Semester - 4

Date 22/4/2020

Data structure using C

Stack is an Abstract Data Type (ADT), commonly used in most programming languages. It is named stack as it behaves like a real world stack for example - a deck of cards or file of plates etc.

Stack ADT allows all data operations at one end only. At any given time, we can only access the top element of stack.

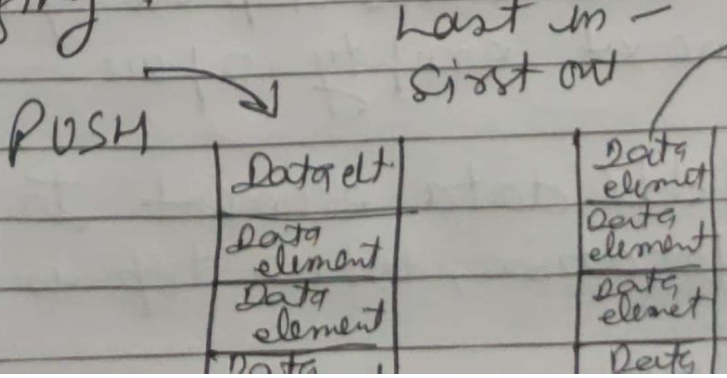
This feature makes it LIFO data structure. LIFO stands for Last-in-First-out.

Here the element which is placed last, is accessed first.

In stack terminology, insertion operation is called PUSH operation and removal operation is called POP operation.

Stack Representation → A stack can be implemented by means of Array, structure, Pointer and linked list.

Stack can either be a fixed size one or it may have a sense of dynamic resizing.



DATE : / 20
PAGE No

Basic Operation

Stack operations may involve initializing the stack, using it and then de-initializing it.

Apart from these, a stack is used for the following two primary operations →

Push() - Pushing (storing) an element on the stack.

Pop() - Removing (accessing) an element on the stack.

Push Operation →

The process of putting a new data element onto stack is known as Push operation.

Push operation involves series of steps →

Step-1 - Checks if the stack is full.

Step-2 - If the stack is full, produces an error and exit.

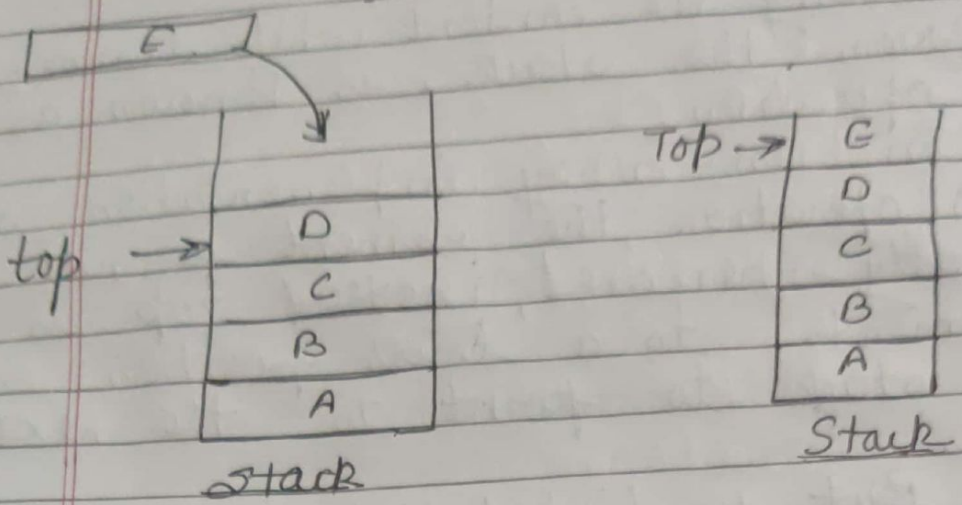
Step-3 - If the stack is not full, increments top to point next empty space.

Step-4 - Adds data element to the stack location, where top is pointing.

Step-4 - Returns success.

DATE : / 20
PAGE No

Push operation



Algorithm for PUSH operation

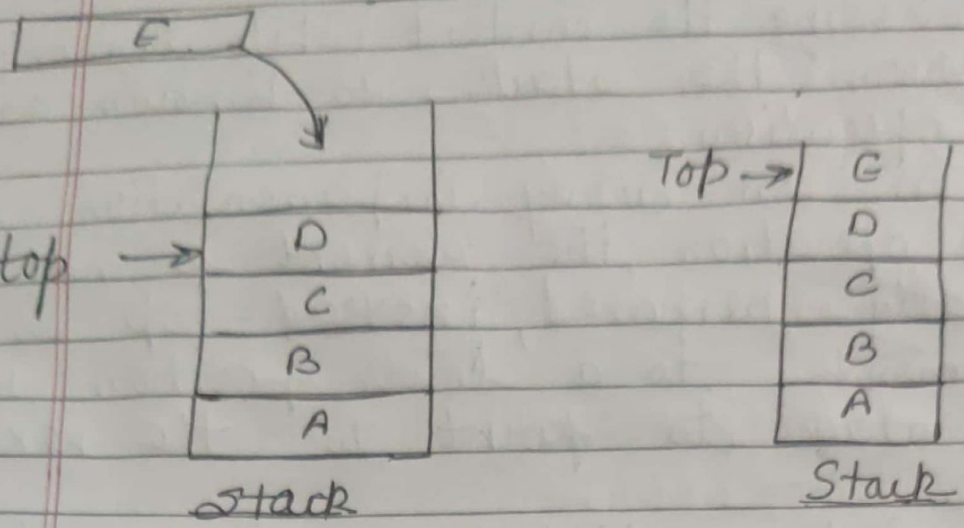
```
begin procedure push : stack , data
    if stack is full
        return null
end if
    top ← top + 1
    stack [top] ← data
end procedure
```

Implementation of this algorithm in C →

```
void push (int data) {
  if (! isFull () ) {
        top = top + 1 ;
    stack [top] = data;
} else {
  printf (" could not insert data,
            stack is full .\n");
    }
}
```

PAGE No

Push Operation



Algorithm for PUSH operation

```
begin procedure push : stack , data
    if stack is full
        return null
end if
    top ← top +1
  stack [top] ← data
end procedure
```

Implementation of this algorithm in C →

```
void push (int data) {
  if (! isFull () ) {
        top = top +1 ;
  stack [top] = data;
} else {
  printf (" could not insert data,
          stack is full .\n");
    }
}
```

DATE: / 20
PAGE No

Pop Operation

Accessing the content while removing it from the stack, is known as Pop operation.

In an array implementation of Pop() operation, the element is not actually removed, instead top is decremented to a lower position in the stack to point to the next value.

But in Linked-list implementation, pop() actually removes data element and deallocates memory space.

A Pop operation may involve the following steps →

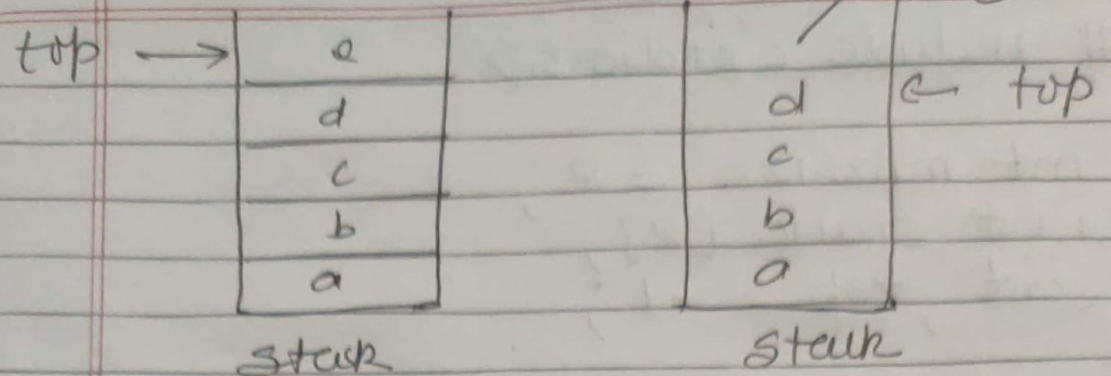
Step 1 – Checks if the stack is empty.

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.



Algorithm for Pop operation →

```
begin procedure pop : stack
    If stack is empty
        return null
end if
    data ← Stack [top]
    top ← top - 1
    return data
end procedure
```

Implementation of algorithm

```
int pop(int data) {
  if (! isempty()) {
    data = stack [TOP];
      top = top - 1;
    return data;
} else {
  printf("could not retreive data, stack is
                                empty\n");
}
}
```