11, April, 2022

DATE

Stacks & Queues

STACK ① → Recursion is based upon it.

② Abstract datatype

(Revision)

1) Introduction

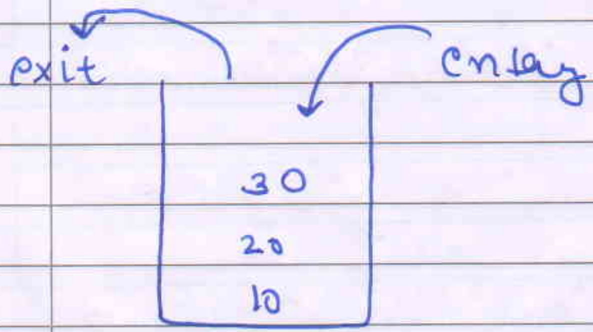
2) Implimentation → ARRAY
→ Linkedlist

3) INBUILD STACK

4) Dynamic stack

5) Templates

Revision → 30/05/22



1) insert → push() → push(10)

2) Delete → pop() → pop()

3) Access top Most element
↓
top() → 10

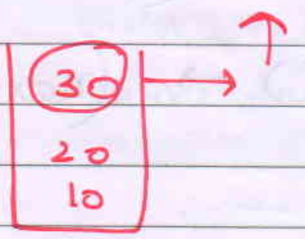
4) size() → 1

5) is empty() → Bool

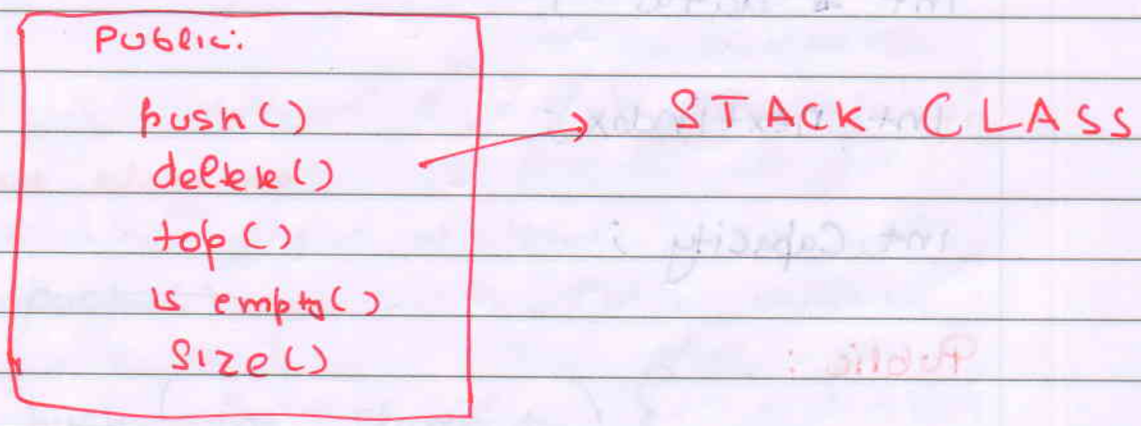
DATE

Stack using array →

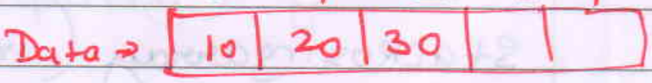
Stack → L I F O
Last In First Out



Stack can be implimented using
↳ ARRAYS
↳ Linked list



push (10);
push (20);
push (30)



next index
~~0~~
~~1~~
~~2~~
3

DATE

- top(); → 30 → [next index - 1]
- pop(); → [next index --]
- size() → (next index)

is empty () → (next index == 0) else

↓ true ↓ false

Stackusingarray.cpp

```
Class StackUSing Array {
Int * data ;
Int nextindex;
Int Capacity ;
Public :
• // Initliasation with size
Stackusingarray (int totalsize) {
data = new int [total size] ;
next index =0 ;
}
```

classmate

PAGE

```
• // to see the size of Stack
int size() {

    return nextindex ;

}
• // Check weather Stack is empty or not/
bool isempty() {

if ( nextindex ==0) return true ;

else return false ;

                    or

   return nextindex == 0 ;

}

• // Insert element

~~Void push()~~

Void push ( int element ) {
    if ( nextindex == Capacity ) {
        Cout << " Stack full " << endl ;
        return ;
    }
    data [next index] = element ;
    next index ++ ;
}
```

DATE

• // Delete Element

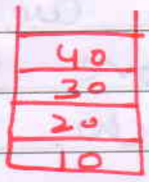
```
int pop () {
if ( isempty (~~t~~)) {
  Cout << "Stack is empty" << endl ;
  return Int_MIN ;
}

next index -- ;
return data [next index ] ;

}
```

• // Display the value at top of Stack.

```
int top () {

if ( isempty () ) {

  Cout << "Stack is empty" << endl;
  return INT_MIN ;

}
return data [next index - 1] ;

}
```

DATE

```
# include < iostream >
using namespace std ;
# include " Stacks using array .CPP " ,

int main () {

StackUsingArray S( 4 ) ;   } Constructor called that we Made

S. push (10) ;    }
S. push (20) ;    }  ->
S. push (30) ;    }
S. push (40) ;    }
S. push (50) ; ——  X "Stack FULL"

Cout << S. top () << endl;   -> 40

Cout << S. pop () << endl ;   }  40
Cout << S. pop () << endl ;   }  30
Cout << S. pop () << endl ;   }  20

Cout << S. size () << endl ;  } -> 1

Cout << S. isempty () << endl;  } -> false
```



Output ->

```
Stack FULL
40
40
30
20
1
0
```