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 списках»**

**Выполнил:** студент группы 381706-1

Лембриков Степан Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc536523907)

[2. Постановка задачи 4](#_Toc536523908)

[3. Руководство пользователя 5](#_Toc536523909)

[4. Руководство программиста 6](#_Toc536523910)

[4.1. Описание структуры программы 6](#_Toc536523911)

[4.2. Описание структур данных 6](#_Toc536523912)

[4.3. Описание алгоритмов 7](#_Toc536523913)

[5. Заключение 8](#_Toc536523914)

[6. Литература 9](#_Toc536523915)

# Введение

**Стек на списках** — структура данных, представляющая собой упорядоченный набор элементов одного типа, фиксированного количества, связанных между собой последовательно посредством указателей, организованных по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»). Каждый элемент стека на списках имеет указатель на следующий элемент. Последний элемент стека на списках указывает на [NULL](https://ru.wikipedia.org/wiki/NULL_(%D0%A1%D0%B8)). Элемент, на который нет указателя, является первым (головным) элементом списка. В стек на списках все новые элементы добавляются в начало.  Вершиной стека на списках является первый элемент в списке.

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

# Постановка задачи

* Разработка и реализация класса стек на списках – TStackList.
* Создание класса MyException для обработки исключений, которые могут возникнуть при выполнении различных операций.
* Программа, демонстрирующая работу класса TStackList.
* Написание Google тестов, позволяющих проверить работу программы в автоматическом режиме.

# Руководство пользователя

Рассмотрим пример использования класса TStackList.

При запуске программы пользователю нужно будет ввести максимальный размер стека . После этого создается стек размером n и автоматически заполняется числами от 1 до n. Добавление каждого числа в стек сопровождается выводом сообщением на экране. После заполнения полученный стек выводится на экран. Затем происходит изъятие элементов из стека до последнего. Изъятие каждого элемента также сопровождается соответствующим выводом сообщения на экран. После того, как будут извлечены все элементы, на экран поступит сообщение о том, что стек опустел. После этого программа завершает свою работу.

# Руководство программиста

## Описание структуры программы

**Проект “ StackList ” включает в себя:**

1. main\_StackList.cpp (в нём находится main)

**Проект “ StackListLib ” включает в себя:**

1. StackList.h (описание и реализация класса “ TStackList ”)
2. StackList.cpp

**Проект “** **QueueTest” включает в себя:**

1. StackListTest.cpp (тесты, разработанные с помощью Google C++ Testing Framework)

**Проект “** **ExceptionLib ” включает в себя:**

1. библиотека, содержащая класс исключений.

## Описание структур данных

#### Класс TStackList

Класс *TStackList* является шаблонным классом. Наследуется от класса *TList* со спецификатором public. В классе *TStackList* определено одно поле, со спецификатором доступа protected, int size – максимальный размер стека.

Далее в публичной зоне (public):

1. Прописаны 2 конструктора и деструктор:

* *TStackList(int \_size = 10)* – конструктор с одним параметром.
* *TStackList(TStackList<T> &A)* – конструктор копирования.
* *~TStackList()* – деструктор.

1. Прописаны методы для работы со стеком на списках:

* *void Put(T A)* – добавить новый элемент A в начало списка.
* *T Get()* – изъять с удалением элемент находящийся в начале списка.
* *void Print()* – вывод элементов стека на списках на экран.
* *bool IsFull()* – проверка стека на списках на полноту.
* *bool IsEmpty()* – проверка стека на списках на пустоту.
* *int GetSize()* – получить текущее количество элементов в стеке на списках.
* *int GetMaxSize()* – получить размер стека.

## Описание алгоритмов

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

При добавлении элемента в стек, он размещается на вершине стека. При изъятии элемента из стека берется элемент, находящийся на вершине стека. Поскольку в данной работе стеки хранится на списках, то целесообразно вершиной стека считать первый элемент списка. То есть добавление и изъятие элементов происходит только из начала списка. В этом случае сложность этих операций составит О(1). Иначе, для добавления или изъятия элементов из конца списка, пришлось бы обходить все список поэлементно, и сложность операций составила O(n).

# Заключение

В результате проведённой работы я понял, что такое очередь, разработал библиотеку StackListLib и реализовал класс стека на списках TQueue.

Также я создал программу – реализацию, демонстрирующую работу основных методов класса TStackList.

Разработал тесты, разработанные для данного программного проекта с использованием Google C++ Testing Framework, и добился их выполнения.

# Литература

* Книги

A.O. Грудзинский. Методы программирования, Издательство Нижегородского госуниверситета, 2006.

* Ссылки в Internet

Гергель В.П. Методические материалы по курсу «Методы программирования 2»: [<http://www.itmm.unn.ru/files/2018/10/Primer-1.3.-Struktury-hraneniya-stack-na-spiscach.pdf>], 2015.

METANIT.COM: стек на списках: [<https://metanit.com/sharp/algoritm/2.3.php>]

Википедия: свободная электронная энциклопедия: на русском языке: [https://ru.wikipedia.org/wiki/Стек\_Списках]