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 №3**

**«Структура хранения данных: Стек»**

**Выполнил:** студент группы 381706-2

Макарихин Семён Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Руководитель:**

Ассистент кафедры МОСТ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2019

Содержание

[1.Введение 3](#_Toc533245674)

[2. Цели и задачи 4](#_Toc533245675)

[3. Руководство пользователя 6](#_Toc533245677)

[4. Руководство программиста 8](#_Toc533245678)

[4.1. Описание структуры программы 8](#_Toc533245679)

[4.2. Описание функций и процедур, их алгоритмов 8](#_Toc533245680)

[5. Эксперименты 10](#_Toc533245681)

[6. Заключение 11](#_Toc533245682)

[7. Литература 12](#_Toc533245683)

[8. Приложения 13](#_Toc533245684)

# 1.Введение

Лабораторная работа направлена на практическое освоение динамической структуры данных Стек.

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[[](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA#cite_note-1). А в 1957 году немцы Клаус Самельсон и Фридрих Л. Бауэр запатентовали идею Тьюринга.

В некоторых языках (например, [Lisp](https://ru.wikipedia.org/wiki/Lisp" \o "Lisp), [Python](https://ru.wikipedia.org/wiki/Python" \o "Python)) стеком можно назвать любой список, так как для них доступны операции pop и push. В языке [C++](https://ru.wikipedia.org/wiki/C%2B%2B) [стандартная библиотека](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0_C%2B%2B) имеет класс с реализованной структурой и методами. И т. д.

# 2. Цели и задачи

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

* добавления элемента в стек,
* извлечения элемента из стека (с удалением),
* проверка стека на полноту/пустоту.

Хранить данную структуру в этой лабораторной работе будем с помощью массива. Такая организация стека удобна, если элемент информации занимает в памяти фиксированное количество длины, например, какую-то одну условную единицу. При этом отпадает необходимость хранения в элементе стека явного указателя на следующий элемент стека, что экономит память.

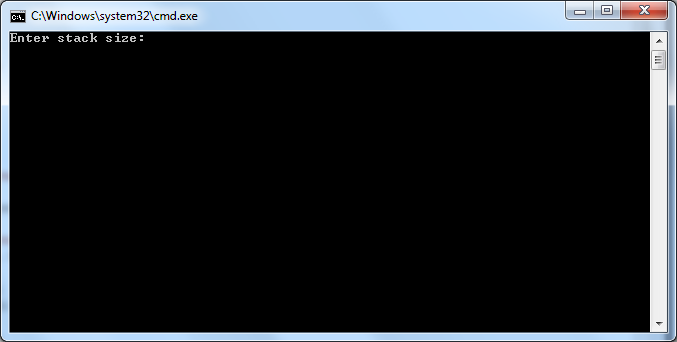
Выполнение работы предполагает решение следующих задач:

1. Реализация методов шаблонного класса TStack согласно заданному интерфейсу.
2. Реализация класса для обработки исключений– TException, которые могут возникнуть при выполнении различных операций, согласно заданному интерфейсу.
3. Обеспечение работоспособности тестов и примера использования.
4. Реализация заготовок тестов, покрывающих все методы TStack.
5. Модификация примера использования в тестовое приложение, позволяющее задавать стеки и осуществлять основные операции над ними.

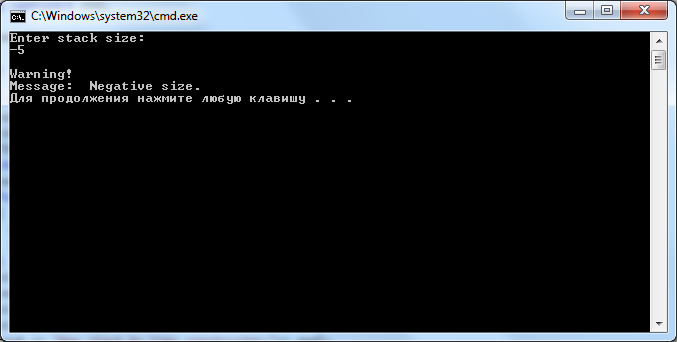
.

# 3. Руководство пользователя

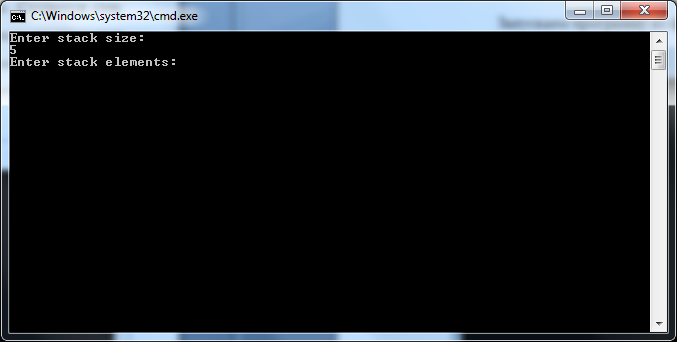
Запускаем программу из файла stack.cpp



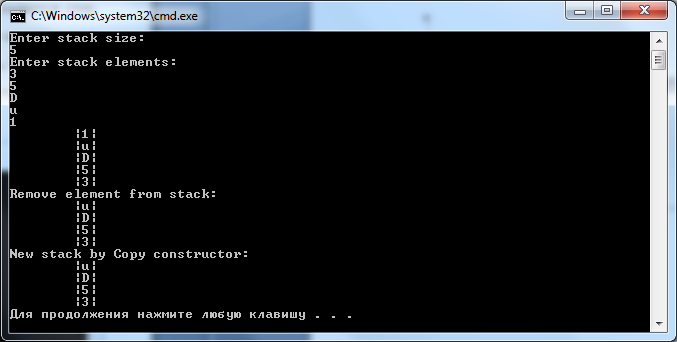
Пользователю предлагается ввести максимальный размер создаваемого стека. Если ввести отрицательное число, бросится исключение



При положительном значении создастся стек заданного размера для хранения значений любого типа. Затем пользователю предлагается заполнить все ячейки стека:



После этого происходит печать стека на консоль. Удаляется один элемент. Снова выполняется печать того, что осталось в стеке и создается точно такой же стек с помощью конструктора копирования:



Алгоритм можно повторить несколько раз, используя другие значения.

# 4. Руководство программиста

Разработка системы вычисления проводились в среде “Microsoft Visual Studio 2017”.

В данной работе будет использовано 2 класса:

* Класс «Стек» (TStack), реализованный с использованием массива
* Класс исключения (TExсeption).

## 4.1. Описание структуры программы

Модульная структура программы:

1. tstack.h– модуль с классом TStack, в котором определен интерфейс шаблонного класса Стек и реализация его методов.
2. exсeption.h – модуль с классом исключения TExсeption.

* sample\_tstack.cpp, sample\_performance\_check.cpp– модуль программы тестирования, с которым работает пользователь, в котором проводятся эксперименты.

1. test\_main.cpp, test\_tstack.cpp – модуль с функциями тестирования для созданных классов. Содержат 20 тестов для класса TStack.

## 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.

# 5. Эксперименты

В качестве примера рассмотрим перегрузку оператора присвоения для класса стека (TStack).

Теоретическая сложность выполнения алгоритма O(1).

Мы провели измерение присваивая стеку разное количество элементов: 10, 100, …, 1000000 элементов. Ниже вы можете увидеть график зависимости времени выполнения операции присвоения от количества элементов стека. 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 - стек, 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. Касперски К. Техника оптимизации программ. Эффективное использование памяти. — СПб.: БХВ-Петербург, 2003. — 464 с.: ил.

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

# 8. Приложение