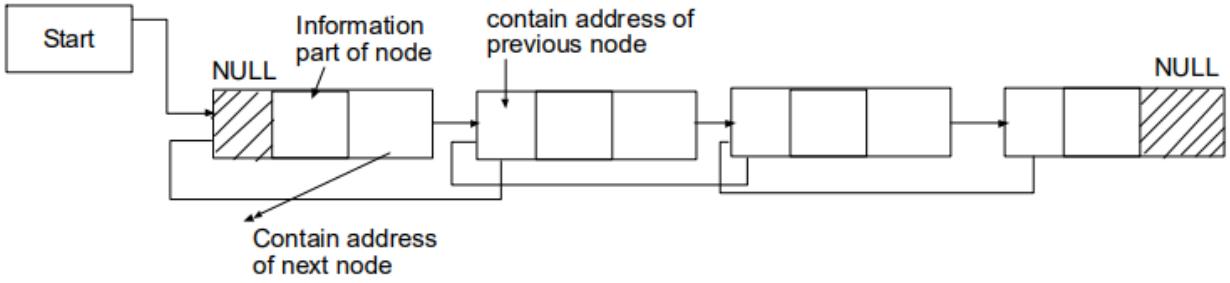
# Doubly Linked List

In single linked list we can traverse only in one direction because each node has address of next node only. Suppose we are in the middle of linked list & we want to operation with just previous node, then we have no way to go in previous node. And we'll again traverse from starting node which is time consuming./ To overcome this problem we use double linked list.

In DLL each node has address of previous & next node along with data.




Algorithm to Insert a new node at the begining of doubly linked list:

Step:1 Create a new node

Step:2 ~~Set~~ Assign an item to the item field of new node

Step:3 Assign previous and next linked to NULL.

Step:4 Make the next link of the new node to point the first node of the list and make previous link of the first node to point the new node.

Step:5 ~~St~~ set the head pointer to point the new node.

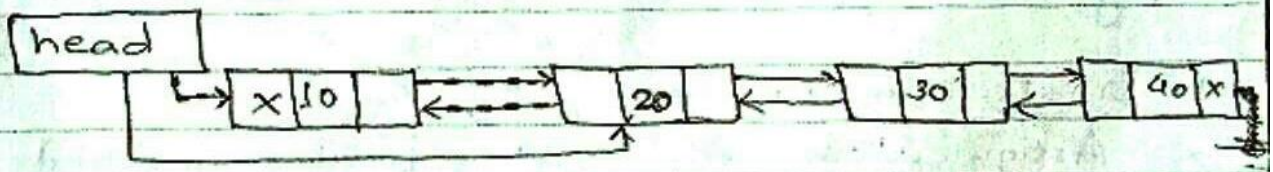
Step:6 Set previous link of new node as head pointer.

## Algorithm to insert a new node at the end of DLL.

Step. 1 ; Create a new node
Step. 2 : Assign an item to the item field of new node
Step. 3 : Set PREVIOUS and NEXT link of new node as NULL.
Step. 4 : If the list is not empty then traverse the list till the last and make the NEXT link of the last node to point to the new node and previous link of new node to point the last node.

## Deleting a node from the beginning (head position) of DLL:

1. If the list is empty print "empty list" and exit.
2. Otherwise, make the head pointer to point to the second node and if the second node is not null then make its ~~LFT~~ PREVIOUS link to point to NULL.
3. Free the first node.




## Deleting a node from the end of DLL:

1. If the list is empty print "empty list" and exit.
2. Otherwise, traverse the list last but not one and make the next link of the node to point to NULL.
3. Free the last node.