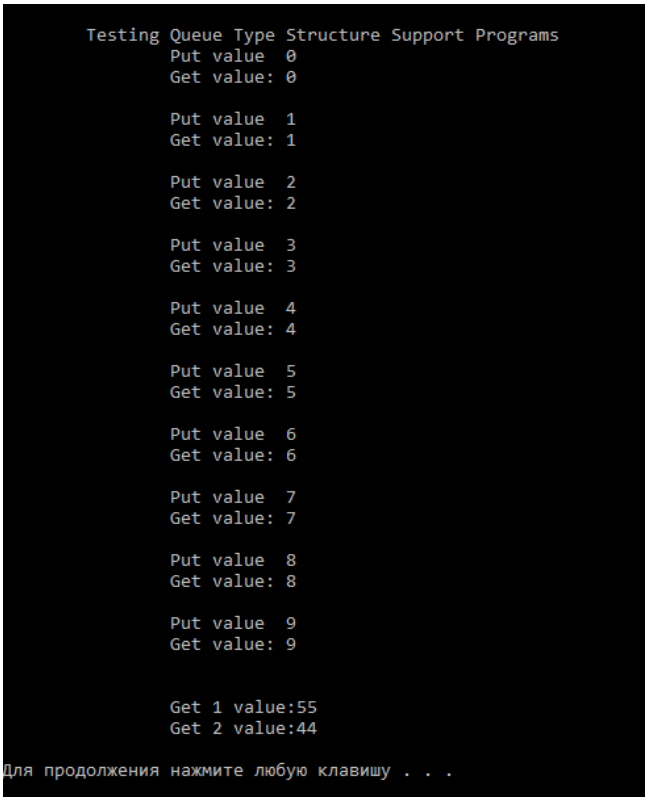

Рис. 3 – Пример использования

Выполняется проверка работы методов Put (добавить элемент) и Get (извлечь элемент).

4. Руководство программиста

4.1. Описание структуры программы

Для реализации алгоритмов будут использованы 2 класса:

1. Класс TQueue
2. Класс TStack

Лабораторная работа состоит из следующих модулей:

- StackLib

Этот модуль состоит из заголовочного файла Stack.h, отвечающего за определение интерфейса класса TStack и содержащего реализацию методов, и файла Stack.cpp.
Реализованы следующие функции: конструкторы, перегружены теоретико-множественные операторы (такие как сравнение, присваивание и т.д.), методы проверок на полноту/пустоту, извлечения/добавления элемента и т. д.

- QueueLib

Наследуется от StackLib. Этот модуль состоит из заголовочного файла Queue.h, отвечающего за определение интерфейса класса TQueue и содержащего реализацию методов, и файла Queue.cpp.
Реализованы следующие функции: конструкторы, методы проверок на полноту/пустоту, извлечения/добавления элемента и т. д.

- Queue

Пример использования программы, рассмотренный в пункте 3. Руководство пользователя.

- Тесты

В Queue_Test.cpp прописаны 10 тестов. Назначение тестов: проверить каждый метод из класса TQueue.

- TException

Этот модуль содержит класс исключений.

4.2. Описание структур данных

Класс TStack

Этот класс является шаблонным. В него включены следующие поля (protected - часть):

int Size; - размер стека

int Top; - элемент, расположенный на вершине стека

T* Mas; - элементы стека

Public – часть:

1) int GetSize() { return Size; }

Возвращает размер стека.

2) TStack(int n = 0);

Конструктор инициализации.

3) TStack(TStack<T> &S);

Конструктор копирования.

4) T Get();

Возвращает элемент, расположенный на вершине стека.

5) void PrintStack();

Выводит стек на экран.

6) void Put(T A);

Добавляет элемент на вершину стека, если стек не полон.

7) bool IsFull();

Проверка на полноту.

8) bool IsEmpty();

Проверка на пустоту. Стек пуст, если в нем нет ни одного элемента, т.е. когда количество элементов равно нулю.

9) int operator!=(const TStack<T>& stack) const;

Принимает ссылку на объект класса TStack, выполняет проверку на неравенство.

10) int operator==(const TStack<T>& stack) const;

Принимает ссылку на объект класса TStack, выполняет проверку на равенство.

11) TStack& operator=(const TStack<T>& stack);

Принимает ссылку на объект класса TStack, приравнивает исходный объект к полученному.

Класс TQueue

Этот класс является шаблонным. В него включены следующие поля (protected - часть):

int Start; - Начало очереди

int Count; - Кол-во элементов в очереди

Public – часть:

1) TQueue(int N = 0); - конструктор с параметром

2)TQueue(TQueue <T> &Q); - конструктор копирования

3) T Get(); - взять элемент

Берет элемент из начала очереди (как и было указано ранее в теоретической части).

4) void Put(T A); - положить в конец очереди

Вводит элемент в очередь.

5)bool IsFull(); - проверка на полноту

6)bool IsEmpty(); - проверка на пустоту

4.3. Описание алгоритмов

В данном разделе не будут рассматриваться тривиальные методы.

Класс TStack

1. Put(T A)

Принцип работы:
При добавлении элемента в стек необходимо переместить указатель вершины стека, записать элемент в соответствующую позицию динамического массива и увеличить количество элементов.

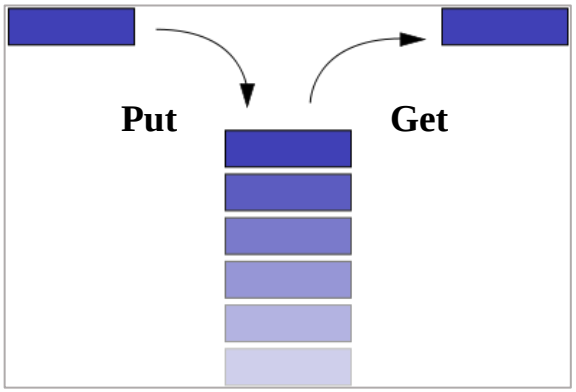

Рис. 4 – схема реализации двух методов (Put и Get)

2. T Get()

Подход аналогичный, из Рис.4 все должно быть ясно.

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

Рассмотрим наглядный пример:

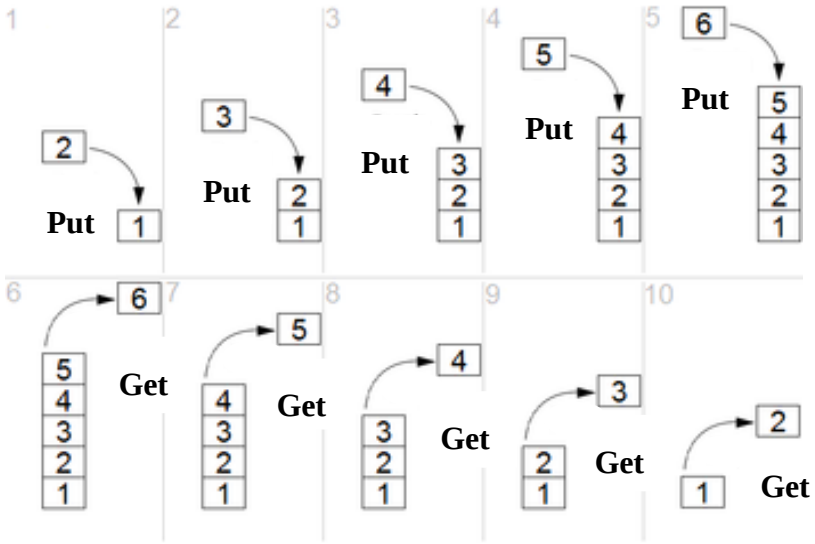

Рис. 5 - Пример

Остальные методы не нуждаются в отдельном рассмотрении.

Класс TQueue

1. void Put(T A)

Сначала необходимо проверить очередь на полноту. После этого переходим к следующему шагу: присваиваем элементу с индексом Start пришедшее значение A.

Таким образом, мы помещаем новый элемент в конец очереди. Далее увеличиваем счетчик элементов.

2. T Get()

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

3. bool IsEmpty()

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

4. bool IsFull()

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

5. Заключение

Было рассмотрены понятие очереди и реализация класса TQueue как наследника класса TStack. Также была освоена техника составления *тестов* путем самостоятельного составления нескольких из них на базе GT.

6. Литература

Книги:

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», 2015.

Интернет – ресурсы:

2. Лекция 31: Динамические структуры данных: очередь и стек, официальный сайт Интуит[https://www.intuit.ru/studies/courses/648/504/lecture/11457?page=2]

3. Статья Е. Вставской [https://prog-cpp.ru/data-queue/]