

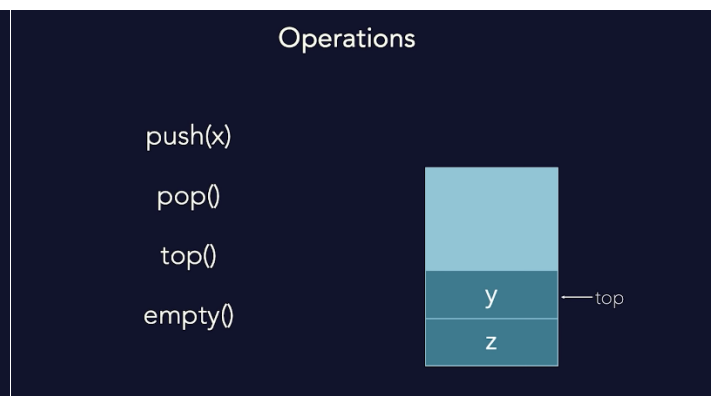
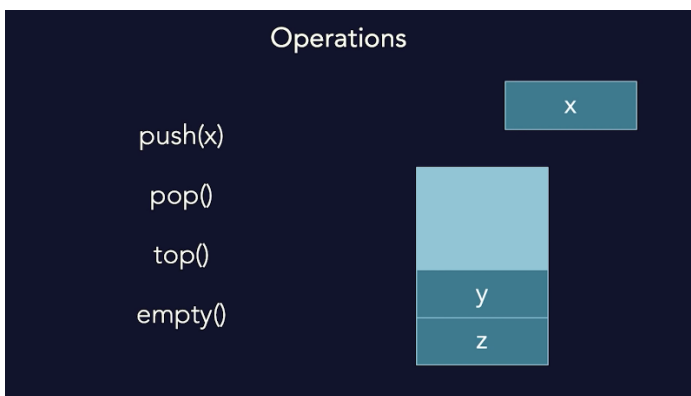
STACKS

Stack Data Structure - Introduction | C++ Placement Course | Lecture 23.1

STACK

Stores a list of items in which an item can be added to or removed from the list only at one **end**

0:15 / 10:13



```
/*STACKS
A stack is a list with the restriction thta insertion and deletion can be done only from one
end called the top.
It works on LIFO
Operations:
Push
Pop
Top
isEmpty
All operations take constant time i.e. O(1)
*/

// To write a program ..for showing implementation of a stack using array
```

ARRAY IMPLEMENTATION OF STACK

```
64 int main(){
65
66     Stack st;
67     st.push(1);
68     st.push(2);
69     st.push(3);
70     cout<<st.Top()<<endl;
71     st.pop();
72     cout<<st.Top()<<endl;
73     st.pop();
74     cout<<st.Top()<<endl;
75     st.pop();
76     st.pop();
77     cout<<st.empty()<<endl;
78     return 0;
79 }

80
81 // 3
82 // Done popping an element
83 // 2
84 // Done popping an element
85 // 1
86 // Done popping an element
87 // Stack is already empty!
88 // Stack is already empty!
89 // 1
```

new > C++ 222_stack-usingArrays.cpp > ...

```
14  // #include <bits/stdc++.h>
15  #include <iostream>
16  using namespace std;
17
18  #define n 100
19
20  class Stack{
21  private:
22      int *arr;
23      int top;
24
25  public:
26      Stack(){
27          arr = new int[n];
28          top = -1;
29      }
30
31      void push(int val){
32          if (top==n-1){
33              cout<<"Stack overflow...\n";
34              return;
35          }
36          top++;
37          arr[top] = val;
38      }
39
40      bool empty(){
41          if(top==-1){
42              cout<<"Stack is already empty!\n";
43              return 1;
44          }
45          return 0;
46      }
47
48      void pop(){
49          if(empty()){
50              return;
51          }
52          cout<<"Done popping an element\n";
53          top--;
54      }
55
56      int Top(){
57          if(empty())
58              return -1;
59
60          return arr[top];
61      }
62  };
--
```

STACK USING LINKED LIST

```

12 // To write a program ..for showing implementation of a stack using linked list
13 // For stack using linked list, we always add or delete at the beginning to get order
14 // of O(1) because for adding or deleting at end we get O(n)
15

```

new > C++ 223_stack-usingLL.cpp > ...

```

17 #include <iostream>
18 using namespace std;
19
20 class Node{
21 public:
22     int data;
23     Node* next;
24
25     Node(int val){
26         data = val;
27         next = NULL;
28     }
29 };
30
31 Node* top= NULL;
32
33 void Push(int v){
34     Node* n = new Node(v);
35     n->next = top;
36     top = n;
37 }
38
39 int isEmpty(){
40     if(top==NULL){
41         cout<<"Stack is Empty!\n";
42         return -1;
43     }
44     return 0;
45 }
46
47 void Pop(){
48     if(isEmpty()==-1)
49         return;
50
51     Node* todel = top;
52     top = top->next;
53     delete todel;
54 }
55
56 int Top(){
57     if(isEmpty()==-1)
58         return -1;
59
60     return (top->data);
61 }
62

```

```

int main(){

    Push(1);
    cout<<Top()<<endl;
    Push(2);
    cout<<Top()<<endl;
    Push(3);
    cout<<Top()<<endl;
    Pop();
    cout<<Top()<<endl;
    Pop();
    cout<<Top()<<endl;
    Pop();
    cout<<isEmpty()<<endl;
    return 0;
}

// 1
// 2
// 3
// 2
// 1
// Stack is Empty!
// Stack is Empty!
// -1

```

```

17 #include <iostream>
18 using namespace std;
19
20 class Node{
21     public:
22         int data;
23         Node* next;
24
25         Node(int val){
26             data = val;
27             next = NULL;
28         }
29 };
30
31 class Stack{
32
33     private:
34         Node* top;
35
36     public:
37         Stack(){
38             top= NULL;
39         }
40
41         void Push(int v){
42             Node* n = new Node(v);
43             n->next = top;
44             top = n;
45         }
46
47         int isEmpty(){
48             if(top==NULL){
49                 cout<<"Stack is Empty!\n";
50                 return -1;
51             }
52             return 0;
53         }
54
55         void Pop(){
56             if(isEmpty()==-1)
57                 return;
58
59             Node* todel = top;
60             top = top->next;
61             delete todel;
62         }
63
64         int Top(){
65             if(isEmpty()==-1)
66                 return -1;
67
68             return (top->data);
69         }
70 };
71

```

```

71
72 int main(){
73
74     Stack st;
75     st.Push(1);
76     cout<<st.Top()<<endl; //1
77     st.Push(2);
78     cout<<st.Top()<<endl; //2
79     st.Push(3);
80     cout<<st.Top()<<endl; //3
81     st.Pop();
82     cout<<st.Top()<<endl; //2
83     st.Pop();
84     cout<<st.Top()<<endl; //1
85     st.Pop(); //Stack is Empty!
86     cout<<st.Top()<<endl; //-1
87     cout<<st.isEmpty()<<endl; //Stack is Empty!
88     //Stack is Empty!
89     return 0;
90 }

```