class

**LinkedList class**

template<typename T>;

LinkedList is a double linked list that stores pointers to other objects within the list. LinkedList can be iterated through both forward and backwards. The elements contained are in different storage locations, the pointers of each element tracks the memory location of the next and previous element. Thus, not requiring the re-allocation of memory as the LinkedList grows or shrinks.

The drawbacks are that you can’t directly access elements within the list. To access an element within the list, it must be iterated through from a known location (front or back). This takes linear time in the distance between the starting location and the element.

**Member Functions**

|  |  |
| --- | --- |
| **(constructor) default** | construct LinkedList |
| **(constructor) LinkedList** | construct LinkedList from another LinkedList |
| **(destructor)** | LinkedList destructor |

**Member Types**

|  |  |  |
| --- | --- | --- |
| int listSize | the current size of the list | starts at 0 |
| head | pointer to the head object |  |
| tail | pointer to tail object |  |

**Capacity**

|  |  |  |
| --- | --- | --- |
| Empty | returns true if list is empty | public |
| Size | returns current size of list | public |

**Modifiers**

|  |  |  |
| --- | --- | --- |
| Insert | Inserts element at head of list | public |
| Delete | deletes specified element from list | public |
| popFront | removes the element at the front of the list | public |

**Element Access**

|  |  |
| --- | --- |
| Find | Returns pointer to object passed in if found in the LinkedList |
| Contains | Returns true of the object is in the LinkedList |
| AtIndex | Returns reference to the object at the specified Index location |