هياكل البيانات

المرحلة الثانية

محاضرة(3)

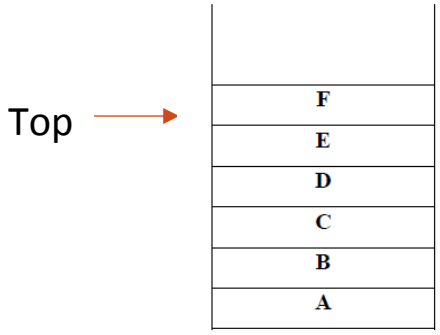
م.م فرح معاذ جاسم

Farahmaath86@uoanbar.edu.iq

1.STACK DATA STRUCTURE

A stack is an ordered collection of items into which new items may be inserted and from which items may be deleted at one end, called the top of the stack.

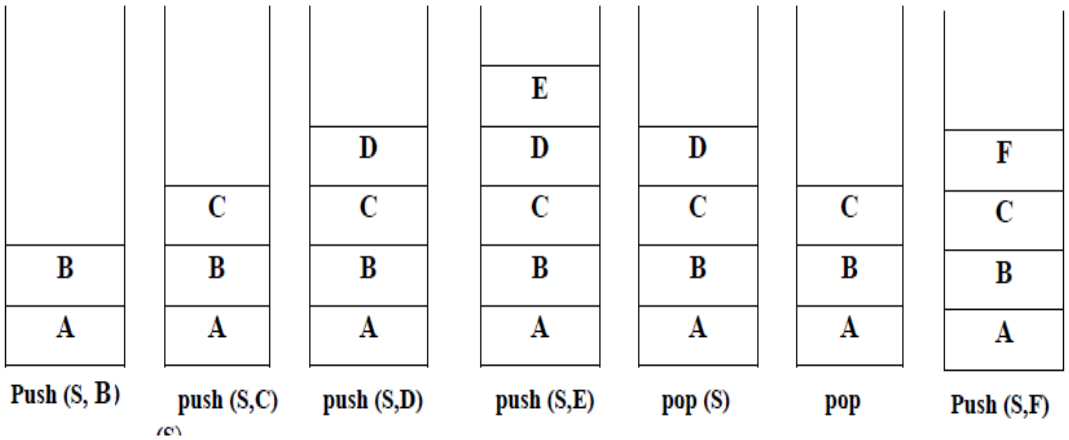
In the figure below, F is physically higher on the page then all the other items in the stack, so F is the current top element of the stack, If any new items are added to the stack they are placed on top of F, and if any items are deleted, F is the first to be deleted. Therefore, a stack, is collected a last – in – first – out.(LIFO) list.



1.2 STACK OPERATIONS

The two main operations which can be applied to a stack are given spatial names (push, pop)

- **push()** − Pushing (storing) an element on the stack.
- **pop()** − Removing (accessing) an element from the stack.



we need to check the status of stack as well. For the same purpose, the following functionality is added to stacks :

- **isFull()** − check if stack is full.
- **isEmpty()** − check if stack is empty.

At all times, we maintain a pointer to the last PUSHed data on the stack. As this pointer always represents the top of the stack, hence named top**. The top** pointer provides top value of the stack without actually removing it.

- **Algorithm of isFull() function**

```
begin procedure isfull
   if top equals to MAXSIZE
      return true
   else
      return false
   endif
end procedure
```

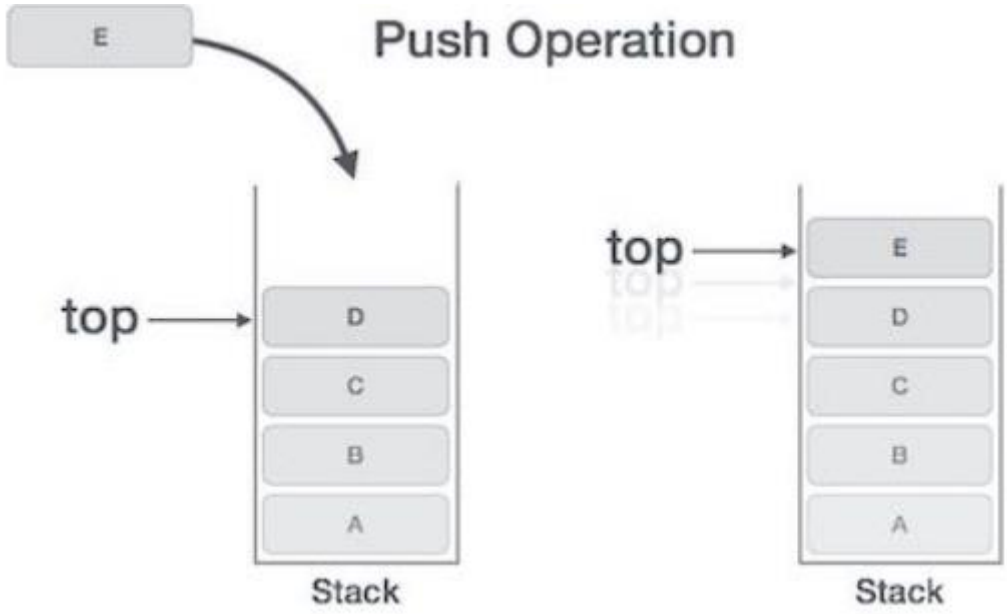
- **Algorithm of isEmpty() function**

```
begin procedure isempty
   if top less than 1
      return true
   else
      return false
   endif
end procedure
```

1.2.1 PUSH OPERATION

The process of putting a new data element onto stack is known as a Push Operation. Push operation involves a series of steps −

- **Step 1** − Checks if the stack is full.
- **Step 2** − If the stack is full, produces an error and exit.
- **Step 3** − If the stack is not full, increments **top** to point next empty space.
- **Step 4** − Adds data element to the stack location, where top is pointing.
- **Step 5** − Returns success.



A simple algorithm for Push operation can be derived as follows :

```
begin procedure push: stack, data

   if stack is full
      return null
   endif

   top ← top + 1
   stack[top] ← data

end procedure
```

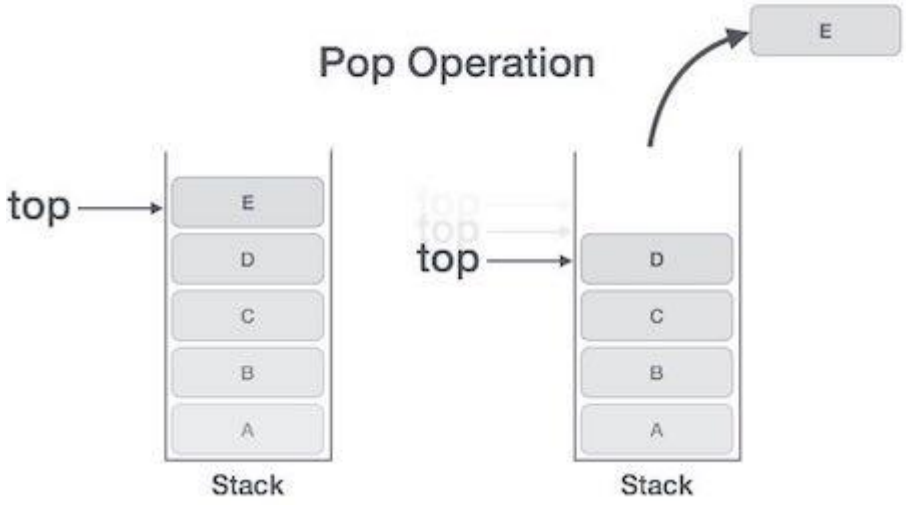
1.2.2 POP OPERATION

Accessing the content while removing it from the stack, is known as a Pop Operation. In an array implementation of pop() operation, the data element is not actually removed, instead top is decremented to a lower position in the stack to point to the next value.

A Pop operation may involve the following steps:

Step 1 − Checks if the stack is empty.
Step 2 − If the stack is empty, produces an error and exit.
Step 3 − If the stack is not empty, accesses the data element at which top is pointing.

Step 4 − Decreases the value of top by 1.
Step 5 − Returns success.



A simple algorithm for Pop operation can be derived as follows:

```
begin procedure pop: stack

   if stack is empty
      return null
   endif

   data ← stack[top]
   top ← top - 1
   return data

end procedure
```

To display al element in stack

```
void display() {
   if(top>=0) {
      cout<<"Stack elements are:";
      for(int i=top; i>=0; i--)
         cout<<stack[i]<<" ";
         cout<<endl;
   }
else
      cout<<"Stack is empty";
}
```