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:**

**список на основе указателей»**

**Выполнил:** студент группы 381303-3

Исхакова Лана Руслановна

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ Лебедев Илья Геннадьевич

Нижний Новгород

2020.

Содержание

[1. Введение 3](#_Toc728565)

[2. Постановка задачи 4](#_Toc728566)

[3. Руководство пользователя 5](#_Toc728567)

[4. Руководство программиста 7](#_Toc728568)

[4.1 Описание структуры программы 7](#_Toc728569)

[4.2 Описание структур данных 7](#_Toc728570)

[4.3 Описание алгоритмов 8](#_Toc728571)

[5. Заключение 11](#_Toc728572)

[6. Литература 12](#_Toc728573)

# Введение

Связный список является простейшим типом данных динамической структуры, состоящей из элементов (узлов). Каждый узел включает в себя в классическом варианте два поля:

* данные (в качестве данных может выступать переменная, объект класса или структуры и т. д.)
* указатель на следующий узел в списке.

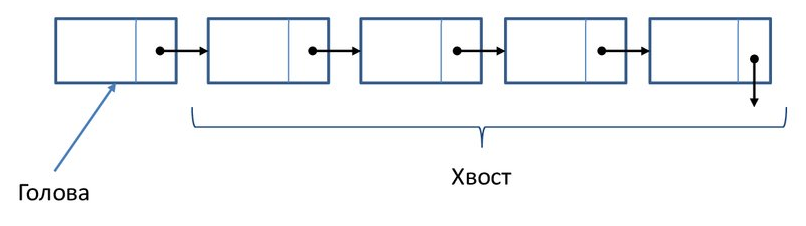
Элементы связанного списка можно помещать и исключать произвольным образом.

**Классификация списков**

По количеству полей указателей различают однонаправленный (односвязный) и двунаправленный (двусвязный) списки:

* Связный список, содержащий только один указатель на следующий элемент, называется односвязным.
* Связный список, содержащий два поля указателя – на следующий элемент и на предыдущий, называется двусвязным.

Начало списка называют головным элементом (головой списка).



*Рисунок 1. Структура односвязного списка.*

В данной лабораторной работе будет реализован односвязный список на основе

указателей.

# Постановка задачи

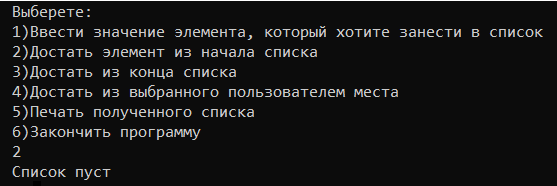
В рамках лабораторной работы ставится задача эффективной реализации структуры данных – линейного односвязного списка на указателях.

Для работы со списком необходимо реализовать методы:

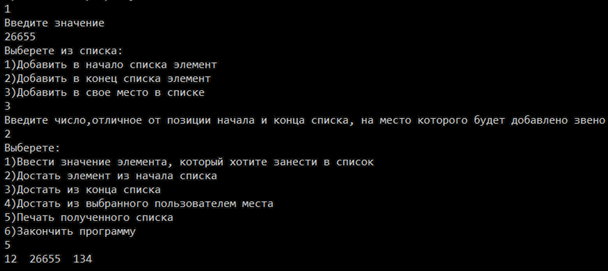
* добавление в начало списка,
* добавление в конец списка,
* добавление в произвольную позицию списка,
* получить значение элемента в произвольную позицию списка с последующим удалением,
* получить значение элемента из начала списка с последующим удалением
* получить значение элемента из конца списка с последующим удалением
* проверка списка на полноту/пустоту.

# Руководство пользователя

Программа начинает свою работу с вывода предложенных функций, которые будут представлены ниже на рисунке 1. Так как изначально список пустой, то при выборе пунктов 2, 3, 4 и 5 будет выведена надпись о том, что список пустой.

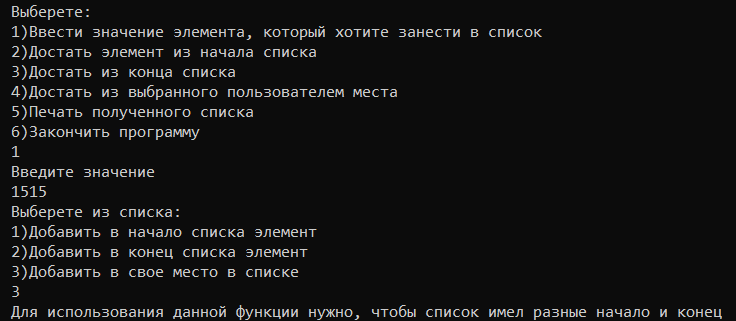


*Рисунок 1. Интерфейс программы.*

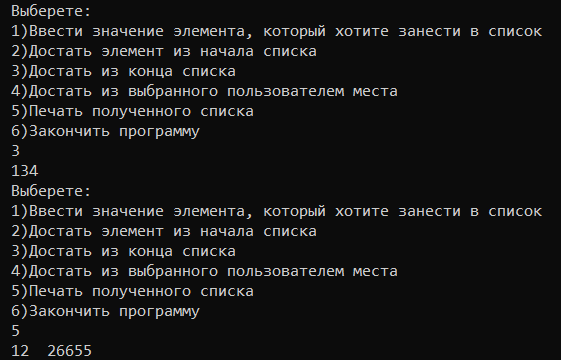
Таким образом, пользователю нужно создать звено списка, для этого нужно выбрать 1 пункт программы, тем самым ввести значение. На втором этапе пользователю предлагается выбрать место, куда записать элемент (Рисунок 2). Если пользователь выберет сначала 3 пункт, программа выдаст об ошибке и попросит выбрать другой пункт.

*Рисунок 2. Пример ввода значения в список.*

После некоторых итераций с 1 пунктом, можно вывести полученный список.

  
*Рисунок 3. Пример вывода списка.*

Также пользователь может получить значение любого звена из списка, при этом удалив его. Так на рисунке 4 показан пример удаление звена и вывод измененного списка.



*Рисунок 4. Пример использования.*

После выполнения всех действий, 6 пункт завершает работу программы.

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль List. Содержит пример использования списка (mainlist.cpp)
* Модуль List – статическая библиотека. Содержит заголовочный файл List.h, в котором реализован шаблонный класс TListElem. А также содержит файл Elem.h, в котором реализован вспомогательный шаблонный класс TElem.
* Модуль test. Содержит тесты, проверяющие правильность работы методов (test\_TList.cpp)

## 4.2 Описание структур данных

В программе определены следующие шаблонные классы: TListElem, TList.

Внутри класса TListElem определены следующие поля:

* T data – данные элемента;
* TListElem\* next – указатель на следующий элемент;
* TListElem\* prev – указатель на предыдущий элемент;

Также определены следующие методы:

* TListElem(T \_data) – конструктор-инициализатор;
* ~TListElem() – деструктор;
* T GetData() – метод для получения данных;
* TListElem\* GetNext() – метод для получения указателя на следующий элемент;
* TListElem\* GetPrev() – метод для получения указателя на предыдущий элемент;
* void SetData(T \_data) – метод для задания данных;
* void SetNext(TListElem\* \_next) – метод для задания указателя на следующий элемент;
* void SetPrev(TListElem\* \_prev) – метод для задания указателя на предыдущий элемент;
* friend ostream& operator<< (ostream& ostr, const TListElem<T1>& A) – оператор вывода;
* friend istream& operator >> (istream& istr, TListElem<T1>& A) – оператор ввода.

Внутри класса TList определены следующие protected-поля:

* TListElem<T>\* root – указатель на головной элемент списка;
* TListElem<T>\* end – указатель на хвостовой элемент списка;
* int count ­– количество элементов в списке;

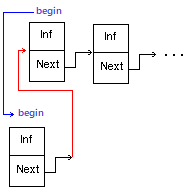
Также определены следующие public-методы:

* TList() – конструктор по умолчанию;
* TList(TList<T>& \_v) – конструктор копирования;
* ~TList() – деструктор;
* TList<T>& operator =(TList<T>& \_v) – оператор =;
* void InsFirst(T d) – метод вставки в начало;
* void InsLast(T d) – метод вставки в конец;
* void Ins(TListElem<T>\* e, T d) – метод вставки на указанную позицию;
* TListElem<T>\* GetFirst() – метод получения указателя на первый элемент списка;
* TListElem<T>\* GetLast() – метод получения указателя на последний элемент списка;
* void DelFirst() – метод удаления элемента с начала;
* void DelLast() – метод удаления элемента с конца;
* void Del(TListElem<T>\* e) – метод удаления элемента с указанной позиции;
* bool IsEmpty(void) const – метод, проверяющий, пустой ли список;
* friend ostream& operator<< (ostream& ostr, const TList<T1> &A) – оператор вывод;
* friend istream& operator >> (istream& istr, TList<T1> &A) – оператор ввода;
* int GetCount() – метод получения количества элементов в списке;
* void reverse() – метод, разворачивающий список;
* void cleanse() – метод, удаляющий одинаковые элементы в списке.

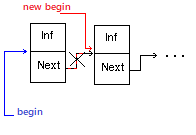
## 4.3 Описание алгоритмов

**Добавление звена в начало списка.**

Чтобы добавить звено в начало списка, нужно создать это звено с помощью конструктора класса TElem, который будет принимать в качестве параметров значение элемента и указатель на следующий, то есть на старый begin. После чего begin присваиваем адрес добавленного звена.

  
*Рисунок 5. Добавление звена в начало списка.*

**Удаление звена из начала списка.**

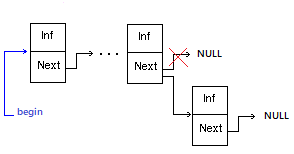
Для удаления звена из начала списка выполняем проверку на пустоту списка. Если список пуст, то бросается исключение. В другом случае создается переменная temp класса TElem<T> \*, которой присваивают значение начала списка. После чего создается переменная шаблонного типа Т, в которую записывается значение элемента начального звена списка. Дальше begin присваиваем новое значение, то есть адрес следующего звена.

*Рисунок 6. Удаление звена из начала списка.*

**Добавление звена в конец списка.**

Для начала делается проверка списка на пустоту, если это так, то данное звено является началом и концом списка. Иначе создается переменная temp класса TElem<T> \*, которой соответствует значение последнего звена списка, то есть полученное звено указывает на NULL, что означает конец списка.

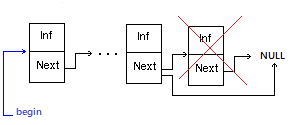
Дальше создается новая переменная i класса TElem<T> \*, которой присваивается значение начала списка, и в цикле ищем последнее звено списка, после чего этому найденному звену присваивается указатель на temp.

  
*Рисунок 7. Добавление звена в конец списка.*

**Удаление звена из конца списка.**

Для удаления звена из конца списка делается проверка на его пустоту. Если список пуст, вызывается исключение. Создается новая переменная i класса TElem<T> \*, которой присваивается значение начала списка. Если список состоит из одного звена, то создается переменная шаблонного типа Т, в которую записывается значение элемента i-го звена, после чего begin присваивается следующее значение звена, то есть NULL.

Иначе, если в списке больше одного звена, в цикле ищем предпоследнее звено списка, создаем переменную шаблонного типа Т и с помощью функции GetNext(), которая возвращает указатель на следующее звено, в нашем случае на последнее, присваиваем этой переменной значение элемента последнего звена. И предпоследнему звену присваиваем указатель на NULL.

  
*Рисунок 8. Удаление звена из конца списка.*

# Заключение

В ходе выполнения данной лабораторной была реализована структура хранения данных в виде односвязного линейного списка. Были реализованы методы, позволяющие производить основные действия над элементами ОЛС, а именно: инициализация списка, добавление узла в список, удаление узла из списка, вывод элементов списка и т.д.

Были разработаны тесты, проверяющие корректную работу методов данных классов, с помощью Google C++ Testing Framework .

# Литература

1. Связный список [Электронный ресурс]

<https://prog-cpp.ru/data-list/>

1. Добавление и удаление звеньев списка [Электронный ресурс]

<https://www.kazedu.kz/referat/1798/1>