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 списке»**

**Выполнил:** студент группы 381706-1

Полетуева Анастасия Николаевна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533186462)

[2. Постановка задачи 4](#_Toc533186463)

[3. Руководство пользователя 5](#_Toc533186464)

[4. Руководство программиста 6](#_Toc533186465)

[4.1. Описание структуры программы 6](#_Toc533186466)

[4.2. Описание структур данных 6](#_Toc533186467)

[4.3. Описание алгоритмов 7](#_Toc533186468)

[5. Заключение 8](#_Toc533186469)

# Введение

Основная цель данной работы – разработать структуру хранения стека на списке.

**Стек на списке** — структура данных, представляющая собой упорядоченный набор элементов, фиксированного количества, связанных между собой последовательно посредством указателей, организованных по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»). По сути, это значит, что мы будем иметь доступ только к последнему добавленному элементу. Каждый элемент стека на списках имеет указатель на следующий элемент, в то время как последний указывает на NULL. **Голова стека на списке** – первый элемент, на который нет указателя, является и его вершиной. Все новые элементы добавляются в голову, как бы над предыдущей головой, сами становясь головой. При этом им присваивается указатель на старую голову. При изъятии элемента из стека на списках извлекается текущая голова, а новой головой становится элемент, на которую указывает изъятый элемент.

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Разработка и реализация класса стека на списке - TStackList.
2. Пример программы, демонстрирующая работу класса TStackList.
3. Написание набора автоматических тестов с использованием Google C++ Testing Framework и проверка работоспособности методов классов.

# Руководство пользователя

При запуске программы пользователя просят ввести размер стека на списке. После чего выводит сформированный стек, заполненный значениями от 0 до введенной длины – 1. А также информирует пользователя, что в настоящий момент стек полон. Затем, из стека на списке извлекается первый элемент для проверки правильности вывода. Стек на списке без первого значения также выводится на экран, после чего пользователя информируют о текущем количестве элементов в стеке на списке и о максимальном. Оставшиеся в стеке на списке элементы последовательно выводятся на экран, вместе с сообщением о том, что стек теперь пуст.

# Руководство программиста

## Описание структуры программы

Программа состоит из следующих модулей:

* Модуль StackListLib. Статическая библиотека. Включает в себя заголовочный файл StackList.h, в котором описаны методы с реализацией шаблонного класса стека на списке TStackList.
* Модуль StackListTest. Набор тестов для класса TStackList. Включает в себя файл *StackListTest.cpp.* Разработаны они с помощью использования Google C++ Testing Framework.
* Модуль StackList. Пример использования стека. Включает в себя файл с реализацией *main.cpp.*

## Описание структур данных

**Класс TStackList наследуется от класса TList***:*

int size; - максимальная длина стека

TStackList<T>(int \_length = 10);

TStackList<T>( (TStackList<T>( &A);

~TStackList();

T Get(); - взять верхний элемент

int GetMaxSize(); - получить длину стека

int GetSize(); - получить текущую длину стека

void Put(T A); - положить элемент в начало стека на списке

bool IsFull(); - проверка стека на списке на полноту

bool IsEmpty(); - проверка стека на списке на пустоту

void Print(); - вывод стека на списке на экран

## Описание алгоритмов

**Добавление и изъятие элемента в стеке на списках.**

Добавляя новый элемент к стеку, мы размещаем его над вершинной, а извлекая, соответственно, достаем текущую вершину стека. При реализации стека на списке за вершину удобно считать голову списка. Добавленные элементы будут размещаться перед головой , сами при этом становясь новой головой. Новый элемент при этом будет указывать на старую голову. Извлекаться, при таком подходе, будет текущая голова, а новой станет элемент, на который указывал изъятый элемент.

Это на порядок выгоднее, чем добавление элементов в конец списка/изъятие из конца списка. Ведь для того, чтобы добраться до конца списка придется пройти через каждый элемент списка, что негативно сказалось бы на времени работы этих операций.

# Заключение

В ходе выполнения был реализован шаблонный класс стека на списке TStackList. Все его методы описаны в специально разработанной библиотеке StakListLib. Для проверки работоспособности всех методов, были написаны автоматические тесты, реализованные с использованием Google C++ Testing Framework, и пример реализации.

1. **Литература**
2. Гергель В.П. Методические материалы по курсу «Методы программирования 2»: [http://www.itmm.unn.ru/files/2018/11/Primer-1.6.-Struktura-hraneniya-neskolkih-stekov-s-ispolzovaniem-spiskov.pdf], 2015
3. Википедия: свободная электронная энциклопедия: на русском языке: https://ru.wikipedia.org/wiki/Стек
4. Университет ИТМО: Викиконспекты:

http://neerc.ifmo.ru/wiki/index.php?title=Стек