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

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.

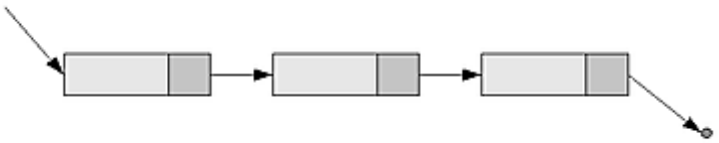

Рис.1 Структура списка

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.

Выглядит это следующим образом:

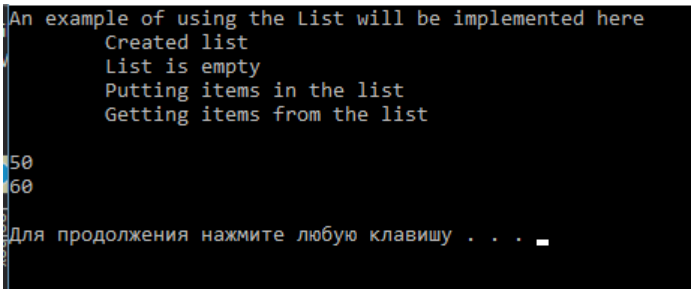

Рис. 2 – Пример использования списков

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. Exception – содержит в себе файл Exception.h, содержащий реализациею класса исключений TException.

4.2. Описание структур данных.

Класс TElement

Этот класс является шаблонным. Защищенная (protected) часть содержит следующие поля:

1. T data; - переменная, хранящая элемент списка

2. TElement <T>* next; - указатель на следующий элемент списка

Рассмотрим public-часть:

1. TElement(T _data = 0, TElement <T>* _next = 0); - конструктор инициализатор

2. TElement(TElement<T> &Elem); - конструктор копирования

3. virtual ~TElement(); - деструктор

4. TElement* GetNext(); - получение указателя на следующий элемент списка

5. T GetData(); - получение значения элемента

6. void Set(T Elem); - установка значения элемента

7. void SetNext(TElement <T>* n); - добавление указателя на следующий элемент списка.

Класс TList

Этот класс является шаблонным. Защищенная(protected) часть:

1. TElement <T>* begin; - Указатель на первое звено списка

2. int size; - размер

Рассмотрим public-часть:

1. TList(); // конструктор по умолчанию

2. TList(TList<T> &List); // конструктор копирования

3. virtual ~TList(); // деструктор

4. T viewData(TElement<T>* ptr) { return ptr->data; }

5. T* viewPtr(TElement<T>* ptr) { return ptr; }

6. int GetSize() { return size; }

7. void PutBegin(T A); - устанавливает начальное звено

8. void PutEnd(T A); - устанавливает последнее звено

9. T GetBegin(); - получает значение первого звена

10. T GetEnd(); - получает значение последнего звена

11. TElement<T> * Delete(T t);

12. bool IsFull(); - проверка на полноту

13. bool IsEmpty(); - проверка на пустоту

4.3. Описание алгоритмов

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

1. *Добавление элемента в конец списка.*

При добавлении элемента списка в конец проверяем, есть ли элементы в списке. Если это так, то создаем указатель на объект класса TElem, записываем в него значение начала списка.

В цикле ищем последний элемент, выделяем память под новой элемент списка и создаем его. Указатель для последнего элемента устанавливаем на следующий.

2. *Добавление элемента в начало списка.*

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

Указатель на начало списка устанавливаем на новый элемент.

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 удаляемый. Очищаем память, занимаемую ранее первым элементом.

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. Гергель 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]