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: мультистек»**

**Выполнил:** студент группы 381903-3

Микийчук Дмитрий Алексеевич

**Научный руководитель:**

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

Нижний Новгород

2020

Оглавление

[1. Введение. 2](#_Toc533106246)

[2. Постановка задачи. 3](#_Toc533106247)

[3. Руководство пользователя. 4](#_Toc533106248)

[4. Руководство программиста. 5](#_Toc533106249)

[4.1. Описание структуры программы. 5](#_Toc533106250)

[4.2. Описание структур данных. 5](#_Toc533106251)

[4.3. Описание алгоритмов. 6](#_Toc533106252)

[5. Заключение. 7](#_Toc533106261)

[6. Литература. 8](#_Toc533106262)

# Введение.

**Стек** — [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 FILO – первым вошел, последним вышел. Чаще всего принцип работы стека сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно снять верхнюю.

Из стека нельзя извлечь произвольный элемент. Есть только вершина стека, и мы можем или положить туда элемент, или забрать его оттуда.

**Мультистек -** [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 набор из некоторого количества стеков. Стеки хранятся в памяти друг за другом.

# Постановка задачи.

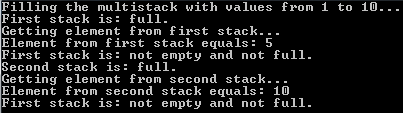
В данной лабораторной работе нужно разработать эффективную библиотеку для хранения и работы с такой структурой данных, как стек.

Для этого нам нужно:

* Описать и реализовать вспомогательный класс **TNewStack**.
* Описать и реализовать класс **TMStack**.
* Протестировать класс **TMStack** с помощью Google Test.
* Реализовать класс **TMyException** для обработки исключений, которые могут возникнуть в результате работы класса **TMStack**.
* Продемонстрировать работу класса **TMStack**.

# Руководство пользователя.

Пример работы класса **TMStack**:



*Рис 1. Пример работы программы*

Программа работает следующим образом:

* Создается мультистек (в данном примере мультистек состоит из двух стеков, размер каждого из которых равен пяти).
* Заполняем первый мультистек значениями от 1 до 10 (первый стек заполняется значениями от 1 до 5, а второй – от 6 до 10).
* Проверяем первый стек на полноту.
* Забираем элемент с вершины первого стека.
* Проверяем первый стек на полноту и пустоту.
* Проверяем второй стек на полноту.
* Забираем элемент с вершины второго стека.
* Проверяем второй стек на полноту и пустоту.

# Руководство программиста.

# Описание структуры программы.

Программа состоит из модулей:

* **MultiStack** – содержит в себе файл **multistack\_main.cpp** с реализацией примера использования класса **TMStack.**
* **MultiStackLib –** содержит в себе файлы **TNewStack.h** и **TMStack**, в которых описаны и реализованы классы **TNewStack** и **TStack** соответственно.
* **MultiStackTest** – содержит в себе файл **test\_multistack.cpp**, в котором находится набор тестов, для проверки работоспособности класса **TMStack**.
* **MyExceptionLib** –содержитв себе файл **MyException.h** с реализацией класса исключений **TMyException**.

# Описание структур данных.

**Класс TNewStack.**

Класс **TNewStack** является шаблонным наследником от класса **TStack** и содержит следующие методы со спецификатором доступа **public**:

* **TNewStack(int \_size, T\* \_mas)** – конструктор-инициализатор.
* **TNewStack(TNewStack<T> &obj)** – конструктор копирования.
* **void SetM(int \_size, T\* \_mas)** – метод, преобразующий массив заданного размера в стек.
* **int GetFreeMem()** – метод, возвращающий количество свободных мест в стеке.
* **int GetSize()** – метод, возвращающий размер стека.
* **int GetTop()** – метод, возвращающий вершину стека (без удаления).
* **void PutElem(T \_elem)** – метод, помещающий элемент в стек.
* **T Get()** – метод, возвращающий элемент из стека (с удалением).

**Класс TNewStack.**

Класс **TMStack** является шаблонным и содержит четыре поля и два метода со спецификатором доступа **protected**:

* **int n** – количество стеков в мультистеке.
* **int size** – размер мультистека.
* **T\* mas** – массив для хранения стеков.
* **TNewStack<T>\*\* mStack** – массив указателей на начало каждого стека.
* **int CountFree()** – количество свободных мест в мультистеке.
* **void Repack(int \_num)** – метод перепаковки стека.

Далее идут методы класса со спецификатором доступа **public**:

* **TMStack(int \_n = 1, int \_size = 5)** – конструкто**-**инициализатор.
* **TMStack(TMStack<T> &obj)** – конструктор копирования.
* **~TMStack()** – деструктор.
* **int GetSize()** – метод, возвращающий размер мультистека.
* **void SetElem(int \_n, T \_elem)** – метод, помещающий элемент в стек с указанным номером.
* **T GetElem(int \_n)** – метод, возвращающий элемент из стека с указанным номером.
* **bool IsFull(int \_n)** – метод, проверяющий стек с указанным номером на полноту.
* **bool IsEmpty(int \_n)** – метод, проверяющий стек с указанным номером на пустоту.

**Класс TMyException.**

Класс **TMyException** содержит одно поле со спецификатором доступа **private**:

* **string str** – строка, хранящая сообщение об ошибке.

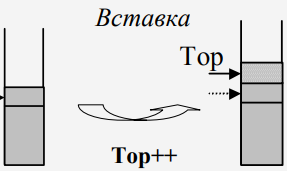
Далее идут методы класса со спецификатором доступа **public**:

* **TMyException(std::string \_str)** – конструктор инициализатор.
* **void What()** – метод вывода ошибки на экран.

# Описание алгоритмов.

**Добавление элемента в стек.**

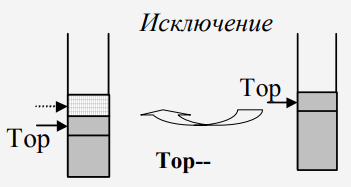
Добавляя элемент в стек с указанным номером, мы помещаем его в вершину стека, на которую указывает поле **top**. После добавления элемента в стек, значение поля **top** увеличивается на 1.



*Рис. 2 Добавление элемента в стек.*

**Удаление элемента из стека.**

Удаляя элемент из стека с указанным номером, мы забираем его из первой непустой ячейки. На эту непустую ячейку указывает поле **top** со значением, уменьшенным на 1.



*Рис. 3 Удаление элемента из стека.*

# Заключение.

В данной лабораторной работе мне удалось реализовать библиотеку для хранения и работы с мультистеком, а именно:

* Удалось реализовать вспомогательный класс **TNewStack**.
* Удалось реализовать вспомогательный класс **TMStack**.
* Удалось протестировать методы класса **TMStack**, а также обеспечить их работоспособность.
* Удалось реализовать класс для обработки исключений **TMyException**.

Таким образом, я смог реализовать структуру данных под названием мультистек.

# Литература.

1. Ссылка из Википедии про стек:

<https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA>

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», 2015.