**Title: Implementation of Stack (Push and Pop)**

**Abstract:** A stack is an Abstract Data Type (ADT), commonly used in most programming languages. A stack is an abstract data structure that contains a collection of elements. Stack implements the LIFO mechanism i.e. the element that is pushed at the end is popped out first. Some of the principle operations in the stack are −

* Push - This adds a data value to the top of the stack.
* Pop - This removes the data value on top of the stack
* Peek - This returns the top data value of the stack

## Methodology: Push Operation

The process of putting a new data element onto stack is known as a Push Operation. Push operation involves a series of steps −

* **Step 1** − Checks if the stack is full.
* **Step 2** − If the stack is full, produces an error and exit.
* **Step 3** − If the stack is not full, increments **top** to point next empty space.
* **Step 4** − Adds data element to the stack location, where top is pointing.
* **Step 5** − Returns success.



If the linked list is used to implement the stack, then in step 3, we need to allocate space dynamically.

### **Algorithm for PUSH Operation**

A simple algorithm for Push operation can be derived as follows −

begin procedure push: stack, data

if stack is full

return null

endif

top ← top + 1

stack[top] ← data

end procedure

## Pop Operation

Accessing the content while removing it from the stack, is known as a Pop Operation. In an array implementation of pop() operation, the data element is not actually removed, instead **top** is decremented to a lower position in the stack to point to the next value. But in linked-list implementation, pop() actually removes data element and deallocates memory space.

A Pop operation may involve the following steps −

* **Step 1** − Checks if the stack is empty.
* **Step 2** − If the stack is empty, produces an error and exit.
* **Step 3** − If the stack is not empty, accesses the data element at which **top** is pointing.
* **Step 4** − Decreases the value of top by 1.
* **Step 5** − Returns success.



### **Algorithm for Pop Operation**

A simple algorithm for Pop operation can be derived as follows −

begin procedure pop: stack

if stack is empty

return null

endif

data ← stack[top]

top ← top - 1

return data

end procedure

()) {

**Discussion:**

* A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle.
* In the pushdown stacks only two operations are allowed: **push** the item into the stack, and **pop** the item out of the stack.
* A stack is a limited access data structure - elements can be added and removed from the stack only at the top.
* **Push** adds an item to the top of the stack, **pop** removes the item from the top.
* A helpful analogy is to think of a stack of books; you can remove only the top book, also you can add a new book on the top.
* A stack is a **recursive** data structure.

**Source Code:**

|  |
| --- |
| #include<stdio.h>  #include<process.h>  #include<stdlib.h>    #define MAX 5 //Maximum number of elements that can be stored    int top=-1,stack[MAX];  void push();  void pop();  void display();    void main()  {  int ch;    while(1) //infinite loop, will end when choice will be 4  {  printf("\n\*\*\* Stack Menu \*\*\*");  printf("\n\n1.Push\n2.Pop\n3.Display\n4.Exit");  printf("\n\nEnter your choice(1-4):");  scanf("%d",&ch);    switch(ch)  {  case 1: push();  break;  case 2: pop();  break;  case 3: display();  break;  case 4: exit(0);    default: printf("\nWrong Choice!!");  }  }  }    void push()  {  int val;    if(top==MAX-1)  {  printf("\nStack is full!!");  }  else  {  printf("\nEnter element to push:");  scanf("%d",&val);  top=top+1;  stack[top]=val;  }  }    void pop()  {  if(top==-1)  {  printf("\nStack is empty!!");  }  else  {  printf("\nDeleted element is %d",stack[top]);  top=top-1;  }  }    void display()  {  int i;    if(top==-1)  {  printf("\nStack is empty!!");  }  else  {  printf("\nStack is...\n");  for(i=top;i>=0;--i)  printf("%d\n",stack[i]);  }  } |

**Output**

*\*\*\* Stack Menu \*\*\**

*1.Push*  
*2.Pop*  
*3.Display*  
*4.Exit*

*Enter your choice(1-4):1*

*Enter element to push:3*

*\*\*\* Stack Menu \*\*\**

*1.Push*  
*2.Pop*  
*3.Display*  
*4.Exit*

*Enter your choice(1-4):1*

*Enter element to push:6*

*\*\*\* Stack Menu \*\*\**

*1.Push*  
*2.Pop*  
*3.Display*  
*4.Exit*

*Enter your choice(1-4):3*

*Stack is…*  
*6*  
*3*

*\*\*\* Stack Menu \*\*\**

*1.Push*  
*2.Pop*  
*3.Display*  
*4.Exit*

*Enter your choice(1-4):2*

*Deleted element is 6*  
*\*\*\* Stack Menu \*\*\**

*1.Push*  
*2.Pop*  
*3.Display*  
*4.Exit*

*Enter your choice(1-4):3*

*Stack is…*  
*3*