

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»

Выполнил: студент группы 381706-1

Митягина Дарья Сергеевна

_____ Подпись

Научный руководитель:

ассистент каф. МОСТ ИИТММ

_____ Лебедев И.Г

Нижний Новгород

2018.

Оглавление

1. Введение	2
2. Постановка задачи	3
3. Руководство пользователя	4
4. Руководство программиста.....	5
4.1. Описание структуры программы.	5
4.2. Описание структур данных.	5
4.3. Описание алгоритмов.....	6
5. Заключение.....	8
6. Список литературы.....	9

1. Введение

Списком называется упорядоченное множество, состоящее из переменного числа элементов, к которым применимы операции включения, исключения.

Экземпляр списка является компьютерной реализацией математического понятия конечной последовательности. Экземпляры значений, находящихся в списке, называются элементами.

Линейный однонаправленный список — это структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством указателей. Каждый элемент списка имеет указатель на следующий элемент. Последний элемент списка указывает на NULL. Элемент, на который нет указателя, является первым (головным) элементом списка. Здесь ссылка в каждом узле указывает на следующий узел в списке.



Рис.1 Структура списка

2. Постановка задачи

В данной лабораторной работе необходимо разработать библиотеку для хранения и работы со списком на указателях.

Для этого нам нужно:

- Описать и реализовать класс элемента списка (TElement).
- Описать и реализовать класс списка (TList).
- Протестировать класс TList с помощью GT.
- Реализовать класс TException для обработки исключений.

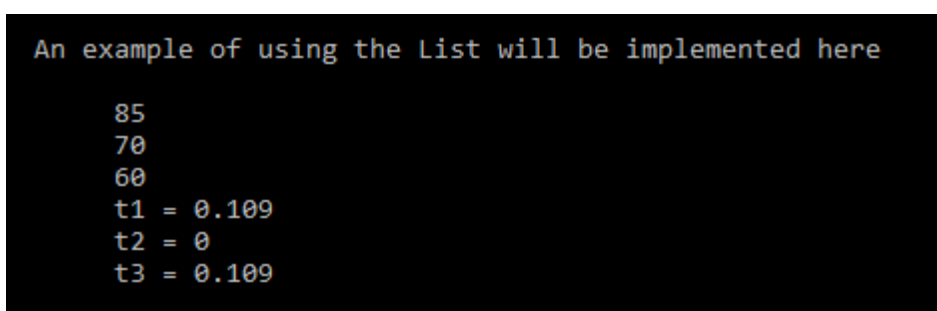
3. Руководство пользователя

При запуске программы пользователю представляется простой пример использования списков.

Создается объект класса TList, список. Затем в него кладутся значения и выводятся на экран.

Далее идет расчёт времени работы отдельных методов, полученные значения выводятся на экран.

Выглядит это следующим образом:



```
An example of using the List will be implemented here  
85  
70  
60  
t1 = 0.109  
t2 = 0  
t3 = 0.109
```

Рис. 2 – Пример использования списков

4. Руководство программиста

4.1. Описание структуры программы.

Программа состоит из модулей:

1. List – содержит в себе файл реализации примера использования класса TList.
2. ListTest – содержит в себе файл test_list.cpp, в котором находится набор тестов, для проверки работоспособности класса TList.
3. ListLib – содержит в себе файлы List.h и Element.h, в которых описаны и реализованы классы TList и TElem.

4.2. Описание структур данных.

Класс TElement

Этот класс является шаблонным.

Рассмотрим public-часть:

1. T data; - переменная, хранящая элемент списка
2. TElement <T>* next; - указатель на следующий элемент списка
3. TElement(T _data = 0, TElement <T>* _next = 0); - конструктор инициализатор
4. TElement(TElement<T> &Elem); - конструктор копирования
5. virtual ~TElement(); - деструктор
6. TElement* GetNext(); - получение указателя на следующий элемент списка
7. T GetData(); - получение значения элемента
8. void Set(T Elem); - установка значения элемента
9. void SetNext(TElement <T>* n); - добавление указателя на следующий элемент списка.

Класс TList

Этот класс является шаблонным. Защищенная(protected) часть:

1. TElement<T> *pFirst, *pLast, *pPrev, *pCurr, *pStop;

2. int size, pos;

Рассмотрим public-часть:

1. TList() конструктор
2. ~TList() деструктор
3. void AddFirst(T a) добавление в начало
4. void AddCurrent(T a) добавление в текущую позицию
5. void AddLast(T a) добавление в конец
6. T viewData() получение элемента
7. T* viewPtr() получение указателя
8. void delFirst() удаление из начала
9. void delCurrent() удаление с текущей позиции

Проверки :

10. bool isEnd()
11. bool isStart()
12. bool isEmpty()
13. bool isFull()

14. T operator[](int m)

Вспомогательные методы:

15. void reset()
16. void goNext()

4.3. Описание алгоритмов

В данной части не будут рассматриваться тривиальные методы, внимание уделим лишь некоторым.

1. *Добавление элемента в начало списка.*

Создаем указатель на объект класса TElem, выделяем память под объект этого класса и создаем очередной элемент списка.

Указатель на начало списка устанавливаем на новый элемент. Инкрементируем значение размера списка и номера позиции.

Алгоритм работает за линейное время, сложность $O(n)$.

2. Добавление элемента на текущую позицию.

Указателю текущего элемента присваиваем пришедшее значение, для которого следующим элементом является элемент, до предыдущего шага являвшийся текущим.

Инкрементируем значение размера списка

3. Удаление элемента списка из начала.

Если список не пуст: создаем указатель на объект класса TElem, которому присваиваем значение текущего начала списка. Создаем временную переменную для хранения значения в первом элементе списка. Начало списка устанавливаем на элемент, следующий за удаляемым. Очищаем память, занимаемую ранее первым элементом.

Аналогично с удалением из конца, с некоторой позиции.

Алгоритм работает за линейное время, сложность $O(n)$.

5. Заключение

В процессе работы над данной лабораторной работой мне удалось реализовать такую структуру хранения данных, как список на указателях.

Кроме того, было достигнуто более глубокое понимание принципов работы с данной структурой, принципов тестирования программы с помощью GT.

6. Список литературы

1. Гегель В.П. Методические материалы по курсу «Методы программирования 2», 2015.
2. Статья, посвященная теме списков [<https://prog-cpp.ru/data-ols/>]
3. Статья в Википедии, посвященная данной теме [https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA#%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D0%B9_%D1%81%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA]