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**

**«Реализация списка на основе указателей»**

**Выполнил:** студент группы 381706-1

Соболева Ю.А.

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

аспирант каф. МОСТ ИИТММ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г.

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533027637)

[2. Постановка задачи 4](#_Toc533027638)

[3. Руководство пользователя 5](#_Toc533027639)

[4. Руководство программиста 6](#_Toc533027640)

4.[1. Описание структуры программы 6](#_Toc533027641)

4.[2. Описание структур данных 6](#_Toc533027642)

4.[3. Описание алгоритмов 7](#_Toc533027643)

[5. Заключение 8](#_Toc533027644)

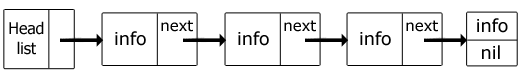
[6. Литература 9](#_Toc533027645)

# **Введение**

**Структура данных** - программная единица, которая определяет метод хранения и обработки различных логически связанных данных в вычислительной технике. Знание структур данных позволяет наиболее компактно и практично расположить данные в памяти компьютера. В данной работе мы рассмотрим такую структуру данных, как **список**.

**Список –** абстрактный тип данных, реализующий упорядоченный набор значений. Списки отличаются от массивов тем, что доступ к их элементам осуществляется последовательно, в то время как массивы – структура данных произвольного доступа. Данный абстрактный тип имеет несколько реализаций в виде структур данных.

**Связный список –** это структура данных, представляющая собой конечное множество упорядоченных элементов, связанных друг с другом посредствам указателей. Каждый элемент структуры содержит поле с какой-либо информацией, а также указатель на следующий элемент.



**Целью данной лабораторной работы** является разработка структуры данных для хранения связных списков с использованием указателей, а также освоение таких инструментов разработки программного обеспечения, как система контроля версий Git и фрэймворк для разработки автоматических тестов Google Test.

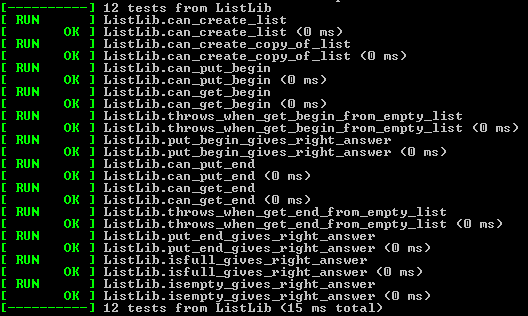
# **Постановка задачи**

Выполнение работы предполагает решение следующих задач:

1. Реализация класса элемента списка TElem.
2. Реализация класса списка TList, использующего класс TElem.
3. Разработка интерфейса для данных классов.
4. Обеспечение работоспособности примера использования.
5. Реализация нескольких тестов на базе Google Test.

# **Руководство пользователя**

При запуске программы на экран выводится сообщение, что все тесты пройдены успешно. Примеры вывода сообщений для класса TList показаны на рис.1 и рис.2 соот ветственно.



# **Руководство программиста**

## **Описание структуры программы**

**Проект “List” состоит из следующих файлов:**

1. mainList.cpp (в нём находится main)

**Проект “ ListLib ” состоит из следующих файлов:**

1. ElemLib.h (описание класса “TElem” реализация методов класса “TElem” и перегрузка операций )
2. ListLib.h (описание класса “TList” реализация методов класса “TList” и перегрузка операций )
3. ListLib.cpp

**Проект “ Test” состоит из следующих файлов:**

1. test\_listlib.cpp (реализация тестов для класса TList)

## **Описание структур данных**

***Класс TElem:***

*Поля:*

* T d; - значение элемента;
* TElem<T>\* next; -указатель на следующий элемент;

*Конструкторы и деструктор:*

* TElem<T>(T \_d = 0, TElem<T>\* n = 0); - конструктор инициализации;
* TElem<T>(TElem<T>& A); - конструктор копирования;

*Методы:*

* T GetD(); - получение значения элемента (если список не пуст);
* TElem<T>\* GetNext(); - получение указателя на следующий элемент (если список не пуст);
* void SetD(T p); - устанавливаем значение элемента (если список не полон);
* void SetNext(TElem<T>\* n); - устанавливаем указатель на следующий элемент (если список не полон);

*Перегруженные операторы:*

Нет перегруженных операторов.

***Класс TList:***

*Поля:*

* TElem<T>\* begin; -указатель на начало списка;

*Конструкторы и деструктор:*

* TList<T> (); - конструктор по умолчанию;
* TList<T> (TList<T>& A); - конструктор копирования;

*Методы:*

* void PutBegin(T A); - добавляет элемент в начало списка (если он не полон);
* void PutEnd(T A); - добавляет элемент в конец списка (если он не полон);
* T GetBegin(); - забирает начальный элемент из списка (если он не пуст);
* T GetEnd(); - забирает последний элемент из списка (если он не пуст);
* T GetElem(T\* A);
* T\* GetInd(T A);
* bool IsFull(); - проверяет полон ли список;
* bool IsEmpty(); - проверяет пуст ли список;

*Перегруженные операторы:*

Нет перегруженных операторов.

## **Описание алгоритмов**

**Метод PutBegin:**

Создаем указатель на объект класса TElem. С помощью конструктора выделяем память и инициализируем новый элемент. Указатель на начало списка переопределяем на только что добавленный элемент.

**Метод GetBegin:**

Если список пуст, то бросаем исключение. Начало списка устанавливаем на следующий за первым элемент. Удаляем бывший первый элемент.

**Метод PutEnd:**

Если список не полон, то создаем указатель \*a на объект класса TElem, в него записываем значение начала списка. В цикле ищем последний элемент, следуя по указателям*.* Как только конец списка будет найден, создаем новый элемент и инициализируем его. Устанавливаем для текущего последнего элемента указатель на следующий – только что созданный.

В том случае, если в списке не было элементов, то указателю на начало списка присваиваем значение, указывающее на звено, созданное с помощью конструктора TElem.

**Метод GetEnd:**

Если список пуст, то бросаем исключение. Иначе создаем указатель на объект класса TElem. В цикле ищем предпоследний элемент, следуя по указателям*.* Создаем новый элемент. В него записываем указатель на последнее звено списка. Получаем данные из этого звена. Удаляем указатель на последнее звенои тем самым освобождаем память, занимаемую последним элементом. Для предпоследнего элемента*,* устанавливаем в качестве следующего за ним 0, т.к. он теперь стал последним.

# **Заключение**

В ходе выполнения лабораторной работы я смогла реализовать такую структуру данных, как список на указателях. Вместе с ней, согласно заданному интерфейсу, я реализовала классы TElem и Tlist. Написание нескольких своих тестов, помогло мне разобраться с системой автоматических тестов Google Test.

В результате проделанной работы у меня получилось

1. Реализовать класс элемента списка TElem.

2. Реализовать класс списка TList.

3. Разработать интерфейс для данных классов.

4. Обеспечить работоспособность примера использования.

5. Реализовать некоторые тесты на базе Google Test

Таким образом, данная лабораторная работа отвечает всем поставленным задачам. Это означает, что цель работы была достигнута.

# **Литература**

1. Лабораторный практикум. Составители: Барышева И.В., Мееров И.Б., Сысоев А.В., Шестакова Н.В. Под редакцией Гергеля В.П. Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. – 105с.

URL: <http://www.unn.ru/books/met_files/Pract_ADS.pdf>

1. http://kvodo.ru/data-structures-introduction.html