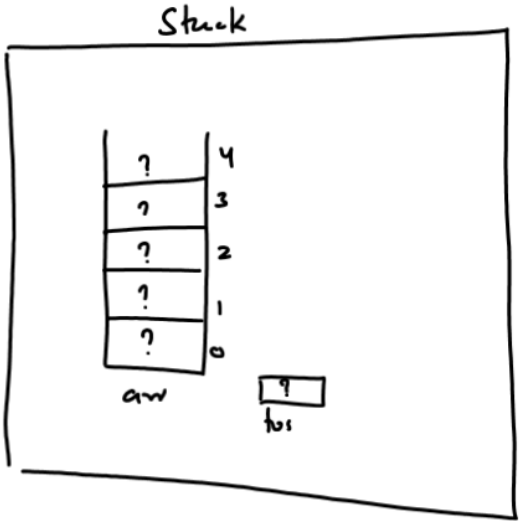
**Implementing Stack In C**




To implement a Stack in c lang , the best approach is to use **Structure.**

Why we prefer Structure for implementing Stack in C ?
==========
As we know , a Stack is a combination of two elements:

a. An array: For holding the data
b. A tos: For holding index

And both these elements ALWAYS WORK TOGETHER. That is , in every Stack operation like PUSH or POP we will require both these elements . So it is good programming practice to keep them in a single unit .

In c lang this SINGLE UNIT is a STRUCTURE.

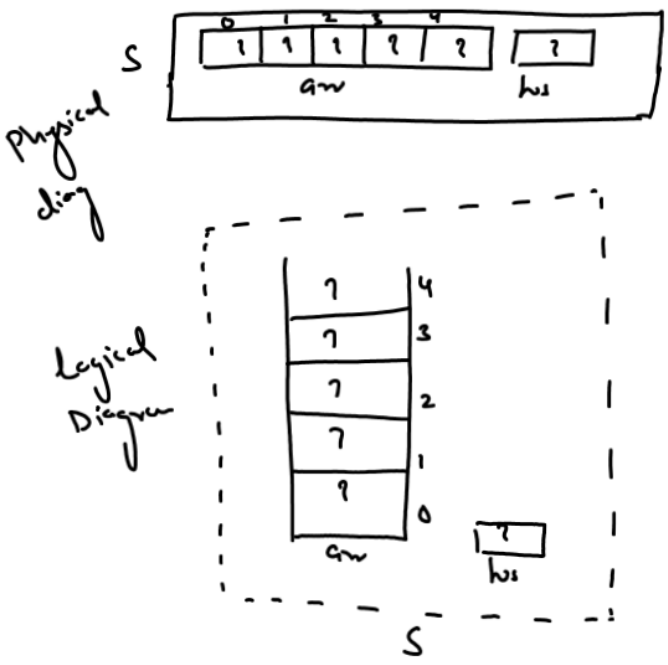
Syntax Of Decl A Structure:
==========
struct <struct_name>
{
    <data type> <var_name>;
    ......
    .......
};

Example:
=======

struct Stack
{
  int arr[5];
  int tos;
};

```
#inlude<stdio.h>
struct Stack
{
   int arr[5];
   int tos;
};

int main()
{
   struct Stack S;
   ........
   .......
   return 0;
}
```



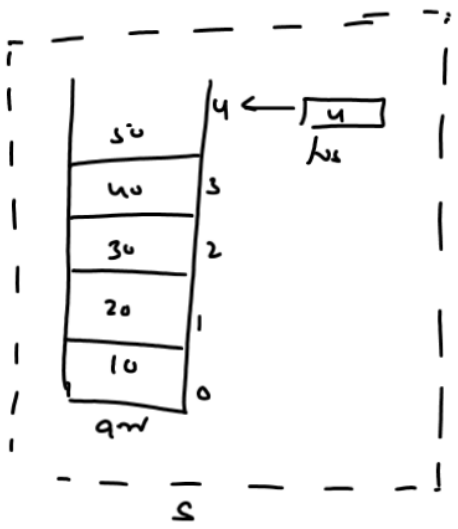
## Code Of Stack-Basic Version

```
#inlude<stdio.h>
struct Stack
{
   int arr[5];
   int tos;
};
int main()
{
   struct Stack S;
   int x;
   tos=-1; // ERROR
   S.tos=-1; // CORRECT
   S.tos=S.tos+1;
   S.arr[S.tos]=10;
  printf("Pushed 10");
   S.tos=S.tos+1;
   S.arr[S.tos]=20;
   printf("\nPushed 20");
   ...
   x=S.arr[S.tos];
   S.tos=S.tos-1;
   printf("\nPopped %d",x);
   ....
   return 0;
```

→ push operation

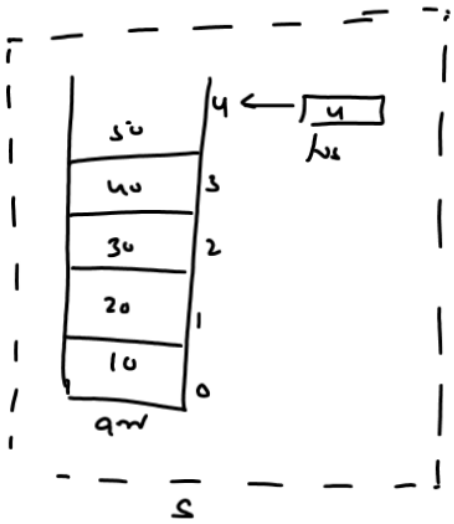
→ push operation

→ pop operation




## Code Of Stack-Second Version

```
#inlude<stdio.h>
struct Stack
{
   int arr[5];
   int tos;
};
int main()
{
   struct Stack S;
   int i,x;
   S.tos=-1;
   for(i=1;i<=5;i++)
   {
      printf("enter no to push:");
      scanf("%d",&x);
      S.tos++;
      S.arr[S.tos]=x;
      printf("\nPushed %d",x);
   }
```



```
  for(i=1;i<=5;i++)
  {
      x=S.arr[S.tos];
      printf("\nPopped %d",x)
     S.tos--;
  }
return 0;
}
```

Algorithm for The Function push()

==========

1.Check for overflow.

2. If Stack is full then display the message STACK OVERFLOW and return, otherwise goto STEP 3.

3. Increment TOS by 1.

4.PUSH the element in STACK at the index pointed by TOS

5. Finish and return

Algorithm for The Function pop()
==========
1.Check for underflow.

2. If Stack is empty then display the message STACK UNDERFLOW and return, otherwise goto STEP 3.

3. POP the element from the STACK from the index pointed by TOS.

4.Decrement TOS by 1.

5. Return the popped element.

6. Finish