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**

**Отчет по учебной практике**

**«Разработка структуры хранения данных: список на массивах»**

**Выполнил:** студент группы 381706-1

Нечаева Екатерина Владимировна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc560926)

[2. Постановка задачи 4](#_Toc560927)

[3. Руководство пользователя 5](#_Toc560928)

[4. Руководство программиста 7](#_Toc560929)

[4.1 Описание структуры программы 7](#_Toc560930)

[4.2 Описание структур данных 7](#_Toc560931)

[4.3 Описание алгоритмов 8](#_Toc560932)

[5. Заключение 10](#_Toc560933)

[6. Литература 11](#_Toc560934)

# Введение

Списком называется упорядоченное множество, состоящее из переменного числа элементов, к которым применимы операции включения, исключения. Список, отражающий отношения соседства между элементами, называется линейным.

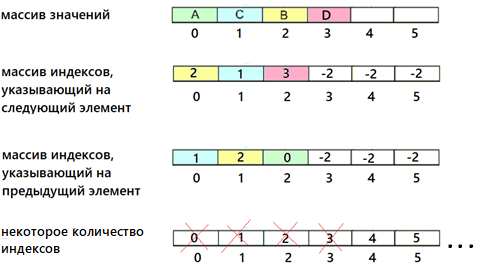
Различают два способа организации списков:

* На массиве
* На указателях

Главное отличие реализации на массиве является то, что память в данном случае ограничена размером массива. Также при реализации списков с помощью массивов элементы списка располагаются в смежных ячейках массива. Это представление позволяет легко просматривать содержимое списка и вставлять новые элементы в его конец.

Для работы со списком требуется два массива индексов, которые отвечают за перемещение по списку, и переменная, отвечающая за свободный индекс в массиве значений. Также, для удобства, нужны две переменные, которые отслеживают начало и конец списка.

Пример:



В данном случае, начало списка начинается в переменной А, конец- D, и передвигаясь по второму массиву, можно сделать вывод, что список выглядит так: A, B, C, D. При этом в массиве значений остались свободные ячейки, значит список можно дополнить какими-нибудь значениями.

# Постановка задачи

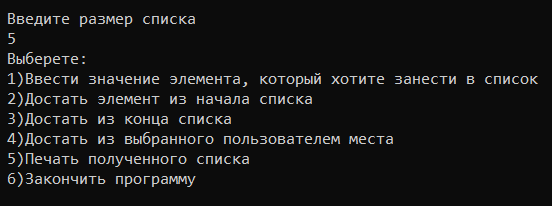
В рамках лабораторной работы ставится задача реализации структуры данных, представляющей собой список на основе массива.

Для этого были реализованы методы:

* добавление в начало списка,
* добавление в конец списка,
* добавление в произвольную позицию списка,
* получить значение элемента в произвольную позицию списка с последующим удалением,
* получить значение элемента из начала списка с последующим удалением
* получить значение элемента из конца списка с последующим удалением
* проверка списка на полноту/пустоту.
* конструкторы: инициализации, копирования, деструктор.

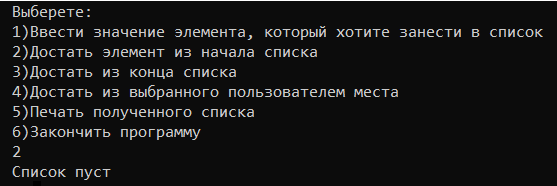
# Руководство пользователя

Программа начинает свою работу с ввода на консоль размера списка. После чего выводит предложенные функции, которые будут представлены ниже на рисунке 1.

**

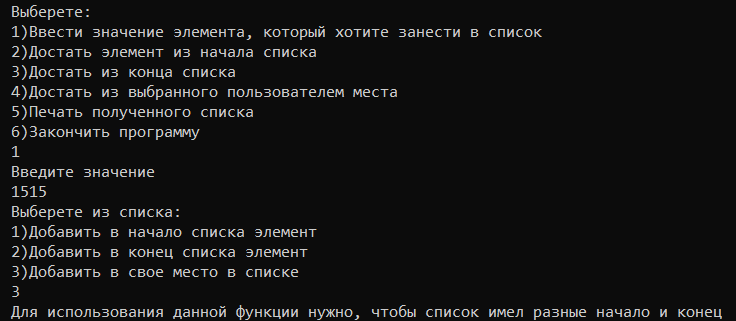
*Рисунок 1. Интерфейс программы в начале работы.*

Так как изначально список пустой, то при выборе пунктов 2, 3, 4 и 5 будет выведена надпись о том, что список пустой.

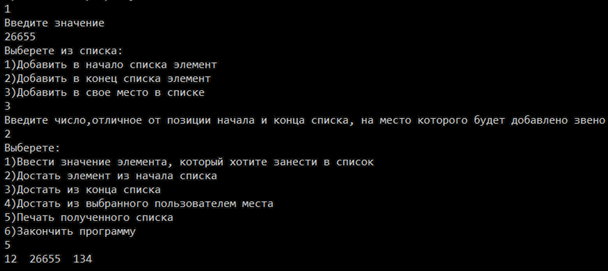


*Рисунок 2. Пример вывода исключения.*

Таким образом, пользователю нужно создать звено списка, для этого нужно выбрать 1 пункт программы, тем самым ввести значение. На втором этапе пользователю предлагается выбрать место, куда записать элемент (Рисунок 2). Если пользователь выберет сначала 3 пункт, программа выдаст об ошибке и попросит выбрать другой пункт.

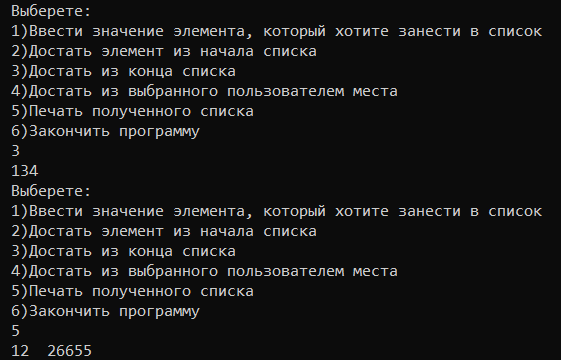


*Рисунок 3. Пример ввода значения в список.*

После некоторых итераций с 1 пунктом, можно вывести полученный список.

*Рисунок 4. Пример вывода списка.*

Также пользователь может получить значение любого звена из списка, при этом удалив его. Так на рисунке 4 показан пример удаление звена и вывод измененного списка.



*Рисунок 4. Пример использования.*

После выполнения всех действий, 6 пункт завершает работу программы*.*

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль arrlist. Содержит пример использования списка (mainarrlist.cpp)
* Модуль libarrlist – статическая библиотека. Содержит заголовочный файл arrlist.h, в котором реализован шаблонный класс TArrList.
* Модуль arrlistTest. Содержит тесты, проверяющие правильность работы методов (arrlistTest.cpp)
* Модуль libthrows – библиотека исключений. Содержит заголовочный файл throws.h, в котором он реализован.

## Описание структур данных

#### Класс TException – класс исключений.

Содержит одно защищенное поле *string str,* переменная строкового типа, которая хранит сообщение об исключении.

Содержит публичные методы

* TException(std::string \_str); //конструктор-инициализатор
* void Print(); //вывод данного исключения на консоль

#### Класс TArrList.

Защищенные поля:

* T\* mas; //массив значений
* int size; //размер списка
* int count; //количество элементов в списке
* int start; //индекс первого элемента списка
* int finish; //индекс последнего элемента списка
* int \*next\_ind; //массив индексов, указывающих на следюущий элемент списка
* int \*prev\_ind; //массив индексов, указывающих на предыдущий элемент списка
* TQueue<int> elem; //очередь индексов для заполнения пустых ячеек в массиве

Содержит публичные методы:

* TArrList(int \_size = 3); //Конструктор инициализатор
* TArrList(TArrList<T> &A); //Конструктор копирования
* ~TArrList(); //Деструктор
* void PutBegin(const T &a); //Положить в начало списка
* void PutEnd(const T &a); //Положить в конец списка
* void PutTek(const T &a, int n); //Положить в любое место списка
* T GetBegin(); //Забрать из начала списка с последующим удалением
* T GetEnd(); //Забрать из конца списка с последующим удалением
* T GetTek(int n); //Забрать из любого места списка с последущим удалением
* bool IsFull(); //Проверка на полноту
* bool IsEmpty(); //Проверка на пустоту
* void PrintArr(); //Печать списка

## Описание алгоритмов

## Добавление элемента в начало и в конец списка.

Для того, чтобы добавить в начало списка, делается проверка на полноту, если список полон, бросается исключение. Иначе, из очереди свободных индексов достается один, и по этому индексу в массив значений добавляется элемент. Дальше, нужно присвоить старт новому индексу. Если список изначально был пуст, то начало и конец списка равны этому элементу, иначе новому элементу в массиве индексов, указывающих на следующий элемент, присваивается индекс на старое начало, а старому старту указываем индекс на предыдущий элемент, то есть на новый добавленный элемент. После присваиваний индексов, присваиваем старту новый элемент и увеличиваем число занятых ячеек на 1.

Для того, чтобы добавить в конец списка, делается проверка на полноту, если список полон, бросается исключение. Иначе, из очереди свободных индексов достается один, и по этому индексу в массив значений добавляется элемент. Дальше, нужно присвоить конец новому индексу. Если список изначально был пуст, то начало и конец списка равны этому элементу, иначе новому элементу в массиве индексов, указывающих на предыдущий элемент, присваивается индекс на старый конец, а старому концу указываем индекс на следующий элемент (в массиве индексов, указывающий на следующий элемент), то есть на новый добавленный элемент. После присваиваний индексов, присваиваем концу новый элемент и увеличиваем число занятых ячеек на 1.

**Извлечение элемента из начала и из конца списка с последующим удалением.**

Чтобы извлечь элемент из начала списка, сначала делается проверка на пустоту, если это так, то бросается исключение. Иначе, создается переменная, которая равна массиву значений с индексом начала списка. После чего, создается еще одна переменная, которая равна индексу следующего элемента, указываем, что предыдущего элемента нет, приравнивая к -2 и новому старту присваиваем эту переменную. Уменьшаем число элементов на 1.

Чтобы извлечь элемент из конца списка, сначала делается проверка на пустоту, если это так, то бросается исключение. Иначе, создается переменная, которая равна массиву значений с индексом конца списка. После чего, создается еще одна переменная, которая равна индексу предыдущего элемента, указываем, что следующего элемента нет, приравнивая к -2 и новому финишу присваиваем эту переменную. Уменьшаем число элементов на 1.

# Заключение

В ходе выполнения данной лабораторной была реализована структура хранения данных с помощью списка, основанного на базе массивов. Разобрали отличия между реализациями разных списков, разобрали методы, позволяющие выполнять действия над списком, а именно: добавление в любое место списка и извлечение с последующим удалением.

Были разработаны тесты, проверяющие корректную работу методов данных классов, с помощью Google C++ Testing Framework.

# Литература

1. Реализация списка на базе массивов [Электронный ресурс]

<http://rsdn.org/article/alg/list.xml>

1. Другие варианты списков [Электронный ресурс]

<http://www.intuit.ru/studies/courses/683/539/lecture/12151?page=3>

1. Списки [Электронный ресурс]

<https://studopedia.ru/14_1558_spiski-realizatsiya-spiskov-na-baze-massiva.html>