**Data Structure - Stack**

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).

**Mainly the following three basic operations are performed in the stack**

* **Push:** Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
* **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
* **Peek or Top:** Returns top element of stack.
* **isEmpty:** Returns true if stack is empty, else false.

**Implementing Stack using Arrays**

/\* C++ program to implement basic stack operations \*/

#include<iostream>

using namespace std;

#define MAX 1000

class Stack

{

int top;

public:

int a[MAX]; //Maximum size of Stack

Stack() { top = -1; }

bool push(int x);

int pop();

bool isEmpty();

};

bool Stack::push(int x)

{

if (top >= (MAX-1))

{

cout << "Stack Overflow";

return false;

}

else

{

a[++top] = x;

return true;

}

}

int Stack::pop()

{

if (top < 0)

{

cout << "Stack Underflow";

return 0;

}

else

{

int x = a[top--];

return x;

}

}

bool Stack::isEmpty()

{

return (top < 0);

}

// Driver program to test above functions

int main()

{

struct Stack s;

s.push(10);

s.push(20);

s.push(30);

cout << s.pop() << " Popped from stack\n";

return 0;

}

Pros: Easy to implement. Memory is saved as pointers are not involved.

Cons: It is not dynamic. It doesn’t grow and shrink depending on needs at runtime.

10 pushed to stack

20 pushed to stack

30 pushed to stack

30 popped from stack

Top item is 20