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 линейном списке»**

**Выполнил:** студент группы 381706-1

Денисов Владислав Львович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc532930116)

[2. Постановка задачи 4](#_Toc532930117)

[3. Руководство пользователя 5](#_Toc532930118)

[4. Руководство программиста 6](#_Toc532930119)

[4.1 Описание структуры программы 6](#_Toc532930120)

[4.2 Описание структур данных 6](#_Toc532930121)

[4.3 Описание алгоритмов 7](#_Toc532930122)

[5. Заключение 9](#_Toc532930123)

[6. Литература 10](#_Toc532930124)

# Введение

Лабораторная работа направлена на практическое освоение динамической структуры данных стек на линейном списке.

Стек (англ. stack — стопка) — абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»).

Чаще всего принцип работы стека сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно снять верхнюю. В цифровом вычислительном комплексе стек называется магазином — по аналогии с магазином в огнестрельном оружии (стрельба начнётся с патрона, заряженного последним).

В 1946 Алан Тьюринг ввёл понятие стека. А в 1957 году немцы Клаус Самельсон и Фридрих Л. Бауэр запатентовали идею Тьюринга.

# Постановка задачи

В рамках лабораторной работы ставится задача эффективной реализации структуры данных – стек на списках.

Хранить данную структуру в этой лабораторной работе будем с помощью списка на указателях. Удобство при таком подходе заключается в том, что размер стека может быть не ограничен.

Заметим, что в нашем случае вершиной стека будет последний добавленный в начало списка элемент. Если указатель на начало списка равен нулю, то стек пуст.



Рисунок 1 Организация стека на односвязном списке.

Для работы со стеком будут реализованы операции:

* добавления элемента в стек,
* извлечения элемента из стека (с удалением),
* печать стека на консоль.

Программное решение будет выглядеть следующим образом:

1. Класс стека – TStackList, построенный на основе существующего класса TList.
2. Класс для обработки исключений – TException, которые могут возникнуть при выполнении различных операций.
3. Программа, демонстрирующая работу класса TStackList.
4. Набор автоматических тестов с использованием Google C++ Testing Framework.

# Руководство пользователя

В качестве примера использования стека предлагается следующее.

После запуска тестовой программы пользователь вводит максимальный размер стека. Создается стек заданного размера для хранения значений целого типа.

Затем пользователю предлагается заполнить все ячейки стека. После этого происходит печать стека на консоль. Происходит удаление одного элемента. Снова выполняется печать того, что осталось в стеке.

Наконец, создается точно такой же стек с помощью конструктора копирования и отображаются адреса старого и нового стеков в памяти, чтобы убедиться, что это два разных объекта.

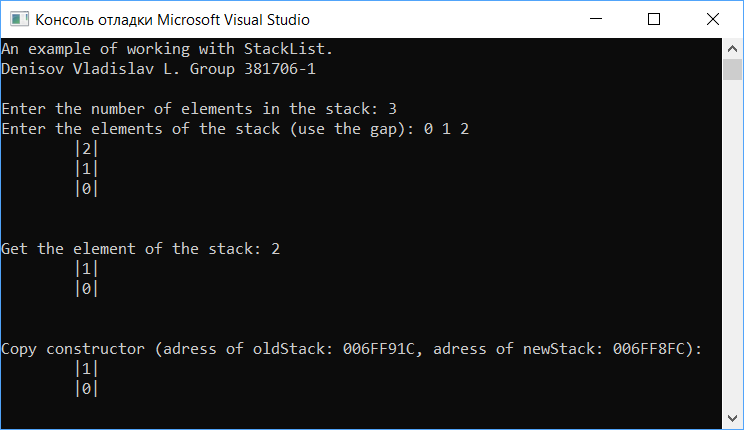


Рисунок 2 Пример работы демонстрационной программы.

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль StackList. Содержит пример использования стека. Реализация в файле *main\_StackList.cpp.*
* Модуль StackListLib – статическая библиотека. Содержит файл StackList.h, в котором описан интерфейс и реализация шаблонного класса *TStackList*.
* Модуль StackListTest. Содержит 4 теста, описанных в файле *StackListTest.cpp* и разработанных с помощью использования Google C++ Testing Framework.
* Модуль ExceptionLib – библиотека, позволяющая создавать собственные исключения.

## Описание структур данных

#### Класс TException – класс исключений.

Класс содержит 1 **private** поле *std::string msg* – переменная, хранящая сообщение об ошибке в виде строки.

И содержит 2 **public** элемента:

*TException(std::string \_str) : msg(\_msg)* – конструктор с одним параметром.

*void Print()* – метод отображения ошибки на консоль.

#### Класс TStackList – стек на списках.

Рассмотрим класс *TStackList* подробно.

template <class T> class TStackList : protected TList<T> {…} – класс стека является шаблонным, что позволяет использовать его для хранения данных любого типа. Кроме того, стек на списке построен с помощью наследования разработанного в прошлой лабораторной работе класса списка TList. Используется спецификатор protected, чтобы запретить использование public-методов класса TList и тем самым сохранить логику работы стека.

**Элементы класса, объявленные со спецификатором protected:**

int count –текущее количество элементов в стеке.

**Элементы класса, объявленные со спецификатором public:**

TStackList() – конструктор по умолчанию.

TStackList(TStackList<T> &s) – конструктор копирования.

virtual ~TStackList() – деструктор.

void Put(T a) – метод, позволяющий добавить новый элемент в стек.

T Get() – метод изъятия элемента из вершины стека с удалением.

void Print() – метод отображения текущих элементов стека.

## Описание алгоритмов

Рассмотрим некоторые алгоритмы, работа которых не очевидна на первый взгляд.

**Добавление элемента в стек.**

При добавлении элемента в стек вызывается метод добавления элемента в начало списка. А именно, в начале выполняется проверка на то, что стек не переполнен. Если проверка прошла успешно, то выделяется память под звено списка, в него записывается значение, которое необходимо положить в стек. В только что созданном звене запоминается указатель на текущее начало стека (а значит и списка, об этом было сказано в постановке задачи). Указатель на начало списка переопределяется на только что добавленный элемент. Следовательно, вершиной списка становится созданное звено.

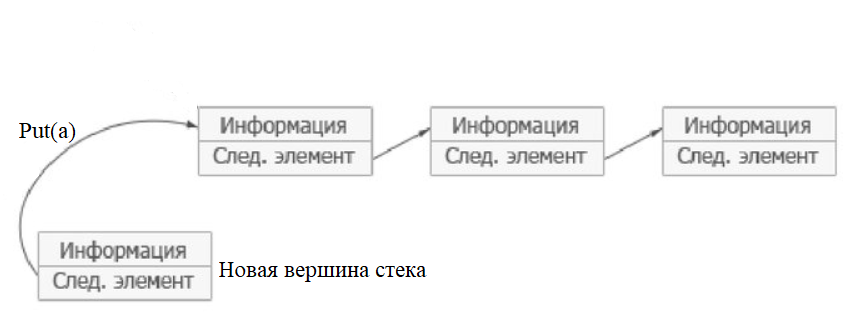


Рисунок 3 Добавление элемента в стек.

**Удаление элемента из стека.**

При удалении элемента из стека вызывается метод удаления элемент из начала списка. А именно, выполняются следующие действия. Во-первых, выполняется проверка стека на пустоту. Для этого вызывается метод проверки списка на пустоту. Если элементов для извлечения нет, то бросаем исключение. Иначе создаем указатель *\*temp* на звено списка, которому присваиваем значение текущего начала стека (списка). Создаем временную переменную *tmp,* в которую записываем значение, хранящееся в первом элементе списка. Начало списка устанавливаем на следующий за удаляемым элемент. Таким образом, на вершине стека оказывается бывший второй элемент. Удаляем указатель *\*temp* для того, чтобы очистить память, занимаемую бывшим первым элементом.

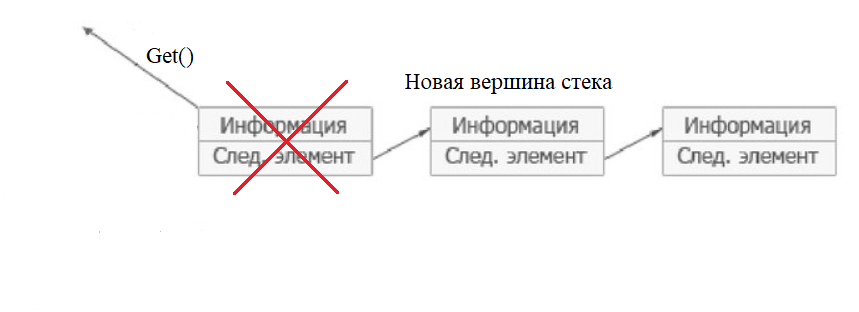


Рисунок 4 Удаление элемента из стека.

# Заключение

В результате лабораторной работы была разработана библиотека, реализующая шаблонный класс стека на линейном односвязном списке. Она позволяет создать объект класса стека и выполнить с ним простейшие операции, задача реализации которых была поставлена в начале данной лабораторной работы.

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

Программное решение было продемонстрировано с помощью простейшего набора операций над стеком. Описание примера работы со стеком было представлено в разделе «Руководство пользователя».

# Литература

1. Википедия: свободная электронная энциклопедия: на русском языке [Электронный ресурс] // URL: https://ru.wikipedia.org/wiki/Стек (дата обращения: 18.12.2018)
2. Методы программирования [Электронный ресурс] // URL: <http://www.itmm.unn.ru/files/2018/10/Primer-1.3.-Struktury-hraneniya-steka.pdf> (дата обращения: 18.12.2018)