# Introduction

A linked list is a data structure that stores information using a series of nodes. The list stores the head property, which contains no data. Each node has a data property and stores a pointer to the next node in the list. If you are using a doubly linked list, then each node will also store a pointer to the previous node. Linked lists provide three main operations: find, insert, and delete. First, you can use the find operation to find the pointer to a specific node. Second, the insert operation allows you to create a new node with specific data. Finally, the delete operation allows you to delete a node by its pointer. I implemented each of these features in my project to create a doubly linked list.

# Programmer’s Guide

My implementation of the doubly linked list uses the generic type “Data”, which allows the linked list to store any type of data. Additionally, it has the following properties:

### Node<Data>\* head

An empty node that marks the beginning of the list. It is used to access other nodes.

### Node<Data>\* tail

An empty node that marks the end of the list. It is used to access other nodes.

## Methods

### LinkedList(Data defaultValue)

The constructor of LinkedList. It creates a head and tail.

Parameters:

* Data defaultValue: The value used to instantiate the head and the tail.

Example:

LinkedList<string> myLinkedList{“”};

### Node<Data>\* getHead()

Returns a pointer to the head node.

Example:

Node<string>\* = myLinkedList.getHead();

### Node<Data>\* Find(Data\* data)

Finds a node by comparing each node’s data value. Returns a pointer to the node, which may be a null pointer if a match was not found.

Parameters:

* Data\* data: A pointer to the data. Used to match the node.

Example:

int myData = 1;

Node<string>\* match = myLinkedList.Find(&myData);

### void Insert(Data data)

Creates a new node and inserts it at the end of the list.

Parameters:

* Data data: the data to make the node with.

Example:

myLinkedList.Insert(“hello world”);

### void Delete(Node<Data>\* node)

Deletes a node from the list. It updates the previous and next node’s pointers accordingly.

Parameters:

* Node<Data>\* node: a pointer to the node to delete.

Example:

Node<string>\* nodeToDelete = myLinkedList.getHead()->getNext();

myLinkedList.Delete(nodeToDelete);