# Linked List

A **linked list** is a linear data structure that consists of a series of nodes connected by pointers.

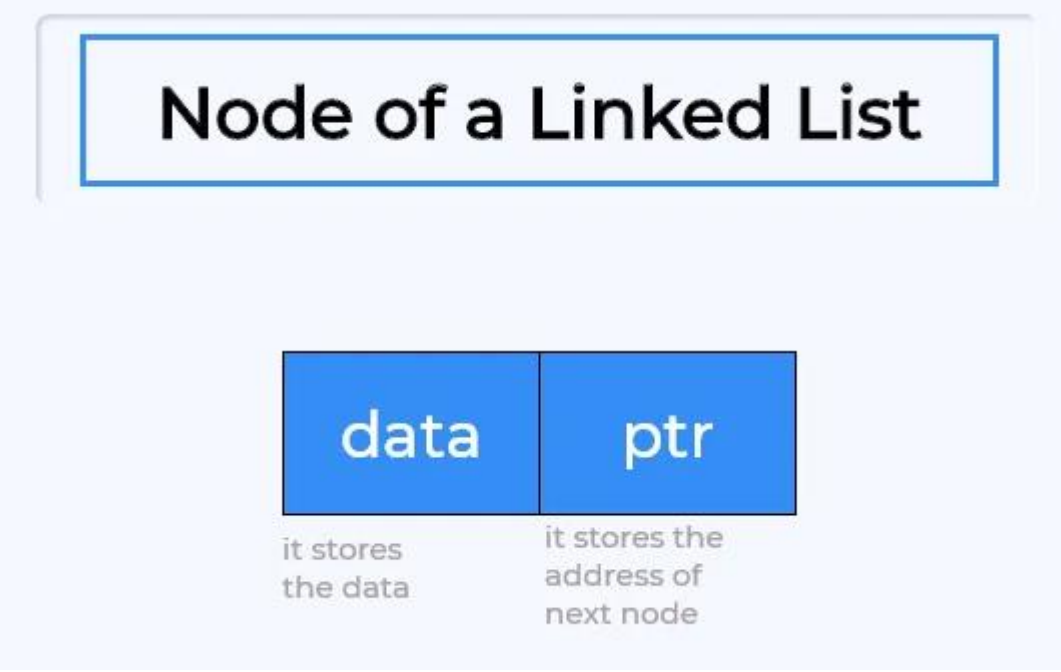
It consists of nodes where each node contains **data** and a **reference (link)** to the next node in the sequence. This allows for dynamic memory allocation and efficient **insertion** and **deletion** operations compared to arrays.

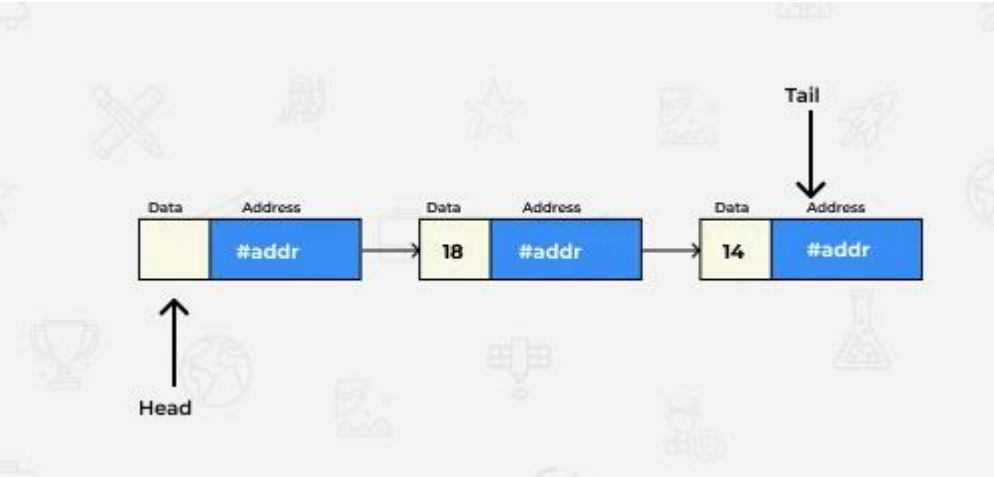
**Linked List Applications**

- Implementing stacks and queues using linked lists.
- Using linked lists to handle collisions in hash tables.
- Representing graphs using linked lists.
- Allocating and deallocating memory dynamically.

Linked List:

- **Data Structure:** Non-contiguous
- **Memory Allocation:** Dynamic
- **Insertion/Deletion:** Efficient
- **Access:** Sequential






Types of Linked List:

a) Singly Linked List
b) Doubly Linked List
c) Circular Linked List

- Singly Linked list
  In C++ the singly linked list can be represented with a class and a **Node** class separately, which has two members, namely data and a **next** pointer which points to the next node.

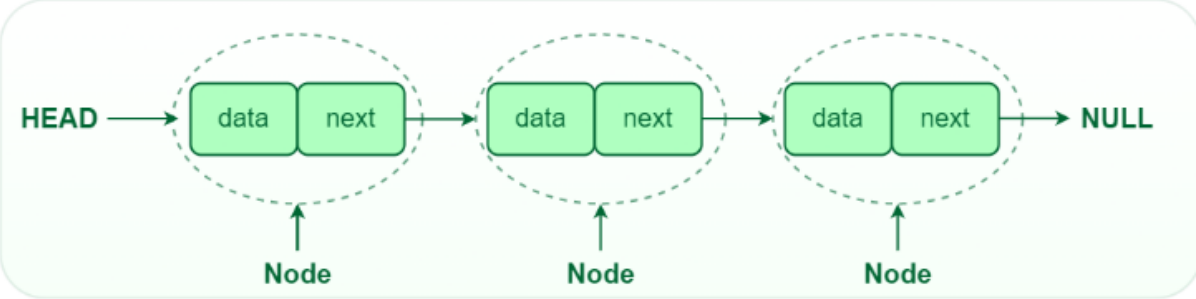
  Operations:
  **InsertNode**: Insertion is done at the end of the list. Follow the steps to insert a node in the linked list.

  - Let's say, **3** is to be inserted on the existing linked list, i.e., **1 -> 2**.The resultant linked list will be **1 -> 2 -> 3.**
  - To insert a new node traverse till the end of the list until NULL node is found.
  - Create a new Node, and link the new node to the last node of the linked list.

**DeleteNode**: Deletion is done using the **index** of the node. Follow the steps to delete a node:

- If the node to be deleted is the head node, store the **head** in **temp** variable. Then update **head** as **head->next**. Delete **temp**.

- If the index of the node to be deleted is greater than the length of the list then return from the function.
- Traverse till the node to be deleted. Delete the node, and link the previous node to the next node of the deleted node.




- Doubly Linked List

**Doubly Linked List** in C++ is very similar to a linked list, except that each node also contains a pointer to the node previous to the current one. This means that in a doubly linked list in C++ we can travel not only in the forward direction, but also in the **backward direction**, which was not possible with a singly linked list.

Inserting a new node in a doubly linked list is very similar to inserting new node in linked list. There is a little extra work required to maintain the link of the previous node. A node can be inserted in a Doubly Linked List in three ways:

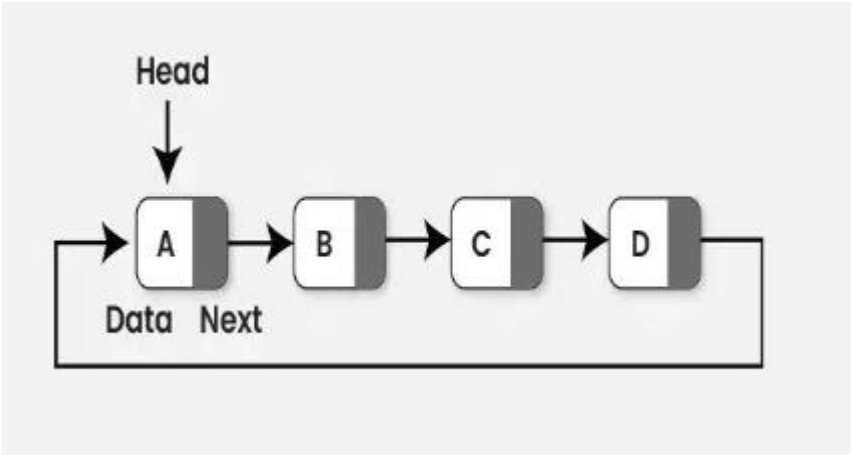
- At the front of the DLL.
- In between two nodes
- At the end of the DLL.

The deletion of a node in a doubly-linked list can be divided into three main categories:

- **Deletion of the head node.**
- **Deletion of the middle node.**
- **Deletion of the last node.**

- Circular Linked List

*The **circular linked list** is a linked list where all nodes are connected to form a circle. In a circular linked list, the first node and the last node are connected to each other which forms a circle. There is no NULL at the end.*


Head
A
B
C
D
Data
Next