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**

**«Стек на списках»**

**Выполнил:** студент группы 381706-1

Корнев Никита Алексеевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533799739)

[2. Постановка задачи 4](#_Toc533799740)

[3. Руководство пользователя 5](#_Toc533799741)

[4. Руководство программиста 6](#_Toc533799742)

[4.1. Описание структуры программы 6](#_Toc533799743)

[4.2. Описание структур данных 7](#_Toc533799744)

[4.3 Описание алгоритмов 8](#_Toc533799745)

[5. Заключение 9](#_Toc533799746)

[6. Литература 10](#_Toc533799747)

# Введение

Структура данных (англ. data structure) — программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных данных в вычислительной технике. Для добавления, поиска, изменения и удаления данных структура данных предоставляет некоторый набор функций, составляющих её интерфейс. Одной из важных структур является список.

Стек (от англ. stack — стопка) — структура данных, представляющая из себя упорядоченный набор элементов, в которой добавление новых элементов и удаление существующих производится с одного конца, называемого вершиной стека. Притом первым из стека удаляется элемент, который был помещен туда последним, то есть в стеке реализуется стратегия «последним вошел — первым вышел» (last-in, first-out — LIFO). Примером стека в реальной жизни может являться стопка тарелок : когда мы хотим вытащить тарелку, мы должны снять все тарелки выше.

Стек можно реализовать на списке. Для этого необходимо создать список и операции работы стека на созданном списке. Стек будем "держать" за голову. Добавляться новые элементы посредством операции Put будут перед головой, сами при этом становясь новой головой, а элементом для изъятия из стека с помощью Get будет текущая голова. После вызова функции Put текущая голова уже станет старой и будет являться следующим элементом за добавленным, то есть ссылка на следующий элемент нового элемента будет указывать на старую голову. После вызова функции Put будет получена и возвращена информация, хранящаяся в текущей голове. Сама голова будет изъята из стека, а новой головой станет элемент, который следовал за изъятой головой.

В данной лабораторной работе нам предстоит написать собственную реализацию данной структуры.

# Постановка задачи

1. Написать класс, реализующий стек на списках.
2. Написать тесты на основе Google Tests для проверки работы класса.

# Руководство пользователя

Чтобы использовать данный класс в своем проекте, необходимо подключить библиотеку «StackList.h». Библиотека позволяет:

* Создавать объекты типа *список*:

StackList st2(10); **конструктор с параметром (1 по-умолчанию)**

StackList st3 (st2); **конструктор копирования**

* Добавлять элементы:

st1.Put(T A);

* Брать элементы:

A = st1.Get();

Проверять на *пустоту*:

A = st1.IsEmpty();

* Проверятьна *полноту*:

A = st1.IsFull();

# Руководство программиста

## Описание структуры программы

Проект состоит из следующих модулей:

1. StackList

Модуль cодержит файл код «main.cpp», в котором продемонстрирован пример использования структуры.

1. StackListLib

Модуль содержит файл заголовок «StackList.h», описывающий структуру «стек на списках, а также файл кода «main.cpp». Так как класс шаблонный, main содержит лишь подключение.

1. StackListTest

Модуль содержит файлы кода «test\_main.cpp», «test\_StackList.cpp». В последнем реализованы тесты для проверки корректности работы методов данного класса.

## Описание структур данных

#### *Структура StackList*

Поля со спецификатором доступа «protected»:

* **int size;** // размер

Поля со спецификатором доступа «public»:

* **TStackList<T>(int \_size = 1);** // конструктор с параметром
* **TStackList<T>(TStackList<T> &A);** // конструктор копирования
* **~TStackList();** // деструктор
* **void Put(T A);** // положить элемент в стек
* **T Get();** // взять элемент из стека
* **int GetMaxSize();** // получить размер стека
* **int GetSize();** // получить кол-во элементов стека на данный момент
* **bool IsEmpty();** // проверка на пустоту
* **bool IsFull();** // проверка на полноту

## Описание алгоритмов

Описание некоторых алгоритмов:

* **TStackList<T>(int \_size = 1);**

Создает стек на списке размера \_size, по-умолчанию равного 1.

* **TStackList<T>(TStackList<T> &A); // конструктор копирования**

Создает стек на списке от стека на списке A.

* **~TStackList(); // деструктор**
* **void Put(T A); // положить элемент в стек**

Использует метод класса списка PutBegin(), чтобы положить элемент в стек.

* **T Get(); // взять элемент из стека**

Использует метод класса списка GetBegin(), чтобы положить элемент в стек.

* **int GetMaxSize(); // получить размер стека**

Возвращает значение count.

* **int GetSize(); // получить кол-во элементов стека на данный момент**

Возвращает значение size.

* **bool IsEmpty(); // проверка на пустоту**

Eсли count равен 0 – возвращает единицу, в противном случае 0.

* **bool IsFull(); // проверка на полноту**

Eсли count равен size – возвращает единицу, в противном случае 0.

# Заключение

В данном лабораторной работе мне удалось:

* Успешно реализовать класс для стека на списках
* Продемонстрировать пример использования данного класса
* Написать тесты на основе Google Tests для проверки корректной работы данного класса

# Литература

Интернет-источники:

1. <https://ru.wikipedia.org/wiki/Стек>
2. [https://ru.wikipedia.org/wiki/Список\_(информатика)](https://ru.wikipedia.org/wiki/Очередь_(программирование))
3. <https://habr.com/ru/post/232009/>

Книги:

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.
2. A.O. Грудзинский. Методы программирования, Издательство Нижегородского госуниверситета, 2006.