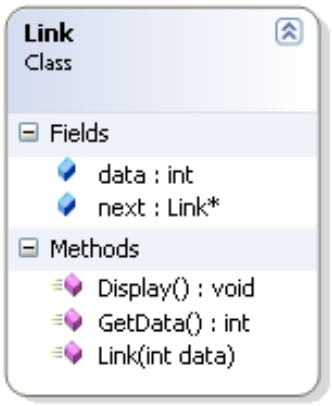
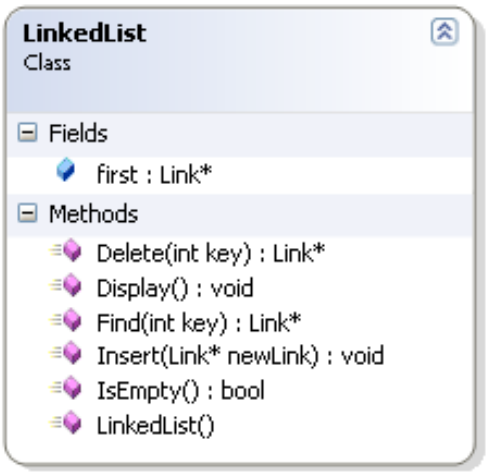
C++ Week 4 Lists

Linked Lists

Linked lists are dynamic data structures second only to arrays in popularity. The data within a linked list is embedded within a link object (usually assigned to a data member within the object). The link also contains a reference to the next link and the LinkedList itself maintains a reference to the first link.





Exercise 1

Within Visual Studio create a new Console Project and create the Link class illustrated above. In this exercise we will be storing int values within the link objects.
The link class consists of

- Data members – data and next.
- A constructor that assigns its parameter to the data attribute
- A member function named display that writes the value of data to the console.

C++ Week 4 Lists

Exercise 2

Now define the LinkedList class illustrated below. Placing the code in the appropriate files (h & cpp). Remember to include the Link header. Within the main function create an instance of the LinkedList class and add 6 Links. Then check the list is saving the data by calling the LinkedList's display member function.

```
// Place in h file
class LinkedList {
public:
    Link *first;
    LinkedList();
    bool IsEmpty();
    void Insert(Link *newLink);
    void Display();
};

// place in cpp file
LinkedList::LinkedList() {
            first = 0;
}

bool LinkedList::IsEmpty(){
                return (first == 0);
}

void LinkedList::Insert(Link *newLink){
                newLink->next = first;
                first = newLink;
}

void LinkedList::Display() {
                Link *current = first;
                while (current != 0) {
                  current->Display();
                  current = current->next;
                }
}
```

The Linked List is a little limited in its functionality, there is no facility to search the list for data values embedded within the links or delete links.

C++ Week 4 Lists

Exercise 3

Below is the pseudo code for the find member function, used to locate values within the linked list. Implement and test the function by invoking Find, passing it the data to locate and then displaying the Link returned.

```
Set current to first link
While the data within current is not equal to key
        If current's next value equals 0
                Return 0
        Otherwise
                Current = current's next
        End if
End while
Return current
```

Exercise 4

The find function will return 0 if it is unable to match the data. Add the logic within main that determines if 0 has been returned and displays an appropriate message.

Exercise 5

Now implement and test the Delete member function, the pseudo code for which is below.

```
Set current to first
Set previous to first
While current's data not equal to key
        If current's next equals 0
                Return 0
        Otherwise
                Set previous to current
                Set current to current's next
        End if
End while
If current equals first
        Set first to first's next
Otherwise
        Set previous next to current's next
End if
Return current
```

C++ Week 4 Lists

Exercise 6

The delete function will return 0 if it is unable to match the data. Add the logic within main that determines if 0 has been returned and displays an appropriate message.

Exercise 7

Implement a delete function that removes the first link within the list.