# Lecture - 20

**Linked list:**

- Insertion easy
- Deletion easy
- No wastage of memory

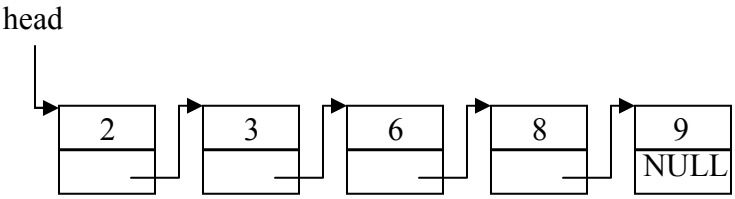
**3 types of linked list:**

- Linear linked list
- Two way linked list
- Circular linked list

**Operation of a linked list:**

- Insertion
- Deletion
- Searching
- Traversing

**Linear linked list:**




```
typedef struct node{
        int a;
        void *next;
}list;
list *head, *current, *previous, *temp ;
```

**Insertion in a linear linked list:**

```
int x , n ;
head = NULL ;
printf("How many element do you want ?");
scanf("%d", &n);
do{
        printf("Please enter a value : ");
        scanf("%d",&x);
        if(head = = NULL){
                head = ( list * ) malloc (sizeof(list));
                head → a = x ;
                current = head ;
        }
        else{
                temp = ( list * ) malloc (sizeof(list));
                temp → a = x ;
                temp → next = NULL ;
                current → next = temp ;
                current = temp;
        }
}while(-- n) ;
```

**Insertion in a shorted linear linked list:**

```
int x, flag = 1 ;
printf("Please enter a value : ");
scanf("%d",&x);

temp = ( list * ) malloc (sizeof(list));
temp → a = x ;

if(!head){
        head = temp ;
        head → next = NULL ;
}
else{
        if(head → a >= x)
                temp → next = head ;
                head = temp ;                          → first insert
        else{
                previous = head ;
                current = head → next ;
                while((current → a < x){
                        previous = current ;
                        current = current → next ;

                        if(current = = NULL){
                                flag = 0 ;

                                current → next = temp ;
                                current = temp ;
                                current → next = NULL ;    → last insert

                                break ;
                        }
                }
                if(flag){

                        previous → next = temp ;
                        temp → next = current ;
                        previous = temp ;                  → middle insert

                }
        }
}
```

**Deletion in a shorted linear linked list:**

```
if(!head)
        printf("\nThere are no data in the list");
else{
        previous = head ;
        current = head → next;
        if(head → a = = x){

                head = head → next ;          ──► first delete
                free(previous) ;

        }
        else{
                int flag = 1 ;
                while(current → a != x){

                        previous = current ;
                        current = current → next ;

                        if((current → next = = NULL){
                                flag = 0;

                                if((current → a = = x){

                                        previous → next = NULL ;   ──► last delete
                                        free(current) ;

                                }
                                else{
                                        printf("\nData not found in the list");
                                break;
                        }
                }
                if(flag){

                        previous → next = current → next ;   ──► middle delete
                        free(current);

                }
        }
}
```

**Searching in a linear linked list:**

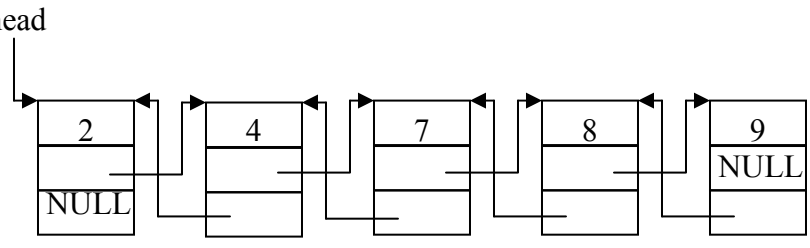
```
int  p = 1, flag = 1;
current = head ;
while(current → a != x){
        current = current → next ;
        p++ ;
        if(current = = NULL){
                flag = 0 ;
                break ;
        }
}
if(flag)
        printf("\nThe value is in the index number %d", p);
else
        printf("\nThe value is not found");
```

**Traversing in a linear linked list:**

```
int  p = 2 ;
if(!head)
        printf("Sorry! list is empty");
else{
        current = head ;
        printf("\n1st element of the list is : %d", current → a);
        while(current → next != NULL){
                current = current → next ;
                printf("\n%d th element of the list is : %d", p++, current → a);
        }
}
```

**Two way linked list:**

```
struct node{
        int a;
        void *flink;
        void *blink;
}list;
```




**Circular linked list:**

```
struct node{
        int a;
        void *next;
}list;
```