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

Шашкин Евгений Вадимович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

**Оглавление**

[1. Введение. 2](#_Toc1080975)

[2. Постановка задачи. 3](#_Toc1080976)

[3. Руководство пользователя. 4](#_Toc1080977)

[4. Руководство программиста. 5](#_Toc1080978)

[4.1. Описание структуры программы. 5](#_Toc1080979)

[4.2. Описание структур данных. 5](#_Toc1080980)

[4.3. Описание алгоритмов. 6](#_Toc1080981)

[5. Заключение. 7](#_Toc1080982)

[6. Литература. 8](#_Toc1080983)

# Введение.

**Список с использование массивов** — 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 элемента списка пустой, то ему присваивается индекс «-2». Если же элемент списка является последним, то ему присваивается индекс «-1».

# Постановка задачи.

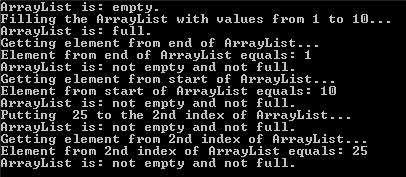
В данной лабораторной работе нужно разработать эффективную библиотеку для хранения и работы с такой структурой данных, как линейный список с использованием массивов.

Для этого нам нужно:

* Описать и реализовать класс списка с использованием массивов **TArrayList**.
* Протестировать класс **TArrayList** с помощью Google Test.
* Реализовать класс **TMyException** для обработки исключений, которые могут возникнуть в результате работы класса **TArrayList**.
* Продемонстрировать работу класса **TArrayList**.

# Руководство пользователя.

Пример работы класса **TArrayList**:

****

*Рис 1. Пример работы программы*

Программа работает следующим образом:

* Создается список.
* Проверяем список на пустоту.
* Заполняем список значениями от 1 до 10.
* Проверяем список на полноту.
* Забираем элемент из конца списка.
* Проверяем список на полноту и пустоту.
* Забираем элемент из начала списка.
* Проверяем список на полноту и пустоту.
* Помещаем в элемент списка с индексом 2 значение 25.
* Проверяем список на полноту и пустоту.
* Забираем значение из элемента списка с индексом 2.

# Руководство программиста.

# Описание структуры программы.

Программа состоит из модулей:

* **ArrList** – содержит в себе файл **arrlist\_main.cpp** с реализацией примера использования класса **TArrayList.**
* **ArrListLib –** содержит в себе файл **ArrList.h**, в котором описан и реализован класс **TArrayList**.
* **ArrListTest** – содержит в себе файл **test\_arrlist.cpp**, в котором находится набор тестов, для проверки работоспособности класса **TArrayList**.
* **MyExceptionLib** –содержитв себе файл **MyException.h** с реализацией класса исключений **TMyException**.

# Описание структур данных.

**Класс TArrayList.**

Класс **TArrayList** является шаблонным классом и содержит восемь полей со спецификатором **protected**:

* **T\* mas** – массив для хранения элементов списка.
* **int\* nextIndex** – массив для хранения индексов, указывающих на следующий элемент списка.
* **int\* prevIndex** - массив для хранения индексов, указывающих на предыдущий элемент списка.
* **int size** – размер списка.
* **int start** – индекс первого элемента списка.
* **int end** – индекс последнего элемента списка.
* **int count** – текущее количество элементов в списке.
* **TQueue<int> fElem** – очередь свободных элементов списка.

Далее идут методы класса со спецификатором доступа **public**:

* **TArrayList(int \_size = 10)** – конструктор инициализатор.
* **TArrayList(TArrayList<T> &obj)** – конструктор копирвания.
* **~TArrayList()** – деструктор.
* **void Put(int \_index, T \_elem)** – метод, помещающий элемент в ячейку списка с указанным индексом.
* **void PutStart(T \_elem)** – метод, помещающий элемент в начало списка.
* **void PutEnd(T \_elem)** - метод, помещающий элемент в конец списка.
* **T Get(int \_index)** – метод, забирающий элемент из ячейки списка с указанным индексом.
* **T GetStart()** – метод, забирающий элемент из начала списка.
* **T GetEnd()** - метод, забирающий элемент из конца списка.
* **bool IsFull()** – метод, проверяющий список на полноту.
* **bool IsEmpty()** – метод, проверяющий список на пустоту.

**Класс TMyException.**

Класс **TMyException** содержит одно поле со спецификатором доступа **private**:

* **string str** – строка, хранящая сообщение об ошибке.

Далее идут методы класса со спецификатором доступа **public**:

* **TMyException(std::string \_str)** – конструктор инициализатор.
* **void what()** – метод вывода ошибки на экран.

# Описание алгоритмов.

**Добавление элемента в начало списка.**

Из очереди свободных элементов списка извлекаем индекс первый свободный элемент и записываем в элемент списка с данным индексом необходимое значение. Затем записываем, что следующим элементом для текущего, будет элемент с индексом **start**, а предыдущим для первого элемента списка делаем добавленный элемент. Увеличиваем количество элементов в списке **count** на 1.

**Удаление элемента списка из начала.**

Сохраняем элемент в начале списка в переменную **elem**. Затем переопределяем индекс первого элемента списка на следующий за ним. После чего добавляем в очередь свободных элементов списка ячейку, освободившуюся после удаления элемента из начала списка и присваиваем ей индекс «-2». Уменьшаем количество элементов в списке **count** на 1.

# Заключение.

В данной лабораторной работе мне удалось реализовать библиотеку для хранения и работы со списком с использованием массивов, а именно:

* Удалось реализовать класс списка **TArrayList**.
* Удалось протестировать методы класса **TArrayList**, а также обеспечить их работоспособность.
* Удалось реализовать класс для обработки исключений **TMyException**.

Таким образом, я смог реализовать структуру данных под названием список на массивах, каждый элемент которого указывает на следующий за ним элемент посредством индекса в массиве индексов элементов списка.

# Литература.

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», 2015.
2. Ссылка из Википедии про списки:

<https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)>