Data Structure
Chapter 3: Linked List

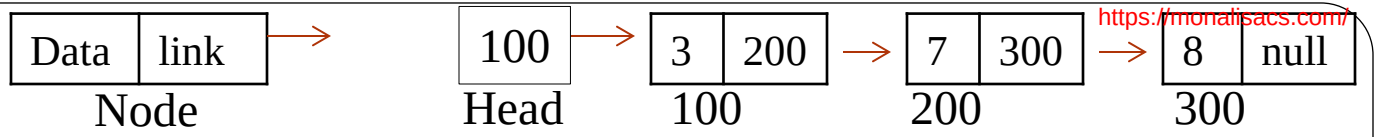
GATE CS Lectures

By

Monalisa Pradhan

- **Section 4: Programming and Data Structures**
  Programming in C. Recursion.Arrays, stacks, queues, linked lists, trees, binary search trees, binary heaps, graphs.

- Chapter 1:Arrays
- Chapter 2: stacks, queues
- Chapter 3: linked lists(Singly,Circular,Doubly linked list,Doubly Circular linked list)
- Chapter 4: trees, binary search trees, binary heaps
- Chapter 5: graphs



- **Singly Linked List:**
- Linked list is collection of node.
- Each node have two part data and address of next node.
- An extra pointer head or start points to the first node.
- Last node address part is null
- Node can store in memory non contiguous manner.
- Extra storage require to store pointer with each data.
- It is of dynamic size.

Self referential structure

```
struct node
{
char data;
struct node * link;
};
struct node *head;
```

Ex: Count number of node in a single linked list.

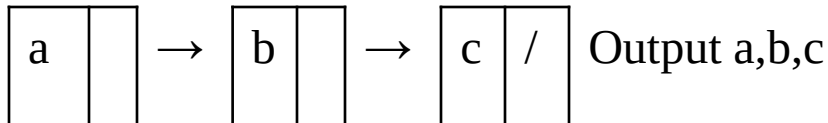
```
int count ()
    { struct node *p=first;
      int count=0
      while (p)
            {++count;
            p=p→link; }
      return count}
```

- Two types of symbol are used for linked list
- Indirect selector (→) :User friendly operator P→data
- Direct selector (.):machine friendly operator (*p).data
- while (p) =>while (p!=null) or while (!p)=> while (p=null)
- while(p→link)=>while (p→link !=null) or while(!p→link)=>while (p→link ==null)
- **Disadvantage of single linked list**
- The link part of last node is not utilized
- The stepping backward is not possible only forward traversal is possible.

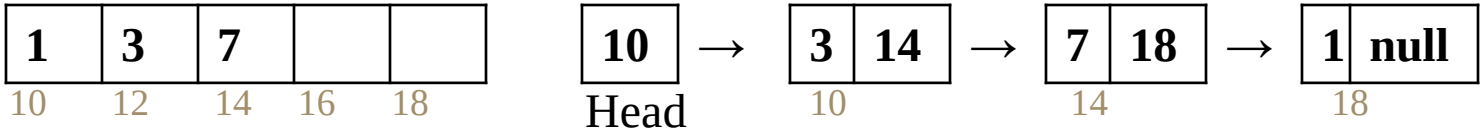
Ex:print data value of single linked list

```
Void main()
{
Do (first);
}
```

```
Void Do (struct node *p)
{if (p)
  {printf("%c",p→data);
   Do (p→link);
   }  }
```

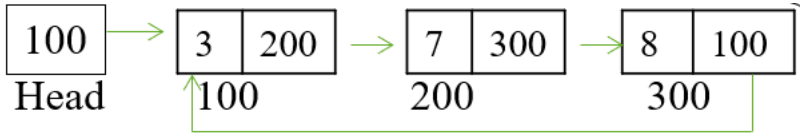


Output a,b,c



	Array	Linked list
1.Cost of accessing an element	O(1)	O(n)
2.Memory utilization	not efficient	efficient
3.Memory allocation	static	dynamic
4.Cost of Insertion or Deletion		
a)at beginning	O(n)	O(1)
b)at end	O(1)	O(n)
c)at i^{th} position	O(n)	O(n)
5.Searching	Linear & Binary	Linear

- **Circular Linked List:**
- Advantage: Last node link part utilize.
- Stepping backward possible
- Disadvantage: It may fall into infinite loop



- **Doubly linked list:**
- Self referential structure
- *struct Dnode*

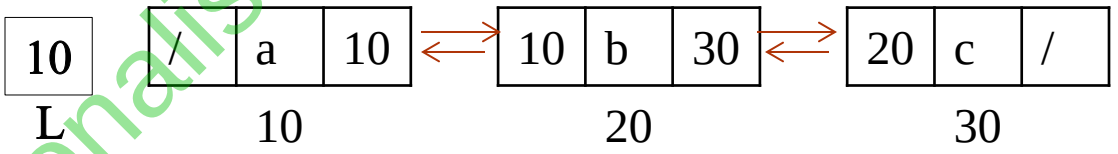
```
{struct Dnode *Llink;
   char data;
   struct Dnode *Rlink;
}; struct node *head;
```

- Insert a node to the right of ‘m’
- where m is some middle node and all steps should be according to new node.
- ***Void insert middle(char x)***

```
{   struct Dnode *n=get Dnode();
      n→data =x;
      n→Llink=m;
      n→Rlink=m→Rlink;
      n→Llink→Rlink=n;
      n→Rlink→Llink=n;     }
```

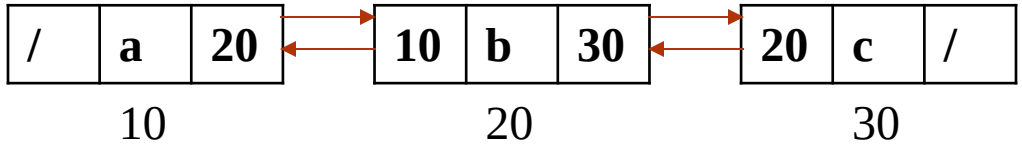
- ***Void insert leftmost(char x)***

```
{       struct Dnode *n=get Dnode();
        n→data =x;
        n→Llink=null;
        n→Rlink=L;
        L→Llink=n;
        L=n;     }
```



- ***Void delate leftmost( )***

```
{       l=l→Rlink;
        free (l→Llink)
        l→Llink=null; }
```



- ***Void delate middle( )***

```
{   m→Llink→Rlink=m→Rlink ;
    m→Rlink→Llink=m→Llink;
    free (m)  }
```

- **Circularly Doubly linked list:**
- Circular Doubly Linked List has properties of both doubly linked list and circular linked list
- The last node points to first node by next pointer and also the first node points to last node by previous pointer.