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 хранения данных: Стек на массиве»**

**Выполнил:** студент группы 381706-1

Мазур Даниил Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533083472)

[2. Постановка задачи 4](#_Toc533083473)

[3. Руководство пользователя 5](#_Toc533083474)

[4. Руководство программиста 6](#_Toc533083475)

[4.1. Описание структуры программы 6](#_Toc533083476)

[4.2. Описание структур данных 6](#_Toc533083477)

[4.3. Описание алгоритмов 7](#_Toc533083478)

[5. Заключение 8](#_Toc533083479)

[6. Литература 9](#_Toc533083480)

# Введение

**Стек на массиве** — структура данных, представляющая собой упорядоченный набор элементов, фиксированного количества, организованных по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»). Это значит, что мы будем иметь доступ только к последнему добавленному элементу.

В стеке мы не можем получить доступ к произвольному элементу. Мы можем только добавлять или удалять элементы с помощью специальных методов. Кроме того, у стека нет итератора. Для того, чтобы понимать, почему на стек накладываются такие ограничения, давайте посмотрим на то, как он работает и как используется.

Наиболее часто встречающаяся аналогия для объяснения стека — стопка тарелок. Вне зависимости от того, сколько тарелок в стопке, мы всегда можем снять верхнюю. Чистые тарелки точно так же кладутся на верх стопки, и мы всегда будем первой брать ту тарелку, которая была положена последней.

**Цель данной лабораторной работы** – разработка структуры хранения стека на массиве.

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Разработка и реализация класса стека – TStack.
2. Создание класса для обработки исключений – MyException, которые могут возникнуть при выполнении различных операций.
3. Разработка программы, демонстрирующей работу классa TStack.
4. Реализация набор автоматических тестов с использованием Google C++ Testing Framework.

# Руководство пользователя

Рассмотрим пример использования класса TStack.

При запуске программы c пользователя спрашивается размер стеков для дальнейшей работы с ними. Затем создаются два идентичных стека Stack и CopyStack размером , полностью заполняются натуральными числами от 0 до , и выводятся на консоль.

Далее осуществляется проверка, что Stack и CopyStack действительно сравнимы. И, если программа работает верно, на экран выводится сообщение что они сравнимы.

Затем из Stack извлекается элемент. Этот элемент выводится на экран и выводится Stack полученный после извлечения.

Поскольку теперь CopyStack и Stack несравнимы, то при корректной работе программы, на экран будет выведено сообщение что они различны. На этом работа программы прекращается.

# Руководство программиста

## Описание структуры программы

Программа состоит из следующих модулей:

* Модуль Stack. Содержит пример использования стека. Реализация в файле *main\_Stack.cpp.*
* Модуль StackLib – статическая библиотека. Содержит файл Stack.h, в котором описан интерфейс и реализация шаблонного класса *TStack* (2 конструктора, деструктор и 9 методов).
* Модуль StackTest. Содержит 18 тестов, описанных в файле *StackTest.cpp* и разработанных с помощью использования Google C++ Testing Framework.
* Модуль ExceptionLib – библиотека, содержащая класс исключений.

## Описание структур данных

#### 4.2.1 Класс TStack

Класс *TStack* является шаблонным классом. В нем определены три поля со спецификатором доступа protected:

* *int size* – максимальный размер стека.
* *int top* – позиция вершины стека.
* *T\* mas* – указатель на область памяти для хранение стека.

Далее в публичной зоне (public):

1. Прописаны 2 конструктора и деструктор:

* *TStack(int n = 0)* – конструктор с одним параметром.
* *TStack(TStack<T> &S)* – конструктор копирования.
* *virtual ~TStack()* – деструктор.

1. Прописаны методы и перегружены операторы для работы со стеком:

* *void Put(T A)* – добавить новый элемент A в стек.
* *T Get()* – изъять с удалением элемент находящийся на вершине стека.
* *void Print()* – вывод элементов стека на экран.
* *bool IsFull()* – проверка стека на полноту.
* *bool IsEmpty()* – проверка стека на пустоту.
* *int operator!=(const TStack<T>& stack) const* – проверка стеков на различие
* *int operator==(const TStack<T>& stack) const* - проверка стеков на идентичность.
* *TStack& operator=(const TStack<T>& stack)* – присваивание стека.

## Описание алгоритмов

**Добавление элемента в стек.**

При добавлении элемента в стек, размещаем его в первую свободную ячейку массива, выделенного под хранение данных (если она имеется). На эту ячейку указывает *top* – вершина стека. Затем значение *top* увеличиваем на 1.

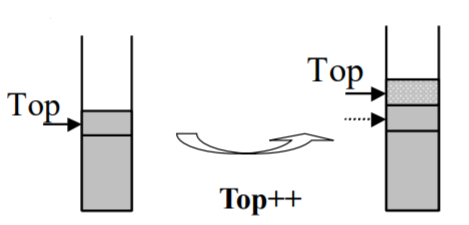


Рисунок 1. Добавление элемента в стек.

**Удаление элемента из стека.**

При удалении элемента из стека уменьшаем значение *top* на 1 – спускаемся к последнему существующему элементу и возвращаем его значение (если стек не пуст). Само значение не затирается, т.к. оно будет перезаписано при следующем добавлении другого элемента.

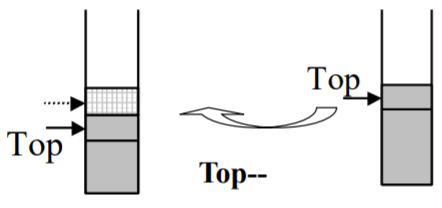


Рисунок 2. Удаление элемента из стека

# Заключение

В ходе выполнения лабораторной был произведен анализ задачи - установлено понятие стека на массиве. Была разработана библиотека StackLib, реализующая шаблонный класс стека TStack. В ней реализованы методы работы со стеком описанные в разделе «Структуры данных».

Программная реализация стека на массиве была продемонстрирована на примере, описывающем основные методы класса TStack.

Разработаны и доведены до успешного выполнения тесты, разработанные для данного программного проекта с использованием Google C++ Testing Framework.

# Литература

* Книги

1. A.O. Грудзинский. Методы программирования, Издательство Нижегородского госуниверситета, 2006.
2. Васильев А.Н. Самоучитель С++ с примерами и задачами. -СПб.: Наука и Техника, 2016. -480с.

* Ссылки в Internet

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2»: [<http://www.itmm.unn.ru/files/2018/10/Primer-1.3.-Struktury-hraneniya-steka.pdf>], 2015.
2. Википедия: свободная электронная энциклопедия: на русском языке: https://ru.wikipedia.org/wiki/Стек