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**

**ЛАБОРАТОРНАЯ РАБОТА №5**

**«Структура хранения данных: Мультистек»**

**Выполнил:** студент группы 381706-2

Макарихин Семён Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Руководитель:**

Ассистент кафедры МОСТ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2019

Содержание

[1. Введение 3](#_Toc536638418)

[2. Цели и задачи 4](#_Toc536638419)

[3. Руководство пользователя 5](#_Toc536638421)

[4. Руководство программиста 6](#_Toc536638422)

[4.1. Описание структуры программы 6](#_Toc536638423)

[4.2. Описание функций и процедур, их алгоритмов 6](#_Toc536638424)

[5. Эксперименты 10](#_Toc536638425)

[6. Заключение 1](#_Toc536638426)1

[7. Литература 1](#_Toc536638427)2

[8. Приложение 1](#_Toc536638428)3

# 1. Введение

Лабораторная работа направлена на практическое освоение динамической структуры данных Мультистек.

Стек (англ.stack– стопка) – это структура данных, в которой новый элемент всегда записывается в ее начало (вершину) и очередной читаемый элемент также всегда выбирается из ее начала. В стеках используется метод доступа к элементам LIFO ( Last Input – First Output, "последним пришел – первым вышел"). Чаще всего принцип работы стека сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно сначала взять верхнюю.

Разработка класса Мультистек обеспечит работу с несколькими стеками.

# 2. Цели и задачи

В рамках лабораторной работы ставится задача эффективной реализации структуры данных – мультистек и выполнение основных операций над ней:

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

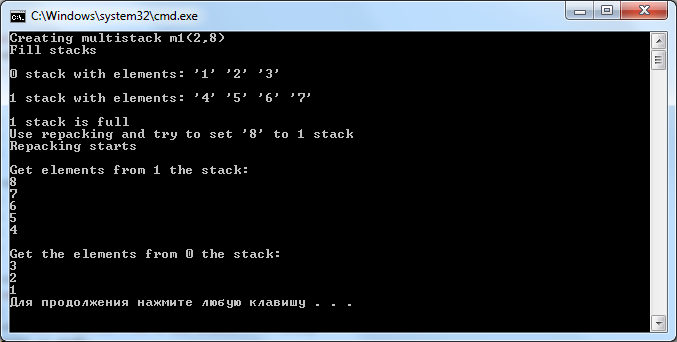
В процессе выполнения лабораторной работы требуется использовать систему контроля версий [Git](https://git-scm.com/book/ru/v2).

Выполнение работы предполагает решение следующих задач:

1. Реализация класса TNewStack, построенного на основе существующего класса TStack.
2. Реализация класса TMStack, построенного на основе класса TNewStack.
3. Реализация класса для обработки исключений– TException, которые могут возникнуть при выполнении различных операций, согласно заданному интерфейсу.
4. Обеспечение работоспособности тестов и примера использования.
5. Реализация заготовок тестов, покрывающих все методы созданных классов.
6. Модификация примера использования в тестовое приложение, позволяющее задавать мультистеки и осуществлять основные операции над ними.

# 3. Руководство пользователя

Запускаем программу из файла multistack.cpp. Программа выведет пример корректной работы основных операций с мультистеком .



# 

# 4. Руководство программиста

Разработка системы вычисления проводились в среде “Microsoft Visual Studio 2010”.

В данной работе будет использовано 4 класса:

* Класс «Стек» (TStack), реализованный с использованием массива.
* Класс TNewStack, построенного на основе класса TStack.
* Класс «Мультистек» (TMStack), построенного на основе класса TNewStack.
* Класс исключения (TExсeption).

## 4.1. Описание структуры программы

Модульная структура программы:

* tstack.h– модуль с классом TStack, в котором определен интерфейс шаблонного класса Стек и реализация его методов.
* tnewstack.h– модуль с классом TNewStack, в котором определен интерфейс и реализация его методов.
* tmultistack.h– модуль с классом TMStack, в котором определен интерфейс шаблонного класса Мультистек и реализация его методов.
* exсeption.h – модуль с классом исключения TExсeption.
* sample\_tmultistack.cpp, sample\_performance\_check.cpp– модуль программы тестирования, с которым работает пользователь, в котором проводятся эксперименты.
* test\_main.cpp, test\_tstack.cpp, test\_ tmultistack.cpp – модуль с функциями тестирования для созданных классов. Содержит 20 тестов для класса TStack и 12 тестов для класса TMStack.

## 4.2. Описание функций и процедур, их алгоритмов

Рассмотрим реализацию методов шаблонного класса TStack:

template <typename StackType>

class TStack

1. TStack(int n = 0) - конструктор класса, принимающий размер стека. По умолчанию создается стек размера 0 с позицией вершины стека 0.

2. TStack(TStack<StackType> &S) - конструктор копирования.

Принимает ссылку на объект класса TStack.

3. ~TStack() - деструктор. Освобождает выделенную под вектор память.

4. TStack& operator=(const TStack<StackType>& stack) – перегрузка оператора присваивания одного стека другому.

Присваивает полям первого объекта класса поля второго объекта класса.

5. int GetSize() - возвращает размер стека.

6. StackType Get() – метод изъятия элемента из вершины стека с удалением.

7. void Put(StackType A) - метод, позволяющий добавить новый элемент в стек.

8. bool IsFull() – метод проверки стека на полноту.

9. bool IsEmpty() – метод проверки стека на пустоту.

10. void PrintStack() - метод отображения текущих элементов стека.

11. bool operator==(const TStack<StackType>& stack) const – перегрузка оператора сравнения.

Выполняется проверка стеков на равенство. Возвращает true, если равенство выполняется, false в противном случае.

12. bool operator!=(const TStack<StackType>& stack) const - перегрузка оператора сравнения.

Выполняется проверка стеков на неравенство. Возвращает true, если неравенство выполняется, false в противном случае.

Рассмотрим реализацию методов шаблонного класса TNewStack:

template <typename NewStackType>

class TNewStack :public TStack<NewStackType>

1. TNewStack(int \_size = 0, NewStackType\* \_memory = 0) - конструктор класса.

2. TNewStack(TNewStack <NewStackType> &NS) - конструктор копирования.

Принимает ссылку на объект класса TNewStack.

3. int GetFreeMemory() – метод для получения свободной памяти (количество свободных позиций)

4. int GetSize() – метод, возвращающий размер.

5. int GetTop() – метод, возвращающий первый элемент.

6. void SetMemory(int \_size, NewStackType\* \_memory) – метод выделении памяти под массив.

7. void Put(NewStackType \_A) – метод добавления элемента.

8. NewStackType Get() – метод изъятия элемента из вершины стека с удалением.

Рассмотрим реализацию методов шаблонного класса TMStack:

template <typename MultiStackType>

class TMStack

1. TMStack(int \_n, int \_size) - конструктор класса, принимающий количество стеков и общий размер мультистека.

2. TMStack(TMStack<MultiStackType> &A) – конструктор копирования.

Принимает ссылку на объект класса TMStack.

3. int GetSize(){ return size; } - возвращает размер стека мультистека.

4. MultiStackType Get(int \_n) - метод изъятия элемента из вершины стека с удалением.

5. void Set(int \_n, MultiStackType p) - метод, позволяющий добавить новый элемент в один из стеков мультистека. Принимает индекс стека и добавляемый элемент.

6. bool IsFull(int \_n) - метод проверки мультистека на полноту.

7. bool IsEmpty(int \_n) – метод проверки мультистека на пустоту.

8. int GetFreeMemory() – метод оценки объема свободной памяти.

9. void Repack(int k) – перепаковка памяти - динамическое распределение памяти путем переписи части хранимых значений в другую область ( другой стек ).

# 5. Эксперименты

В качестве примера рассмотрим перегрузку оператора присвоения для класса мультистека (TMStack).

Теоретическая сложность выполнения алгоритма O(1).

Мы провели измерение присваивая мультистеку разное количество элементов: 10, 100, …, 100000 элементов. Ниже вы можете увидеть график зависимости времени выполнения операции присвоения от количества элементов мультистека. По приведенным данным можно сделать вывод, что практическая сложность выполнения алгоритма равна теоретической.

По горизонтали – количество присваиваемых элементов.

По вертикали - время выполнения программы.

# 6. Заключение

В результате лабораторной работы была разработана структура данных - мультистек, а также освоены такие инструменты разработки программного обеспечения, как система контроля версий [Git](https://git-scm.com/book/ru/v2).

# 7. Литература

1. Лабораторный практикум: Учебно-методическое пособие / Мееров И.Б. [и др.] – Нижний Новгород: Нижегородский госуниверситет , 2017. – 105с.
2. <https://ru.wikipedia.org/wiki/Стек>
3. [http://www.programmersforum.ru](http://www.programmersforum.ru/)

# 8. Приложение