МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Национальный исследовательский**

**Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**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

Колесова Кристина Юрьевна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018.

Оглавление

[1. Введение 3](#_Toc534392616)

[2. Постановка задачи 4](#_Toc534392617)

[3. Руководство пользователя 5](#_Toc534392618)

[4. Руководство программиста 6](#_Toc534392619)

[4.1. Описание структуры программы 6](#_Toc534392620)

[4.2. Описание структур данных 6](#_Toc534392621)

[4.3. Описание алгоритмов 6](#_Toc534392622)

[5. Заключение 9](#_Toc534392623)

[6. Список литературы 10](#_Toc534392624)

# 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. 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

Выполнение работы предполагает решение следующих задач:

1. Разработка и реализация класса стека на списках – TStackList
2. Реализация класса для обработки исключений – EexceptionLib
3. Реализация тестов на базе TStackList.
4. Пример использования класса TStackList.

# Руководство пользователя

При запуске программы пользователь сможет увидеть простейший пример работы со стеком на списках.

Пользователя просят ввести число n, затем создается стек размера n. Далее необходимо ввести элементы стека. После того, как все элементы будут введены, полученный стек выводится на экран. Затем из стека удаляется элемент, полученный в результате этого стек также выводится на экран.

# Руководство программиста

## Описание структуры программы

В программе содержатся следующие модули:

* Модуль StackLib – статическая библиотека. Содержит файл Stack.h, в котором описан интерфейс и реализация шаблонного класса *TStack*.
* Модуль ExceptionLib – библиотека, позволяющая создавать собственные исключения.
* Модуль MStackLib – содержит файл MStack.h, в котором реализованы классы TMStack и TNewStack
* Модуль MStack – содержит в себе файл реализации примера использования класса TMStack.
* Модуль MStackTest – содержит в себе файл MStackTest.cpp, в котором находится набор тестов, для проверки работоспособности класса TMStack.

## Описание структур данных

* Класс TNewStack

Класс TNewStack является шаблонным.

1. Элементы класса, объявленные со спецификатором public:

TNewStack (int \_size = 0, T\* \_mas = 0) - конструктор

TNewStack(TNewStack <T> &NS) - конструктор копирования

int GetFreeMemory() - получение свободной памяти (кол-во свободных позиций)

int GetSize() - получение размера

int GetTop() - получение первого элемента

void SetMas(int \_size, T\* \_mas) - задать массив размера \_size, содержащий элементы \_mas.

void Put(T \_A) - положить элемент \_A

T Get() - получить значение элемента

~ TNewStack() – деструктор.

* Класс TMStack

1) Элементы класса, объявленные со спецификатором protected:

int size - размер

T\* mas - массив элементов

int n - кол-во стеков

TNewStack<T>\*\* newS - массив указателей на начало каждого стека в мультистеке

int GetFreeMemory() – возвращает число свободных элементов

void Repack(int k) – перепаковка (память k-ого стека увеличивается)

2) Элементы класса, объявленные со спецификатором public:

TMStack(int \_n, int \_size) - конструктор

TMStack(TMStack<T> &A) - конструктор копирования

int GetSize() - возвращает размер

T Get(int \_n) - возвращает значение из n-ого стека

void Set(int \_n, T p) - положить значение в \_n-ый стек

bool IsFull(int \_n) - проверка стека на полноту \_n-ого стека

bool IsEmpty(int \_n) - проверка стека на пустоту \_n-ого стека

~TMStack() - деструктор

## Описание алгоритмов

* Конструктор инициализации

Размер и количество стеков определяются пришедшими значениями. Создается массив (smth), элементы которого, кроме первого, равны целой части от size/ n; первый же элемент больше на остаток от деления на n. Затем мы задаем значения стеков мультистека через конструктор инициализатор. Для первого элемента: newS[0] = new TNewStack<T>(smth[0], &mas[0]); Для всех остальных: int temp = smth[0] + (i - 1)\*smth[i]; newS[i] = new TNewStack<T>(smth[i], &mas[temp]);

* Перепаковка k-ого стека

Запоминаем старые, новые индексы начала, а также новый размер каждого из стеков. Распределяем свободную память поровну между стеками, кроме k-ого, ему выделяется память, размер которой определяется выражением: sizeNewOne[k] += FreeForNow % n, где FreeForNow – число свободных позиций. Индекс начального элемента каждого стека рассчитывается следующим образом: индекс начала предыдущего стека + его новый размер. Не изменяется лишь начало первого стека. Затем выполняем проверку на выполнение одного из следующих условий: а) Если индекс нового начала i-го стека startNewOne[i] не больше, чем индекс старого начала i-го стека startOldOne [i], то копируем элементы по порядку, в котором они хранятся в старом стеке. б) Иначе идем по новым позициям стеков до тех пор, пока не выполняется пункт а). Затем копируем элементы, в котором они хранятся в старом стеке, но в обратном порядке.

# Заключение

В результате выполнения лабораторной работы была разработана библиотека, позволяющая работать с шаблонным классом мультистека. С помощью этой библиотеки можно создать объект класса мультистека и некоторые операции операции над ним.

Были реализованы тесты для проверки работоспособности класса мультистека на базе GoogleTest. А также приведен пример работы стека.

# Список литературы

1. Википедия. Статья «Стек»: [https://ru.wikipedia.org/wiki/Стек].
2. Васильев А.Н. Самоучитель С++ с примерами и задачами. -СПб.: Наука и Техника, 2016. -480с.
3. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.