Lab 12 – 08227 Advanced Programming

This tutorial introduces the reader to linked lists in C++.

1.0 Linked Lists Exercise 1

Download **Lab12.zip** from the module site and extract the contents to the folder **G:/08227/Lab12/**.

Previously, we implemented a simple Address Book application that added People's names and ages to it. We are going to implement this Address Book as a Single Linked List (SLL).

The **PersonNode** class will be instantiated to create nodes for our SLL that will hold the **name** (m_name) and **age** (m_age) of each person, and will also hold a pointer to the **next** (m_next) **PersonNode** in the SLL. Therefore, add these three private member data variable to the **PersonNode** class. Remember to initialize these data members to appropriate values in all constructors (preferred to do this on the initialisation list of the constructor).

Add other constructors and member methods to the **PersonNode** so that the **name** and **age** of the **PersonNode** can be created, set and returned (setters and getters).

We will be required to set and return the **m_next** pointer to the **AddressBookSLL** class so that we can navigate through the SLL. The **m_next** member data member pointer should be made private, which means that if we add the functionality for the **PersonNode** class to return this pointer then it should be returned as a const pointer. This will unfortunately mean that we cannot navigate through the SLL very easily. Therefore this is one occasion were the use of the keyword **friend** is applicable, especially as these two classes are clearly coupled together and cannot be used without each other. Therefore make the **AddressBookSLL** class a private **friend** of the **PersonNode** class.

2.0 Linked Lists Exercise 2

The **AddressBookSLL** class is required to have the functionality to allow for manipulation of the **PersonNode**s that are contained within the SLL.

The first thing that the **AddressBookSLL** class requires is a head **PersonNode**. Therefore add a private data member to **AddressBookSLL** which is a **PersonNode** pointer called **m_head**. Remember to initialize this data member to an appropriate value in all constructors (i.e. m_head = 0).

The second thing that is required is the functionality to add a new person's details into the **AddressBookSLL**. Therefore implement the functionality of a public member method using the following declaration:

```
void AddPerson(const string &name, int age);
```

This **AddPerson()** method should add a new **PersonNode** to the SLL according to the following pseudo code:

1. If the **m_head** pointer is **0** then assign a new **PersonNode** to the **m_head** pointer (i.e. the SLL was empty).
2. Otherwise, if the **m_head** pointer is not **0** and the **PersonNode** pointed to by the **m_head** pointer has an **m_next** pointer that is **0**, assign the new **PersonNode** to this **m_next** pointer (i.e. there is only one element in the SLL).
3. Otherwise, if the **m_head** pointer is not **0** then navigate through each linked **PersonNode** in the SLL until we find a **PersonNode** that has an **m_next** pointer that is **0**, then assign the new **PersonNode** to this **m_next** pointer.

We are instantiating new **PersonNode** on the heap therefore we are required to take care of our own memory management. Therefore implement the functionality in the **AddressBookSLL**'s **deconstructor** that will delete any memory that has been created in the **AddressBookSLL** and linked **PersonNode**.

3.0 Linked Lists Exercise 3

Add the following functionality to the **AddressBookSLL** class:

- The ability to find a person from the SLL by using their name:

```
bool FindPerson(const string &name);
```

  This method should return true if the PersonNode is found or return false if it is not found.

- The ability to delete a person from the SLL by using their name:

```
bool DeletePerson(const string &name);
```

  This method should return true if the PersonNode was deleted or return false if it is not deleted.

- The ability to output all of the people's names and ages that our in the AddressBookSLL to an ostream.

4.0 ADVANCED: Linked Lists Exercise 4

Add any other functionality to the **AddressBookSLL** class that would improve its usability.