Lab 2: Singly Linked List

CLO:

02

Objectives:

To write a program that implements the basic operations of the linked list in C++.

Linked List Overview:

A linked list is a way to store a collection of elements. Like an array these can be character or integers. Each element in a linked list is stored in the form of a node. A linked list is formed when many such nodes are linked together to form a chain.

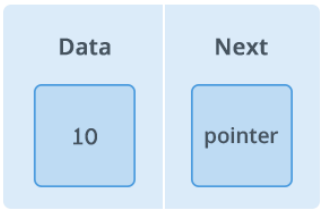


Figure 1. Node

It is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:

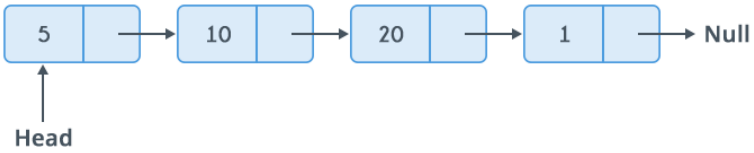


Figure 2. Linked list

- ✓ List is a collection of components, called nodes. Every node (except the last node) contains the address of the next node. Every node in a linked list has two components:
  - ➢ One to store the relevant information (that is, data)
  - ➢ One to store the address, called the link or next, of the next node in the list.
- ✓ The address of the first node in the list is stored in a separate location, called the head or first.
- ✓ The address of the last node in the list is stored in a separate location, called the tail or last.

There are 3 types of linked list:

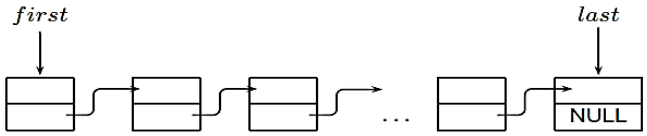
- Singly Linked List
- Circular Linked List
- Doubly Linked List

In this lab we will discuss singly linked list.

Singly linked list:

A Singly-linked List Model

A singly-linked list may be depicted as in Figure 1.1.



Sample Code:

Inserting a Node in a Singly-linked List

The node to be inserted may be created. We will assume here that the node to insert is pointed to by p.

```
// Example program
#include <iostream>
using namespace std;
class Node {
public:
  int data;
  Node *next;

  Node (int val) {
            data = val;
            next = NULL;
  }
};
class LinkList {
public:
  Node* head;

  LinkList () {
            head = NULL;
  }

  void insertHead (int val) {
            Node *mynode= new Node(val);
            if(head == NULL){
            head=mynode;
            }
            else{
            mynode->next=head;
            head=mynode;
            }
  }

  void display(){
            if(head == NULL){
            cout<<"List is empty"<<endl;
            }
            else{
```

```
            Node *temp = head;
            while(temp != NULL){
            cout<<temp->data<<"\t";
            temp=temp->next;
            }
            }
  }
};

int main(){
  LinkList L;
  L.insertHead(20);
  L.insertHead(30);
  L.insertHead(40);
  L.insertHead(50);
  L.display();
}
```

LAB TASKS

As you have learned how to insert and delete a node at different positions in the link list now you are required to perform the following tasks.
Suppose we have the following singly linked list

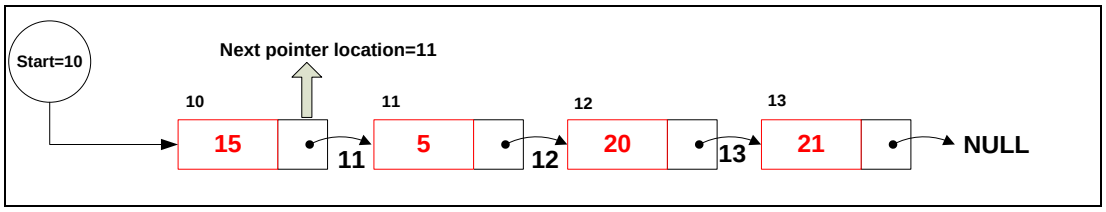


Figure 3. Singly Linked list

1. Create your own class of link list which will have the following functions.
   a. Function called **InsertAtBegin** to add node at the begging of list.
   After insertion of value 12 at the beginning of the linked list, it will become as following:

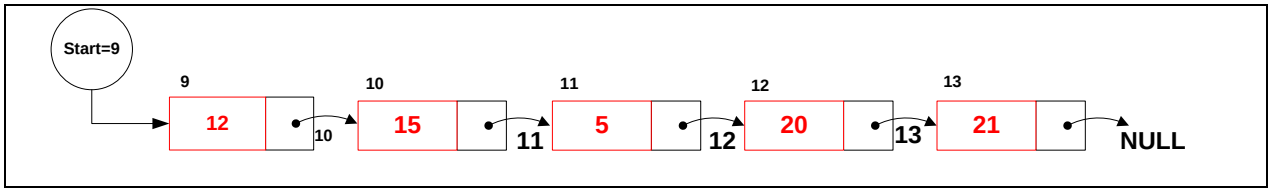


Figure 4. Insertion at beginning of singly linked list

   b. Function called **InsertAtEnd**to add node at the last of list.
   After insertion of value 25 at the end of the linked list, it will become as following:

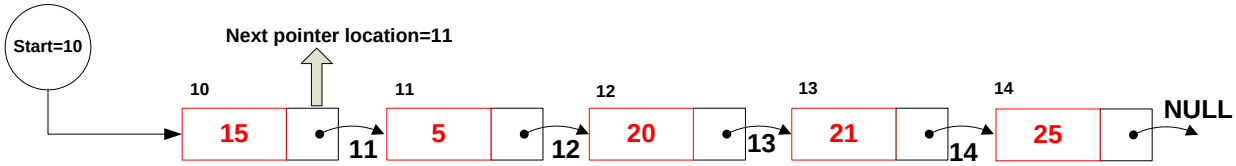
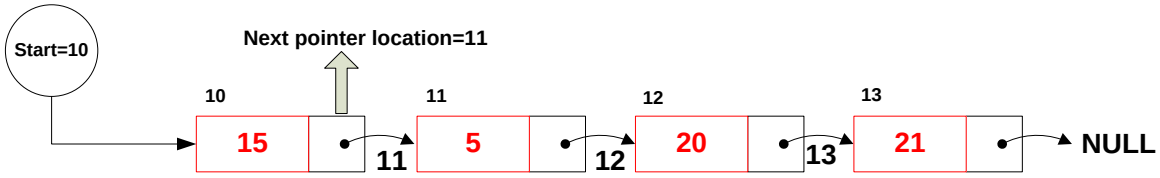


Figure 5. Insertion at the end of the singly linked list

c. Function called update to update the value.



If we want to update the value 20 residing at location 12 to 25, our linked list will display as figure 6.

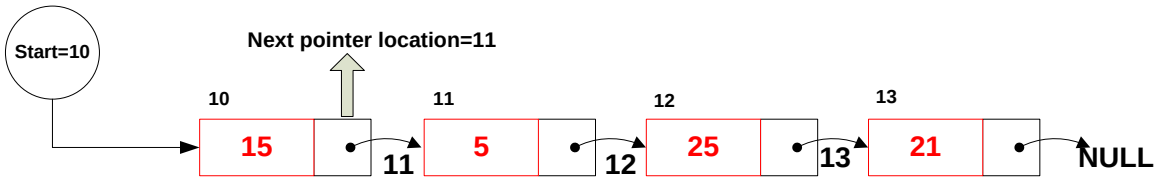
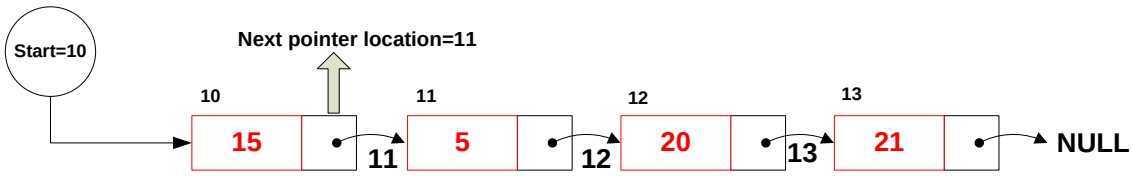


Figure 6. Updated linked list

d. Function called display to display the list.
In this part display the content of the following linked list



e. Function called search to search the value.

f. Function called insertAfter to insert any value after certain value or key.

g. Sort the created linked list (ascending or descending)