LINKED LIST

In C++, if you want to store a list of related data we usually use arrays. When an array is created we specify the size initially. The compiler then reserves the memory space specified even though we might not need so much memory. If the specified number of elements for array are filled up, we cannot push any more elements in the array. This is the major drawback of using arrays.

To overcome this situation we use the concept of Linked Lists. A Linked list is a data structure made up of nodes which hold the data in your list. Every linked list must have a root node which points to the first data node, each data node contains the data for that item in the list and a reference to the next node in the list.

Each node does not have to be one after the other in the memory because there is a pointer in each node which points to the memory location of the next node. This means we can dynamically add nodes to the list whenever we feel like.

The most basic implementation of a linked list will involve the following operations as a minimum:

1. Inserting an element: To add an element you need to make sure you don’t break the continuous nature of the list. When you insert a node in the beginning of the list, we have to reference the pointer of the root to the new node. When you insert a new node between 2 nodes, you need to copy the reference of the previous node to the new nodes reference. So now the new node points to the previous node’s reference to next node. Now you need to connect the previous node and the new node. So add the reference of the new node to the previous node’s next pointer.

New Node Contains:

1. Data

2. Pointer to next node

Next node contains:

1. Data

2. Pointer to next node

Previous node contains:

1. Data

2. Pointer to next node

This reference is copied into the new nodes pointer

1. Deleting an element: To remove an element you need to copy the reference of the next node of the node that you are deleting and copy it into the previous node’s next pointer.

Deleted Node Contains:

1. Data

2. Pointer to next node

Next node contains:

1. Data

2. Pointer to next node

Previous node contains:

1. Data

2. Pointer to next node

This reference is copied into the previous node’s pointer

3

1. Traversing the list: You start with the root and get to the first node. Then you read the node’s next pointer, if there is a location it points to, go to that location and search for the next location that node is pointing to. Repeat the process till you find that a node in the list is not pointing to any location.
2. Modifying an element: To modify an element we need to traverse through the list till you find the element you want to modify. Once you have found the element, you can modify the element.

Please check out our codes on Linked Lists which will be found in our Downloads Section !