Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

Высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий математики механики

Стек

Отчет по лабораторной работе

Выполнил:

студент ИИТММ гр. 381706-2

Антипин А.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

ассистент каф. МОСТ, ИИТММ

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

Нижний Новгород

2018 г.

**Содержание**

[Введение 3](#_Toc820585)

[Постановка целей и задач 4](#_Toc820586)

[Руководство пользователя 5](#_Toc820587)

[Руководство программиста 6](#_Toc820588)

[Описание структуры программы 6](#_Toc820589)

[Описание структур данных 7](#_Toc820590)

[Описание алгоритмов 7](#_Toc820591)

[Заключение 9](#_Toc820592)

[Литература 10](#_Toc820593)

# Введение

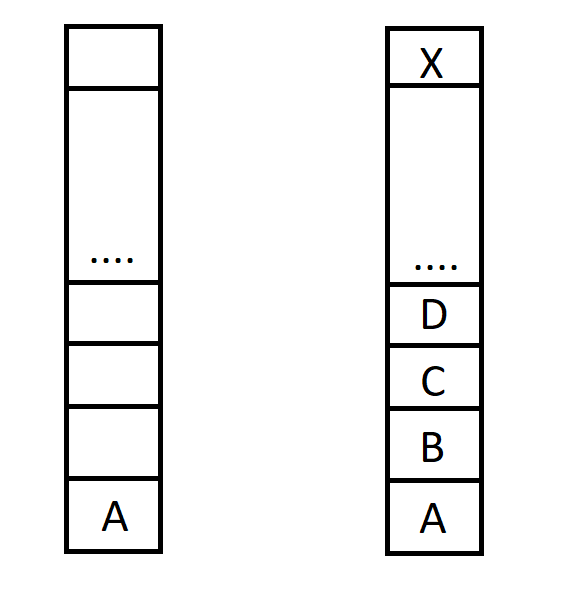
Одной из самых распространенных структур данных является стек. Стек – это структура, при которой элементы хранятся по принципу «первым вошел, последним вышел». Что это значит? Если у нас есть пустой стек, и в него кладется элемент, а за ним еще несколько, то чтобы добраться до нужного нам, придется извлечь все элементы, которые были положены после него (рис. 1).

рис. 1 (представление элементов в стеке. 1-вый столбец – положили один элемент в стек, 2-ой столбец – заполненный стек)

Практическое применение стек нашел в алгоритмах поиска в «глубину» и в «ширину». Таким образом стек – это инструмент решения алгоритмических (и не только) задач.

# Постановка целей и задач

Основной целью лабораторной работы является создание структуры данных «Стек» и реализация таких методов, как доступ к элементам стека:

* Положить элемент в стек;
* Вернуть элемент с удалением;
* Вернуть элемент без удаления;

Для реализации алгоритмов будет использоваться шаблонный класс Stack.

Для проверки правильности работы этого класса будут написаны тесты с использованием фреймворка Google Test, а также тестовый образец программы, которая использует данный класс.

# Руководство пользователя

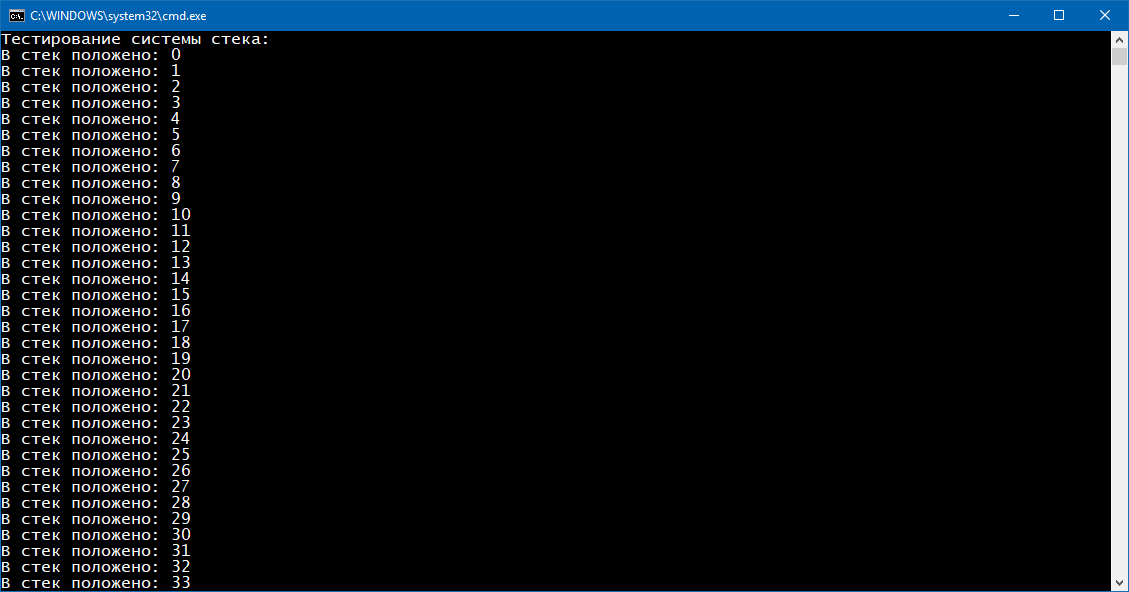
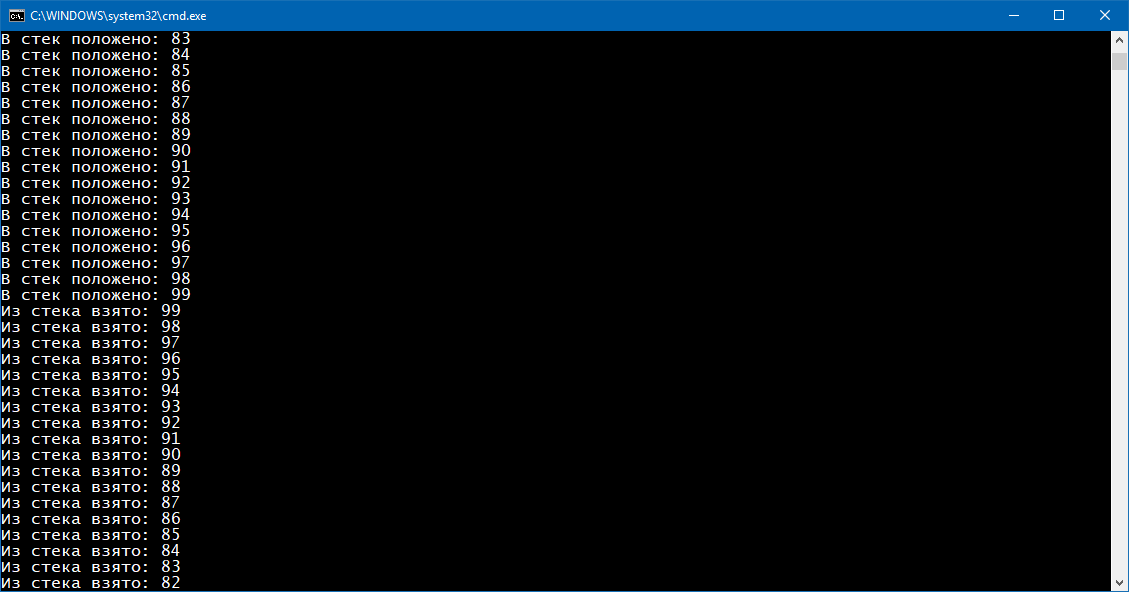
После запуска программы пользователя встречает консольное окно (рис. 2):

рис. 3 (вывод работы стека для пользователя, из стека забираются элементы с 99 до 0)

рис. 2 (вывод работы стека для пользователя, в стек кладутся числа от 0 до 99)

В стек кладутся числа от 0 до 99, а затем из стека забираются эти элементы, причем сначала забирается 99, 98, и т.д., а уже потом 0 (рис. 3).

# Руководство программиста

## Описание структуры программы

Объект типа стек будет выделять необходимое количество памяти под объекты определенного типа, которое будет задаваться при его создании. Т.е. максимальное число объектов в стеке будет ограниченно.

Для реализации алгоритмов будет создан класс Stack, который будет включать в себя все алгоритмы для работы с ним.

А также проект использующий фреймворк Google Test, для проверки правильности работы класса стек, а также тесовый проект, который будет показываться пользователю.

Древо классов

**Класс gtest**

**test\_main.cpp**

**Stack\_test.cpp**

**Проект для пользователя**

**main.cpp**

**stack.h**

**Класс Stack**

**Класс Stack:**

В этом классе реализованы методы работы со стеком, такие как: положить элемент, забрать элемент, вернуть текущее количество элементов в стеке, проверить является ли стек пустым или полным в данный момент.

**Класс gtest:**

Класс gtest реализует тестирование класса Stack, по средствам фреймворка Google Test. Тесты пишутся для каждого метода классов, каждого ветвления этих методов и для всех возможных исключений этих методов.

**Проект Stack:**

В данном проекте реализован примет использования стека для заполнения элементами и их последующего забора.

## Описание структур данных

**Класс Stack:**

template <class T> - шаблон класса T

int Len; - максимальная длина стека:

T \*Mem; - указатель на первый элемент стека типа Т;

int Index; - индекс текущего элемента в стеке;

Exceptions\_from\_stack\_queue\_multystack except; - переменная для вызова исключений.

**Описание методов:**

|  |  |
| --- | --- |
| Метод: | Описание: |
| Stack<T>::Stack() | Конструктор по умолчанию стека. |
| Stack<T>::Stack(const int \_Len) | Конструктор с параметром, который принимает длину стека. |
| Stack<T>::Stack(const Stack<T>& A) | Конструктор копирования стека, который принимает константную ссылку на другой стек. |
| Stack<T>::~Stack() | Деструктор для класса стек |
| int Stack<T>::GetIndex() | Метод, который возвращает текущий индекс стека. |
| int Stack<T>::GetLen() | Метод, который возвращает длину стека. |
| void Stack<T>::Put(const T& A) | Метод, который позволяет положить элемент в стек. |
| T Stack<T>::Get() | Возвращение элемента из стека с его удалением. |
| T Stack<T>::GetWithoutDelete() | Возвращение элемента из стека без удаления. |
| bool Stack<T>::IsFull() | Проверка стека на полноту. |
| bool Stack<T>::IsEmpty() | Проверка стека на пустопу. |

## Описание алгоритмов

**Подробное описание некоторых методов**

Добавление элемента в стек:

* Проверка на полноту, если стек полон, то выдается ошибка о полноте стека;
* Если в стеке есть свободное место, то элемент кладется в стек, а индекс увеличивается на 1.

Возврат элемента из стека:

* Проверка на пустоту, если стек пустой, то выдается ошибка о пустоте стека;
* Если в стеке есть элементы, то элемент возвращается, а индекс уменьшается на 1.

# Заключение

Подводя итоги можно сказать, что стек имеет огромное практическое применение, а его реализация не составляет труда. Также нельзя не отметить скорость работы стека, т.к. в алгоритмах его реализации не присутствует циклов, а значит сложность O(1). Все эти неоспоримые преимущества показывает доказывают эффективность применения структуры данных «стек» в практических задачах.

# Литература

* Учебные материалы к учебному курсу «Методы программирования» - Гергель В.П.
* http://www.cyberforum.ru/cpp-beginners/thread587712.html