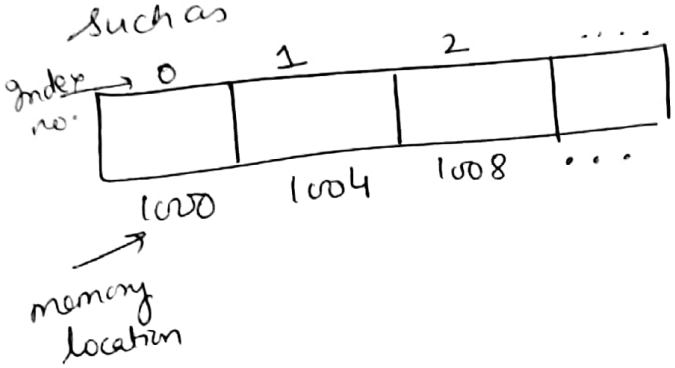
Lecture - 14

Array Vs Link List

Arrays

① Contiguous Memory location.

such as



② We can directly access an element with help of its index no.

eg a[1] ← index no.

a is name of array

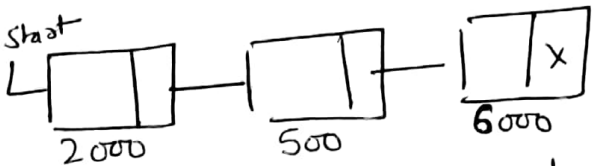
③ As in array elements are stored at successive or contiguous memory location. Therefore, We can compute the location of middle element.

i.e In array Both Linear & Binary search are possible.

④ Relatively Expensive to insert and delete elements in an array.

Link List

① Un Contiguous memory location. such as



i.e nodes are scattered at different memory location.

or you can say that nodes can be allocated anywhere in the memory.

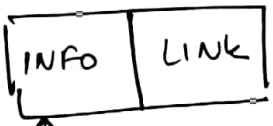
② In case of link list we cannot directly access the 5th node or 3rd node

③ As list are not stored in successive memory location. So we cannot find the location of middle node of a list. Therefore Binary Search is not possible.

④ As elements did not occupy adjacent space in memory so it make easier to insert & delete elements in the list.

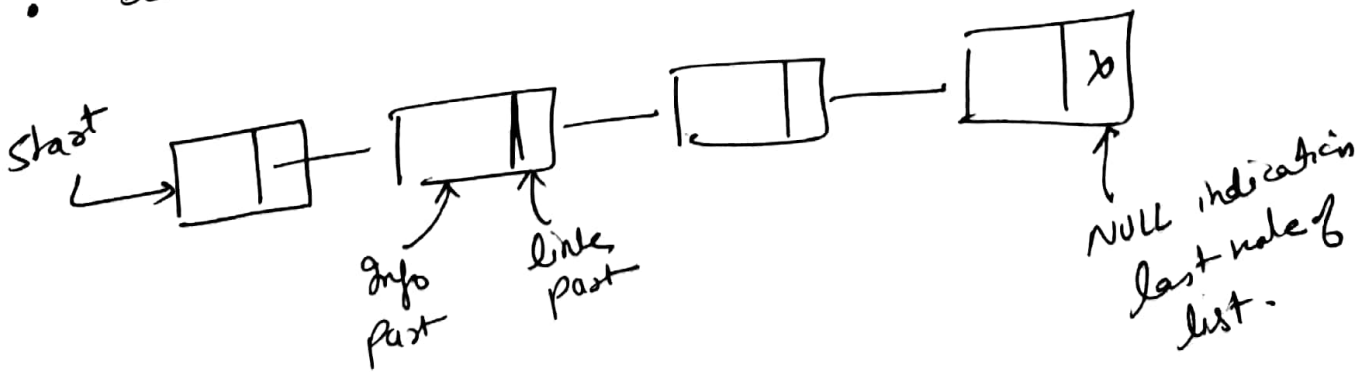
Linked List :-

- Is a linear collection of data elements called nodes
- Linear order is given by means of pointers.
- Each Node is divided into two parts
- First part :- Contains the information of the element.
- Second part :- Contains the address of the next node in the list.



- It may contain an entire record of data items. such as name, address, price

- Last node contains a special value called Null pointer [• you can use 0 or -ve value to indicate NULL]
- Start or Head pointer which contains the address of the first node in the list
- ~~Start~~ If the list has no nodes. Such a list is called the null list or empty list.
- Denoted by Start = NULL.



Operations :-

- Insertion
- Deletion
- Traversing
- Searching
- Sorting
- Merging.

Considering a small code for creating just 2 nodes & then how can we connect them.

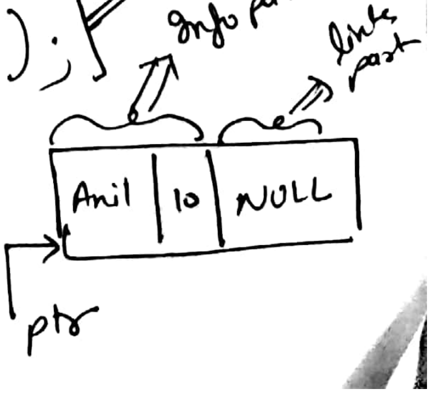
```
typedef struct list
{
    int roll no ;          } => Info part
    char name [20] ;       }
    struct list * next ;   } => Link part

} node ;
```

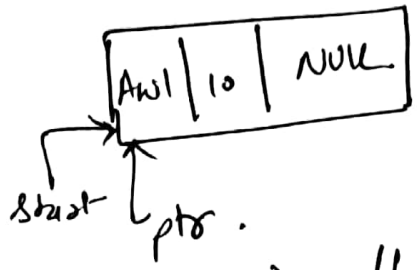
↳ Now this will act as a data type of struct list type.

```
main ( )
{
  node * start , * ptr , * temp ;

ptr = (node *) malloc ( size of (node)) ;   => creating 1st node
  strcpy ( ptr -> name , "Anil" ) ;
  ptr -> roll no = 10 ;
  ptr -> next = NULL ;
```



start = ptr; //

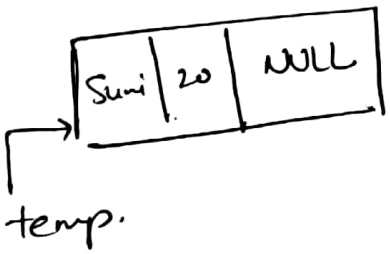


temp = (node *) malloc (size of (node)); // creating 2nd node

strcpy (temp → name, "Sumit");

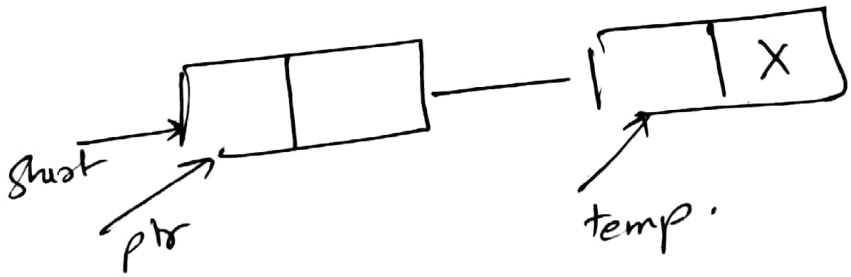
temp → roll no = 20;

temp → next = NULL



// Now we want to connect these two nodes

ptr → next = temp.; // connected these nodes.



In the same way, Now you can try to create list of n number of nodes.