# Practical # 05

## Objective:

***Discuss** and Analyze implementation of Singly Linked List Data Structure. Write a C++ program to create a singly linked list of integers.*

## Theory:

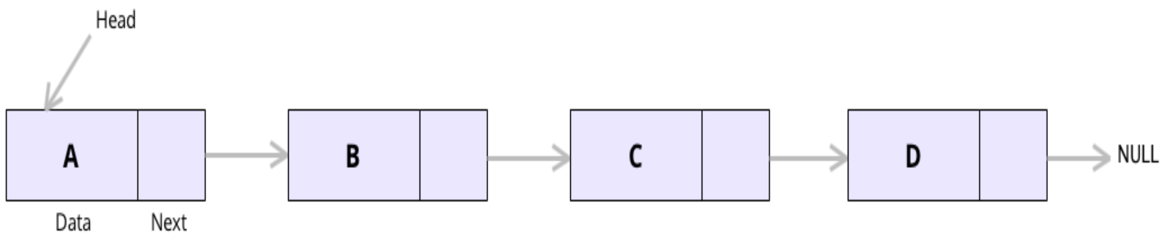
In this Lab, we will introduce the singly Linked list and possible operations performed on singly linked list.

## Lab Objectives:

- To be able to declare a singly linked list.
- To be able to perform fundamental operations on singly linked list.

## Introduction:

The **singly-linked** list contains nodes that only point to the next node. A node has two parts: the data part and the next part. The **data** part contains the stored data, and the **next** part provides the address of the next node. The first node of a linked list is called the **head**, and the last node is called the **tail**. The list starts traversing from the head, while the tail ends the list by pointing at NULL.



## Declaring a Node type

To set up a linked list, the first thing we'll need is a structue that represents a single node in the list. For simplicity, let's assume that a node contains nothing but an ineger(the node's data) plus a pointer to the next node in the list. Here's what our node structue look like:

```
struct node{
        int value;                          /* data stored in the node */
        struct node * pointer;                      /* pointer to the next node */
};
```

Notice that the ***next member*** has type ***struct node *,*** which means that it can store a pointer to a node structure. There's nothing about the name node, by the way; it's just an ordinary structure tag.
Now that we have the node structure declared, we'll need a way to keep track of where the list begins. In other words, we'll need a variable that always points to the first node in the list. Let's name the variable start:

```
struct node  *start=NULL;
```

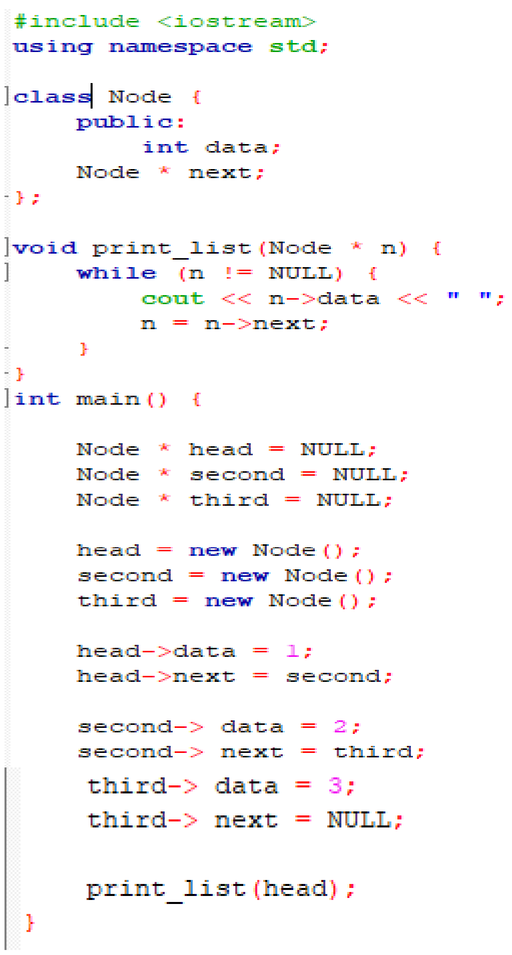
Setting start to null indicates that the list is initially empty.

## Creating a Node

As we construct a liked list, we want to create node one by one, adding each node to the list. Creating a node requires three steps.

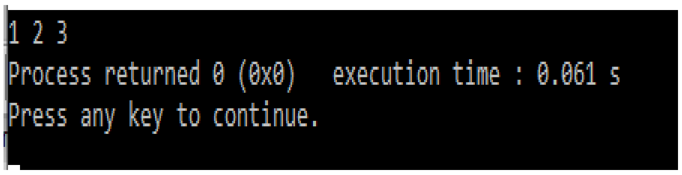
1. Allocate memory for the node.
2. Store data in the node.
3. Insert node in to the list.

***Example program:*** *Declare and print elements of singly linked list.*



```
#include <iostream>
using namespace std;

class Node {
    public:
        int data;
    Node * next;
};

void print_list(Node * n) {
    while (n != NULL) {
        cout << n->data << " ";
        n = n->next;
    }
}
int main() {

    Node * head = NULL;
    Node * second = NULL;
    Node * third = NULL;

    head = new Node();
    second = new Node();
    third = new Node();

    head->data = 1;
    head->next = second;

    second-> data = 2;
    second-> next = third;
     third-> data = 3;
     third-> next = NULL;

     print_list(head);
 }
```

**OUTPUT**



```
1 2 3
Process returned 0 (0x0)   execution time : 0.061 s
Press any key to continue.
```

## Review Questions/ Exercise:

1. Write a program to find the number NUM of elements in a linked list.

2. Write a program to insert a new node at the beginning of liked list data structure.

3. Write a program to insert new node at given position of the linked list data structure.

4. Write a program to find the location LOC of the last node in a sorted list

5. Write a program to delete node from linked list.

**Name:** ____________________

**Roll #:** ____________________

**Date:** ____________________

**Subject Teacher**

**Remarks:**