## Introduction

This program is a sample implementation of a doubly-linked list using generic programming in C++. It will store- a value in a node, and maintain a list of nodes in the order they were created, with each node containing a pointer to the node previous to and immediately following itself.

The user interacts with the program through a simple console-based menu. They are able to perform the following functions:

* **Insert** a new node at the end of the linked list.
* **Delete** a node from the linked list that has a key value matching a value of the user’s choosing. The program will only delete the first instance of the inputted value (it uses the Find function to locate this node).
* **Find** a node containing a key value.matching a value of the user’s choosing. The program will only find the first instance of the inputted value.
* Determine the lowest (**minimum**) value stored in the linked list.
* Determine the highest (**maximum**) value stored in the linked list.
* **Display** the contents of the linked list, in both **forward** and **reverse** orders.
* **Auto-populate** a random number of nodes with a randomized value.
* **Display** a **detailed** list of the contents of the linked list. (Each node’s key value, as well the key value and memory address of that node’s adjacent nodes.
* **Exit** the program.

## Programmer’s Guide

### <LinkedList.h>

template <typename T> class LinkedList

A custom-built doubly-linked list class. The class is templated as to allow for use with a wide variety of data types.

|  |  |  |
| --- | --- | --- |
| **Attributes/Data Members (private)** | | |
| Name | Type | Description |
| nodeCount | int | Stores a count of the number of nodes in the list. Initializes to 0. |
| headNode | LinkedList\_Node<T>\* | Stores a pointer to the first node in the list. Initializes to nullptr. |
| tailNode | LinkedList\_Node<T>\* | Stores a pointer to the last node in the list. Initializes to nullptr. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Member Functions (private)** | | | |
| Signature | |  | |
| Type | Name | Description & Parameters | |
| void | DecreaseNodeCount | Decreases the count of the number of nodes stored in the linked list. | |
| void | IncreaseNodeCount | Increases the count of the number of nodes stored in the linked list. | |
| void | PrintDetails | Outputs the key values and addresses of the current node and its adjacent nodes.  LinkedList\_Node<T>\* targetNode - A pointer to the node object that will have its details printed to the console. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Member Functions (public)** | | | |
| Signature | |  | |
| Type | Name | Description & Parameters | |
| (Constructor) | LinkedList | Default constructor. | |
| (Destructor) | ~LinkedList | Default destructor. | |
| void | Delete | Deletes a node from the linked list.  LinkedList\_Node<T>\* delNode - A pointer to the node object to be deleted. | |
| LinkedList\_Node<T>\* | Find | Finds the first instance of a key value in the linked list. If no match is found, returns nullptr.  T dataToFind - The key value to look for in the linked list.  bool suppressMsg - Tells the function whether or not to display output messages to the console. Defaults to false . | |
| int | GetNodeCount | Returns the current count of the number of nodes in the linked list. | |
| void | Insert | Creates a new node, and inserts it at the end the linked list.  T newData - The key value to be stored in the new node. | |
| bool | IsEmpty | Determines if the linked list is empty. Returns true if empty, returns false if not. | |
| T | Maximum | Returns the largest value stored in the nodes contained in the linked list. | |
| T | Minimum | Returns the smallest value stored in the nodes contained in the linked list. | |
| void | PrintList | Outputs the key values of each node to the console.  bool reverse - Determines the order in which to print the list. false prints from head to tail; true prints from tail to head (reverse). Defaults to false  bool detailed - Detailed mode will display each node’s value, as well as the value and address of the current node’s adjacent nodes, if desired. true prints the extra data; false only prints the key values. Defaults to true. | |

### <LinkedList\_Iterator.h>

template <typename T> class LinkedList\_Iterator

A custom iterator for LinkedList\_Node objects stored in an LinkedList object. Uses its member functions GetPrevNode() and GetNextNode() to traverse the linked list. The class is templated as to allow for use with a wide variety of data types.

|  |  |  |
| --- | --- | --- |
| **Attributes/Data Members (private)** | | |
| Name | Type | Description |
| currentNode | LinkedList\_Node<T>\* | Stores a pointer to the current node the iterator is referencing. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Member Functions (public)** | | | |
| Signature | |  | |
| Type | Name | Description & Parameters | |
| (Constructor) | LinkedList\_Iterator | LinkedList\_Node<T>\* startNode - The node at which this iterator should point to upon creation.  Default constructor. | |
| (Destructor) | ~LinkedList\_Iterator | Default destructor. | |
| LinkedList\_Node<T>\* | GetCurrentNode | Returns a pointer to the current node the iterator is referencing. | |
| T | GetNodeData | Returns the key value stored in the current nodethe iterator is referencing. | |
| LinkedList\_Node<T>\* | IterateBack | Attempts to move to the previous node in the linked list. If the iterator is at the beginning of the list, returns nullptr. Otherwise, updates the iterator’s currentNode pointer to point to the previous node in the list, and returns a pointer to the new node. | |
| LinkedList\_Node<T>\* | IterateFwd | Attempts to move to the next node in the linked list. If the iterator is at the end of the list, returns nullptr. Otherwise, updates the iterator’s currentNode pointer to point to the next node in the list, and returns a pointer to the new node. | |
| void | SetCurrentNode | Sets the node this iterator is pointing to.  LinkedList\_Node<T>\* targetNode - The node at which this iterator should point to upon creation. | |

### <LinkedList\_Node.h>

template <typename T> class LinkedList\_Node

A custom-built node class for use in a doubly-linked list. The class is templated as to allow for use with a wide variety of data types.

|  |  |  |
| --- | --- | --- |
| **Attributes/Data Members (private)** | | |
| Name | Type | Description |
| nextNode | LinkedList\_Node<T>\* | Stores a pointer to the node immediately following this node in the linked list. Initializes to nullptr. |
| nodeData | T | The data being held by this node. |
| prevNode | LinkedList\_Node<T>\* | Stores a pointer to the node immediately preceding this node in the linked list. Initializes to nullptr. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Member Functions (public)** | | | |
| Signature | |  | |
| Type | Name | Description & Parameters | |
| (Constructor) | LinkedList\_Node | Default constructor. |  |
| (Destructor) | ~LinkedList\_Node | Default destructor. | |
| void | ClearNextNode | Sets the pointer to the node immediately following this node in the linked list to nullptr. | |
| void | ClearPrevNode | Sets the pointer to the node immediately preceding this node in the linked list to nullptr. | |
| LinkedList\_Node<T>\* | GetNextNode | Returns a pointer to the node immediately following this node in the linked list. | |
| T | GetNodeData | Returns the key value stored in this node. | |
| LinkedList\_Node<T>\* | GetPrevNode | Returns a pointer to the node immediately preceding this node in the linked list. | |
| void\* | SetNextNode | Changes the pointer to the node immediately following this node in the linked list.  LinkedList\_Node<T>\* nodePtr - A pointer to the node that will be designated as immediately following the current node in the linked list. | |
| void\* | SetNodeData | Changes the key value to be stored in this node.  T dataToStore - The key value to store in this node. | |
| void\* | SetPrevNode | Changes the pointer to the node immediately preceding this node in the linked list.  LinkedList\_Node<T>\* nodePtr - A pointer to the node that will be designated as immediately preceding the current node in the linked list. | |