ARRAY IMPLEMENTATION OF STACK

A better implementation of Stack is usually using linked list unless you are sure of the number of elements in array. But if you are just starting with data structures and are not familiar with linked list, you can try implementing stack in an array. While using array to implement stack is easy, it has limitation that the stack cannot grow beyond a fixed size (of array).

Operations

There can be many operations on stack like this post, but we will discuss only three basic operations on the stack

Push – inserting element at the top of stack.

Pop – removing the top element of stack.

isEmpty – check is stack is empty or not.

Code

For simplicity reason lets assume the array (which hold stack) to be global and accessible to all the functions.

```
1    // Max number of elements a stack can hold
2    #define MAX 100
3
```

```
4   // Array to hold the stack
5   int s[MAX];
6
7   // will hold the index of top element of the stack.
8   // top == -1 means stack is empty
9   int top = -1;
10
11  /** Push the element if there is a space in the Stack
12   * If Stack is full then does nothing.
13   */
14  void  push(int value)
15  {
16      if (top == MAX-1)
17      {
18          printf("Stack is full cannot push more elements.");
19          return;
20      }
21      s[++top] = value;
22  }
23
24  /** Pop the top element from stack and returns it. If stack is empty return -1.
25   */
26  int pop()
```

```
27      {
28          if (top == -1)
29          {
30              printf("Empty Stack. Nothing to return.");
31              return -1;
32          }
33          return s[top--];
34      }
35      /** returns true if stack is empty, else false.
36       */
37      bool isEmpty()
38      {
39          return (top == -1);
40      }
```

Understanding above code

Above code is in way a good code. There are lot of problems, to start with pop returns -1 when stack is empty which means that -1 cannot be a valid member of the stack.

Push function should ideally return something to tell its calling function whether or not it was able to insert element in the stack. printf is not a good way in production code, because most often it ends up printing in logs.

The size of Array is 100, if we just need 10 element stack, then 90 element memory will be wasted. In other case, if we want a stack with 500 element, we will have to change MAX value, which means program need to be compiled again. The solution to this can be to get the size of stack from the calling function and allocate memory using malloc (or new in C++). But then we should be cautious about deallocating memory when done, else it will leave memory leaks.

Time Complexity

All the above operations are constant time operations and takes O(1) time. In fact, almost all the operations of Stack and Queue are constant time operations, except for operations like printing all elements, etc.

Source: http://www.ritambhara.in/array-implementation-of-stack/