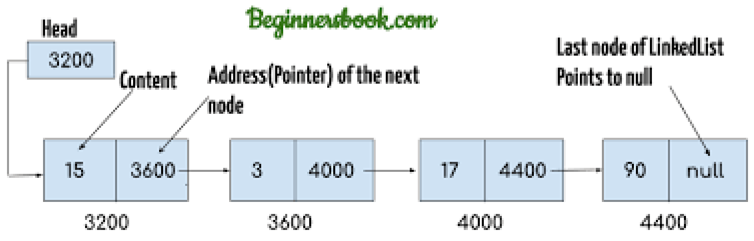
Linked List

Linked List: Linked list is a linear data structure. It is a collection of nodes stored at non-contiguous memory location.

Each node is divided into two parts:

- Data Field: Which contains data.
- Pointer Field: which contains the address of the next node or adjacent node of the list.

Last node of Linked list points to Null.


Beginnersbook.com
Head
3200
Content
Address(Pointer) of the next node
Last node of LinkedList Points to null
15
3600
3
4000
17
4400
90
null
3200
3600
4000
4400

Uses of Linked List

i. The list is not required to be contiguously present in the memory. The node can reside any where in the memory and linked together to make a list. This achieves optimized utilization of space.
ii. list size is limited to the memory size and doesn't need to be declared in advance.
iii. Empty node can not be present in the linked list.
iv. We can store values of primitive types or objects in the singly linked list.

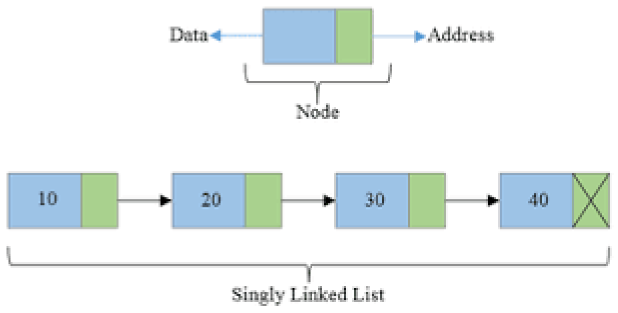
Operations on Linked List.

- **Creation-** To create a new node in the linked list.
- **Insertion**: To add a new node in a linked list.
- **Deletion**: To delete a node from the linked list.
- **Searching**: To search an element(s) by value.
- **Updating**: To update a node.

- **Sorting:** To arrange nodes in a linked list in a specific order.
- **Merging:** To merge two linked lists into one.
- **Traversal**: To traverse all the nodes one after another.

Types of Linked List

1. **Singly Linked List:** A Singly-linked list is a collection of nodes linked together in a sequential way where each node of the singly linked list contains a data field and an address field that contains the reference of the next node.

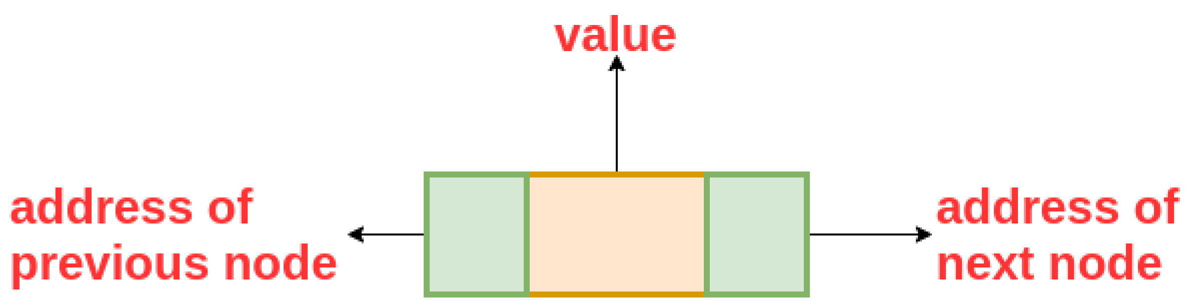


2. **Doubly Linked List:** A Doubly-linked list is a collection of nodes linked together in a sequential way where each node is divided into three parts:

i. **Data Field:** Contains the data.

ii. **Previous Pointer:** address of the previous node in the list.

iii. **Next Pointer:** address of the next node in the list.



Node in DLL

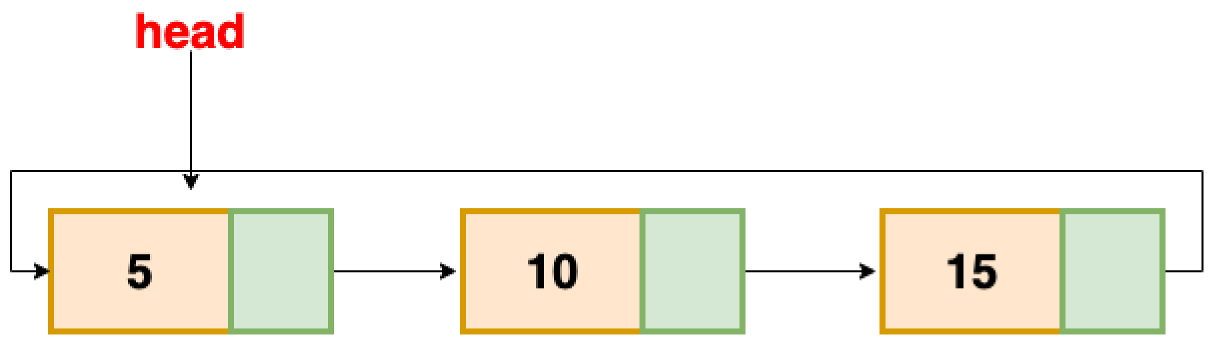
Advantages over Singly Linked List-

- It can be traversed both forward and backward direction.
- The delete operation is more efficient if the node to be deleted is given.
- The insert operation is more efficient if the node is given before which insertion should take place.

Disadvantages over Singly Linked List-

- It will require more space as each node has an extra memory to store the address of the previous node.
- The number of modification increase while doing various operations like insertion, deletion, etc.

3. **Circular Linked List:** A circular linked list is either a singly or doubly linked list in which there are no **NULL** values. Here, we can implement the Circular Linked List by making the use of Singly or Doubly Linked List. In the case of a singly linked list, the next of the last node contains the address of the first node and in case of a doubly-linked list, the next of last node contains the address of the first node and prev of the first node contains the address of the last node.



Circular Linked List

Advantages of a Circular linked list

- The list can be traversed from any node.
- Circular lists are the required data structure when we want a list to be accessed in a circle or loop.
- We can easily traverse to its previous node in a circular linked list, which is not possible in a singly linked list.

Disadvantages of Circular linked list

- If not traversed carefully, then we could end up in an infinite loop because here we don't have any **NULL** value to stop the traversal.
- Operations in a circular linked list are complex as compared to a singly linked list and doubly linked list like reversing a circular linked list, etc**.**