Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе №5**

**«Структура хранения данных: мультистек»**

Выполнила:

студентка ф-та ИТММ ПМИ – 381903-3

Семибабнова Анна Владимировна

Проверил:

ассистент кафедры МОСТ

Лебедев Илья Геннадьевич

Нижний Новгород

2020

Содержание

[Введение 3](#_Toc23876641)

[1. Постановка задачи 4](#_Toc23876642)

[2. Руководство пользователя 5](#_Toc23876643)

[3. Руководство программиста 6](#_Toc23876644)

[3.1. Описание структуры программы 6](#_Toc23876645)

[3.2. Описание алгоритмов 6](#_Toc23876646)

[Заключение 7](#_Toc23876647)

[Литература 8](#_Toc23876648)

# Введение.

**Стек** – это динамическая структура сохранения данных, которая работает по принципу «последний пришел — первый вышел» (Last-In First-Out). В стеке добавление новых элементов и удаление существующих элементов производится с одного конца, который называется «вершиной стека».

**Мультистек –** это динамическая структура сохранения данных, представляющая собой набор из некоторого количества стеков. Стеки хранятся в памяти друг за другом.

Организация данных с помощью стека эффективна, когда нужно реализовать:

* обмен данными между методами приложения с помощью параметров;
* синтаксический анализ разнообразных выражений.

В программировании стек можно реализовывать разными способами, например:

* в виде статического массива;
* в виде динамического массива;
* в виде односвязного списка;
* в виде двусвязного списка.

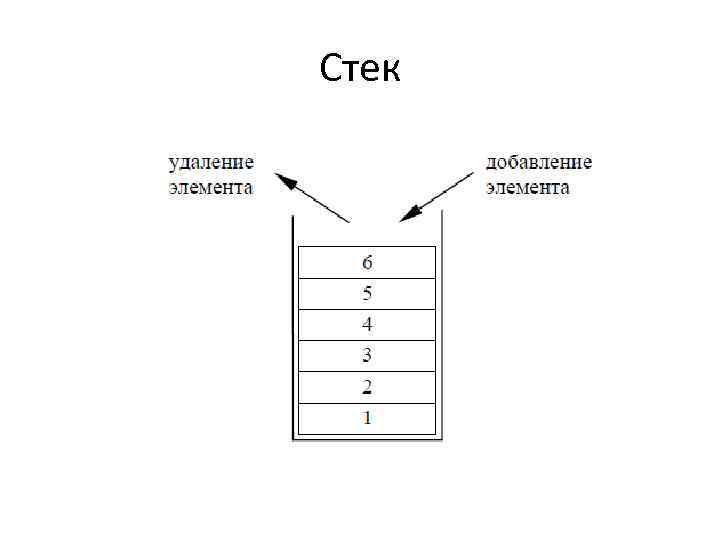


Рис.1 Интерпретация стека.

# Постановка задачи

В рамках лабораторной работы ставится задача создания программных средств, поддерживающих эффективную реализацию структуры данных – мультистек и выполнение основных операций над ней:

* добавления элементов в мультистек,
* извлечения элементов из мультистека (с удалением),
* проверка мультистека на полноту/пустоту.

# Руководство пользователя

Использование программы пользователем:

1. Зайти в интересуемый репозиторий на GitHub.
2. Перейти в нужную ветку под названием Lab5.
3. Скачать код программы. Это легко можно сделать, скачав файлы программы архивом. Для этого нужно нажать на кнопку “Code”, затем на кнопку “Download ZIP”. После этих действий появится окно проводника. Нужно сохранить данный файл на рабочий стол и извлечь из него файлы программы (нажать правой кнопкой мыши на файл, выбрать команду “Извлечь все”).

# Открыть папку с программой на рабочем столе.

1. Создать папку “build”.
2. Перейти в эту папку, нажать правой кнопкой мыши в поле проводника, нажать на “Git Bash Here” (предварительно нужно скачать программу Git с официального сайта).
3. Благодаря действиям из предыдущего пункта на экране появится консоль, в которой нужно прописать команду “cmake ../” (предварительно нужно скачать программу CMake, позволяющую автоматически собрать программу из исходного кода).
4. Теперь проект MultiStack.sln можно открыть в Visual Studio.
5. Пользователю нужно нажать на проект “ MultiStackPrime” правой кнопкой мыши и выбрать пункт “Назначить автозагружаемым проектом”.
6. Затем нужно нажать на сочетание клавиш CTRL+F5 на клавиатуре. Откроется консольное приложение, демонстрирующее работу мультистека.

## Руководство программиста

## Описание структуры программы

Программа написана на языке программирования C++.

Состоит из четырех файлов:

* **MultiStack.h** – класс мультистеков и методы реализации данного класса,
* **MultiStackPrime.cpp** – файл программы тестирования,
* **Test\_MultiStack.cpp –** файл тестирования функций для классов,
* **Test\_main.cpp -** файл тестирования функций для классов

В программе реализованы следующие классы:

1. **Класс «MultiStack»**

Описание: структура данных Мультистек.

* MultiStack(int \_StackCount, int CeilCount) – 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 ShiftRight(int index, int count) - задать массив,
* void Put(T \_A) - положить элемент \_A,
* T Get() - получить значение элемента,
* ~ MultiStack() – деструктор.

## Описание алгоритмов

1. **Добавление элемента в стек:**

Процедура добавление элемента в стек с указанным номером, позволяет поместить его на вершину стека, на которую указывает поле top. После добавления элемента в стек, значение поля top увеличивается на 1.

1. **Удаление элемента из стека:**

В процедуре удаления элемента из стека с указанным номером забирается его из первой непустой ячейки. На эту непустую ячейку указывает поле top со значением, уменьшенным на 1.

# Заключение

При выполнении лабораторной работы реализовано хранение и операции с мультистеком.

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

# Литература

1. Лафоре Р. Объектно-ориентированное программирование в C++. Классика Computer Science. 4-е изд. – СПБ.: Питер, 2018. – 928 с.
2. Страуструп Бьерн Язык программирования C++. Специальное издание. Пер. с англ. – М.: Издательство Бином, 2017 г. – 1136 с.
3. Шилдт Г. С++ Базовый курс. 3-е изд. – М.: Издательство Вильямс, 2018. 624 с.