МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Национальный исследовательский**

**Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**Отчет по учебной практике**

**«Структура хранения данных: односвязный линейный список с использованием указателей»**

**Выполнил:** студент группы 381706-1

Колесова Кристина Юрьевна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

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

Нижний Новгород

2018.

Оглавление

[1. Введение 3](#_Toc2009015)

[2. Постановка задачи 4](#_Toc2009016)

[3. Руководство пользователя 5](#_Toc2009017)

[4. Руководство программиста 6](#_Toc2009018)

[5.1. Описание структуры программы 6](#_Toc2009019)

[5.2. Описание структур данных 6](#_Toc2009020)

[5.3. Описание алгоритмов 7](#_Toc2009021)

[5. Заключение 9](#_Toc2009022)

[6. Список литературы 10](#_Toc2009023)

# Введение

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

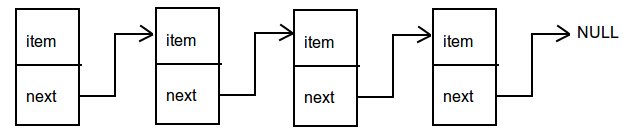
Начало списка называют головным элементом, а звенья списка - узлами. Каждый узел односвязного списка помимо лежащего в нем значения, содержит поле указателя на следующий узел. Поле указателя последнего узла содержит нулевое значение (указывает на NULL).

Рисунок 1. Структура списка

В односвязном списке вставка и удаление узлов производится либо в начало, либо в конец списка.

Структура списка ограничивает доступ к его узлам по индексу. Список нельзя индексировать, как массив. Чтобы попасть на некоторый узел односвязного списка, необходимо последовательно пройти весь путь от головного элемента до нужного узла.

Цель данной лабораторной работы – реализация структуры данных односвязного списка.

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Разработка и реализация вспомогательного класса – TElem
2. Разработка и реализация класса списка - TList
3. Реализация класса для обработки исключений – EexceptionLib
4. Реализация тестов на базе Google Test
5. Пример использования классаTList

# Руководство пользователя

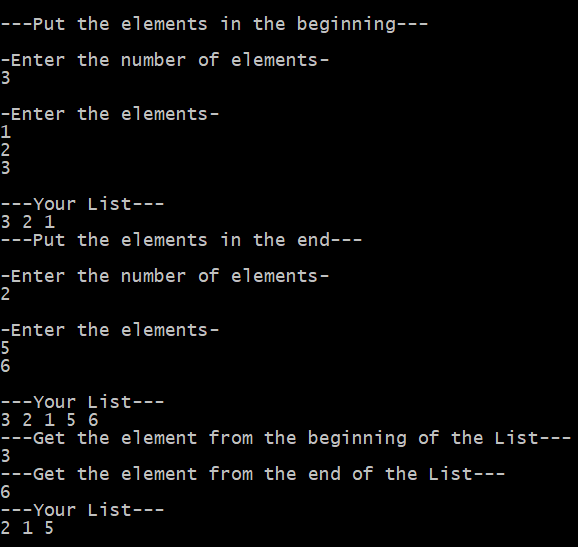
Рассмотрим пример работы программы. После запуска программы пользователю предлагается положить элементы в начало списка, требуется ввести количество элементов и сами элементы. Затем на экран выводится сам список и предлагается положить элементы в конец списка. Далее демонстрируется работа методов, позволяющих забрать элемент из начала и конца списка.

Рисунок . Пример работы программы

# Руководство программиста

## Описание структуры программы

В программе содержатся следующие модули:

* Модуль ListLib – содержит заголовочные файлы файлы List.h и Elem.h, в которых описаны и реализованы классы TList и TElem, и файлы List.cpp, Elem.cpp
* Модуль ExceptionLib – библиотека, позволяющая создавать собственные исключения.
* Модуль List – содержит файл main\_List.cpp, в котором реализован пример использования класса TList;
* Модуль ListTest – содержит файл test\_List.cpp, в котором реализованы тесты для классов TElem и Tlist

## Описание структур данных

* Класс TElem

Класс TElem является шаблонным.

Элементы класса, объявленные со спецификатором public:

T data; - переменная, хранящая элемент списка

TElem<T>\* next; - указатель на следующий элемент списка

TElem(T \_data = 0, TElem <T>\* \_next = 0) - конструктор.

TElem(TElem<T> &A) - конструктор копирования.

virtual ~TElem() - деструктор.

TElem\* GetNext() - метод, возвращающий указатель на следующий элемент.

T GetData() - метод, возвращающий значение элемента.

void SetData(T \_data) - метод, позволяющий установить значение элемента.

void SetNext(TElem <T>\* \_next) – метод, позволяющий установить указатель на следующий элемент списка.

* Класс TList

Класс TList является шаблонным.

Элементы класса, объявленные со спецификатором public:

TElem <T>\* begin – указатель на начало списка.

int size – число элементов в списке.

TList() - конструктор .

TList(TList<T> & A) - конструктор копирования.

virtual ~TList() - деструктор.

void Print() – печать списка на экран.

void Put(int n, T A) - положить элемент на позицию n.

T Get (int n) - взять элемент из списка с номером n.

void PutBegin(T A) – метод, позволяющий положить элемент в начало списка.

void PutEnd(T A) – метод, позволяющий положить элемент в конец списка.

T GetBegin() –метод, позволяющий взять элемент с удалением из начала.

T GetEnd() – метод, позволяющий взять элемент списка с удалением из конца.

bool IsEmpty() - проверка списка на пустоту.

bool IsFull()- проверка полон ли список.

* Класс TException

Класс содержит 1 private поле:

string msg – переменная, хранящая сообщение об ошибке в виде строки.

Содержит 2 public элемента:

msg(\_msg) – конструктор.

void Print() – метод отображения ошибки на консоль.

## Описание алгоритмов

**Добавление элемента в начало списка.**

При добавлении элемента в начало списка мы создаем указатель на объект класса TElem, далее выделяем память под объект этого класса и создаем элемент списка. Указатель на начало списка переопределяем на созданный элемент списка. Алгоритм работает за линейное время, сложность O(n).

**Удаление элемента из начала списка.**

При удалении элемента из начала списка (если он не пуст) создаем указательна объект класса TElem, которому присваиваем значение текущего начала списка. Создаем временную переменную*,* в которую записываем значение первого элемента. Начало списка устанавливаем на следующий за удаляемым элемент. Удаляем указатель, указывающий на старое начало списка. Алгоритм работает за линейное время, сложность O(n).

**Добавление элемента в конец списка.**

Проверяем есть ли элементы в списке. Если есть, то создаем указатель на объект класса TElem, в него записываем значение начала списка. В цикле ищем последний элемент*.* Затем выделяем память под новый элемент списка и создаем его. Устанавливаем для найденного последнего элемента указатель на созданный элемент. Если в списке нет элементов, то создаем элемент и указателю на начало списка присваиваем значение, указывающее на созданный элемент.

**Удаление элемента из конца списка.**

Проверяем список на пустоту, если список не пуст проверяем больше ли одного элемента в списке. Если указатель на следующий за первым элемент равен 0, то возвращаем первый элемент и обнуляем указатель на начало списка. В случае, когда элементов больше одного, создаем указатель \*a на объект класса TElem. Ищем в цикле предпоследний элемент списка. Создаем указатель \*bна объект класса TElem. В него записываем указатель на последний элемент списка. Получаем данные из этого элемента. Удаляем указатель \*b. Для \*aустанавливаем указатель, на следующий элемент 0.

# Заключение

В ходе выполнения лабораторной был произведен анализ задачи - установлено понятие списка на указателях. Была разработана библиотека, реализующая шаблонный класс списка. Она позволяет при работе со списком выполнять базовые операции извлечения/добавление элементов списка.

Были реализованы тесты для проверки работоспособности класса стека на базе GoogleTest. А также приведен пример работы стека.

# Список литературы

1. Википедия. Статья «Список»: [https://ru.wikipedia.org/wiki/Список\_(информатика)].
2. Васильев А.Н. Самоучитель С++ с примерами и задачами. -СПб.: Наука и Техника, 2016. -480с.
3. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.