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**

**«Мультистек»**

**Выполнил:** студент группы 381706-1

Суслов Егор Игоревич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533083472)

[2. Постановка задачи 4](#_Toc533083473)

[3. Руководство пользователя 5](#_Toc533083474)

[4. Руководство программиста 6](#_Toc533083475)

[4.1. Описание структуры программы 6](#_Toc533083476)

[4.2. Описание структур данных 6](#_Toc533083477)

[4.3. Описание алгоритмов 7](#_Toc533083478)

[5. Заключение 8](#_Toc533083479)

[6. Литература 9](#_Toc533083480)

# Введение

# Мультистек — структура данных, представляющая собой упорядоченный набор стеков, фиксированного размера.

Стек ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *stack* — стопка; читается *стэк*) — [абстрактный тип данных](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), представляющий собой [список элементов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), организованных по принципу [*LIFO*](https://ru.wikipedia.org/wiki/LIFO) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *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](https://ru.wikipedia.org/wiki/%D0%A6%D0%92%D0%9A) стек называется магазином — по аналогии с [магазином в огнестрельном оружии](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD_(%D1%87%D0%B0%D1%81%D1%82%D1%8C_%D0%BE%D1%80%D1%83%D0%B6%D0%B8%D1%8F)) (стрельба начнётся с патрона, заряженного последним).

В 1946 [Алан Тьюринг](https://ru.wikipedia.org/wiki/%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3,_%D0%90%D0%BB%D0%B0%D0%BD) ввёл понятие стека. А в 1957 году немцы Клаус Самельсон и Фридрих Л. Бауэр запатентовали идею Тьюринга.

Стек является общей структурой данных для представления данных, которые должны обрабатываться в определенном порядке. Например, когда функция вызывает другую функцию, которая, в свою очередь, вызывает третью функцию, важно, чтобы третья функция вернулась на вторую функцию, а не первую.

Один из способов реализации такого порядка обработки данных — это организовать своего рода очередь вызовов функций. Последняя добавленная в стек функция, будет завершена первой и наоборот, первая добавленная в стек функция будет завершена последней. Таким образом, сама структура данных обеспечивает надлежащий порядок вызовов.

# Постановка задачи

В рамках лабораторной работы ставится задача эффективной реализации структуры данных – мультистека.

Для работы со мультистеком необходимо реализовать операции:

* добавления элемента в мультистек,
* извлечения с элемента из мультистека,
* проверка мультистека на полноту/пустоту.

Программное решение будет выглядеть следующим образом:

1. Разработка и реализация вспомогательного класса стека – TNewStack.
2. Разработка и реализация класса мультистека – TMultiStack.
3. Подключение класса для обработки исключений – MyException.
4. Разработка программы, демонстрирующей работу классa TMultiStack (получается при выполнении файла кода «MultiStack\_main.cpp».
5. Реализация набора автоматических тестов с использованием Google C++ Testing Framework.

# Руководство пользователя

1. Создается мультистек состоящий из двух стеков в каждом из которых по 6 элементов.
2. Мультистек с введенными числами выводится на экран
3. Далее берется по 2 элемента из каждого стека
4. Затем добавляется «5» во второй стек
5. Вывод мультистека
6. Закрытие программы

# Руководство программиста

## Описание структуры программы

Программа состоит из следующих модулей:

* Модуль MultiStack. Содержит пример использования мультистека. Реализация в файле «MultiStack\_main.cpp*».*
* Модуль MultiStackLib – статическая библиотека. Содержит файл MultiStack.h, в котором описан интерфейс и реализация шаблонного класса TMultiStack(2 конструктора и 8 методов), и «TNewStack.h», в котором описан интерфейс и реализация шаблонного класса TNewStack(2 конструктора и 7 методов)
* Модуль MultiStackTest. Содержит 27 тестов, описанных в файле «MultiStackTest.cpp» и разработанных с помощью использования Google C++ Testing Framework.
* Модуль ExceptionLib – библиотека, содержащая класс исключений.

## Описание структур данных

#### Класс TNewStack

Класс TNewStack является шаблонным классом. Он наследуется от класса TStack со спецификатором public. Внутри класса определены 2 конструктора и 7 методов со спецификатором доступа public:

* TNewStack(int \_size, T\* \_mas) - конструктор по умолчанию.
* TNewStack(TNewStack<T>& A) – конструктор копирования.
* int GetFreeMem() – получить количество свободных ячеек в стеке.
* T Get() – забрать элемент из стека.
* void Push(T \_A) – положить элемент в стек.
* int GetSize() – получить размер стека.
* int GetTop() - получить позицию вершины стека.
* void SetMas(int \_size, T\* \_mas) – преобразование массива mas размером size в стек.
* void PrintNewStack() – вывод стека на консоль.

**Класс TMultiStack**

Класс TMultiStack является шаблонным классом. Внутри класса определены 4 поля и 2 метода со спецификатором доступа protected:

* int size – размер мультистека.
* T\* mas – указатель на область память под хранение мультистека.
* int n – количество стеков в мультистеке.
* TNewStack<T>\*\* stackMas – массив указателей на начало каждого стека в мультистеке.
* int CountFree() – количество свободных элементов в мультистеке.
* void Repack(int \_n) – перепаковка стека, с увеличением свободной памяти в n-м стеке.

В публичной зоне описаны 2 конструктора 6 методов:

* TMStack(int \_n = 1, int \_size = 10) - конструктор по умолчанию.
* TMStack(TMStack &A) - конструктор копирования.
* int GetSize() – возвращает размер мультистека.
* void Set(int \_n, T \_elem); - положить в n-й стек элемент elem.
* T Get(int \_n) – взять элемент из n-го стека.
* bool IsFull(int \_n) – проверка на полноту n-го стека.
* bool IsEmpty(int \_n) – проверка на пустоту n-го стека.
* void PrintMStack() – вывод мультистека на консоль.

## Описание алгоритмов

**Добавление звена списка в начало и в конец.**

Сначала проверяем не заполнен ли список. Если он заполнен, то бросаем исключение. Если нет, то в очереди свободных позиций freeElem, берем первую свободную ячейку i. По полученному индексу в массив mas записываем значение, которое хотим положить в список. Определяем, что следующим для этого элемента, будет элемент с текущим индексом start, то есть nextInd[i] = start. Если, перед добавлением, список не был пуст, то предыдущим для первого элемента списка делаем только что добавленный элемент, то есть: predInd[start] = i. Если же список был пуст, то определяем, что добавленный элемент является и последним элементом в списке. Затем индекс start переопределяем на только что добавленный элемент: start = i. Увеличиваем количество элементов в списке count++.

Для добавления элемента в конец списка, рассуждения аналогичны, с поправкой на то, что добавляем в конец списка.

# Заключение

Все поставленные задачи были выполнены это подтверждается Google C++ Testing Framework (все 26 тестов, 17 на класс TMultiStack и 9 на вспомогательный класс TNewStack, успешно выполняются) и работой с файлом кода «MultiStack \_main.cpp» использующего наш класс.

# Литература

* Книги:

1. A.O. Грудзинский. Методы программирования, Издательство Нижегородского госуниверситета, 2006.
2. Топп У., Форд У. Структуры данных в С++. - М. Бином, 1999.
3. Мейн М., Савитч У. Структуры данных и другие объекты в С++. - М.: Издательский дом “Вильямс”, 2003.
4. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.

* Ссылки в Internet:

1. Учебно-методическое пособие из электронной библиотеки ННГУ: «ЛАБОРАТОРНЫЙ ПРАКТИКУМ» : [http://www.unn.ru/books/met\_files/Pract\_ADS.pdf].