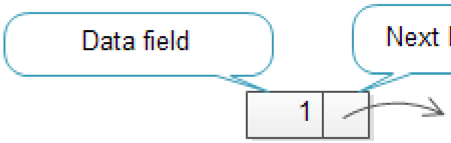
Linked list

القوائم المرتبطة

المحاضرة الخامسة **Data structure** **م.وجدان ياسين**

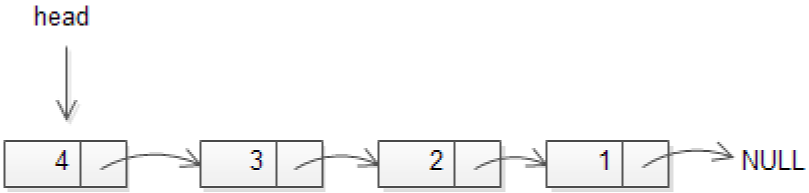
A linked list is a data structure that consists of sequence of nodes. Each node is composed of two fields: **data field** and **reference field** which is a **pointer** that points to the next node in the sequence.



Linked List Node

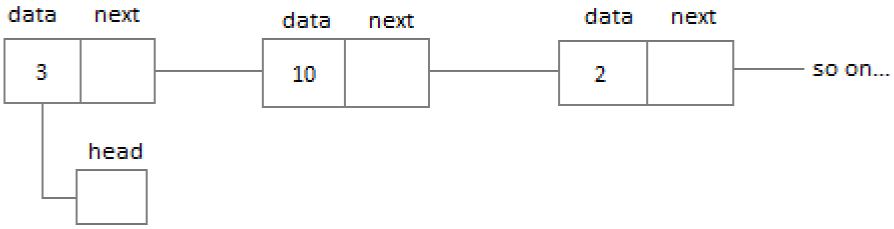
Each node in the list is also called an element. The reference field that contains a pointer which points to the next node is called **next pointer** or **next link**.

A **head** pointer is used to track the first element in the linked list, therefore, it always points to the first element

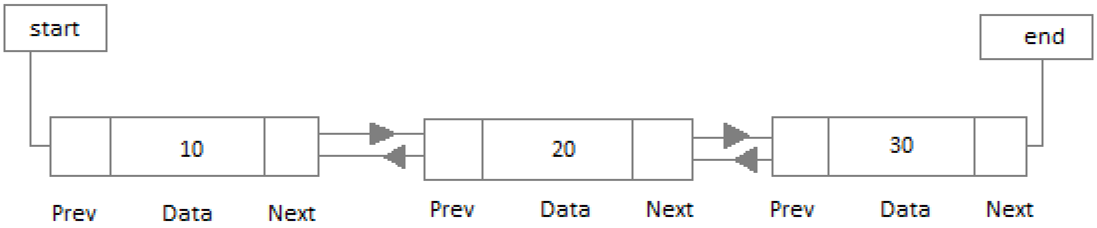


Types of Linked Lists:

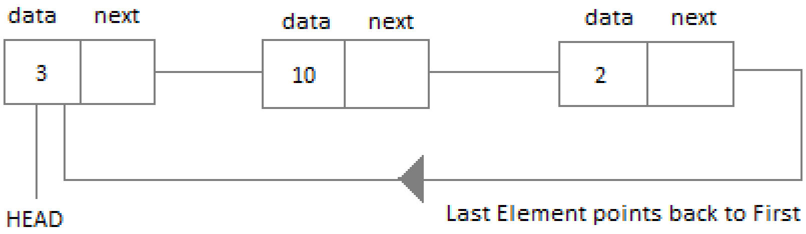
- **Singly Linked List :** Singly linked lists contain nodes which have a data part as well as an address part i.e. next, which points to the next node in sequence of nodes. The operations we can perform on singly linked lists are insertion, deletion and traversal.



- **Doubly Linked List :** In a doubly linked list, each node contains two links the first link points to the previous node and the next link points to the next node in the sequence.



- **Circular Linked List :** In the circular linked list the last node of the list contains the address of the first node and forms a circular chain.

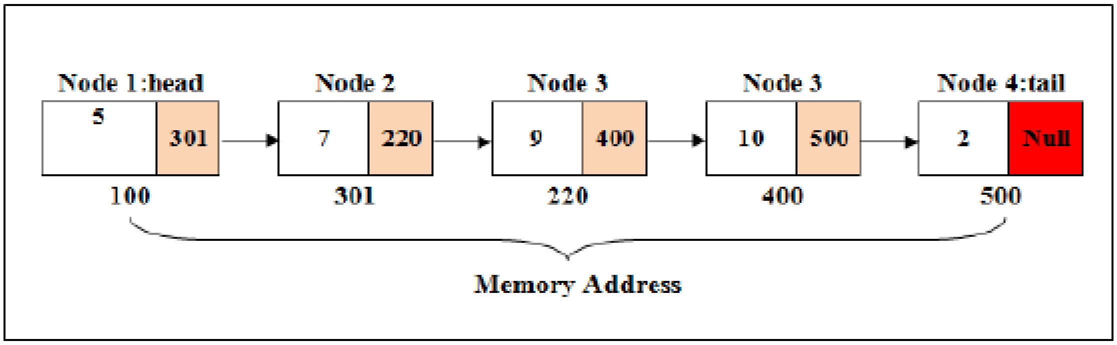


Linked List implementation in C++:

We can model a node of the linked list using a **structure** as follows:

```
struct node{
   int  data;
   node*  next;
};
```

- Data: stores the information
- next: pointer holds the address of the next node



Operations Of a Single Linked List:

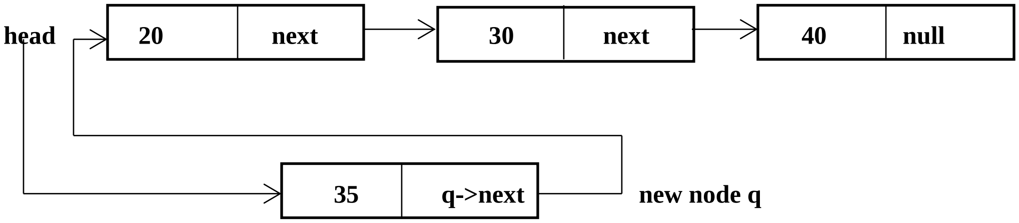
1. Insert – Inserts a new node to the linked list.
2. Delete – Deletes any node from the linked list.
3. Find – Finds any node in the linked list.
4. **print** – Prints the linked list

Create a Linked List (of one node):

```
  Struct node {

      int data;

      node* next;

}*p, *q,  *z, *head;

Void  creatlist( int  value)                              //enter value 20

{

    p=new node;

    p->data=value;

    p->next=NULL;

    head=p;

}
```

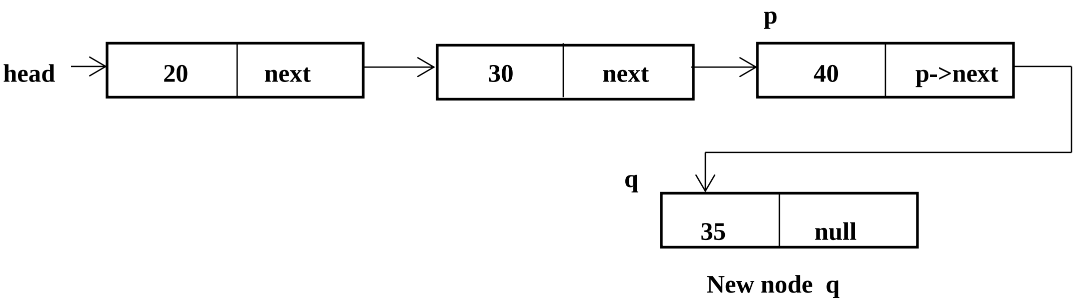
Inserts Node at the beginning of the Linked List:

```
void   insertfirst( int  value)                           //insert value 35
{
   q=new node;
   q->data = value;
   q->next=head;
   head=q;
}
```



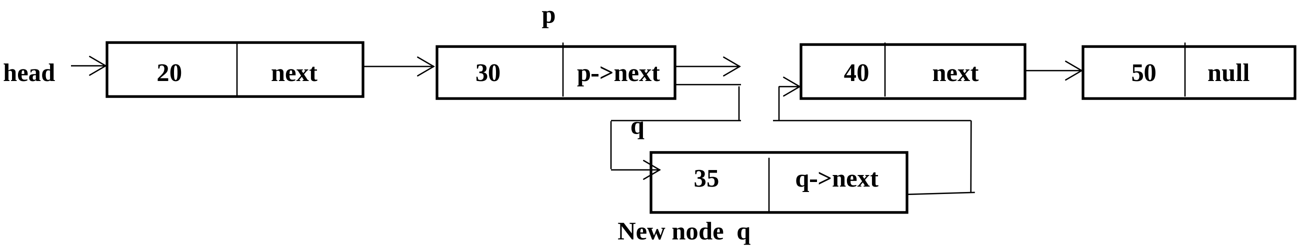
Inserts Node at the End of the Linked List:

```
void insertlast( int value)                    //insert   value  35
{
   q=new node;
   q->data = value;
   p->next=q;
   q->next=null;
}
```



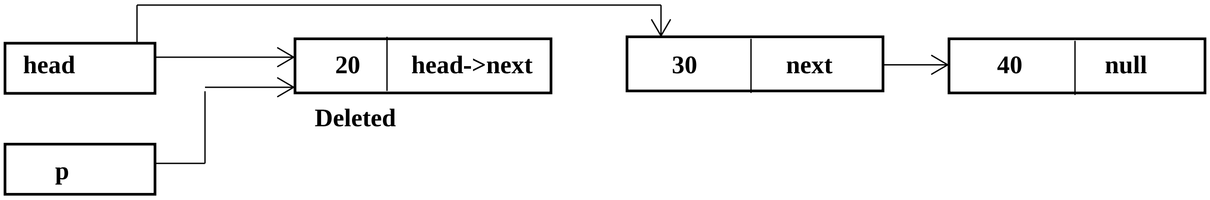
Inserts Node at the Middle of the Linked List:

```
void insertmiddle( int value)                    //insert value  35
{
   q=new node;
   q->data = value ;
   q->next=p->next;
   p->next=q;
}
```



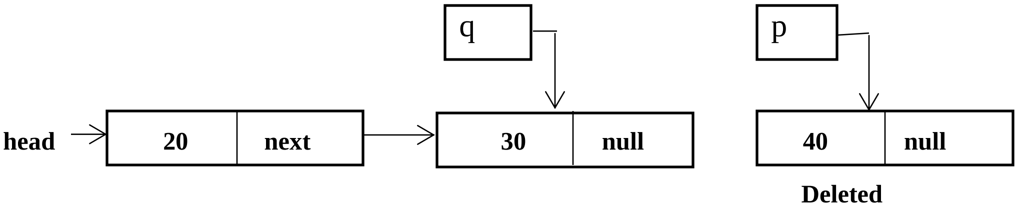
Deletes Node from the beginning of the Linked List:

```
void deletefirst()
{
   p=head;
   head=(head)->next;
   delete(p);
}
```



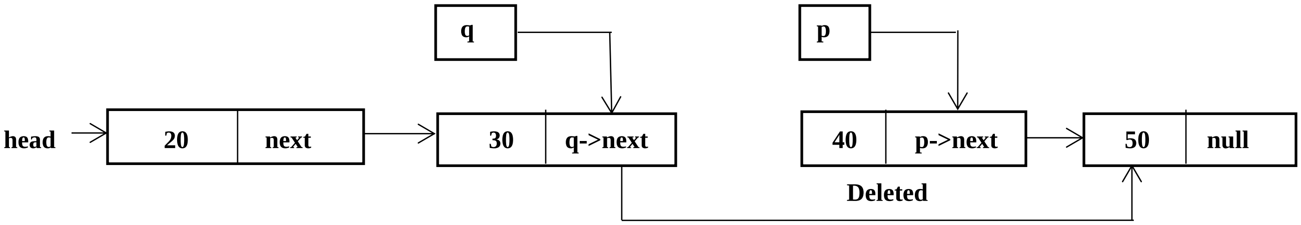
Deletes Node from the End of the Linked List:

```
void deletelast()
{   p=head;
    if(p->next==NULL)
       { delete(p); head=NULL; }
   else
       { while(p->next!=NULL)
             { q=p; p=p->next; }
         q->next=NULL;
         delete(p);
       }

}
```



Deletes Node from the Middle of the Linked List:

```
void deletemiddle( int value)                    // delete value  40
{   p=head;
    while(p->data!=value)
       { q=p; p=p->next; }
    q->next=p->next;
    delete(p);
}
```



Prints the Linked List:

```
void displaylist( )
{
  z=head;
  cout<<"the list:"<< endl;
  while(z)
  {
    cout<< endl<< z->data;
    z= z->next;
  }

}
```

Sorting linked list :

```
 Void sorting()
{
    Node * temphead;
    Node * tempnode;
   Int tempdata=0;
   For (temphead=head;temphead!=NULL;temphead=temphead->next)
    {
     For (tempnode=temphead->next;tempnode!=NULL;tempnode=tempnode->next)
     {
         If (temphead->data > temnode->data)
        {
            Tempdata=temphead->data;
           Temphead->data=tempnode->data;
          Tempnode->data=tempdata;
        } } } }
```