**Project: Linked List**

template<typename T>

class LinkedList

{

}

**Linked List**: Linked lists are a linear data structure where the data is linked using pointers.



**Doubly Linked List**: For this project I used a doubly linked list which works almost the same as a linked list. The only difference is you have a head and a tail not just a head. You also have an extra pointer called previous pointer. This is so you can go back and forth on the list. Not just forward like on the linked list.



**Linked list class structure:**

template<typename T>

class LinkedList

{

public:

LinkedList();

~LinkedList();

Node<T>\* get\_head();

Node<T>\* get\_tail();

void insert(T data);

Node<T>\* find(T data\_find);

void remove(T data\_delete);

Node<T>\* maximum();

Node<T>\* minimum();

private:

Node<T>\* head;

Node<T>\* tail;

};

|  |  |
| --- | --- |
| LinkedList() | Constructor |
| ~LinkedList() | Deconstructor |
| get\_head() | Gets the head pointer of the list(Refer back to linked list or doubly linked list diagram example on page 1) |
| get\_tail() | Gets the tail pointer of the list(Refer back to doubly linked list diagram example on page 1) |
| insert(T data) | Inserts data type T data to the linked list |
| find(T data\_find) | Returns a specific data type T data pointer from the list |
| remove(T data\_delete) | Deletes specific data type T data from the list |
| maximum() | Returns the maximum data type T value in the linked list |
| minimum() | Returns the minimum data type T value in the linked list |
| head | Head pointer |
| tail | Tail Pointer |

**Member functions:**

**Usage:** Linked lists are great for organizing different types of data. Get\_head or get\_tail could be used if you’re trying to access the last or newest data that was inserted. This would depend on how you structure your linked list, whether it be inserted from the head or the tail. Find would help you look up the data you want to access. Remove can be used to get rid of data that isn’t needed anymore. The minimum and maximum have an infinite amount of uses, it depends on the type of data that was stored. If you’ve stored debts, you could use maximum to return the biggest debt. Head and tail work as boundaries for the data structure to work within.