Array Implementation of Stack

In array implementation, the stack is formed by using the array. All the operations regarding the stack are performed using arrays. Lets see how each operation can be implemented on the stack using array data structure.

Adding an element onto the stack (push operation)

Adding an element into the top of the stack is referred to as push operation. Push operation involves following two steps.

1. Increment the variable Top so that it can now refere to the next memory location.
2. Add element at the position of incremented top. This is referred to as adding new element at the top of the stack.

Stack is overflow when we try to insert an element into a completely filled stack therefore, our main function must always avoid stack overflow condition.

Algorithm:

```
begin
   if top = n then stack full
   top = top + 1
   stack (top) : = item;
end
```

implementation of push algorithm in C language

```
1. void push (int val,int n) //n is size of the stack
2. {
3.    if (top == n )
4.    printf("\n Overflow");
5.    else
6.    {
7.    top = top +1;
8.    stack[top] = val;
9.    }
10. }
```

Deletion of an element from a stack (Pop operation)

Deletion of an element from the top of the stack is called pop operation. The value of the variable top will be incremented by 1 whenever an item is deleted from the stack. The top most element of the stack is stored in an another variable and then the top is decremented by 1. the operation returns the deleted value that was stored in another variable as the result.

The underflow condition occurs when we try to delete an element from an already empty stack.

Algorithm :

```
1. begin
2.     if top = 0 then stack empty;
3.     item := stack(top);
4.     top = top - 1;
5. end;
```

Implementation of POP algorithm using C language

```
1.  int pop ()
2.  {
3.      if(top == -1)
4.      {
5.          printf("Underflow");
6.          return 0;
7.      }
8.      else
9.      {
10.         return stack[top - - ];
11.     }
12. }
```

Visiting each element of the stack (Peek operation)

Peek operation involves returning the element which is present at the top of the stack without deleting it. Underflow condition can occur if we try to return the top element in an already empty stack.

Algorithm :

PEEK (STACK, TOP)

```
1. Begin
```

```
2.    if top = -1 then stack empty
3.    item = stack[top]
4.    return item
5.  End
```

Implementation of Peek algorithm in C language

```
1.  int peek()
2.  {
3.     if (top == -1)
4.     {
5.        printf("Underflow");
6.        return 0;
7.     }
8.     else
9.     {
10.       return stack [top];
11.    }
12. }
```

C program

```
1.  #include <stdio.h>
2.  int stack[100],i,j,choice=0,n,top=-1;
3.  void push();
4.  void pop();
5.  void show();
6.  void main ()
7.  {
8.     printf("Enter the number of elements in the stack ");
9.     scanf("%d",&n);
10.    printf("**********Stack operations using array**********");
11.
12. printf("\n----------------------------------------------\n");
13.    while(choice != 4)
```

```
14.    {
15.        printf("Chose one from the below options...\n");
16.        printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");
17.        printf("\n Enter your choice \n");
18.        scanf("%d",&choice);
19.        switch(choice)
20.        {
21.            case 1:
22.            {
23.                push();
24.                break;
25.            }
26.            case 2:
27.            {
28.                pop();
29.                break;
30.            }
31.            case 3:
32.            {
33.                show();
34.                break;
35.            }
36.            case 4:
37.            {
38.                printf("Exiting....");
39.                break;
40.            }
41.            default:
42.            {
43.                printf("Please Enter valid choice ");
44.            }
45.        };
46.    }
47. }
```

```
48. void push ()
49. {
50.     int val;
51.     if (top == n )
52.     printf("\n Overflow");
53.     else
54.     {
55.         printf("Enter the value?");
56.         scanf("%d",&val);
57.         top = top +1;
58.         stack[top] = val;
59.     }
60. }
61. void pop ()
62. {
63.     if(top == -1)
64.     printf("Underflow");
65.     else
66.     top = top -1;
67. }
68. void show()
69. {
70.     for (i=top;i>=0;i--)
71.     {
72.         printf("%d\n",stack[i]);
73.     }
74.     if(top == -1)
75.     {
76.         printf("Stack is empty");
77.     }
78. }
```