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 №8**

**«Стек на списке»**

**Выполнил:** студент группы 381706-2

Макарихин Семён Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Руководитель:**

Ассистент кафедры МОСТ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2019

Содержание

[1.Введение 3](#_Toc718796)

[2. Цели и задачи 4](#_Toc718797)

[3. Руководство пользователя 5](#_Toc718799)

[4. Руководство программиста 7](#_Toc718800)

[4.1. Описание структуры программы 7](#_Toc718801)

[4.2. Описание функций и процедур, их алгоритмов 7](#_Toc718802)

[5. Эксперименты 1](#_Toc718803)0

[6. Заключение 1](#_Toc718804)1

[7. Литература 1](#_Toc718805)2

[8. Приложения 1](#_Toc718806)3

# 1.Введение

Лабораторная работа направлена на практическое освоение динамической структуры данных Стек на списке.

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.

Стек – это список, у которого доступен один элемент (одна позиция). Этот элемент называется вершиной стека. Взять элемент можно только из вершины стека, добавить элемент можно только в вершину стека. Например, если записаны в стек числа 1, 2, 3, то при последующем извлечении получим 3,2,1.

# 2. Цели и задачи

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

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

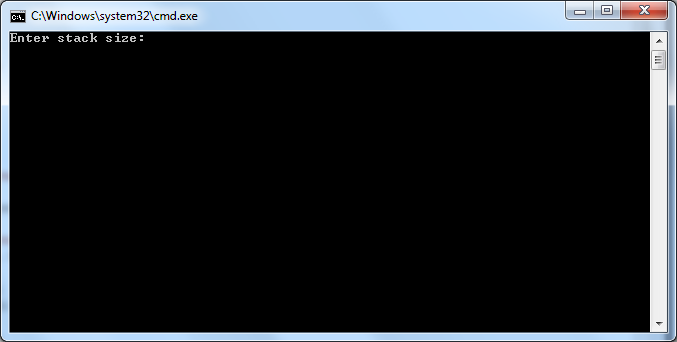
В процессе выполнения лабораторной работы требуется использовать систему контроля версий [Git](https://git-scm.com/book/ru/v2) и фрэймворк для разработки автоматических тестов [Google Test](https://github.com/google/googletest).

Выполнение работы предполагает решение следующих задач:

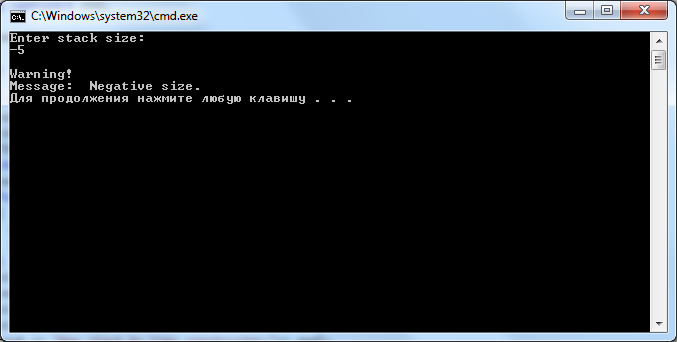
1. Реализация методов шаблонного класса TStackList согласно заданному интерфейсу.
2. Реализация класса для обработки исключений– TException, которые могут возникнуть при выполнении различных операций, согласно заданному интерфейсу.
3. Обеспечение работоспособности тестов и примера использования.
4. Реализация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

# 3. Руководство пользователя

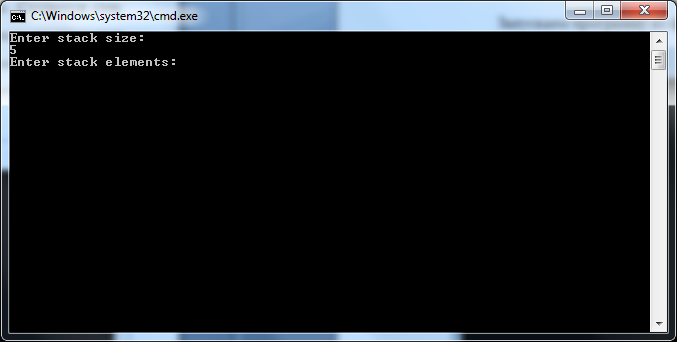
Запускаем программу stacklist.cpp:



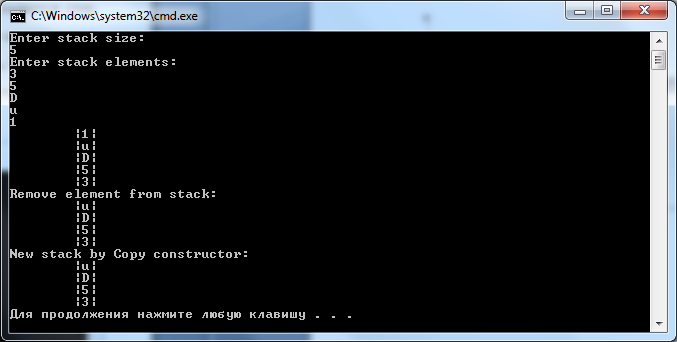
Пользователю предлагается ввести максимальный размер создаваемого стека. Если ввести отрицательное число, бросится исключение:



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 2010”.

В данной работе будет использовано 4 класса:

* Класс «Элемент» (TElement) – реализует узел списка.
* Класс «Список» (TList), реализованный с использованием указателей.
* Класс «Стек» (TStackList), реализованный на основе класса TList.
* Класс исключения (TExсeption).

## 4.1. Описание структуры программы

Модульная структура программы:

1. telement.h– модуль с классом TElement, в котором определен интерфейс шаблонного класса Элемент и реализация его методов.
2. tlist.h– модуль с классом TList, в котором определен интерфейс шаблонного класса Список и реализация его методов.
3. tstacklist.h– модуль с классом TStackList, в котором определен интерфейс шаблонного класса Стек и реализация его методов.
4. exсeption.h – модуль с классом исключения TExсeption.

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

1. test\_main.cpp, test\_tstacklist.cpp – модуль с функциями тестирования для созданных классов. Содержат 18 тестов для класса TStackList.

## 4.2. Описание функций и процедур, их алгоритмов

Рассмотрим реализацию методов шаблонного класса TElement:

template <class ElementType>

class TElement

1. TElement(ElementType \_elem = 0, TElement<ElementType>\* \_next = 0) - конструктор класса с параметрами, принимающий значение элемента и указатель на следующий элемент. По умолчанию значения установятся в ноль .

2. TElement(TElement<ElementType> &Elem) - конструктор копирования.

Принимает ссылку на объект класса TElement.

3. ElementType GetElem() - метод класса, позволяющий получить значение элемента.

4. TElement\* GetNext() – метод класса, позволяющий получить указатель на следующий элемент.

5. void Set(ElementType \_elem) - метод класса, позволяющий установить значение элемента.

6. void SetNext(TElement<ElementType>\* next) - метод класса, позволяющий установить указатель.

Рассмотрим реализацию методов шаблонного класса TList:

template <class ListType>

class TList

1. TList() – конструктор класса по умолчанию.

2. TList(TList<ListType> &List) - конструктор копирования.

Принимает ссылку на объект класса TList.

3. ~TList() - деструктор. Освобождает выделенную память.

4. void Put(int \_n, ListType elem) - метод, позволяющий добавить новый элемент в список на определенную позицию.

5. void PutStart(ListType List) - метод, позволяющий добавить новый элемент в начало списка.

6. void PutEnd(ListType List) - метод, позволяющий добавить новый элемент в конец списка.

7. ListType Get(int \_n) - метод изъятия элемента на определенной позиции из списка с удалением.

8. ListType GetStart() - метод изъятия элемента из начала списка с удалением.

9. ListType GetEnd() - метод изъятия элемента с конца списка с удалением.

10. bool IsFull() - метод проверки списка на полноту.

11. bool IsEmpty() - метод проверки списка на пустоту.

12. void PrintList() - метод отображения текущих элементов списка.

Рассмотрим реализацию методов шаблонного класса TStackList:

template <class StackListType>

class TStackList : public TList<StackListType>

1. TStackList<StackListType>(int \_size = 10) - конструктор класса, принимающий размер стека. По умолчанию создается стек размера 10 с позицией вершины стека 0.

2. TStackList<StackListType>(TStackList<StackListType> &A) - конструктор копирования.

Принимает ссылку на объект класса TStackList.

3. ~TStackList() - деструктор. Освобождает выделенную память.

4. void Put(StackListType A) - метод, позволяющий добавить новый элемент в стек.

4. StackListType Get() - метод изъятия элемента из вершины стека с удалением.

5. int GetMaxSize() - возвращает максимальный размер стека.

6. int GetSize() - возвращает размер стека.

7. bool IsEmpty() - метод проверки стека на пустоту.

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

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

# 5. Эксперименты

В качестве примера рассмотрим операции добавления/изъятия элемента для класса стека (TStackList).

Теоретическая сложность выполнения алгоритмов O(1).

Мы провели измерение скорости добавления/изъятия элемента, при разном количестве элементов в стеке: 10, 100, …, 1000000 элементов. Ниже вы можете увидеть таблицу зависимости времени выполнения операции от количества элементов списка.

По приведенным данным можно сделать вывод, что практическая сложность выполнения алгоритмов равна теоретической.

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | метод Put () | метод Get() |
| 10 | 0 | 0 |
| 100 | 0 | 0 |
| 1000 | 0 | 0 |
| 10000 | 0 | 0 |
| 100000 | 0 | 0 |

# 6. Заключение

В результате лабораторной работы была разработана структура данных – стек на списке, 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://www.intuit.ru/studies/courses/648/504/lecture/11457>
3. <https://neerc.ifmo.ru/wiki/index.php?title=%D0%A1%D1%82%D0%B5%D0%BA>

# 8. Приложение