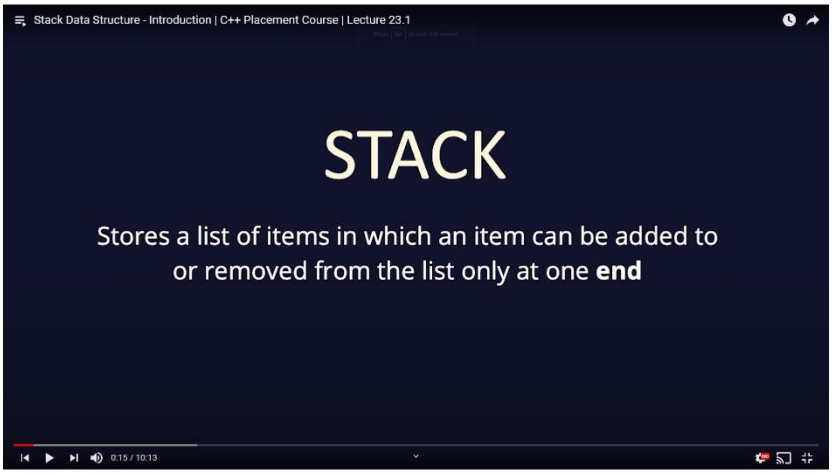
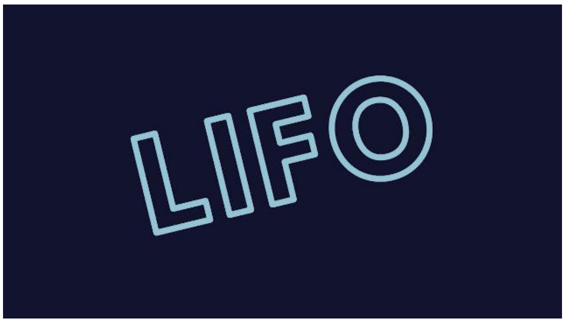
STACKS

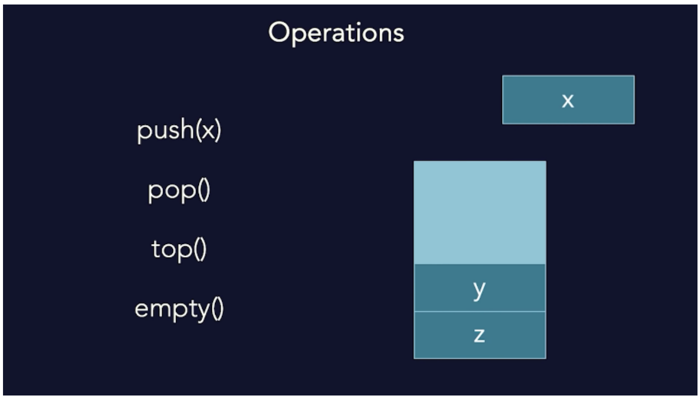


Stack Data Structure - Introduction | C++ Placement Course | Lecture 23.1
0:15 / 10:13

STACK

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





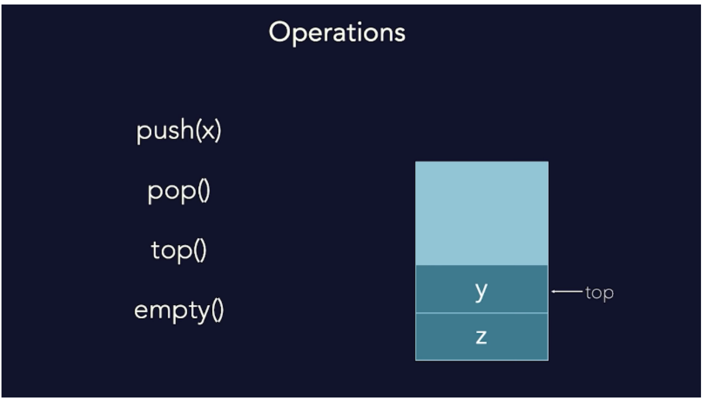
Operations

push(x)

pop()

top()

empty()



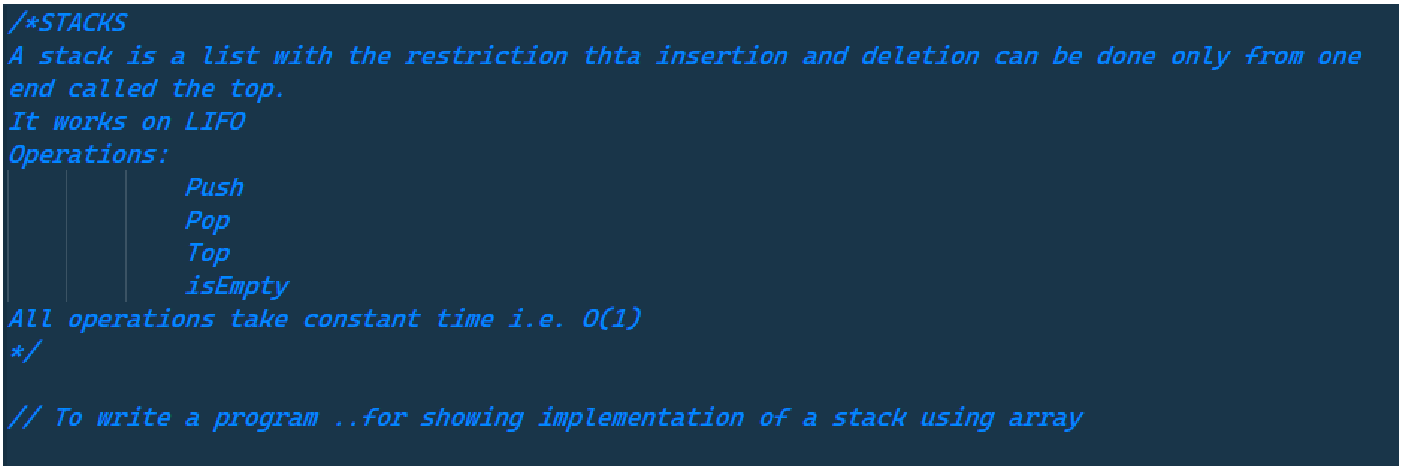
Operations

push(x)

pop()

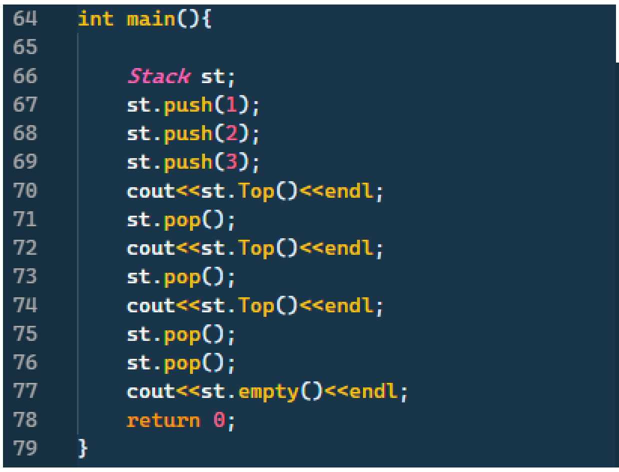
top()

empty()

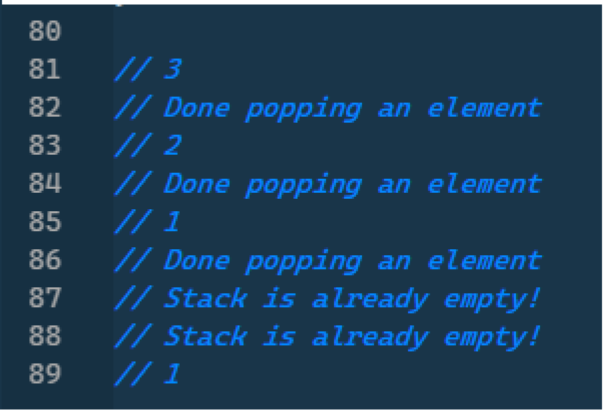


```
/*STACKS
A stack is a list with the restriction thta insertion and deletion can be done only from one
end called the top.
It works on LIFO
Operations:
			Push
			Pop
			Top
			isEmpty
All operations take constant time i.e. O(1)
*/

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

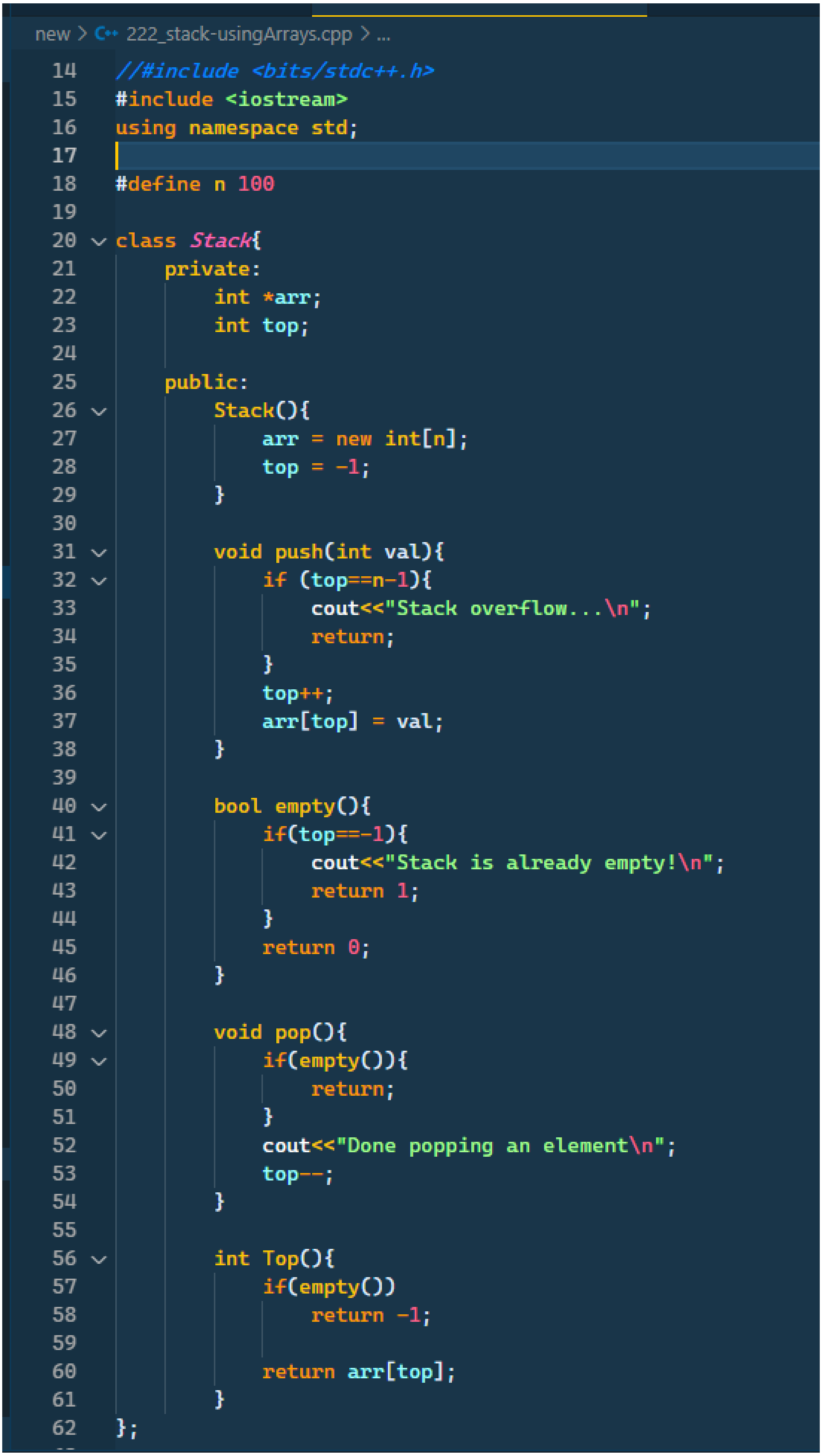
ARRAY IMPLEMENTATION OF STACK



```
int main(){

    Stack st;
    st.push(1);
    st.push(2);
    st.push(3);
    cout<<st.Top()<<endl;
    st.pop();
    cout<<st.Top()<<endl;
    st.pop();
    cout<<st.Top()<<endl;
    st.pop();
    st.pop();
    cout<<st.empty()<<endl;
    return 0;
}
```

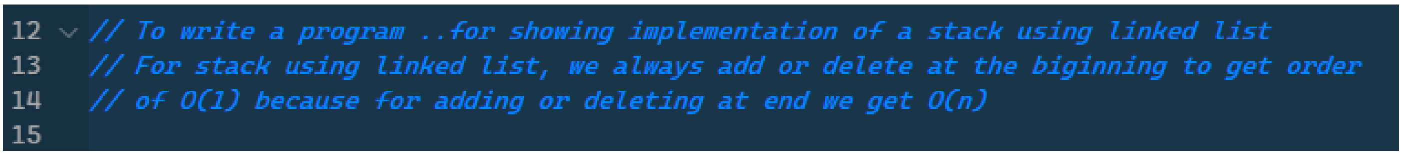


```

// 3
// Done popping an element
// 2
// Done popping an element
// 1
// Done popping an element
// Stack is already empty!
// Stack is already empty!
// 1
```

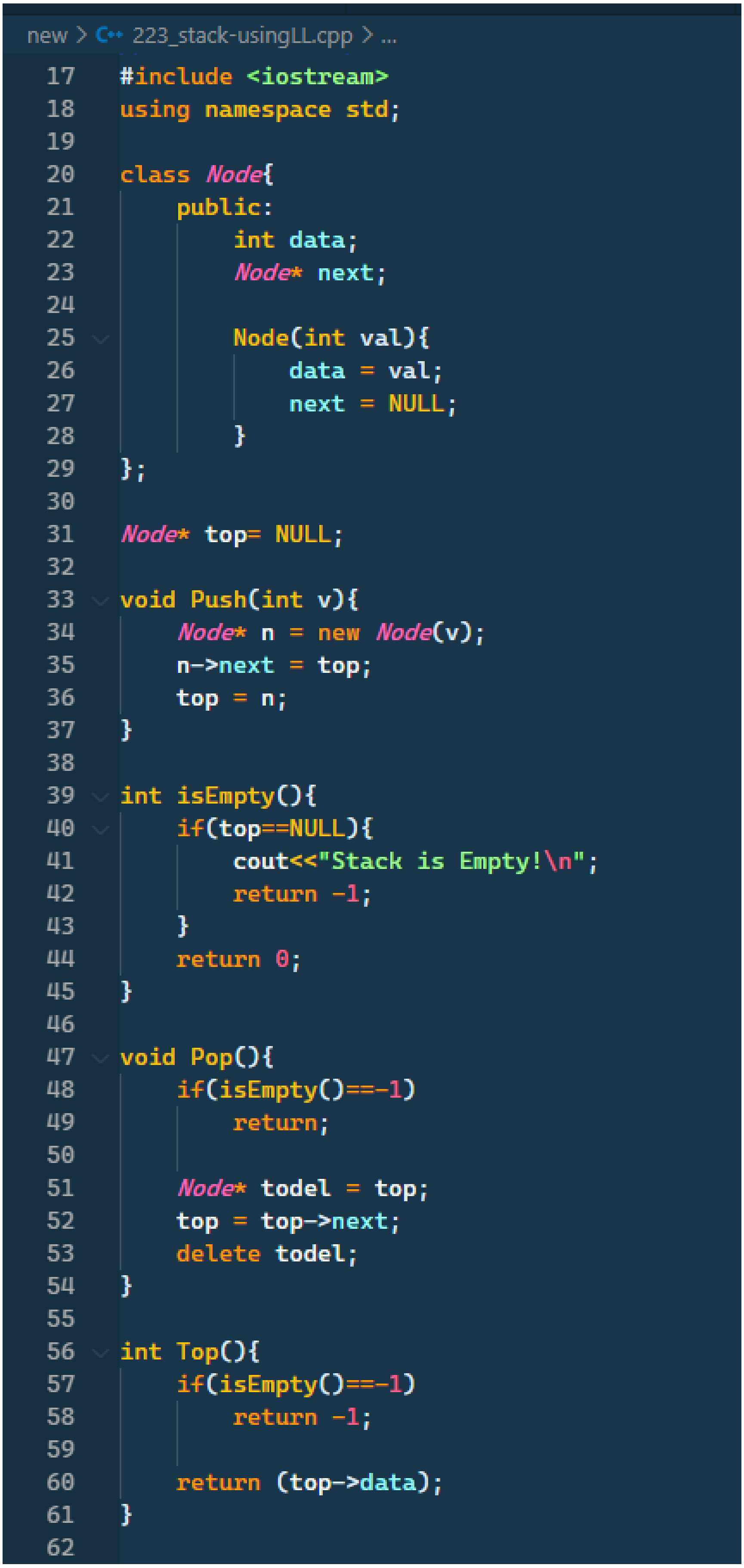


```cpp
//#include <bits/stdc++.h>
#include <iostream>
using namespace std;

#define n 100

class Stack{
    private:
        int *arr;
        int top;

    public:
        Stack(){
            arr = new int[n];
            top = -1;
        }

        void push(int val){
            if (top==n-1){
                cout<<"Stack overflow...\n";
                return;
            }
            top++;
            arr[top] = val;
        }

        bool empty(){
            if(top==-1){
                cout<<"Stack is already empty!\n";
                return 1;
            }
            return 0;
        }

        void pop(){
            if(empty()){
                return;
            }
            cout<<"Done popping an element\n";
            top--;
        }

        int Top(){
            if(empty())
                return -1;

            return arr[top];
        }
};
```

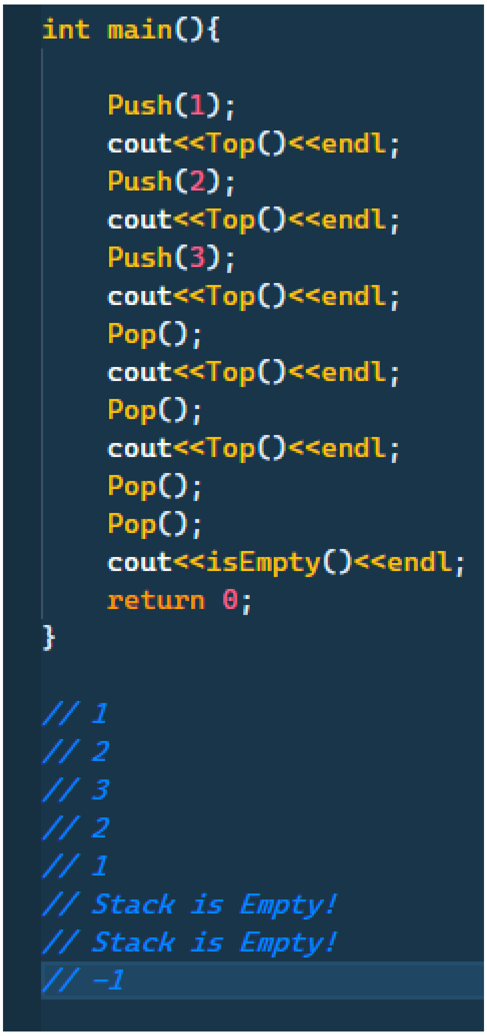
STACK USING LINKED LIST



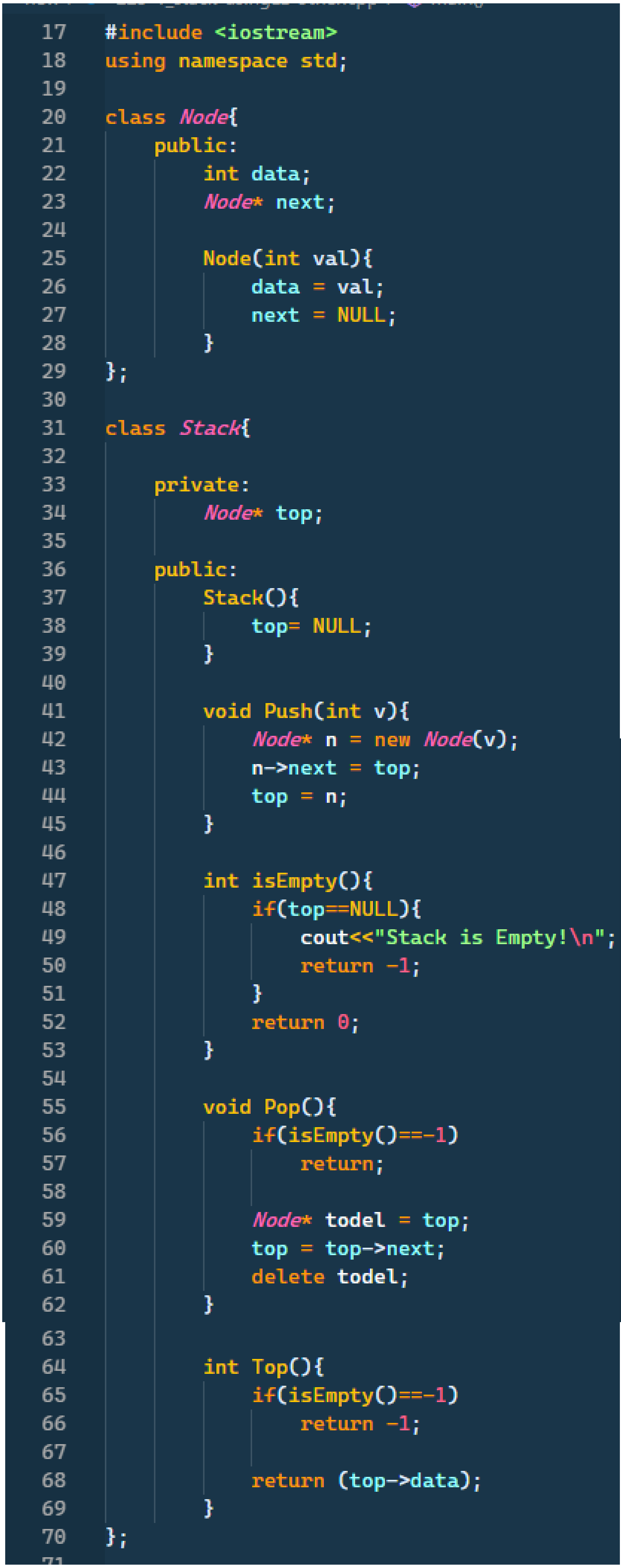
```cpp
// To write a program ..for showing implementation of a stack using linked list
// For stack using linked list, we always add or delete at the biginning to get order
// of O(1) because for adding or deleting at end we get O(n)

```



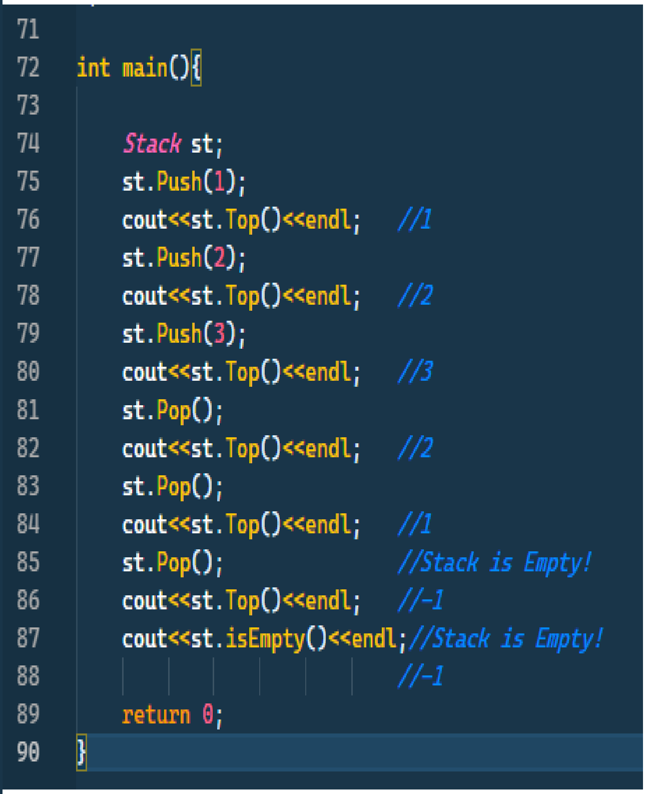
```cpp
#include <iostream>
using namespace std;

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

        Node(int val){
            data = val;
            next = NULL;
        }
};

Node* top= NULL;

void Push(int v){
    Node* n = new Node(v);
    n->next = top;
    top = n;
}

int isEmpty(){
    if(top==NULL){
        cout<<"Stack is Empty!\n";
        return -1;
    }
    return 0;
}

void Pop(){
    if(isEmpty()==-1)
        return;

    Node* todel = top;
    top = top->next;
    delete todel;
}

int Top(){
    if(isEmpty()==-1)
        return -1;

    return (top->data);
}

```



```cpp
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();
    Pop();
    cout<<isEmpty()<<endl;
    return 0;
}

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



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

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

        Node(int val){
            data = val;
            next = NULL;
        }
};

class Stack{

    private:
        Node* top;

    public:
        Stack(){
            top= NULL;
        }

        void Push(int v){
            Node* n = new Node(v);
            n->next = top;
            top = n;
        }

        int isEmpty(){
            if(top==NULL){
                cout<<"Stack is Empty!\n";
                return -1;
            }
            return 0;
        }

        void Pop(){
            if(isEmpty()==-1)
                return;

            Node* todel = top;
            top = top->next;
            delete todel;
        }

        int Top(){
            if(isEmpty()==-1)
                return -1;

            return (top->data);
        }
};
```



```cpp

int main(){

    Stack st;
    st.Push(1);
    cout<<st.Top()<<endl;   //1
    st.Push(2);
    cout<<st.Top()<<endl;   //2
    st.Push(3);
    cout<<st.Top()<<endl;   //3
    st.Pop();
    cout<<st.Top()<<endl;   //2
    st.Pop();
    cout<<st.Top()<<endl;   //1
    st.Pop();               //Stack is Empty!
    cout<<st.Top()<<endl;   //-1
    cout<<st.isEmpty()<<endl;//Stack is Empty!
                            //-1
    return 0;
}
```