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**

**«Односвязный список с использованием массивов»**

**Выполнил:** студент группы 381706-1

Суслов Егор Игоревич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533083472)

[2. Постановка задачи 4](#_Toc533083473)

[3. Руководство пользователя 5](#_Toc533083474)

[4. Руководство программиста 6](#_Toc533083475)

[4.1. Описание структуры программы 6](#_Toc533083476)

[4.2. Описание структур данных 6](#_Toc533083477)

[4.3. Описание алгоритмов 7](#_Toc533083478)

[5. Заключение 8](#_Toc533083479)

[6. Литература 9](#_Toc533083480)

# Введение

**Односвязный список на массивах** – это структура данных, представляющая собой упорядоченный набор значений, в котором некоторое значение может встречаться более одного раза.

Для создания такого списка используется два массива одинакового размера: содержательный массив узлов и массив индексов определяющий порядок следования узлов. Эти массивы связанны друг с другом по индексу. По фиксированному значению содержательный массив узлов хранит значение элемента списка (узла), а массив индексов, содержит индекс следующего элемента списка в содержательном массиве.

Если список заполнен не полностью, то для пустых ячеек в содержательном массиве, в соответствующих ячейках в массиве индексов лежит значение «-2». Для последнего элемента списка, в соответствующей ячейке массива индексов лежит значение -1.

Как правило, стоит дополнительно хранить индекс первого элемента списка в содержательном массиве.

# Постановка задачи

В рамках лабораторной работы ставится задача эффективной реализации структуры данных – линейного односвязного списка на массивах.

Для работы со списком необходимо реализовать операции:

* 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.

Программное решение будет выглядеть следующим образом:

1. Класс списка – TArrList.
2. Класс для обработки исключений – MyException, которые могут возникнуть при выполнении различных операций.
3. Программа, демонстрирующая работу класса TArrList (получается из файла кода «ArrList\_main.cpp».
4. Набор автоматических тестов с использованием Google C++ Testing Framework.

# Руководство пользователя

При запуске консольного приложения «ArrList\_main.cpp» пользователю выводятся на консоль сначала список заполненный с одной стороны 5 элементами добавлением в начало и аналогично 5 элементами добавлением в конец. Итого список состоит из 10 элементов. Затем дважды выводятся на консоль и соответственно из списка первый и последний элементы. Затем выводится список уже без этих элементов (т.е. из 6 элементов).

# Руководство программиста

## Описание структуры программы

Программа состоит из следующих модулей:

Программа состоит из следующих модулей:

* Модуль ArrList. Содержит пример использования списка на массивах. Реализация в файле *main\_ArrList.cpp.*
* Модуль ArrListLib – статическая библиотека. Содержит файл ArrList.h, в котором описан интерфейс и реализация шаблонного класса *TArrList*.
* Модуль ArrListTest. Содержит 23 теста, описанных в файле *ArrListTest.cpp* и разработанных с помощью использования Google C++ Testing Framework.
* Модуль ExceptionLib – библиотека, позволяющая создавать собственные исключения.

## Описание структур данных

#### Класс TArrList – класс списка на массивах.

Класс TArrList является шаблонным. В классе всего 8 полей, объявленных со спецификатором доступа private:

* T\* mas - указатель на область памяти для хранения элементов списка.
* int \*nextInd - указатель на область памяти для хранения индексов, указывающих на следующий элемент списка.
* int \*predInd - указатель на область памяти для хранения индексов, указывающих на предыдущий элемент списка.
* int size - максимальный размер списка.
* int count - текущее количество элементов в списке.
* int start - индекс первого элемента списка.
* int end - индекс последнего элемента списка.
* TQueue <int> freeElem - Очередь свободный ячеек в массиве mas.

**Конструкторы и методы класса, объявленные со спецификатором public:**

* TArrList(int \_size = 10) - конструктор по умолчанию.
* TArrList(TArrList<T> &A) - конструктор копирования.
* ~ TArrList() – деструктор.
* void PutStart(T elem) - метод, позволяющий добавить элемент в начало списка.
* void PutEnd(T elem) - метод, позволяющий добавить элемент в конец списка.
* void Put(int n, T elem) - добавить промежуточный элемент на позицию n в списке
* T Get(int n) - извлечь из списка элемент на позиции n.
* T GetStart() - метод, позволяющий получить с удалением элемент из начало списка.
* T GetEnd() - метод, позволяющий получить с удалением элемент из конца списка.
* bool IsFull() - проверка списка на полноту.
* bool IsEmpty() - проверка списка на пустоту.
* void Print() - вывод элементов списка на консоль.

## Описание алгоритмов

**Добавление звена списка в начало и в конец.**

Сначала проверяем не заполнен ли список. Если он заполнен, то бросаем исключение. Если нет, то в очереди свободных позиций freeElem, берем первую свободную ячейку i. По полученному индексу в массив mas записываем значение, которое хотим положить в список. Определяем, что следующим для этого элемента, будет элемент с текущим индексом start, то есть nextInd[i] = start. Если, перед добавлением, список не был пуст, то предыдущим для первого элемента списка делаем только что добавленный элемент, то есть: predInd[start] = i. Если же список был пуст, то определяем, что добавленный элемент является и последним элементом в списке. Затем индекс start переопределяем на только что добавленный элемент: start = i. Увеличиваем количество элементов в списке count++.

Для добавления элемента в конец списка, рассуждения аналогичны, с поправкой на то, что добавляем в конец списка.

# Заключение

Все поставленные задачи были выполнены это подтверждается Google C++ Testing Framework (все 25 тестов на класс TArrList успешно выполняются) и успешным выполнением и работой с файлом кода «ArrList\_main.cpp» использующего наш класс.

# Литература

* Книги:

1. A.O. Грудзинский. Методы программирования, Издательство Нижегородского госуниверситета, 2006.
2. Топп У., Форд У. Структуры данных в С++. - М. Бином, 1999.
3. Мейн М., Савитч У. Структуры данных и другие объекты в С++. - М.: Издательский дом “Вильямс”, 2003.
4. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.

* Ссылки в Internet:

1. Учебно-методическое пособие из электронной библиотеки ННГУ: «ЛАБОРАТОРНЫЙ ПРАКТИКУМ» : [http://www.unn.ru/books/met\_files/Pract\_ADS.pdf].