LINKED LIST IMPLEMENTATION OF STACK IN C++

Write a class which will implement the Stack data structure in C++. Give the declaration & definition of the class. Also define the Node. For simplicity, you may assume stack to hold integer data type.

Solution:

There can be multiple type of implementations of Stack. Two most popular ones are

1. Array implementation:

- Stack is stored in an Array.
- Easy to implement.
- Is of fixed size (i.e if the array is of size 100, then it will occupy 100 memory locations even if there are, say 2 elements in the stack, also we cannot store more that 100 elements in the Stack).

2. Linked List Implementation:

- Stack is stored in a Linked List.
- Have to manipulate pointers.
- The size of this implementation is dynamic and it takes memory proportional to the actual number of elements in Stack.

In both the implementations, the push and pop operations are constant-time operations, i.e O(1)

In this post we will only consider the Linked List implementation of Stack (you may try Array implementation on your own). For Linked list implementation, the stack is

A linked list where Insertion (Push) and Deletion (Pop) are performed from the same end (lets say at head of the list).

Lets first define the Node structure :

```
1   struct Node{
2       int data;
3       Node* link;
4
5       /* Constructors. Both the properties (data & link) are initialized
6        * in the Member Initialization List itself. So the body of constructors is empty
7       Node():data(0), link(NULL) {}
8       Node(int n):data(n), link(NULL) {}
9   };
```

Given this definition of Node of Linked List, The declaration of Stack class (may go in mystack.h , file) will be as below:

```
1   class MyStack
2   {
3     private:
4       // Will always point to the head of the list (First element)
5       Node * top;
6     public:
7       // Default Constructor
8       MyStack():top(NULL){}
9
10      void push(int);
11      int pop();
12
13      // returns the top-most element.
14      int getTop();
15
16      // returns true if the Stack is Empty, else return false.
17      int isEmpty();
18  };
```

Definition of the functions of Stack class will be in the .cpp file (mystack.cpp)

```
1   /** Function to push element in the Stack.
2    *  Will Create a new Node and put it at the head of the linked list.
3    */
```

```
4   void MyStack::push(int d)
5   {
6       Node* temp = new Node(d);
7       temp->link = top;
8       top = temp;
9   }
10
11  /** Function to Pop element in the Stack.
12   *  Will delete the node pointed to by top of the list and return its value.
13   *  If Stack is NULL then returns -1
14   */
15  int MyStack::pop()
16  {
17      int retValue = -1;
18
19      if(top != NULL){
20          retValue = top->data;
21
22          Node * temp = top;
23          top = top->link;
24          delete temp;
25      }
26      return retValue;
```

```
27  }
28
29  /** Returns the element at the Top of the Stack
30   *  Top is the first element of the Stack.
31   *  If Stack is NULL then returns -1
32   */
33  int MyStack::getTop()
34  {
35      if(top)
36          return top->data;
37      else
38          return -1;
39  }
40
41  /** Check if the Stack is Empty or Not
42   *  If top is NULL then the Stack is Empty.
43   */
44  int MyStack::isEmpty()
45  {
46      return ( (top == NULL)? true : false);
47  }
```

Source: http://www.ritambhara.in/linked-list-implementation-of-stack-in-c/