Stack

- A stack is one of the most important and useful non-primitive linear data structure in computer science.
- It is an ordered collection of items into which new data items may be added/inserted and from which items may be deleted at only one end, called the **top** of the stack.
- As all the addition and deletion in a stack is done from the **top** of the stack, the last added element will be first removed from the stack.
- That is why the stack is also called **Last-in-First-out (LIFO)**.
- The most frequently accessible element in the stack is the **top** most elements, whereas the least accessible element is the **bottom** of the stack.
- The insertion (or addition) operation is referred to as **push**, and the deletion (or remove) operation as **pop**.
- A stack is said to be empty or underflow, if the stack contains no elements.
- At this point **the top of the stack** is present at the bottom of the stack. And it is overflow when the stack becomes full, i.e., no other elements can be pushed onto the stack. At this point the **top pointer is at the highest location of the stack**.

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.
* MakeNull: Make stack empty stack.
```

Algorithm for PUSH

Suppose STACK [SIZE] is a one-dimensional array for implementing the stack, which will hold the data items. TOP is the pointer that points to the top most element of the stack. Let DATA is the data item to be pushed.

```
1. If TOP = SIZE – 1, then:
    (a) Display “The stack is in overflow condition”
    (b) Exit
2. TOP = TOP + 1
3. STACK [TOP] = ITEM
4. Exit
```

Algorithm for POP

Suppose STACK [SIZE] is a one-dimensional array for implementing the stack, which will hold the data items. TOP is the pointer that points to the top most element of the stack. DATA is the popped (or deleted) data item from the top of the stack.

```
1. If TOP < 0, then:
    (a) Display “The Stack is empty”
    (b) Exit
2. Else remove the Top most element
    a. DATA = STACK[TOP]
    b. TOP = TOP – 1
5. Exit
```

Application of Stack :

```
* Parsing
* Recursive Function
* Calling Function
* Expression Evaluation
* Expression Conversion
    a. Infix to Postfix
    b. Infix to Prefix
    c. Postfix to Infix
    d. Prefix to Infix
* Towers of hanoi
```

Program implementing stack using array

```
#include <stdio.h>
#include <stdlib.h>
#define size 100
struct Stack{
```

```
    int stack[size];
    int top;
}s;
void push(int);
void pop();
void show();

int main(){
    s.top = -1;
    int ch;
    while(1){
        printf("*** Menu ***\n");
        printf("Press 1. to perform push operation \n");
        printf("Press 2. to perform pop operation \n");
        printf("Press 3. to show data\n");
        printf("Press 4. to exit from program \n");
        printf("Enter your choice: ");
        scanf("%d",&ch);
        switch(ch){
            case 1:
                {
                    int number;
                    printf("Enter the number you want to put on stack: ");
                    scanf("%d",&number);
                    push(number);
                }
            break;
            case 2:
                pop();
            break;
            case 3:
                show();
            break;
            case 4:
                printf("\nExited");
                exit(0);
            default:
                printf("Enter approprite number\n");
            break;
        }
    }
}
void push(int x){
    if(s.top >= size-1){
        printf("Stack is full!!!\n");
        exit(0);
    }
    else{
        s.top++;
        s.stack[s.top] = x;
    }
}

void pop(){
    int x;
    if(s.top==-1){
        printf("The stack is empty!!!\n");
        exit(0);
    }
    else{
        x = s.stack[s.top];
        s.top--;
        printf("poped element is %d\n",x);
    }
}

void show(){
    for(int i=0;i<=s.top;i++){
        printf("The entered number/s:\n");
        printf("%d\n",s.stack[i]);
    }
```

```
}
```

Output

```
*** Menu ***
Press 1. to perform push operation
Press 2. to perform pop operation
Press 3. to show data
Press 4. to exit from program
Enter your choice: 1
Enter the number you want to put on stack: 10
*** Menu ***
Press 1. to perform push operation
Press 2. to perform pop operation
Press 3. to show data
Press 4. to exit from program
Enter your choice: 1
Enter the number you want to put on stack: 20
*** Menu ***
Press 1. to perform push operation
Press 2. to perform pop operation
Press 3. to show data
Press 4. to exit from program
Enter your choice: 3
The entered number/s:
10
The entered number/s:
20
*** Menu ***
Press 1. to perform push operation
Press 2. to perform pop operation
Press 3. to show data
Press 4. to exit from program
Enter your choice: 2
poped element is 20
*** Menu ***
Press 1. to perform push operation
Press 2. to perform pop operation
Press 3. to show data
Press 4. to exit from program
Enter your choice: 4
```