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 линейный список с использованием указателей»**

**Выполнил:** студент группы 381706-1

Кольтюшкина Янина Вадимовна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc533088478)

[2. Постановка задачи 4](#_Toc533088479)

[3. Руководство пользователя 5](#_Toc533088480)

[4. Руководство программиста 6](#_Toc533088481)

[4.1 Описание структуры программы 6](#_Toc533088482)

[4.2 Описание структур данных 6](#_Toc533088483)

[4.3 Описание алгоритмов 7](#_Toc533088484)

[5. Эксперименты 9](#_Toc533088485)

[6. Заключение 10](#_Toc533088486)

[7. Литература 11](#_Toc533088487)

# Введение

**Целью** данной лабораторной работы является практическое освоение динамической структуры данных – линейного односвязного списка с использованием указателей.

Ознакомимся со следующими базовыми понятиями.

**Односвязный список** – это базовая динамическая структура данных, состоящая из упорядоченного набора элементов одного типа, последовательно связанных между собой.

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).

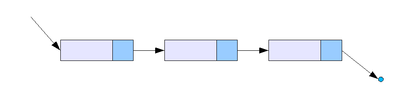


Рисунок 1 Структура односвязного списка.

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

Элементам списка нельзя задать индекс, как мы это делали в массивах. Структура списка ограничивает доступ к его узлам по индексу. Чтобы иметь доступ к определенному узлу односвязного списка, необходимо последовательно пройти весь путь от головного элемента до нужного узла.

# Постановка задачи

Главной задачей данной лабораторной работы является эффективная реализация структуры данных линейного односвязного списка на указателях.

Программное решение будет выглядеть следующим образом:

1. Разработка и реализация вспомогательного класса узла списка - TElem.
2. Разработка и реализация класса списка – TList.
3. Класс для обработки исключений – MyException.
4. Пример программы, демонстрирующая работу класса TList.
5. Написание набора автоматических тестов с использованием Google C++ Testing Framework и проверка работоспособности методов классов.

# Руководство пользователя

При запуске программы пользователя просят ввести количество элементов списка, которое он хочет видеть



Рисунок Ввод количества элементов списка

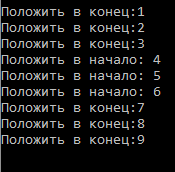
Затем создается список и пользователь может видеть, как он поэтапно заполняется 

Рисунок Заполнение списка

Далее выводится список полностью.



Рисунок Созданный список

В следующих двух шагах мы можем видеть, как изымаются первый и последний элементы списка, их значения.



Рисунок Первый и последний элементы

Далее выводится начальный список без первого и последнего элементов.



Рисунок Конечный вид списка

На этом работа программы прекращается.

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль List. Содержит пример использования списка. Реализация в файле *main\_List.cpp.*
* Модуль ListLib – статическая библиотека. Содержит файл List.h, в котором описан интерфейс и реализация шаблонного класса *TList*. А также содержит файл Elem.h, в котором представлен интерфейс и реализация вспомогательного шаблонного класса *TElem*.
* Модуль ListTest. Содержит 24 теста, описанных в файле *ListTest.cpp* и разработанных с помощью использования Google C++ Testing Framework.
* Модуль ExceptionLib – библиотека, позволяющая создавать собственные исключения.

## 4.2 Описание структур данных

#### Класс TElem – класс узла списка

Класс TElem является шаблонным. Всего в классе два поля, объявленные со спецификатором доступа protected: T data – переменная под хранение данных в узле и TElem <T>\* next – указатель на следующий узел списка.

**Конструкторы и методы класса со спецификатором public:**

* *TElem(T \_data = 0, TElem <T>\* \_next = 0)* – конструктор по умолчанию.
* *TElem(TElem<T> &L)* – конструктор копирования.
* *virtual ~TElem()* – деструктор.
* *TElem\* GetNext()* – получить указатель на следующий узел.
* *T GetData()* – получить значение из узла.
* *void Set(T e)* – установить значение в звено списка.
* *void SetNext(TElem <T>\* n)* – установить указатель на следующее звено списка.

#### Класс TList – класс списка.

Класс TList списка является шаблонным. В классе всего два поля *TElem <T>\* start* – указатель на начало списка и *int count* – количество элементов в списке, объявленное со спецификатором доступа protected.

**Конструкторы и методы класса, объявленные со спецификатором public:**

* *TList()* – конструктор по умолчанию.
* *TList(TList<T> &L)* – конструктор копирования.
* *virtual ~TList()* – деструктор.
* *void Put(int \_n, T elem)* – положить элемент в списке на позицию *\_n*.
* *void Get(int \_n)* – извлечь элемент в списке на позиции *\_n*.
* *void PutStart(T a)* – положить элемент в начало списка.
* *void PutEnd(T a)* – положить элемент в конец списка.
* *T GetStart()* – взять элемент с удалением из начала.
* *T GetEnd()* – взять элемент списка с удалением из конца.
* *bool IsEmpty()* – метод проверки списка на пустоту.
* *void Print()* – метод вывода списка на консоль.

## 4.3 Описание алгоритмов

**Добавление звена списка в начало.**

При добавлении звена в начало списка мы создаем указатель на объект класса TElem. Далее выделяем память под объект этого класса и с помощью конструктора с параметрами для TElem, передав туда значение, которое необходимо положить в начало списка, и указатель на текущее начало, создаем очередное звено списка. Указатель на начало списка переопределяем на только что добавленный элемент.

**Удаление звена списка из начала.**

Для удаления звена списка из начала выполняем проверку на пустоту списка. Если список пуст, то бросаем исключение. Иначе создаем указатель *\*temp* на объект класса TElem, которому присваиваем значение текущего начала списка. Создаем временную переменную *tmp,* в которую записываем значение, хранящееся в первом элементе списка. Начало списка устанавливаем на следующий за удаляемым элемент. Удаляем указатель *\*temp* для того, чтобы очистить память, занимаемую бывшим первым элементом.

**Добавление звена списка в конец.**

При добавлении звена списка в конец проверяем, есть ли элементы в списке. Если есть, то создаем указатель \*a на объект класса TElem, в него записываем значение начала списка. В цикле ищем текущий последний элемент*.* Как только конец списка будет найден, выделяем память под новое звено списка и с помощью конструктора по умолчанию TElem создаем 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.

**Удаление звена списка из конца.**

Для удаления звена списка из конца выполняем проверку на пустоту списка. Если список пуст, то бросаем исключение. Иначе необходимо проверить: в списке больше одного элемент или ровно один. Для этого смотрим на следующий за первым элемент. Если указатель на него равен нулю, то мы возвращаем только данные из первого элемента списка, начало списка обнуляем.

Ели элементов больше одного, то создаем указатель *\*temp* на объект класса TElem. Ищем в цикле предпоследнее звено списка. Создаем еще один указатель *\*temp1* на объект класса TElem. В него записываем указатель на последнее звено списка. Получаем данные из этого звена. Удаляем указатель *\*temp1* и тем самым освобождаем память, занимаемую бывшим последним элементом. Для *\*temp,* устанавливаем в качестве следующего за ним 0, т.к. он теперь стал последним.

# Эксперименты

Эксперименты проводились на ПК с следующими параметрами:

1. Операционная система: Windows 10
2. Процессор: Intel(R) Core™ i7-4710MQ CPU @ 2.50 GHz
3. Версия Visual Studio: 2017

|  |  |  |
| --- | --- | --- |
| Кол-во элементов в списке | Время работы метода *PutStart()* в млс.  O(1) | Время работы метода *PutEnd()* в млс.  O(n) |
| 10 000 | 0 | 0 |
| 100 000 | 0 | 2 |
| 740 000 | 0 | 9 |
| 1 000 000 | 0 | 15 |
| 3 000 000 | 0 | 41 |

Таблица 1. Время работы методов добавления элементов в начало и в конец списка

Как видно из таблицы метод *PutStart(),* работает значительно быстрее метода *PutEnd()*. Это происходит за счет того, что при добавлении элемента в начало списка, к памяти мы обращаемся всего 1 раз, за O(1), а для добавления элемента в конец списка, мы должны пройти по всем n элементам списка, тем самым мы n раз обратимся к памяти, и сложность алгоритма равна O(n), что существенно замедляет работу.

Делаем вывод, при работе со списками в приоритете стоит использовать метод *PutStart().*

# Заключение

В этой лабораторной работе я смогла справиться со всеми поставленными задачами. Разобравшись в устройстве работы списка, сначала я реализовала вспомогательный класс TElem, а затем и класс TList. Для этого была реализована библиотека ListLib, в которой полностью описан данный шаблонный класс. Написанный мной набор автоматических тестов, проверяет работоспособность всех реализованных методов и делает это весьма успешно. Пример использования данных методов класса очередь для пользователя полностью функционирует и готов к использованию.

# Литература

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2»: [http://www.itmm.unn.ru/files/2018/11/1.5.-Struktury-hraneniya-s-ispolzovaniem-ukazatelej-spiski.pdf], 2015.
2. Википедия: свободная электронная энциклопедия: на русском языке: [https://ru.wikipedia.org/wiki/Список\_(информатика)]