1- **Stack:**

- A stack is an ordered group of homogeneous items (elements), in which the removal and addition of stack items can take place only at the top of the stack.
- To add (push) an item to the stack, it must be placed on the top of the stack.
- To remove (pop) an item from the stack, it must be removed from the top of the stack too.
- Thus, the last element that is pushed into the stack is the first element to be popped out of the stack.

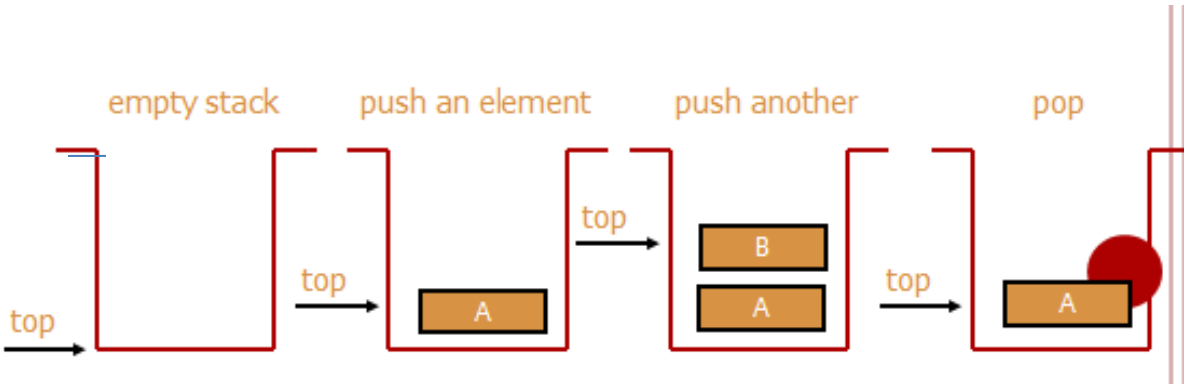
  i.e., last in First out (LIFO)

Operations of Stack:

- Emptystack: return(one) if stack is empty, return (zero) otherwise
- Fullstack: return (one) if stack is full, return (zero) otherwise
- Top: return the element at the top of stack
- Push: add an element to the top of stack
- Pop: delete the element at the top of stack
- Display Stack: print all the data in the stack

Primary operations:

- Push
  - Add an element to the top of the stack
- Pop
  - Remove the element at the top of the stack



Push Algorithm:

1-[overflow]

if Top>=N

Then Over flow

2-[increment pointer]

Top ⟵ Top+1

3-[insert element]

Stack [top] ⟵ new element

Pop Algorithm:

1-[underflow]

if Top<=-1

Then under flow

2-[unstack element]

element ⟵ stack[top]

3-[decrement pointer]

Top ⟵ Top-1

Implementation:

```
int fullstack()                    int emptystack()
{                                  {
 if(top>=size-1)                   if(top==-1)
 return(1);                        return(1);
 else return(0);                   else return(0);
}                                  }
```

---

void push(char stack[n],int&top,char x) // بناء دالة الإضافة

```
                {If ( fullstack() )
                  cout<<"stack over flow \n";
                        else
                                {
                                top++;
                                stack[top]=x;
                                }
                }
```

void pop(char stack[n],int&top,char x) // بناء دالة الإضافة

```
          {
           if (emptystack())
                                cout<<"stack is under flow\n";
                        else
                         {  x=stack[top];
                         top--;
       cout<<"\n the value pop is :"<<x;
```

```
                    }
        }
```

Full stack program:

```
#include<iostream.h>
#define n 30
```

يكون & لأن عملية السحب والإضافة يتغير هذا المؤشر داخل الدالة لذلك يجب أن َيعرف هذا التغير البرنامج الرئيسي//

```
void push(char stack[n],int&top,char x)   // بناء دالة الإضافة
            { if(top>=n-1) cout<<"stack over flow \n";
                    else
                            {
                            top++;
                            stack[top]=x;} }
```

```
void pop(char stack[n],int&top,char x) // بناء دالة الإضافة
        {
         if (top==-1)
                            cout<<"stack is under flow\n";
                    else
                    {  x=stack[top];
                    top--;
      cout<<"\n the value pop is :"<<x;
                    }
        }
```

```
void clear(int &top) هذه الداله لتصفير
  {
```

```
 top=-1;
  }
int full(int top)؟ تختبر هل المكدس ممتلئ أم لا
{
        if(top==n-1)
                return 1;
         else
                 return 0;
}
int empty (int top) // ؟ تختبر هل المكدس فارغ أم لا
{
if(top==-1)
                return 1;
        else
                return 0;

void main()  {
char stack[n],x;
int top= -1 ,c;
do{
cout<<"1: clear stack\n";
cout<<"2: push for the stack\n";
cout<<"3: pop from the stack\n";
cout<<"4: stack full or No\n";
```

```
cout<<"5: stack empty or No\n";
cout<<"6: Exist\n";
cout<<" inter your choice :";
cin>>c;
 switch(c)  {
        case 1: clear(top);  break;
        case 2: cout <<"enter the value : ";
                                cin>>x;    // إدخال القيمة المراد إضافتها
                                push(stack,top,x);
                                cout<<"\n";   break;
        case 3: pop (stack,top,x);
                                 cout<<"\n";   break;
        case 4: if(full(top))  // هنا استدعاء للدالة التي تختبر هل أن المكدس ممتلئ لو لا
              cout<<" :::stack full\n";
                                        else cout<<"\n stack not full\n";  break;
        case 5: if((empty (top)) )  // هنا استدعاء للدالة التي تختبر هل أن المكدس فارغ لو لا
                            cout<<":::stack empty\n";
                                 else cout<<"\n :::stack not empty \n"; break;
        case 6: break; }
} while(c!=6);
} }  //out put
```

```
1: clear stack
2: push for the stack
```

```
3: pop from the stack

4: stack full or No

5: stack empty or No

6: Exist

enter your choice :2

enter the value : a

1: clear stack

2: push for the stack

3: pop from the stack

4: stack full or No

5: stack empty or No

6: Exist

enter your choice :6
```