Stacks and its applications

BY

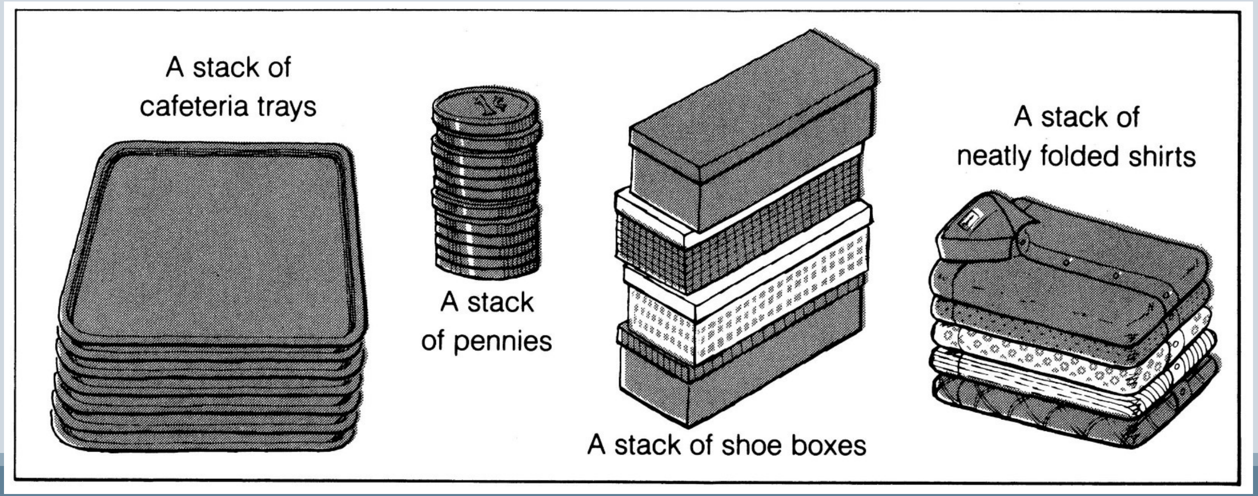
DR. RASHMI NAVEEN RAJ, I&CT DEPT.,

rashmi.naveen@manipal.edu

LECTURE- 6, OCT 13, 2021

What is a stack?

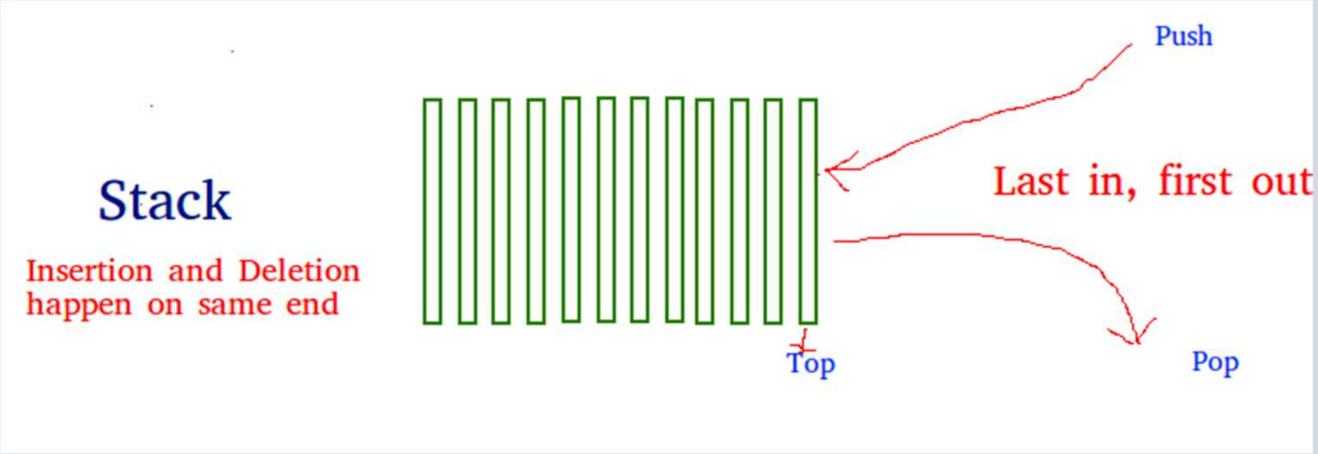
- It is an ordered group of homogeneous items of elements.
- Elements are added to and removed from the top of the stack (the most recently added items are at the top of the stack).
- The last element to be added is the first to be removed (**LIFO**: Last In, First Out).



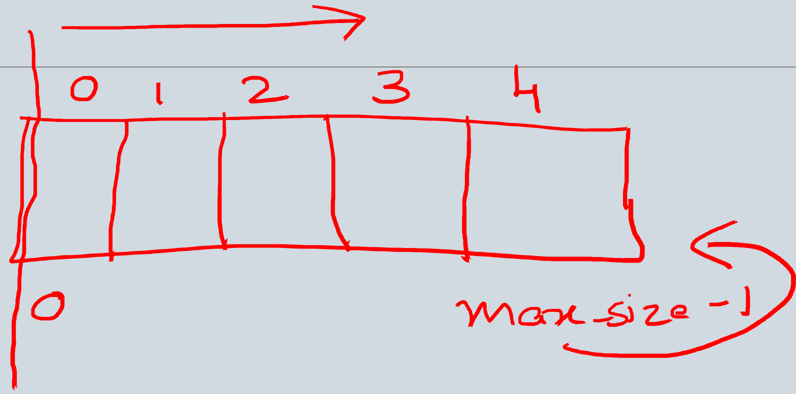
STACK

Stack is a linear data structure which follows a particular order in which the operations are performed.

The order may be LIFO(Last In First Out) or FILO(First In Last Out).



Examples to illustrate stack operations:

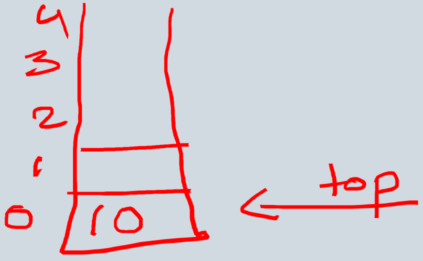


top = int variable

top = -1

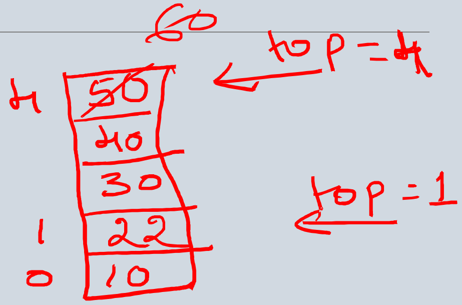
① Insert 10, PUSH 10

top = top +1



a[0]

② PUSH 22

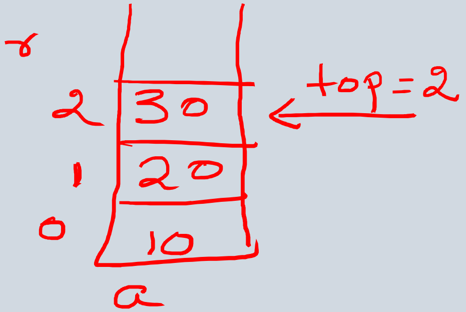


```
void push (int ele)
{
    if (top = =maxsize -1)
        cout<<"stack is full";
    else
    {
        top = top + 1;
        a[top] = ele;
    }
}
```

Stack ~~as an ADT~~

```
int pop( )
{   if(top==-1)
       { cout<<"stack is empty";
                   return -1; }
    else
       return(a[top--]);
       // a[top] is returned
       // top=top-1;
```

consider



```
POP: 30, top=top-1, top=1 }
     20,    "     , top=0
     10,    "     , top=-1
pop() ?
```

```
#define maxsize 10 // macro
```

Stack Program

```
enum Boolean{FALSE, TRUE};
               0      1
```

```
class stack{
        int top;
        int a[max-size];
public:
        stack(){top=-1;}
        Boolean Isfull();
        Boolean Isempty();
        void push(int);
        int pop();
        void display();
};
```

```
Boolean stack::Isfull()
{   if(top==max-size-1)
            return TRUE;
    return FALSE;
}
Boolean stack::Isempty()
{   if(top==-1)
            return TRUE;
}   return FALSE
}
```

Stack Program

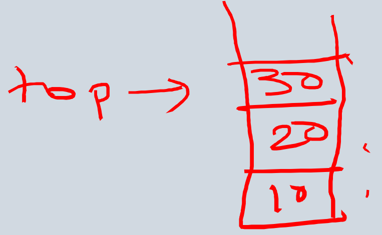
```
void stack:: push(int ele)
{     if(Isfull())
         { cout<<"stack is full\n";
   else
        a[++top] =ele;
}
int stack:: pop()
{ if (Isempty())
      {cout<<"stack is empty\n"
           return (-1);
      }
 else
      return (a[top--]);
}
```

```
void stack:display ()
{ if(Isempty())
      cout<<"stack is empty\n";
  else cout<<"stack content: ";
  { for(int i=0; i<=top; i++)
        cout<<a[i]<<"  ";
  } cout<<"\n";         }
}
```

30, 20, 10

```
for(int i=top; i>=0; i--)
```

c



30 20 10

Stack Program

Main program :

1. PUSH
2. POP
3. DISPLAY
4. EXIT

option : ?

Stack main() Program

```
int main()
 { stack1 s;  //object
   int n,ele;

  do{
     cout<<"1. PUSH \n 2. POP \n 3. DISPLAY \n 4. EXIT \n";
     cout<<"Enter option: ";
     cin>>n;
     switch(n)
     {
        case 1: {cout<<"enter the element to be pushed:";      cin>>ele; s.push(ele);break;}
        case 2: cout<<s.pop()<<" "; break;
        case 3: s.display(); break;
        case 4: exit(0);
     }
  }while(1);

  return 0;
 }
```

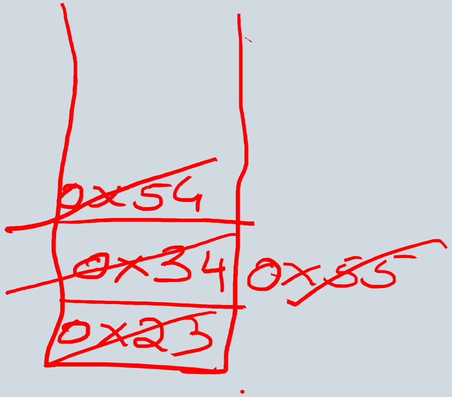
Application1: function call

```
void fn1( )
{
  cout<<"In fn1 \n";
  fn2();
0x34 fn3();
0x35 }

void fn2()
{ cout<<"In fn2";
  fn3();
0x54
}

void fn3()
{ cout<<"In fn3 ";
}
```

```
int
~~void~~ main()
{
  fn1();
0x23 cout<<"at the end of main";
  return 0;
}
```

O/P :

```
In fn1
In fn2
In fn3
In fn3
at the end of main
```



Application2: Recursive function call

```
int fact(int n)
{    if(n==1 || n==0)
          return 1;
     return (n*fact(n-1));
}
```

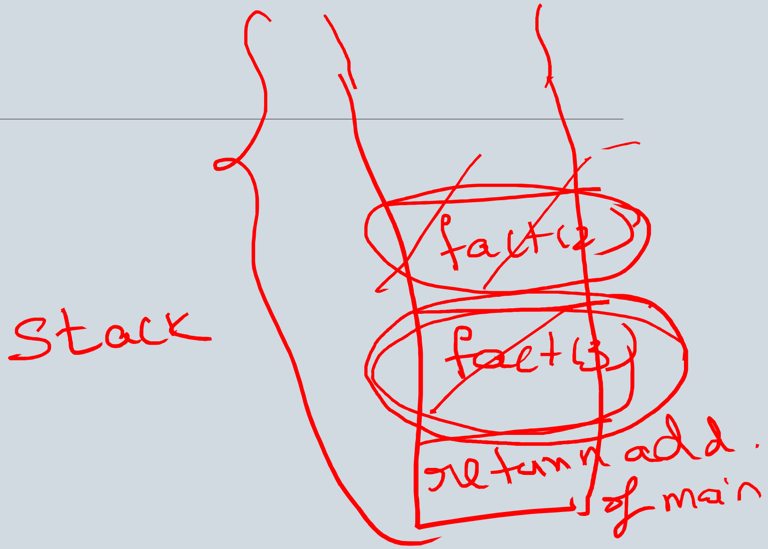
① n = 3

```
int fact(3)
{
→ return (3 * fact(2))
}
```

n

$\text{fact}(3) \xrightarrow{n=2} \text{fact}(2) \longrightarrow \text{fact}(1)$

$3 * \underbrace{\text{fact}(2)}_{2} \leftarrow 2 \quad 2 * \overset{1}{\text{fact}(1)} \quad 1$



Application3: Number base conversion

$(N)_{10} \Rightarrow (\ )_b$

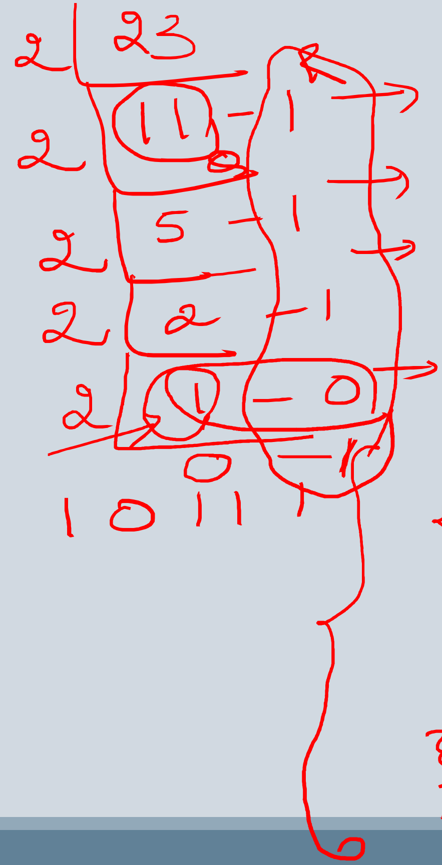
$(23)_{10} \Rightarrow (\ )_8$

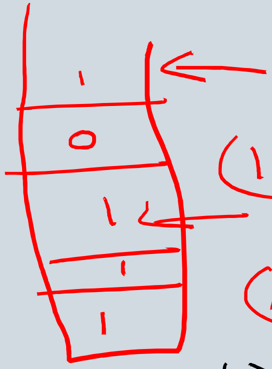
$N = 23 \quad b = 8$

$8 \,|\, 23 \quad R$

$\boxed{2} - 7$

$(27)_8$





$(10111)_2$

$(N)_{10} \quad (\ )_b$

```
while(N>0)   ~~N >= b~~
{   R = N%b;
    N = N/b;
    ~~N = 0~~
    S.push(R);
}
S.display(); // i=top
```

Application 4: To check is string is palindrome

```
int main()
 { stack1 s;
   char str[10],str_rev[10];
    int n,ele,i;

  cout<<"Enter the string\n"; cin>>str;

  for(i=0;str[i]!='\0';i++)
       s.push(str[i]);
  n=i;

cout<<"string length is :"<<n<<endl;
```

manipal'\0'

l
a
p
i
n
a
m

```
for(i=0;i<n;i++)
       str_rev[i]=s.pop();
  str_rev[i]='\0';

cout<<"Reversed string is :"<<str_rev<<endl;

  if(strcmp(str,str_rev)==0)
      cout<<"String is a   palindrome\n";
  else
       cout<<"String is not a palindrome\n";
  return 0;
   }
```

lapinam'\0'

string.h

ABC
ABF   -1

ABF
ABC   2

Evaluation of Arithmetic expressions

- The Representation and evaluation of expression is of great interest to computer scientists.
- An expression will have operators with different precedence and different associativity

  20

- Ex: 8-3*4 = ?

  -4

- Expression needs to be evaluated based on the precedence of operators
- 3 types of expressions:
  1. Infix Expression : A+B
  2. Postfix Expression : AB+
  3. Prefix expression : +AB

Infix to Postfix/prefix expression conversion:examples

- *, /, % → same precedence, left to right associativity---level1
- +, - → same precedence , , left to right associativity ---level 2

Postfix (operand1 operand2 opr) Prefix

① 2+3-4

(23+) -4

23+4-

② 2+3*4

2 + (34*)

opnd1 opnd2

234*+

③ 2*3+4

23* + 4

23*4+