



Fundamentals of
Data Structures using C

Array Implementation of Stack

B.Bhuvaneswaran, AP (SG) / CSE



9791519152



bhuvaneswaran@rajalakshmi.edu.in



RAJALAKSHMI
ENGINEERING COLLEGE

Introduction

- In this implementation each stack is associated with a top pointer, which is -1 for an empty stack.

Push

- To push an element X onto the stack, top pointer is incremented by 1 and then set:

$\text{Stack}[\text{top}] = X.$

Pop

- To pop an element from the stack, the Stack [top] value is returned and the top pointer is decremented by 1.

Check whether a Stack is Full

```
int IsFull()
{
    if(top == MAX - 1)
        return 1;
    else
        return 0;
}
```

Check whether a Stack is Empty

```
int IsEmpty()  
{  
    if(top == -1)  
        return 1;  
    else  
        return 0;  
}
```

Push an Element on to the Stack

```
void Push(int ele)  
{  
    if(IsFull())  
        printf("Stack Overflow...\n");  
    else  
    {  
        top = top + 1;  
        Stack[top] = ele;  
    }  
}
```

Pop an Element from the Stack

```
void Pop()  
{  
    if(IsEmpty())  
        printf("Stack Underflow...\n");  
    else  
    {  
        printf("%d\n", Stack[top]);  
        top = top - 1;  
    }  
}
```


Return Top of Stack

```
void Top()  
{  
    if(IsEmpty())  
        printf("Stack Underflow...\n");  
    else  
        printf("%d\n", Stack[top]);  
}
```

Display Stack Elements

```
void Display()
{
    int i;
    if(IsEmpty())
        printf("Stack Underflow...!\n");
    else
    {
        for(i = top; i >= 0; i--)
            printf("%d\t", Stack[i]);
        printf("\n");
    }
}
```

Queries?

Thank You!