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**

**«Структура хранения данных:**

**Односвязный список с использованием массивов»**

**Выполнил:** студент группы 381706-1

Остапович Денис Евгеньевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533097873)

[2. Постановка задачи 4](#_Toc533097874)

[3. Руководство пользователя 5](#_Toc533097875)

[4. Руководство программиста 6](#_Toc533097876)

[4.1 Описание структур данных 6](#_Toc533097877)

[4.2 Описание структур данных 6](#_Toc533097878)

[4.3 Описание алгоритмов 7](#_Toc533097879)

[5. Заключение 9](#_Toc533097880)

[6. Литература 10](#_Toc533097881)

# Введение

Односвязный список на массивах — это абстрактный тип данных, представляющий собой упорядоченный набор значений, в котором некоторое значение может встречаться более одного раза. Экземпляр списка является компьютерной реализацией математического понятия конечной последовательности. Экземпляры значений, находящихся в списке, называются элементами; если значение встречается несколько раз, каждое вхождение считается отдельным элементом.

Реализуется данная структура хранения данных на базе двух массивов одинакового размера. В одном массиве хранятся элементы списка, а в другом – индекс следующего элемента списка. Если элемент списка пустой, то ему присваивается индекс «-2». Если же элемент списка является последним, то ему присваивается индекс «-1».

**Цель лабораторной работы – практическое освоение структуры данных: линейного односвязного списка на массивах.**

# Постановка задачи

Необходимо разработать библиотеку для хранения и взаимодействия с такой структурой данных, как линейный список с использованием массивов.

Для этого нужно:

* Описать класс списка с использованием массивов TArrayList.
* Продемонстрировать работу класса TArrayList.
* Создать набор автоматических тестов с использованием Google C++ Testing Framework.

# Руководство пользователя

Рассмотрим пример использования класса TArrayList.

После запуска программы пользователю предлагается заполнить два списка. Один с начала, а другой с конца. Далее получившиеся списки выводятся на экран. После этого происходит поэлементное извлечение элементов из списка с выводом их на экран. Программа завершается.

# Руководство программиста

## Описание структуры программы

Программа состоит из следующих модулей:

* Модуль ListOnArray. – cодержит пример использования списка на массивах. Реализация в файле *main.cpp.*
* Модуль ListOnArrayLib – статическая библиотека. Содержит файл listonarray.h, в котором описан интерфейс и реализация шаблонного класса *TArrayList*.
* Модуль ListOnArrayTest. Содержит 23 теста, описанных в файле *ListOnArray.cpp* и разработанных с помощью использования Google C++ Testing Framework.

## Описание структур данных

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

Класс TArrayList 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 \*prevInd* - 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:**

* *TArrayList* *(int \_size = 10)* - конструктор по умолчанию.
* *TArrayList* *(TArrList<T> &A)* - конструктор копирования.
* *~ TArrayList* *()* – деструктор.
* *void PushFront(T elem)* – метод, позволяющий добавить элемент в начало списка.
* *void PushBack(T elem)* – метод, позволяющий добавить элемент в конец списка.
* *void Push(int n, T elem)* - добавить промежуточный элемент на позицию n в списке
* *T Pop(int n)* - извлечь из списка элемент на позиции n.
* *T PopFront()* – метод, позволяющий получить с удалением элемент из начало списка.
* *T PopBack()* – 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 в 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: *prevInd[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.

**Удаление звена списка из начала/конца.**

Сначала проверяем не пуст ли список. Если он пуст, то бросаем исключение. Если нет, то в переменную *elem* записываем элемент в начале списка *mas[start].* Затем переопределяем индекс первого элемента списка, он будет равен индексу элемента следующим за первым: *newstart = nextInd[start].* Затем в очередь свободных позиций *freeElem*, добавляем освободившуюся после изъятия первого элемента свободную ячейку *start*. Идентифицируем пустыми соответствующие ячейки массивов *nextInd* и *prevInd*: *nextInd[start] = prevInd[start]* = -2. Если после извлечения первого элемента, список не пуст, то предыдущего элемента для нового первого элемента не существует: *predInd[newstart] = -1*.Затем индекс *start* переопределяем на *newstart*: *start = newstart*. Уменьшаем количество элементов в списке *count--*.Извлечение из конца списка производится аналогично.

# Заключение

В ходе выполнения лабораторной был произведен анализ задачи - установлено понятие списка на массивах, разработана библиотека, реализующая шаблонный класс TArrayList. Также были разработаны тесты, проверяющие корректность работы методов данной библиотеки.

# Литература

* Книги

1. A.O. Грудзинский. Методы программирования, Издательство Нижегородского госуниверситета, 2006.
2. Bjarne Stroustrup The C++ programming language

* Ссылки в Internet

1. Википедия: свободная электронная энциклопедия: на русском языке: [https://ru.wikipedia.org/wiki/Список\_(информатика)]
2. Гергель В.П. Методические материалы по курсу «Методы программирования 2»: [ http://www.itmm.unn.ru/files/2018/11/Primer-1.7.-Razrabotka-obshhego-predstavleniya-linejnogo-spiska.pdf], 2015.