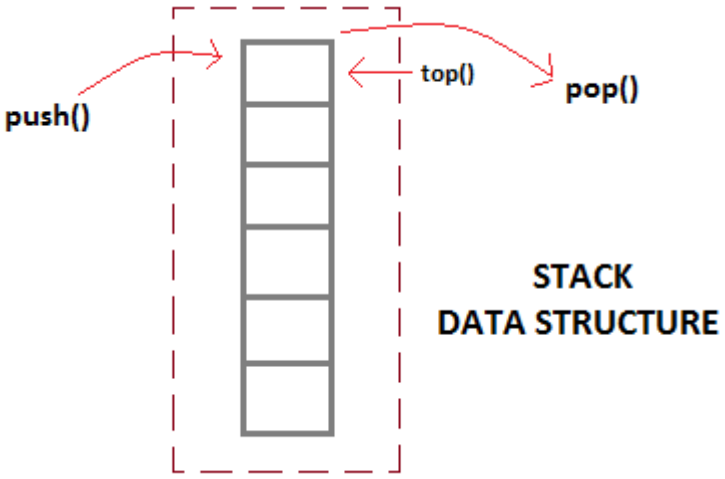
What is Stack Data Structure?

Stack is an abstract data type with a bounded(predefined) capacity. It is a simple data structure that allows adding and removing elements in a particular order. Every time an element is added, it goes on the **top** of the stack and the only element that can be removed is the element that is at the top of the stack, just like a pile of objects.



Basic features of Stack

1. Stack is an **ordered list** of **similar data type**.
2. Stack is a **LIFO**(Last in First out) structure or we can say **FILO**(First in Last out).
3. `push()` function is used to insert new elements into the Stack and `pop()` function is used to remove an element from the stack. Both insertion and removal are allowed at only one end of Stack called **Top**.
4. Stack is said to be in **Overflow** state when it is completely full and is said to be in **Underflow** state if it is completely empty.

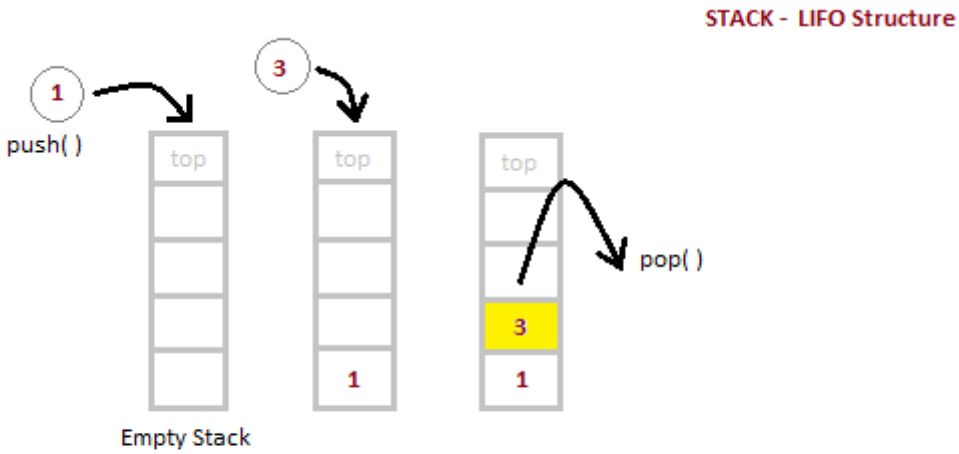
Applications of Stack

The simplest application of a stack is to reverse a word. You push a given word to stack - letter by letter - and then pop letters from the stack.
There are other uses also like:

1. Parsing
2. Expression Conversion(Infix to Postfix, Postfix to Prefix etc)

Implementation of Stack Data Structure

Stack can be easily implemented using an Array or a Linked List. Arrays are quick, but are limited in size and Linked List requires overhead to allocate, link, unlink, and deallocate, but is not limited in size. Here we will implement Stack using array.



In a Stack, all operations take place at the "**top**" of the stack. The "**push**" operation adds an item to the top of the Stack.
The "**pop**" operation removes the item on top of the stack.

Algorithm for PUSH operation

1. Check if the stack is **full** or not.
2. If the stack is full, then print error of overflow and exit the program.
3. If the stack is not full, then increment the top and add the element.

Algorithm for POP operation

1. Check if the stack is empty or not.
2. If the stack is empty, then print error of underflow and exit the program.
3. If the stack is not empty, then print the element at the top and decrement the top.

```
/* stack using array */

#include<stdio.h>
int stack[10],top=-1,size=10;
void main()
{
int opt=0;
 while(1)
 {
clrscr();
printf("\nstack Operation");
printf("\n1. Push");
printf("\n2. Pop");
printf("\n3. Peep");
printf("\n4. Exit");
printf("\nEnter your option");
scanf("%d",&opt);
  switch(opt)
  {
   case 1:
```

```
    push();
    break;
   case 2:
    pop();
    break;
   case 3:
    peep();
    break;
   case 4:
    exit();
   default:
printf("\nInvalid option");
  }
 }
}

push()
{
int item;

  if(top==size)
printf("\nStack Full");
  else
 {
printf("enter an item");
scanf("%d",&item);
  stack[++top]=item;
 }
}

 peep()
 {
int count=top;
  while(count>=0)
 {
printf("\n%d",stack[count]);
  count--;
 }
getch();
}

pop()
{
```

```
 if (top<0)
printf("Stack Empty");
 else
printf("\npopped item is %d",stack[top--]);
getch();
}
```