MySirG's Youtube

15, March, 2021

DATE

LINKED List

why linked list - ?

- Linked list is a data structure used for storing Collection of data

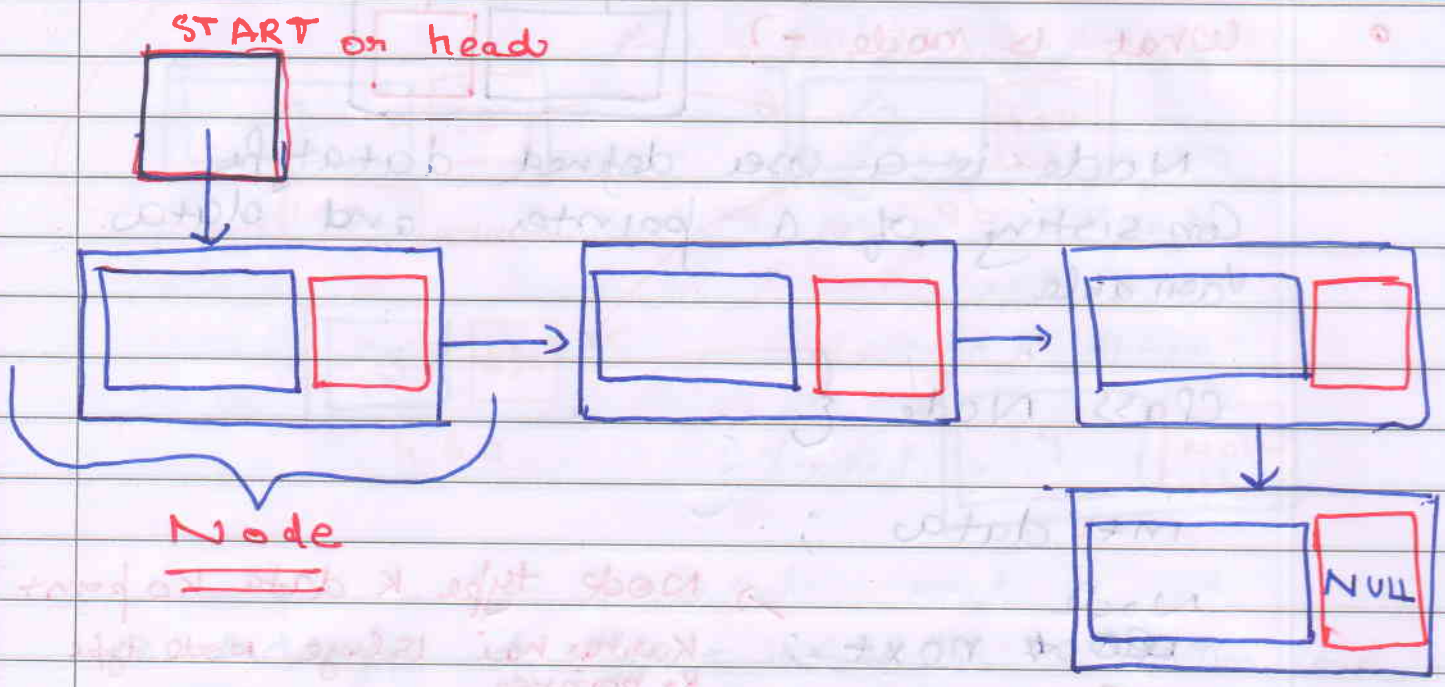
- Properties →

- Successive Elements are connected by Pointers
- Last element Points to null
- It Can grow or shrink during execution of a program
- It Can be Made just as long as required
- It does not waste Memory space

Representation →

PTO

DATE

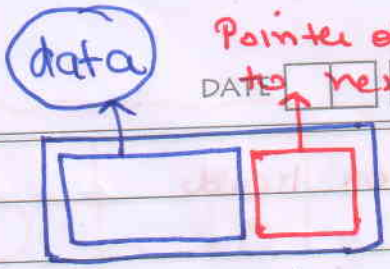


- Three main operations on linked list →
  - Insertion
  - Deletion
  - Traversing

Need of linked list -?

Size of array cannot be increased during runtime.

Size of linked list can be increased / shrink during execution



- What is node -?

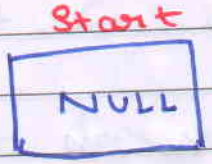
Node is a user defined datatype consisting of a pointer and data variable

```
Class Node {

int data ;
Node * next ;
}
```

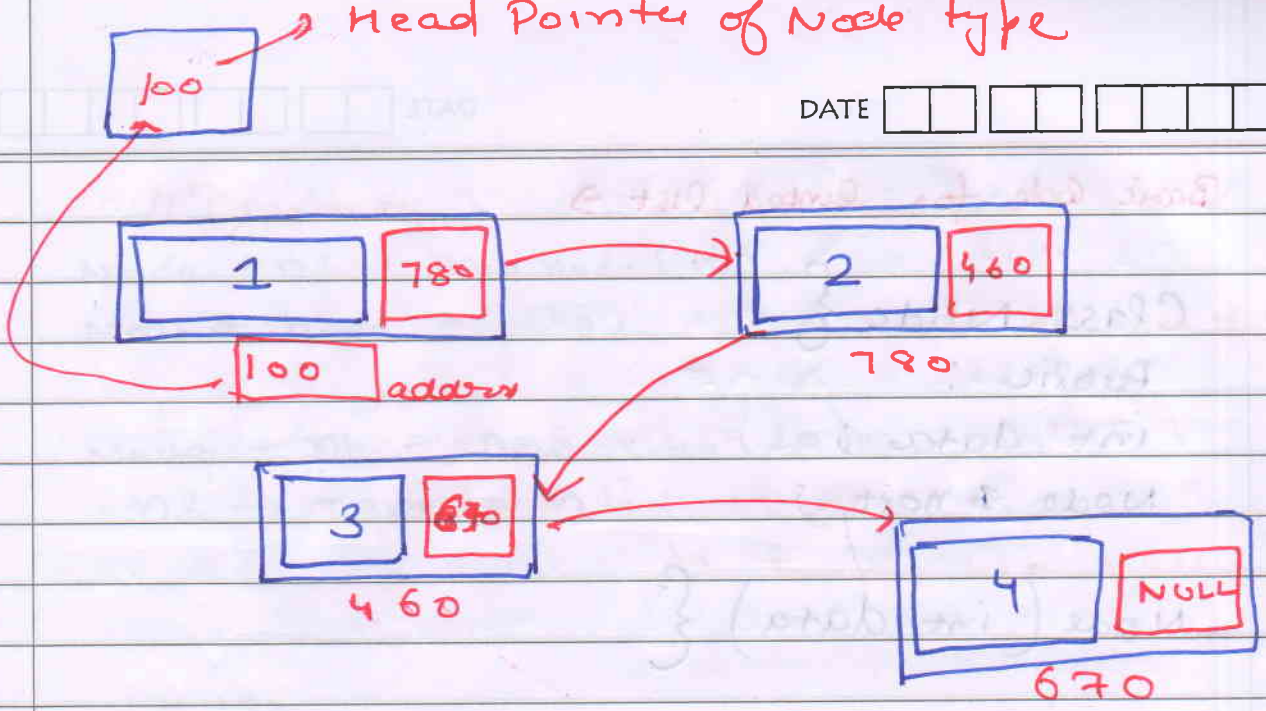
→ Node type k data ko point karwa hai isliye node type k pointer

↑ since it will point to a node variable

- Why Start initially contain null -?



Because if we will not put Null then it will point to any other Memory location and if it is Pointing another memory location then we cannot say it is empty but at the beggning the Linked list is empty.



DATE

→ Address of first Node is Stored in Head

⇒ Previous Node Contains Address of next node

→ Last Node Points to NULL.

✡ Hamare Pass koi inbuild data type nai hai in which we can store int variable and a Pointer variable so we need to create our own datatype using classes

In Case of Linked list we create Node Consisting of a int variable and a pointer variable

Role of int variable → TO Store data

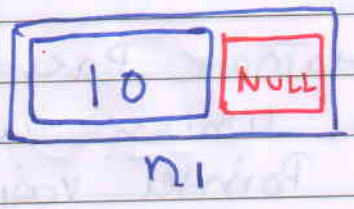
Role of Pointer variable → To Store address of next node.

DATE

Basic Code for Linked List →

```
Class Node {
Public :
int data ;
Node *next ;

Node ( int data) {

this → data = data
next = NULL ;

}

};
```

```
int Main ()
{ // initialisation of Node
   Node n1 (10) ;
   Node * head = &n1 ;
      node n2 (20) ;
// Connecting both the nodes →
n1. next = &n2
// Print data of n1 using head
Cout << head → data.
```



head is storing the address of first node

DATE

// Dynamically

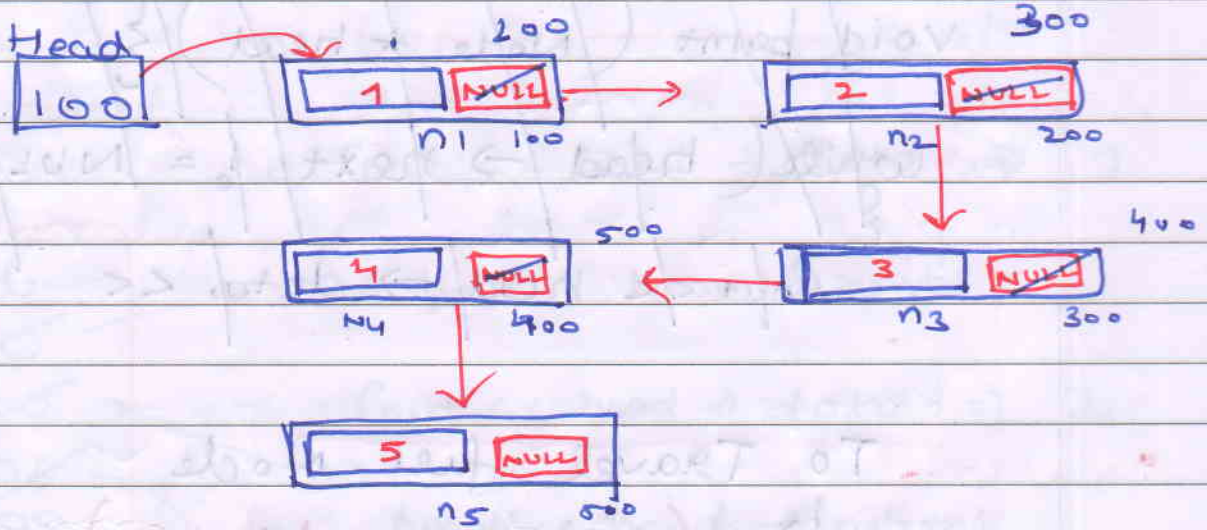
```
Node *n3 = new node(10);
Node * Head = n3;
```

→ Storing address of first Node

```
Node * n4 = new node(20);
n3 → next = n4;
```

↳ Joining both nodes

Q) Create a linked list whose,



Code →

```
Int Main() {
 Node n1(1);
Node * Head = &n1;

 Node n2 (2);
 Node n3 (3);
 Node n4 (4);
 Node n5 (5);
```

DATE

```
n1.next = &n2;
n2.next = &n3;
n3.next = &n4;
n4.next = &n5;
// Till Now linked list is Created
// Now printing our linked list
Print ( Head );
}

void print (Node * Head){
while( Head -> next != NULL)
{
Cout << head -> data <<
```

To Travel full Node
↳ (head != NULL)

To Stop at last node
↳ (head -> next != NULL)

PAGE

classmate

DATE

```
void print(node * head){
while(head != NULL){
  Cout << head -> data << endl;
    head = head -> next;
  }
}
```

So what are we doing here ?

Cout << head -> data => 1

Head
100
200
300
400
NULL

head = head -> next;

Cout << head -> data => 2

head = head -> next;

Cout << head -> data => 3

head = head -> next;

We are repeating the same task until head becomes = NULL So we can use loops to print our linked list.

Basically we are overwriting our head in above case

classmate

PAGE

DATE

Note → There is a problem in Void print()

Basically I am Changing values of head and at the end of while loop Head will point to null and if head will point to null we cannot print our linked list again since our head is ~~&~~ lost in one traversal.

What Should be done -?

Ideally, we Should Create a Copy of head pointer into a temporary variable and use that variable in place of head this will not disturb the address of first Node in head.

Rectified Code for traversal ↓

```
Void print ( Node * head ){

  Node * temp = head;

  While ( temp != NULL ){

    Cout << head → data << " ";
    temp = temp → next;
  }

}
```

classmate

PAGE

16, March, 2022

DATE

TAKING INPUT

will return head

```
Node * takeinput() {
    int data;
    cin >> data;
    Node * head = NULL;
    while(data != -1) {
        Node * newnode = new node(data)
        if (head == NULL) {
            head = newnode;
        }
        else {
            Node * temp = head;
            while(temp -> next != NULL) {
                temp = temp -> next;
            }
            temp -> next = newnode;
        }
        cin >> data;
    }
    return head;
}
```

-1 is used for Terminator purpose

// initially Head is NULL So this line will be executed and Head will be assigned the address of new node

this will only get executed one time

temp will always start from begning

temp will go to last Node

address of newnode will be inserted in last Node

data entered

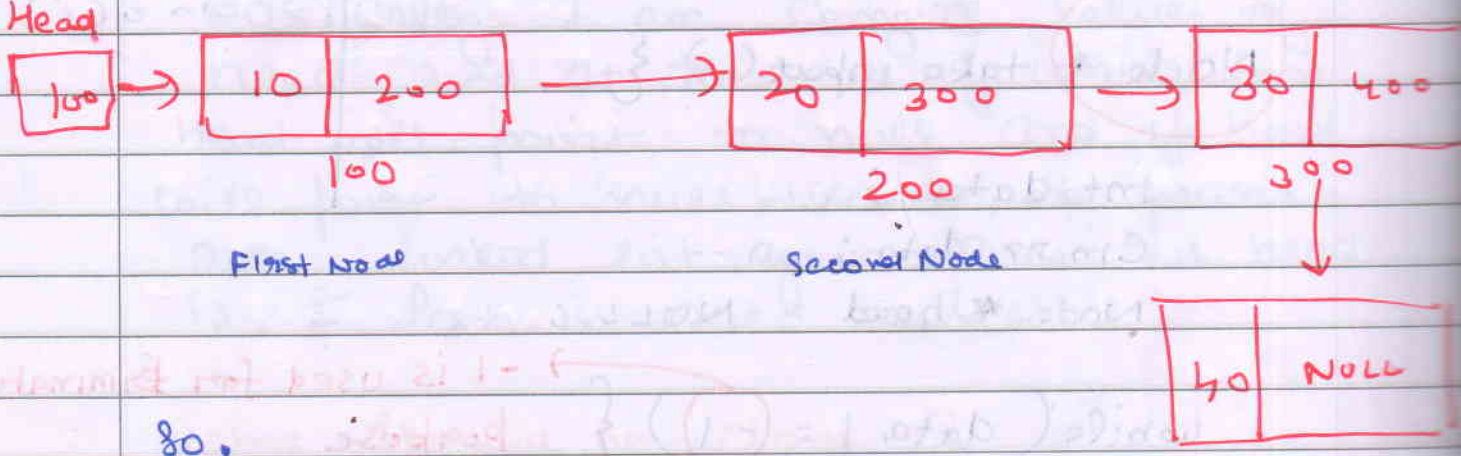
Head is returned

IN ARRAYS.

```
cin >> n
for (i = 0 — n)
    cin a[i]
```

DATE

DRY RUN



So,

(1) Initially newnod will be created

(2) Address of newnode is inserted in Head (only one)

(3) Now, a new node will be created and data will be inserted into it

(4) Now I want to insert address of ~~first~~ second node into first Node

↓

We will make a temp and point it to Head

We will iterate from first Node to that Node which has Null in next.

Now temp will point to that Memory block

Put the address of New created node in Next of temp

classmate ie → temp → next = newnode

PAGE

DATE

3, March, 2022

NOTE → Our previous code has complexity $O(n^2)$ which is a bad complexity

```
while (data != -1) {        → first loop
else →
    while (temp → next != NULL)   → second loop
```

$$O(n \times n) = O(n)^2$$

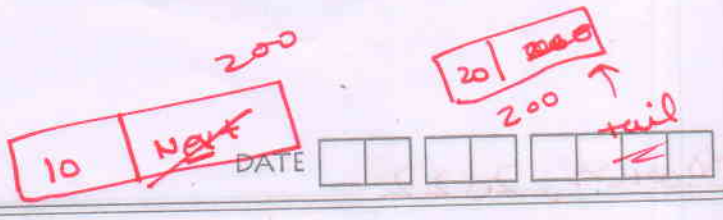
Taking Input In Better Way

```
~~Void~~
Node * takeinputbetter {

    int data;
    Cin >> data;
Node * head = NULL;   Node * tail = NULL;

while (data != -1) {

Node* newnode = new node (data);

If (Head == NULL) {

    head = New node;
    tail = new node;
```

} tail and head will point to first Memory block initially.

}

PAGE



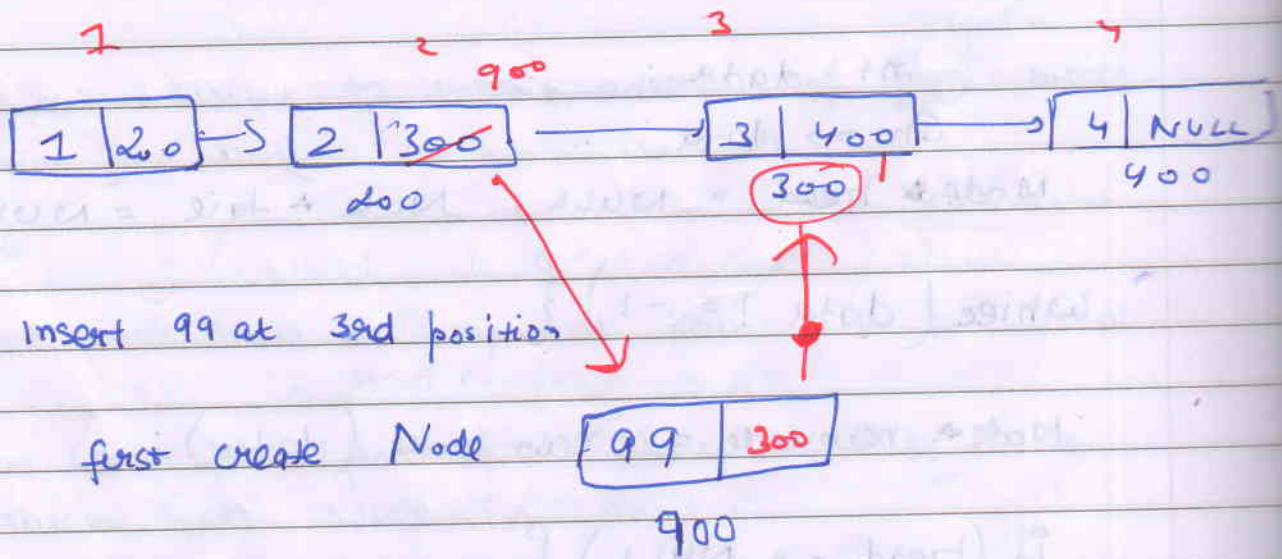
DATE

```
else {
    tail -> next = newnode;
    tail = tail -> next ;
```

↳ tail will point to next node address.

```
}
    cin >> data ;
}
return head ;
}
```

Insert Node at ith position



DATE

Code →

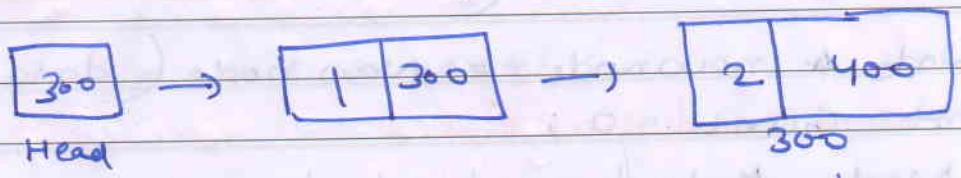
```
Node * insertNode ( Node * head , int position , int data )

Node * newnode = New node ( data ) ;
int Count = 0 ;
Node * temp = head ;

if ( position == 0 ) {
    newnode → next = head ;
    head = New Node ;
    return head ;
}

while ( temp != NULL && Count < i - 1 ) {
    temp = temp → next
    Count ++ ;
}

if ( temp != NULL ) {
    Node * a = temp → next ;
    temp → next = Newnode ;
    newnode → next = a ;
}

Return Head ;
}
```

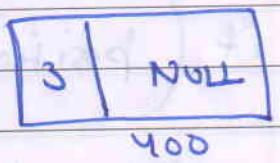
Node * a = head
head = newnode
newnode → next =
node * a = temp → nex
OR temp → next = newn
newnode → next =

DATE

During insertion

* When (position == 0)





① Create a Node and put data

Put Head in

and point head to newnode.

* # Deletion in Linked List

Code ⇒

classmate

PAGE

31 March, 2021

DATE

```
Delete_node(node * head, int position)
{
    Node * temp = head;
    int Count = 0
    while (temp != NULL && Count < position-1)
    {
        temp = temp -> next;
        Count ++;
    }
    if (temp -> next != NULL)
    {
        Node * a = temp -> next;
        Node * b = a -> next;
        temp -> next = b;
        delete (a);
    }
    return head;
}
```

if position exceed length of linked list we will come out and not do anything

Because inside it I am trying to access temp -> next -> next

temp -> next = temp -> next -> next

DRY RUN → 1, 2, 3, 4, 5, 6

delete element at 6th position

after while loop temp will point to 5

then we wrote

a = 5 -> Next ie a = 6

b = a -> next ie b = NULL

temp -> next = NULL

This will delete last element.

DATE

Another way → Two pointers

```
Node * delek(Node * head , int position)
{
  if (position == 0) {
    Node * a = head;
    head = head -> next;
    delek (a);
    return head
  }
else
    Node * prev = NULL;
    Node * curr = Head;
    int count = 0;
    while (curr != NULL && position < count-1
{
        prev = curr;
        curr = curr -> next
          count++
    }
if ( curr != NULL)
{
  prev -> next = curr -> next
}
return Head
}
```

classmate

PAGE