



Software Engineering Dept.- Second year

Linked List

by

Dr.Iaheeb M. Ibrahim



Type of data structure

There are two type of data structure

1- Static data structure

2- Dynamic data structure



- There are four differences between Static data structure and Dynamic data structure

1-The storage space

2-Insert and delete operation

3-Random access to the element

4- Merge and split operation



What is Linked List?

Linked list: is a linear data structure that contains sequence of elements such that each element links to its next element in the sequence.

Linked list has three types :

- 1- Single Link List (SLL)
- 2- Double Link List (DLL)
- 3- Circular Link List (CLL)



What is Single Link List?

Single link list: is a sequence of elements in which every element has link to its next element in the sequence.

The individual element is called as "**Node**".

Every "**Node**" contains two fields, **data** and **next**.

Data field: is used to store actual value of that node

Link Field: and next field is used to store the address of the next node in the sequence.



T

Stores Address of next node

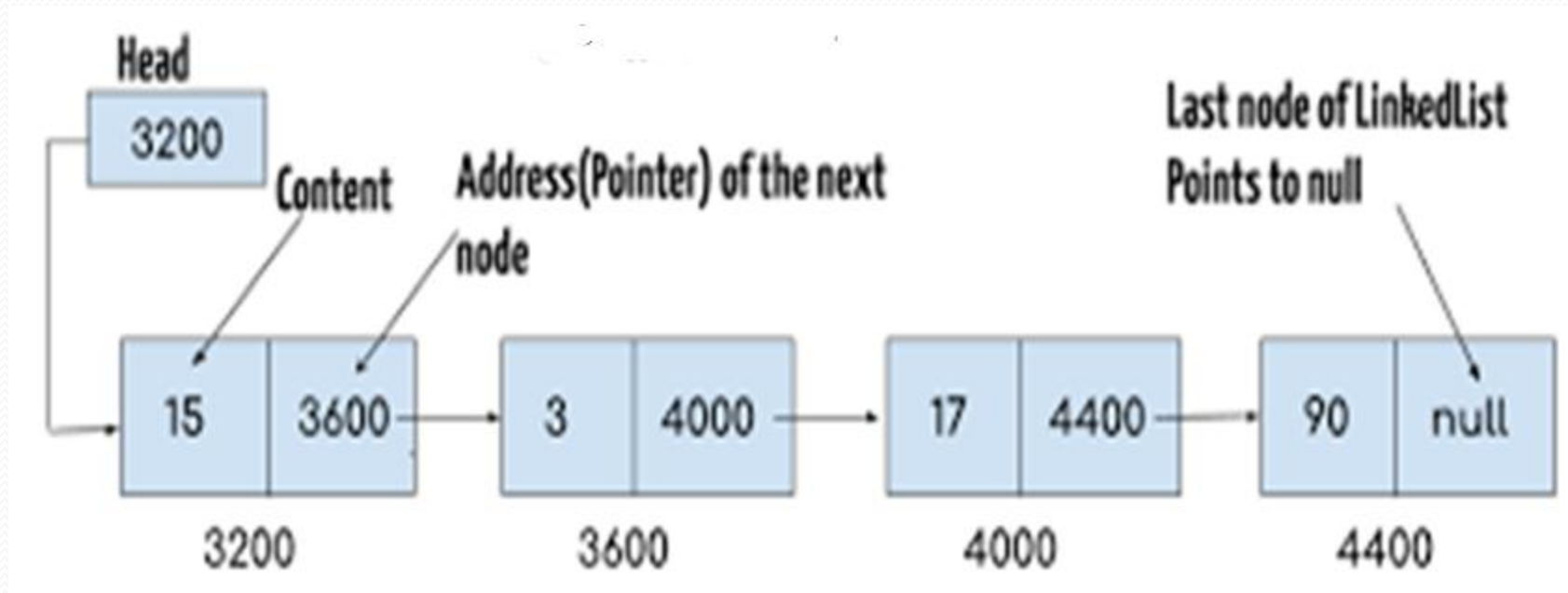


Stores Actual value

NOTE

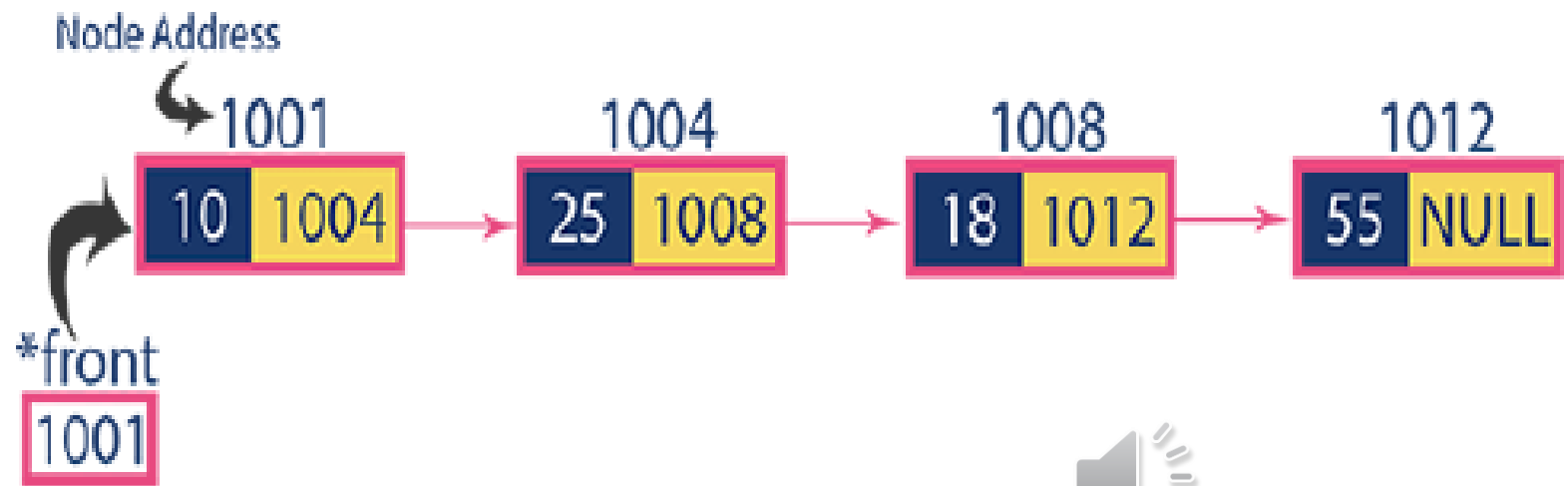
- ✧ In a single linked list, the address of the first node is always stored in a reference node known as "front" (Some times it is also known as "head").
- ✧ Always next part (reference part) of the last node must be NULL.



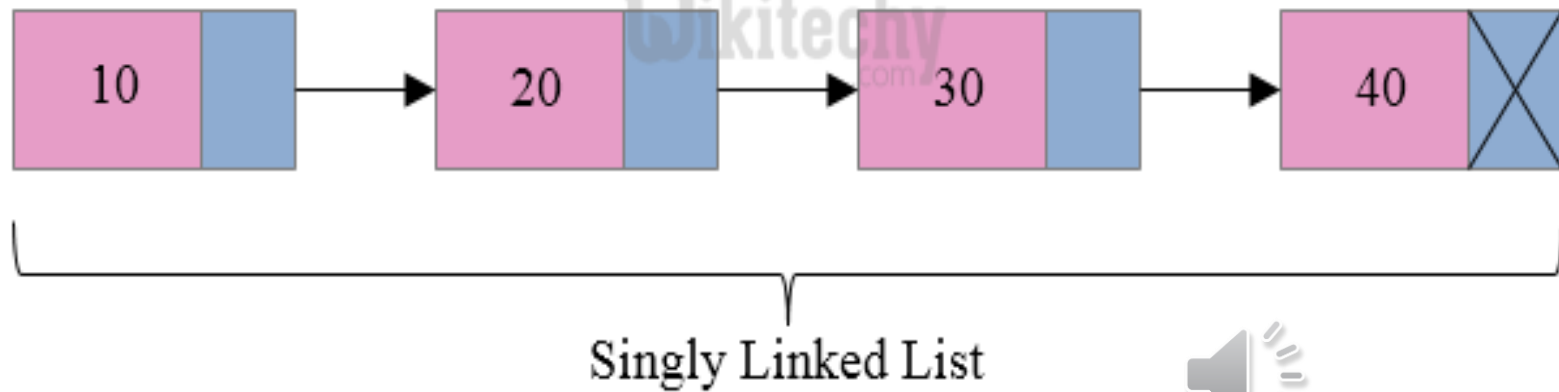
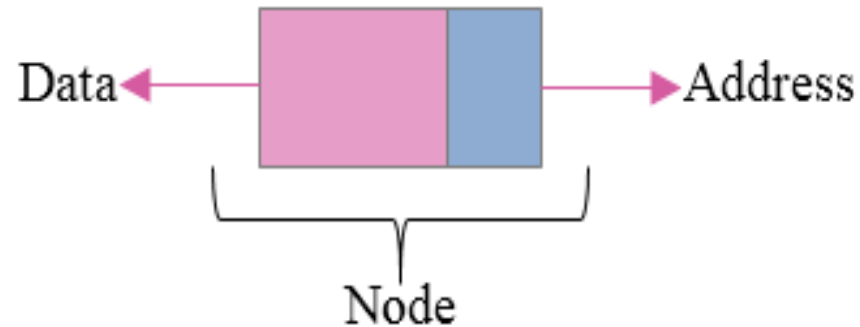


Example for SLL

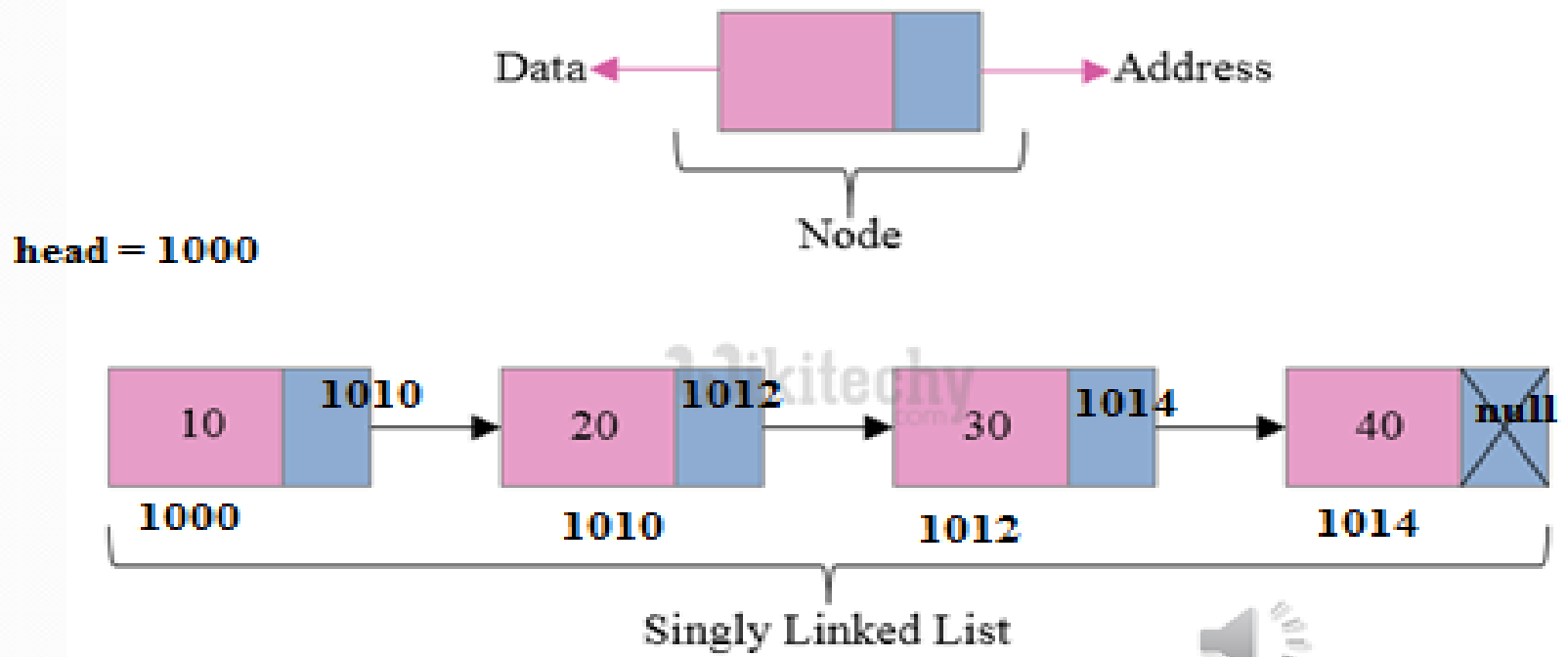
Example



Example for SLL



solution



Operations in single linked list

In a single linked list we perform the following operations.

1. Insertion
2. Deletion
3. Display



Before we implement actual operations, first we need to setup empty list. First perform the following steps before implementing actual operations.

Step 1: Include all the **header files** which are used in the program.

Step 2: Declare all the **user defined** functions.

Step 3: Define a **Node** structure with two members **data** and **next**

Step 4: Define a Node pointer '**head**' and set it to **NULL**.

Step 4: Implement the **main** method by displaying operations menu and make suitable function calls in the main method to perform user selected operation.



Insertion in SLL

In a single linked list, the insertion operation can be performed in three ways. They are as follows...

1. Inserting At Beginning of the list
2. Inserting At End of the list
3. Inserting At Specific location in the list



Inserting At Beginning of the SLL

- We can use the following steps to insert a new node at beginning of the single link list

Step 1: Create a **newNode** with given value.

Step 2: Check whether list is **Empty** (**head == NULL**)

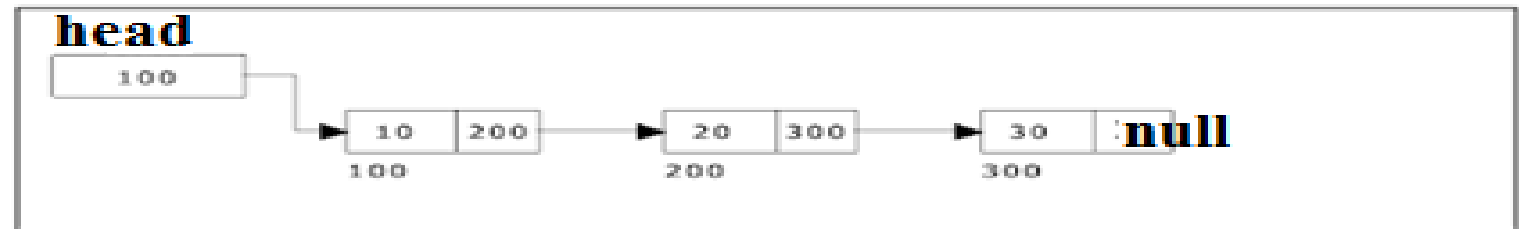
Step 3: If it is **Empty** then, set **newNode**→**next** = **NULL** and **head** = **newNode**.

Step 4: If it is **Not Empty** then, set **newNode**→**next** = **head** and **head** = **newNode**.

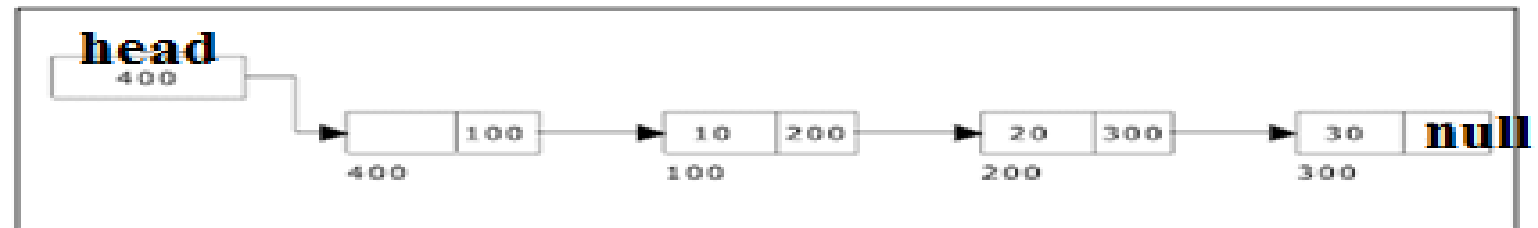


Inserting At Beginning of the SLL

Inserting At Beginning of the SLL



Single Linked List without a header node



Single Linked List with header node



Inserting At End of the SLL

We can use the following steps to insert a new node at end of the single linked list...

Step 1: Create a **newNode** with given value and **newNode** → **next** as **NULL**.

Step 2: Check whether list is **Empty** (**head == NULL**).

Step 3: If it is **Empty** then, set **head = newNode**.

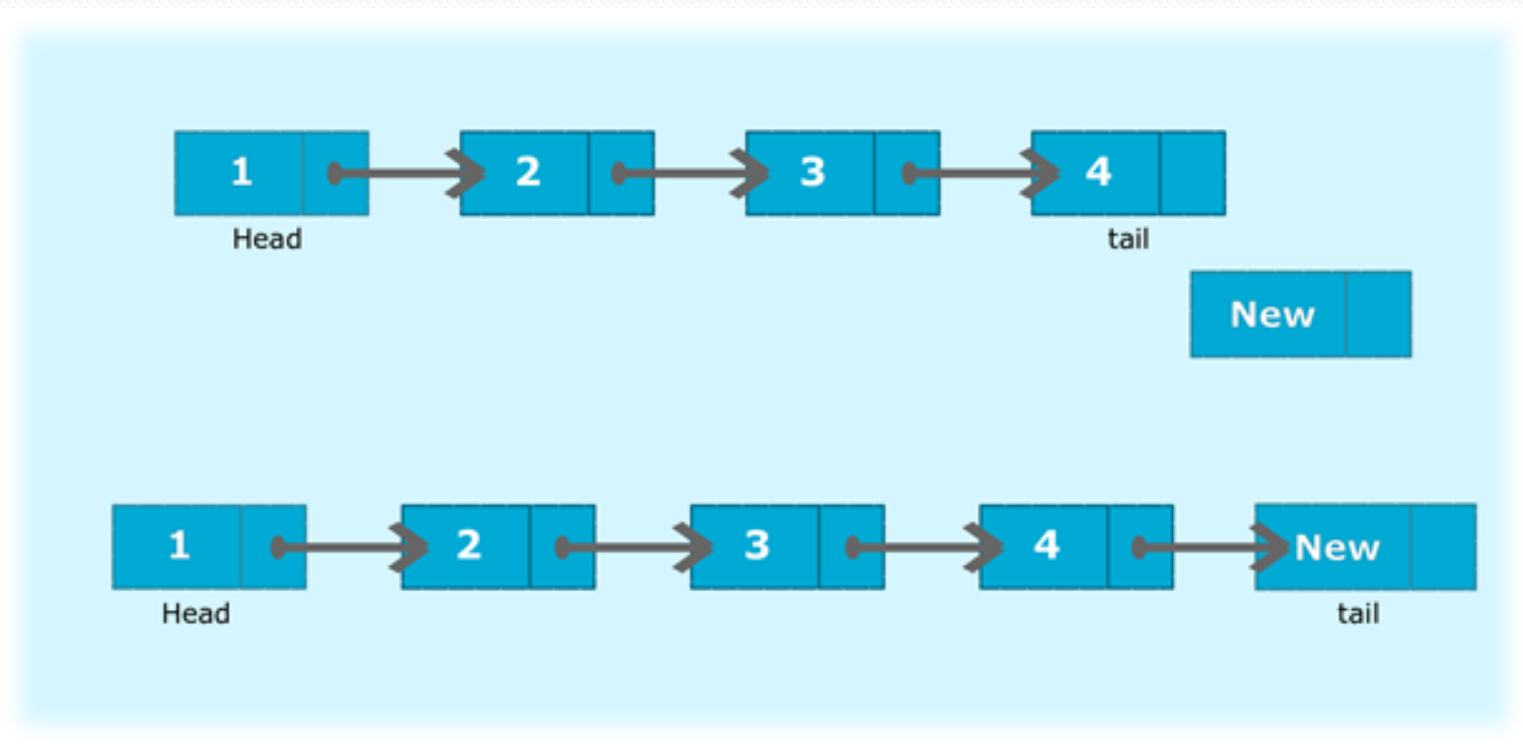
Step 4: If it is **Not Empty** then, define a node pointer **temp** and initialize with **head**.

Step 5: Keep moving the **temp** to its next node until it reaches to the last node in the list (until **temp** → **next** is equal to **NULL**).

Step 6: Set **temp** → **next = newNode**.



Inserting At End of the SLL



Displaying a Single Linked List

We can use the following steps to display the elements of a single linked list...

Step 1: Check whether list is **Empty** (**head == NULL**)

Step 2: If it is **Empty** then, display '**List is Empty!!!**' and terminate the function.

Step 3: If it is **Not Empty** then, define a Node pointer '**temp**' and initialize with **head**.

Step 4: Keep displaying **temp** → **data** with an arrow (--->) until **temp** reaches to the last node

Step 5: Finally display **temp** → **data** with arrow pointing to **NULL** (**temp** → **data** ---> **NULL**).

