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**

**«Реализация N-стеков (на общей памяти)»»**

**Выполнил:** студент группы 381706-2

Банденков Даниил Викторович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2019.

# Содержание

[Содержание 2](#_Toc534333789)

[1 Введение 3](#_Toc534333790)

[2 Постановка задачи 4](#_Toc534333791)

[3 Руководство пользователя 5](#_Toc534333792)

[4 Руководство программиста 6](#_Toc534333793)

[4.1 Описание структуры программы 6](#_Toc534333794)

[4.2 Описание структур данных 6](#_Toc534333795)

[4.3 Описание алгоритмов 7](#_Toc534333796)

[5 Заключение 8](#_Toc534333797)

[6 Литература 9](#_Toc534333798)

# Введение

Стек (англ. stack — стопка; читается стэк) — абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»). Принцип работы стека сравнивают со стопкой листов бумаги: чтобы взять второй сверху, нужно снять верхний.

В стеке нет индексов как в массиве, а значит, вы не можете обратиться к определенному элементу. Все потому что, стек построен на связных списках. Зачем тогда использовать связные списки, если с таким же успехом можно было использовать простой массив? Все достоинство шаблонного стека заключается в добавлении и удалении элементов. Эти операции происходят за константное время (это хороший плюс).

Мультистек - Структура хранения нескольких стеков в общей памяти

# Постановка задачи

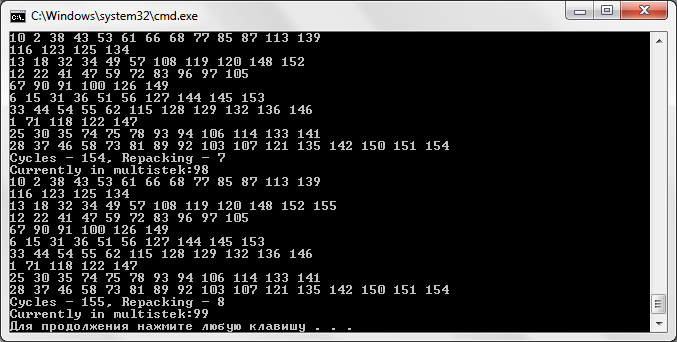
Цель данной лабораторной работы — разработать на языке программирования С++ статическую библиотеку, реализующую структуру данных для хранения N-стеков (на общей памяти).

Реализация класса мультистек:

1. Структура памяти для размещения нескольких стеков
2. Начальное распределение памяти
3. Ситуация локального переполнения памяти
4. Оценка свободной памяти
5. Динамическое перераспределение памяти при помощи перепаковки данных
6. Схема наследования и последовательность разработки программ
7. Реализация перепаковки

# Руководство пользователя

Данная программа тестирует динамическую структуру мультистек. В мультистек поочередно загружаются или удаляются элементы до тех пор, пока их число не достигнет 100. После каждого действия на экран выводится мультистек, число элементов в нем и число перепаковок. (см Рис 1).

Рис 1.Пример использования программы.

# Руководство программиста

## Описание структуры программы

*В решении содержатся следующие модули:*

1. multistacklib (TMultiStack.h, TMultiStack.cpp, Mstack.h)– модуль реализующий класс мультистек.
2. multistack(main.cpp)– модуль реализации программы для тестирования динамической структуры мультистек.
3. multistacktest(multistack\_test.cpp)- модуль тестирования класса мультистек при помощи Google C++ Testing Framework.

## Описание структур данных

**Структура:**

*Класс MStack является наследником класса TStack и имеет дополнительные*

*методы:*

*Методы:*

MStack(ValType\*ind=NULL, int Size=0) - перегруженный конструктор.

~Mstack() - перегруженный деструктор.

void SetMem(ValType\* ind, int Size) - функция для управления памятью.

**Класс TMultiStack:**

**Структура:**

*Поля:*

pStack - стеки - память выделяется из StackMem.

MemSize - всего выделено памяти.

CurrentCount — число элементов в мультистеке в данный момент.

StackCount- число стеков.

StackMem- память под стек.

pStackMem- базовые адреса для памяти стеков.

*Методы:*

int IsEmpty(int ns) const - контроль пустоты СД.

int IsFull(int ns) const - контроль переполнения СД.

int operator==(const TMultiStack &ms) - сравнение.

TMultiStack & operator= (const TMultiStack &ms) - присванивание.

ValType Get(int ns) - взять из стека с удалением.

void Put(int ns, const ValType &Val) - положить в стек.

## Описание алгоритмов

*Добавление в стек n элемента a (псевдокод)*

Если нет свободной памяти

Ошибка

Если стек n полон

Перепаковка

Положить в стек n значение a

# Заключение

В ходе работы реализована структура памяти для размещения N-стеков (на общей памяти), в нем реализованы функции контроля пустоты и переполнения, добавление и извлечение элементов, а так же динамическое перераспределение памяти при помощи перепаковки данных.

# Литература

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.
2. <https://codelessons.ru/cplusplus/realizaciya-steka-stack-v-c.html>

(Дата обращения 02.01. 2019)

1. <https://ru.wikipedia.org/wiki/Стек> (Дата обращения 02.01. 2019)