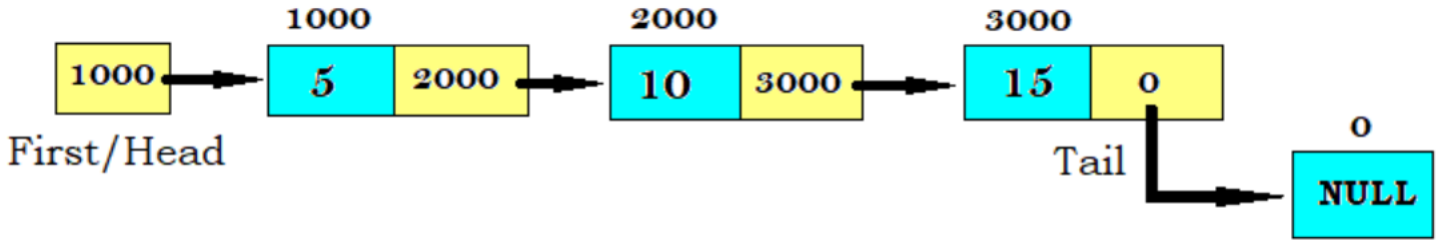
Data Structures and Algorithms

Lecture 8: Linked List – Traversal

Linked List - Traversal

Traversing means visiting each node of the list once in order to perform some operation on that.



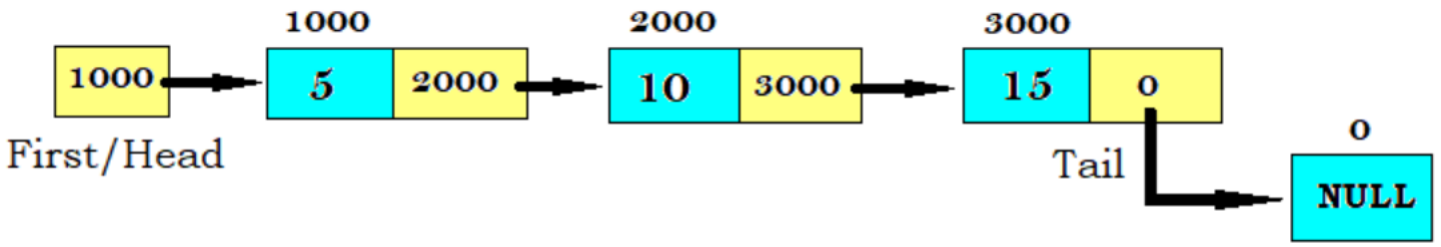
- Traversing - Accessing or printing the values of a singly linked list exactly once until the end node is reached.
- And, This accessing and printing is sometimes called “visiting” the linked list.

Traversing a linked list

```
struct node * temp = start;
printf("\n list empty are-");
while (temp!= NULL)
{
        printf('%d ", temp -> data)
        temp=temp -> next;
}
```

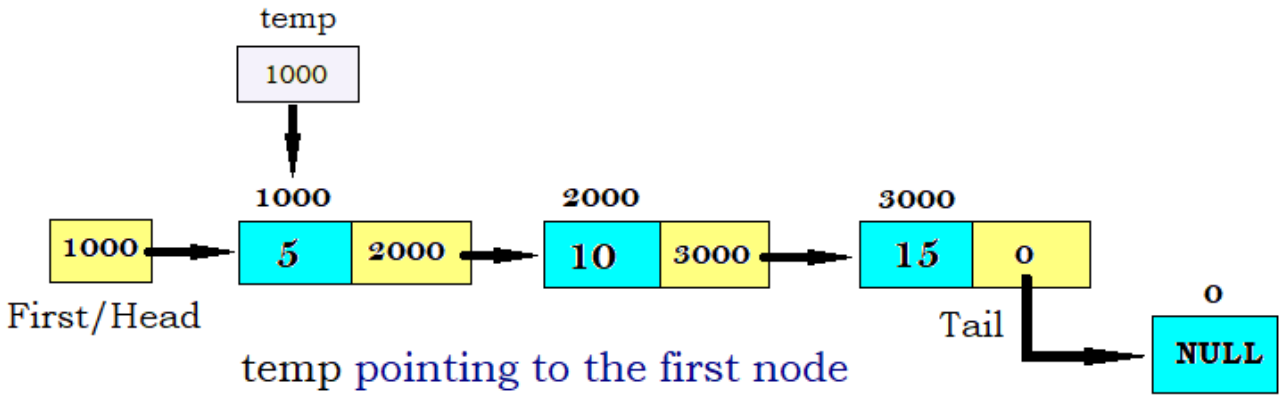
When the temp is null, it means you traversed all the nodes, and you reach the end of the linked list and get out from the while loop.

Linked List - Traversal



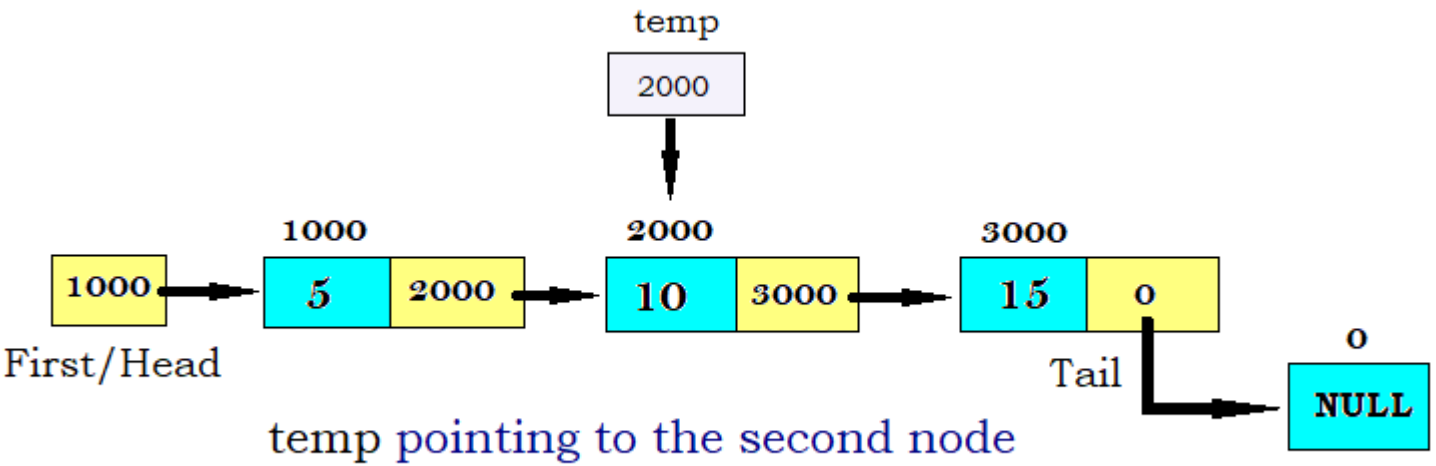
- Head pointer contains the address of the first node, so if you want to access every value from the first node to the last node, you have to do it sequentially.
- First of all you have to create a head pointer of node type which will contain the address of the first node of the list.
- We are creating temp because we don’t want to make any changes inside head.
- Because, we have to do traversing, visit each node, extract its value.

Linked List - Traversal

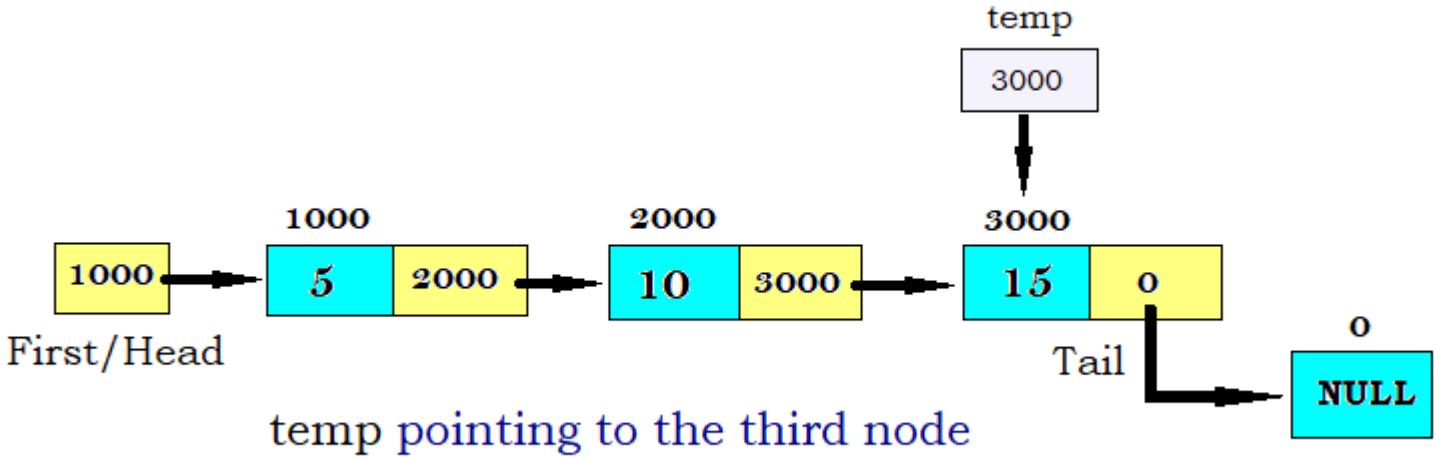


- We do not want the head value to be disturbed, because if head does not have the address of the first node then we will not be able to access this linked list.

Linked List - Traversal



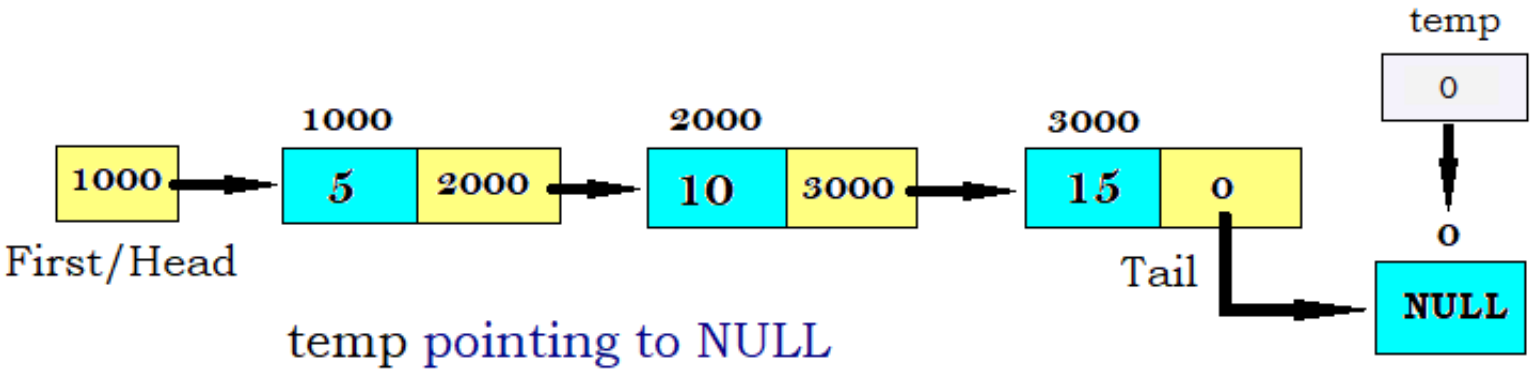
temp pointing to the second node



temp pointing to the third node

Linked List - Traversal

If the temp pointer points to NULL, program come outside the body of while. Hence all the nodes are visited i.e., traversed.



Recall:

- Linked list – Traversal