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 №7**

**«Список на массивах»**

**Выполнил:** студент группы 381706-2

Макарихин Семён Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Руководитель:**

Ассистент кафедры МОСТ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2019

Содержание

[1.Введение 3](#_Toc536814977)

[2. Цели и задачи 5](#_Toc536814978)

[3. Руководство пользователя 5](#_Toc536814980)

[4. Руководство программиста 1](#_Toc536814981)1

[4.1. Описание структуры программы 1](#_Toc536814982)1

[4.2. Описание функций и процедур, их алгоритмов 1](#_Toc536814983)1

[5. Эксперименты 1](#_Toc536814984)3

[6. Заключение 1](#_Toc536814985)4

[7. Литература 1](#_Toc536814986)5

[8. Приложение 1](#_Toc536814987)6

# 1.Введение

Списки являются чрезвычайно гибкой структурой, так как их легко сделать большими или меньшими, и их элементы доступны для вставки или удаления в любой позиции списка. Списки также можно объединять или разбивать на меньшие списки. Списки регулярно используются в приложениях, например в программах информационного поиска, трансляторах программных языков или при моделировании различных процессов.

В математике список представляет собой последовательность элементов определенного типа ( elementtype ). Представим список в виде последовательности элементов, разделенных запятыми: a1, а2, ..., аn, где n≥ 0 и всё ai имеют тип elementtype. Количество элементов n - длина списка. Если n ≥ 1, то а1 называется первым элементом, а аn—последним элементом списка. В случае n = 0 имеем пустой список, который не содержит элементов.

Важное свойство списка заключается в том, что его элементы можно линейно упорядочить в соответствии с их позицией в списке. Говорим, что элемент ai предшествует ai+1 для і = 1, 2, ..., n — 1 и ai следует за ai-1 для і =2, 3, .., n. Также будем говорить, что элемент аi имеет позицию і.

# 2. Цели и задачи

В рамках лабораторной работы ставится задача эффективной реализации структуры данных – линейный односвязный список на массивах и выполнение основных операций над ним:

* добавления элемента в начало списка,
* добавления элемента в конец списка,
* добавления, извлечения с удалением промежуточного элемента списка,
* извлечения с удалением элемента из начала списка,
* извлечения с удалением элемента из конца списка,
* проверка списка на полноту/пустоту.

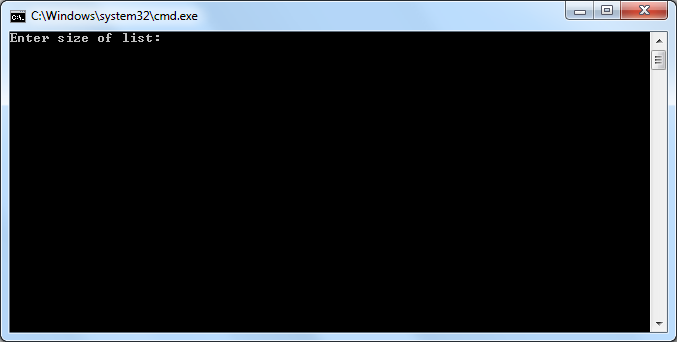
В процессе выполнения лабораторной работы требуется использовать систему контроля версий [Git](https://git-scm.com/book/ru/v2) и фрэймворк для разработки автоматических тестов [Google Test](https://github.com/google/googletest).

Выполнение работы предполагает решение следующих задач:

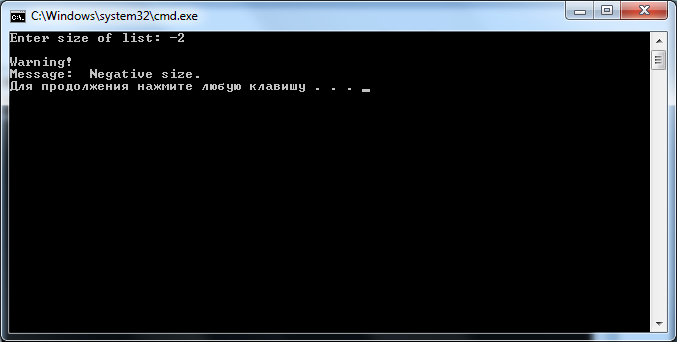
1. Реализация методов шаблонного класса TArrList согласно заданному интерфейсу.
2. Реализация класса для обработки исключений– TException, которые могут возникнуть при выполнении различных операций, согласно заданному интерфейсу.
3. Обеспечение работоспособности тестов и примера использования.
4. Реализация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

# 3. Руководство пользователя

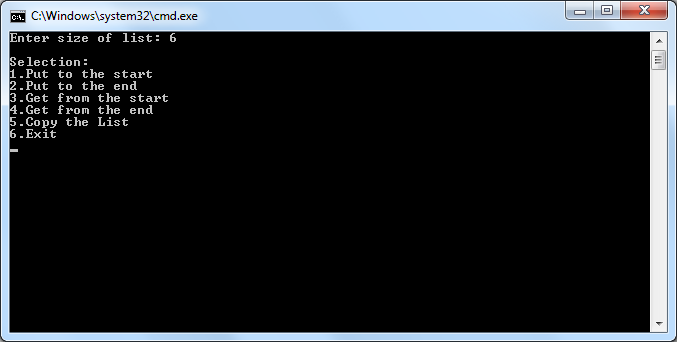
Запускаем программу из файла arrlist.cpp.



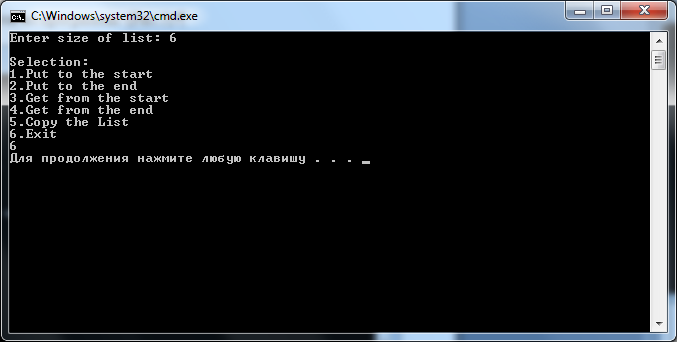
Пользователю предлагается ввести максимальный размер создаваемого списка. Если ввести отрицательное число, бросится исключение:



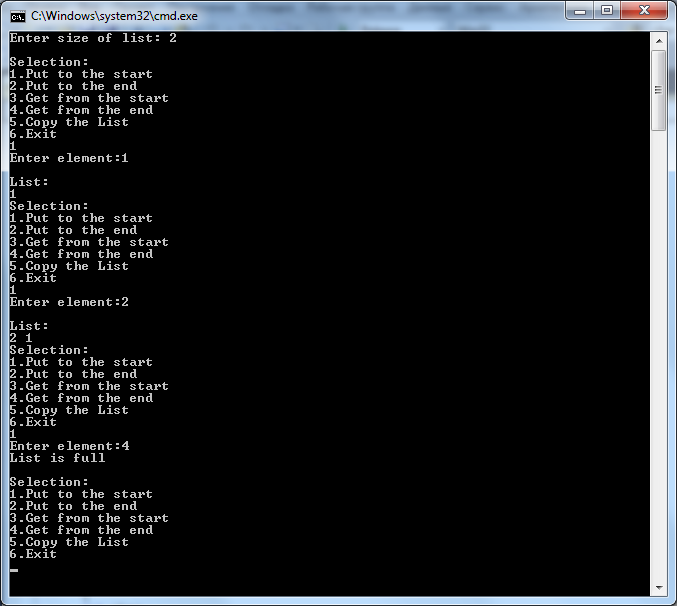
При положительном значении создастся список заданного размера с пустыми ячейками, и пользователю предоставляется выбор операций со списком.

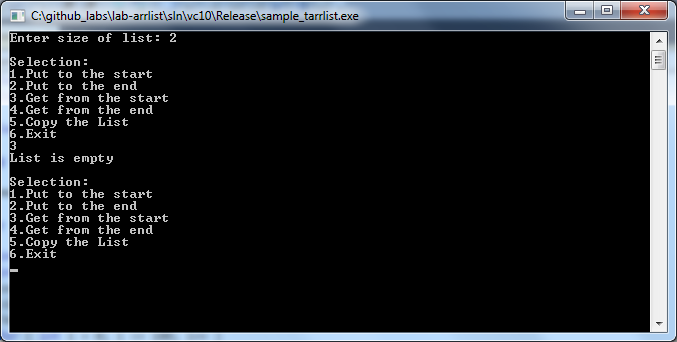


Выбрав ‘Exit’, можно выйти из программы.

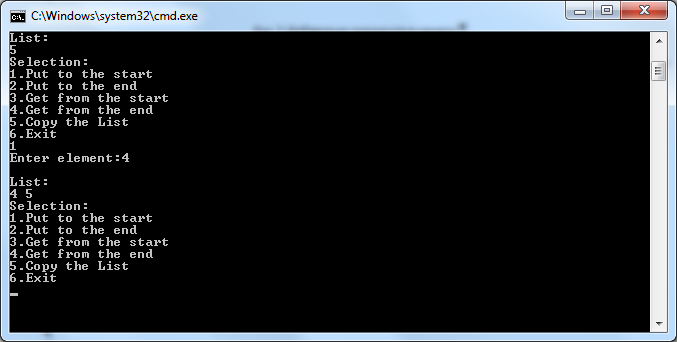


Если удалять из пустого списка или добавлять в полный, программа выведет ошибку и не выполнит операцию.

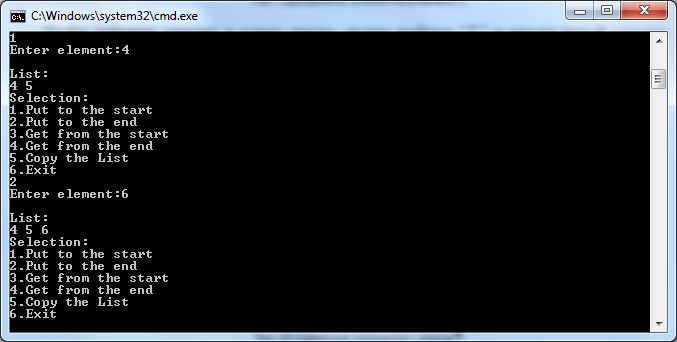




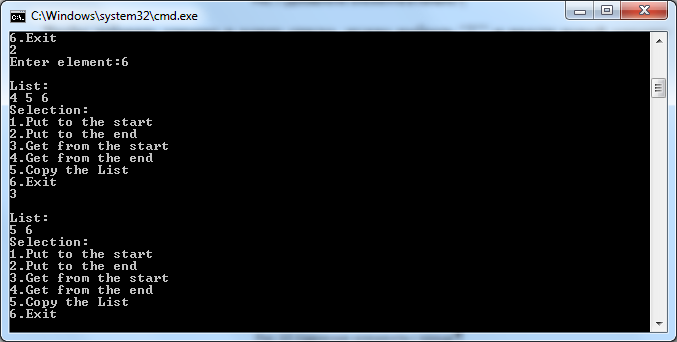
Чтобы добавить элемент в начало списка, нужно выбрать “1” и ввести новый элемент.



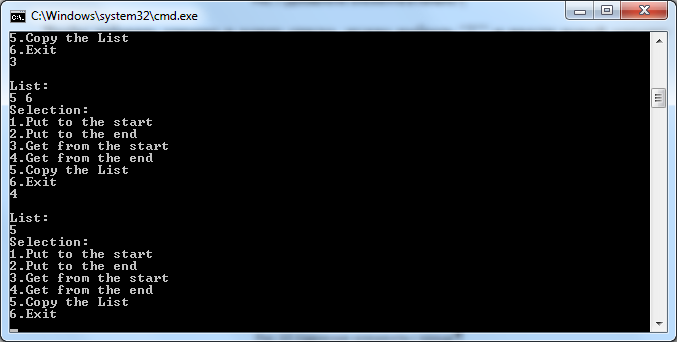
Чтобы добавить элемент в конец списка, нужно выбрать “2”” и ввести новый элемент.



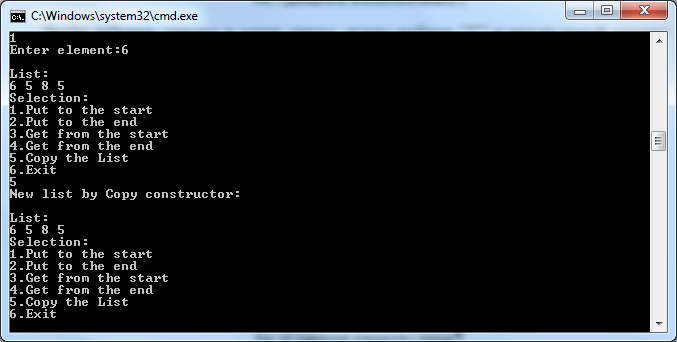
Чтобы удалить элемент из начало списка, нужно выбрать “3”.



Чтобы удалить элемент с конца списка, нужно выбрать “4” .



Чтобы скопировать список, нужно выбрать “5”.



Алгоритм можно повторять много раз, не выходя из программы.

# 4. Руководство программиста

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

В данной работе будет использовано 4 класса:

* Класс «Список» (TArrList), реализованный с использованием массивов и использующий класс TQueue.
* Класс «Стек» (TStack), реализованный с использованием массива.
* Класс «Очередь» (TQueue), построенного на основе класса TStack.
* Класс исключения (TExсeption).

## 4.1. Описание структуры программы

Модульная структура программы:

1. tartlist.h– модуль с классом TArrList, в котором определен интерфейс шаблонного класса Список и реализация его методов.
2. tstack.h– модуль с классом TStack, в котором определен интерфейс шаблонного класса Стек и реализация его методов.
3. tqueue.h– модуль с классом TQueue, в котором определен интерфейс шаблонного класса Очередь и реализация его методов.
4. exсeption.h – модуль с классом исключения TExсeption.
5. sample\_tattlist.cpp, sample\_performance\_check.cpp– модуль программы тестирования, с которым работает пользователь, в котором проводятся эксперименты.
6. test\_main.cpp, test\_tarrlist.cpp – модуль с функциями тестирования для созданных классов. Содержит 24 теста для класса TArrList.

## 4.2. Описание функций и процедур, их алгоритмов

Рассмотрим реализацию методов шаблонного класса TArrList:

template <class ArrListType>

class TArrList

1. TArrList(int \_size = 10) – конструктор класса, принимающий размер списка. По умолчанию создается список размера 10.

2. TArrList(TArrList<ArrListType> &List) - конструктор копирования.

Принимает ссылку на объект класса TArrList.

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

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

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

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

7. ArrListType Get(int n) - метод изъятия элемента на определенной позиции из списка с удалением.

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

9. ArrListType 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 Print() - метод отображения текущих элементов списка.

# 5. Эксперименты

В качестве примера рассмотрим операцию добавления элемента в конец и в начало списка для класса TArrList.

Теоретическая сложность выполнения алгоритмов O(1).

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

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

|  |  |  |
| --- | --- | --- |
| Кол-во элементов | метод PutEnd() | метод PutStart() |
| 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 – односвязный линейный список на массивах, 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. Д. Кнут. Искусство программирования. (3-е издание) Т.1.

# 8. Приложение