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 мультистэка»**

**Выполнил:** студент группы 381706-1

Соболева Ю.А.

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

аспирант каф. МОСТ ИИТММ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г.

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533027637)

[2. Постановка задачи 4](#_Toc533027638)

[3. Руководство пользователя 5](#_Toc533027639)

[4. Руководство программиста 6](#_Toc533027640)

4.[1. Описание структуры программы 6](#_Toc533027641)

4.[2. Описание структур данных 6](#_Toc533027642)

4.[3. Описание алгоритмов 7](#_Toc533027643)

[5. Заключение 8](#_Toc533027644)

[6. Литература 9](#_Toc533027645)

# **Введение**

**Структура данных** - программная единица, которая определяет метод хранения и обработки различных логически связанных данных в вычислительной технике. Знание структур данных позволяет наиболее компактно и практично расположить данные в памяти компьютера. В данной работе мы рассмотрим такую структуру данных, как **мультистэк**.

**Стек** – это последовательный список с переменной длиной, в котором включение и исключение элементов происходит только с одной стороны – с его вершины. Стек функционирует по принципу: последним пришел – первым ушел, Last In – First Out (LIFO).

**Мультистэк** – это массив с особой организацией памяти. Каждым элементом этого массива является стэк. Особенность мультистэка заключается в том, что при переполнении любого из стэков, свободные ячейки памяти можно забрать у других, не заполненных до конца стэков.

Программная реализация мультистека возможна на основе различных структур данных, например, с использованием статических или динамических массивов (векторов) и линейных списков.

**Целью данной лабораторной работы** является разработка структуры данных для хранения мультистеков с использование динамических массивов, а также освоение таких инструментов разработки программного обеспечения, как система контроля версий Git и фрэймворк для разработки автоматических тестов Google Test.

# **Постановка задачи**

Выполнение работы предполагает решение следующих задач:

1. Реализация класса стэка TNewStack, унаследованного от класса TStack.
2. Реализация класса мультистэка TMStack
3. Разработка интерфейса для данных классов.
4. Обеспечение работоспособности примера использования.
5. Реализация нескольких тестов на базе Google Test.

# **Руководство пользователя**

При запуске программы на экран выводится сообщение, что все тесты пройдены успешно. Примеры вывода сообщений для класса TMStack показаны на рис.1.

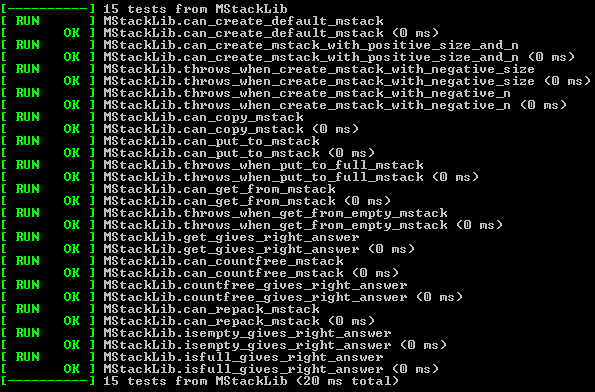


Рис. 1

В качестве примера реализации я выбрала метод Repack. Сначала мы создаем мультистэк на 3 стэка и 6 ячеек. Затем заполняем один из стэков до конца. Применяем метод Repack. И теперь мы снова можем класть значения в тот же стэк. На экран мы выводим количество свободных ячеек и затем выводим последний элемент стэка, с которым мы работали (рис. 2).



Рис 2.

# **Руководство программиста**

## **Описание структуры программы**

**Проект “MStack” состоит из следующих файлов:**

1. mainStack.cpp (в нём находится main)

**Проект “ MStack Lib ” состоит из следующих файлов:**

1. MStack.h (описание класса “TMStack” реализация методов класса “TMStack” и перегрузка операций )
2. NewStack.h (описание класса “TNewStack” реализация методов класса “TNewStack” и перегрузка операций )
3. MStackLib.cpp

**Проект “ Test” состоит из следующих файлов:**

1. test\_mstacklib.cpp (реализация тестов для класса TMStack)

## **Описание структур данных**

***Класс TNewStack:***

*Поля:*

Нет полей.

*Конструкторы и деструктор:*

* TNewStack<T> (int n, T\* mas); - конструктор инициализации;
* TNewStack<T> (TNewStack<T> &A); - конструктор копирования;

*Методы:*

* T GetTop();- забирает элемент из стека (если стек не пуст);
* void PutTop(T A); - добавляет элемент в стек (если стек не полон);
* int CountFreeE(); - возвращает количество свободных элементов массива;
* void SetM (int n, T\* mas); - устанавливает массив для стэка;
* int GetSize(); - возвращает размер массива;

*Перегруженные операторы:*

Нет перегруженных операторов.

***Класс TMStack:***

*Поля:*

* int n; - количество подстеков;
* int size; - размер мультистэка;
* T\* mas; - массив для хранения элементов;
* TNewStack<T>\*\* m; - массив стэков;
* *Конструкторы:*
* TMStack<T> (int \_n = 0, int l = 0); - конструктор инициализации
* TMStack<T> (TMStack<T> &A); - конструктор копирования;

*Методы:*

* T Get(int i) ;- забирает элемент из стека i(если стек не пуст);
* void Put(int i, T а); - добавляет элемент в стек i(если стек не полон);
* bool IsFull(); - проверяет полон ли мультистек;
* bool IsEmpty(); - проверяет пуст ли мультистек;
* int CountFree(); - считает количество свободных элементов;
* void Repack(int k); - меняет количество элементов в стэках;

*Перегруженные операторы:*

Нет перегруженных операторов.

## **Описание алгоритмов**

**Метод Repack.**

Ищем количество свободных ячеек, которые можно добавить в каждый стек: делим количество свободных ячеек во всем мультистеке на количество стеков. Увеличиваем старые размеры стеков на это количество. Если количество свободных ячеек во всем мультистеке не кратно количеству стеков n, то оставшиеся свободные ячейки добавляем в переполненный стек. Затем определяем новые индексы начала стэков. Если требуется перемещение элементов стэка, то действуем по следующему правилу:

* Если индекс нового начала i-го стека не больше, чем индекс старого начала i-го стека, то копируем элементы по порядку, в котором они хранятся в старом стеке.
* Иначе идем по новым позициям стеков до тех пор, пока не выполнится первое правило. Затем копируем элементы, в котором они хранятся в старом стеке в обратном порядке.

# **Заключение**

В ходе выполнения лабораторной работы я смогла реализовать такую структуру данных, как мультистэк. Вместе с ней, согласно заданному интерфейсу, я реализовала класс TMStack. Написание нескольких своих тестов, помогло мне разобраться с системой автоматических тестов Google Test.

В результате проделанной работы у меня получилось

1. Реализовать класса стэка TNewStack, унаследованного от класса TStack.

2. Реализовать класса мультистэка TMStack

3. Разработать интерфейс для данных классов.

4. Обеспечить работоспособность примера использования.

5. Реализовать некоторые тесты на базе Google Test.

Таким образом, данная лабораторная работа отвечает всем поставленным задачам. Это означает, что цель работы была достигнута.

# **Литература**

1. Лабораторный практикум. Составители: Барышева И.В., Мееров И.Б., Сысоев А.В., Шестакова Н.В. Под редакцией Гергеля В.П. Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. – 105с.

URL: <http://www.unn.ru/books/met_files/Pract_ADS.pdf>

1. Национальный открытый университет «Интуит». Курс «Основы программирования». Лекция 11: Структуры данных: общее понятие, реализация. Простейшие структуры данных: очередь, стек. Использование стека и обратная польская запись.

URL: <https://www.intuit.ru/studies/courses/2193/67/lecture/1980?page=3>