Pointer

- A pointer is a variable that stores a memory address. Pointers are used to store the addresses of other variables or memory items. Pointers are very useful for another type of parameter passing, usually referred to as Pass by Address. Pointers are essential for dynamic memory allocation.
- A Pointer in C language is a variable which holds the address of another variable of same data type.
- Pointers are used to access memory and manipulate the address

Address in C

- Whenever a variable is defined in C language, a memory location is assigned for it, in which it's value will be stored. We can easily check this memory address, using the & symbol.
- If var is the name of the variable, then &var will give it's address.
- Let's write a small program to see memory address of any variable that we define in our program.

```
#include<stdio.h>

void main()
{
    int var = 7;
    printf("Value of the variable var is: %d\n", var);
    printf("Memory address of the variable var is: %x\n", &var);
}
```

OUTPUT:

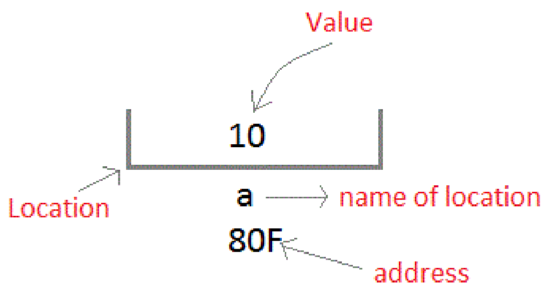
```
Value of the variable var is: 7

Memory address of the variable var is: bcc7a00
```

Whenever a variable is declared in a program, system allocates a location i.e an address to that variable in the memory, to hold the assigned value. This location has its own address number, which we just saw above.

Let us assume that system has allocated memory location 80F for a variable a.

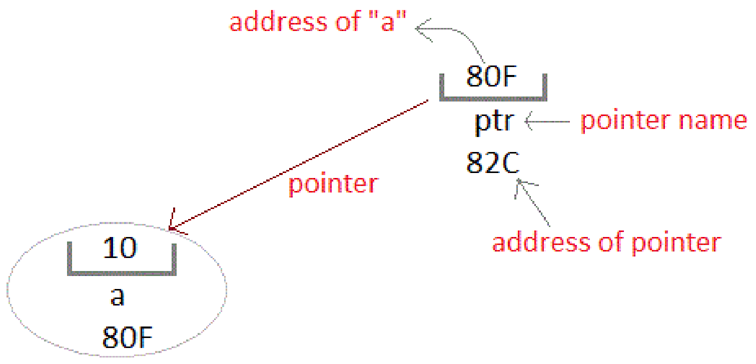
int a = 10;



We can access the value 10 either by using the variable name a or by using its address 80F.

The question is how we can access a variable using it's address? Since the memory addresses are also just numbers, they can also be assigned to some other variable. The variables which are used to hold memory addresses are called Pointer variables.

A pointer variable is therefore nothing but a variable which holds an address of some other variable. And the value of a pointer variable gets stored in another memory location.



Benefits of using pointers

1) Pointers are more efficient in handling Arrays and Structures.
2) Pointers allow references to function and thereby helps in passing of function as arguments to other functions.
3) It reduces length of the program and its execution time as well.
4) It allows C language to support Dynamic Memory management.

Tower of Hanoi

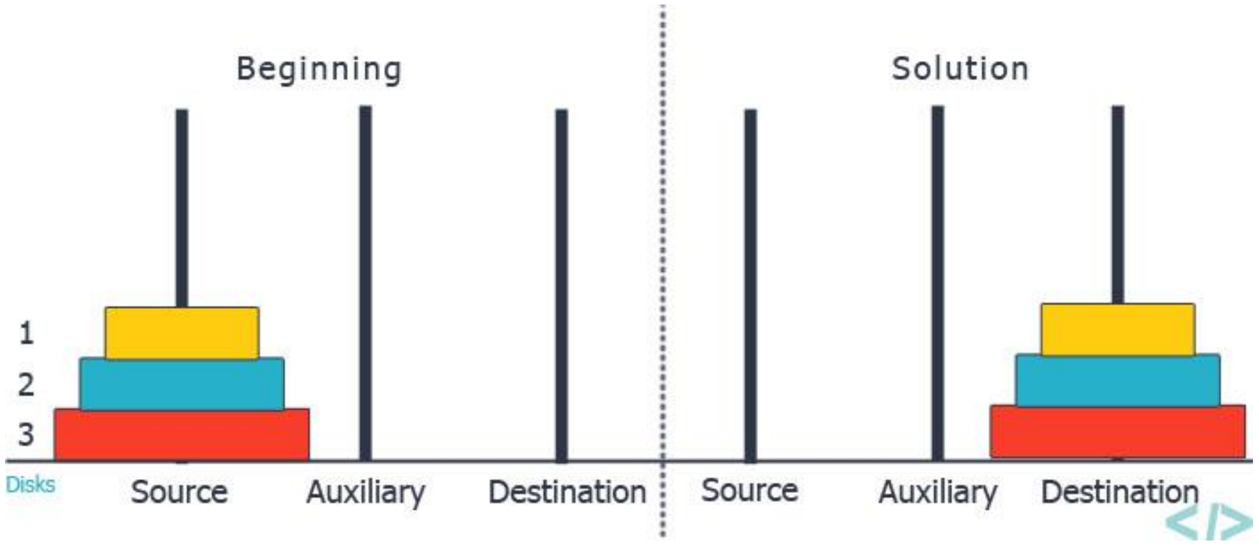
Tower of Hanoi is a mathematical puzzle where we have three rods and n disks. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.

Rules

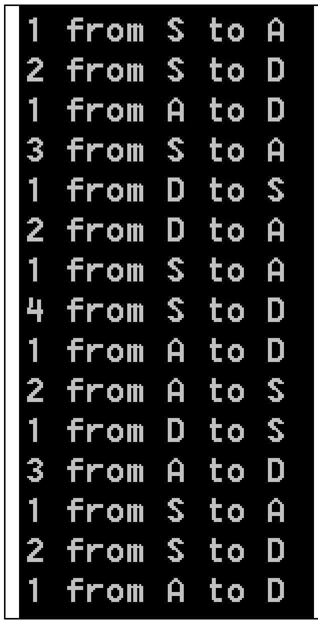
The mission is to move all the disks to some another tower without violating the sequence of arrangement. A few rules to be followed for Tower of Hanoi are −

1. Only one disk can be moved among the towers at any given time.
2. Only the "top" disk can be removed.
3. No large disk can sit over a small disk.



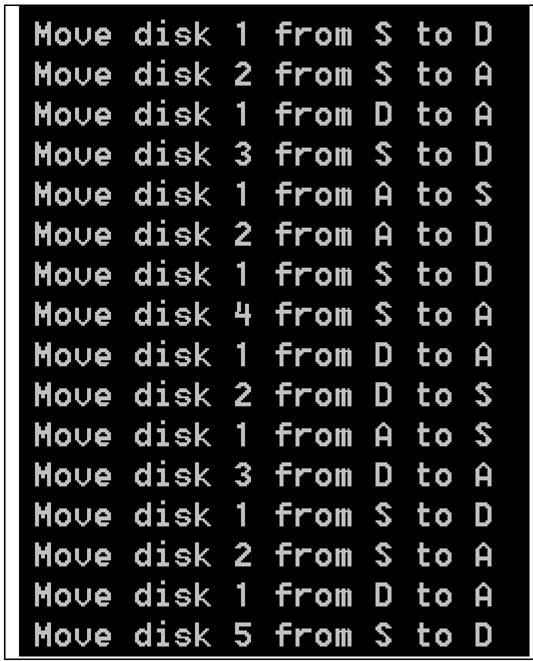
Step 1: S to D (1) Step 2: S to A (2) Step 3: D to A (1) Step 4: S to D (3)	Step 5 : A to S (1) Step 6: A to D (2) Step 7 : S to D (1)

Tower of Hanoi for 4 Disk:

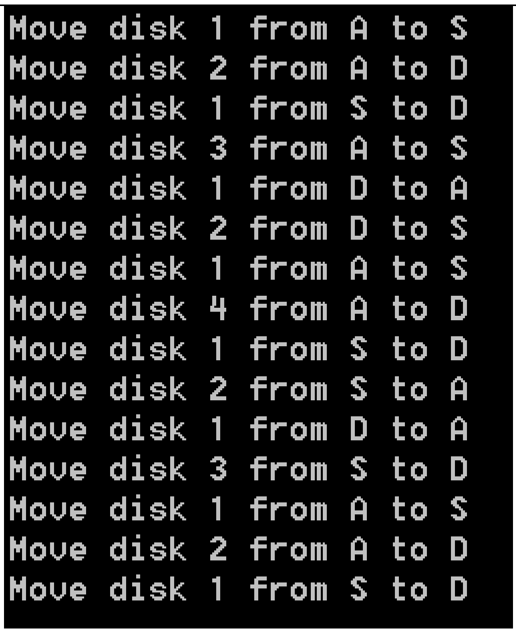


```
1 from S to A
2 from S to D
1 from A to D
3 from S to A
1 from D to S
2 from D to A
1 from S to A
4 from S to D
1 from A to D
2 from A to S
1 from D to S
3 from A to D
1 from S to A
2 from S to D
1 from A to D
```

Tower of Hanoi for 5 Disk:



```
Move disk 1 from S to D
Move disk 2 from S to A
Move disk 1 from D to A
Move disk 3 from S to D
Move disk 1 from A to S
Move disk 2 from A to D
Move disk 1 from S to D
Move disk 4 from S to A
Move disk 1 from D to A
Move disk 2 from D to S
Move disk 1 from A to S
Move disk 3 from D to A
Move disk 1 from S to D
Move disk 2 from S to A
Move disk 1 from D to A
Move disk 5 from S to D
```



```
Move disk 1 from A to S
Move disk 2 from A to D
Move disk 1 from S to D
Move disk 3 from A to S
Move disk 1 from D to A
Move disk 2 from D to S
Move disk 1 from A to S
Move disk 4 from A to D
Move disk 1 from S to D
Move disk 2 from S to A
Move disk 1 from D to A
Move disk 3 from S to D
Move disk 1 from A to S
Move disk 2 from A to D
Move disk 1 from S to D
```

Algorithm

To write an algorithm for Tower of Hanoi, first we need to learn how to solve this problem with lesser amount of disks, say → 1 or 2. We mark three towers with
name, source, destination and aux (only to help moving the disks). If we have only one disk, then it can easily be moved from source to destination peg.

If we have 2 disks −

- First, we move the smaller (top) disk to aux peg.
- Then, we move the larger (bottom) disk to destination peg.
- And finally, we move the smaller disk from aux to destination peg.

Our ultimate aim is to move disk n from source to destination and then put all other (n1) disks onto it. We can imagine to apply the same in a recursive way for all given set of disks.

The steps to follow are −

- Step 1 − Move n-1 disks from source to aux (by using destination)
- Step 2 − Move nth disk from source to dest
- Step 3 − Move n-1 disks from aux to dest (by using source)

A recursive algorithm for Tower of Hanoi can be driven as follows −

```
Procedure Hanoi(disk, source, aux, dest)

  IF disk = 1, THEN
    move disk from source to dest

  ELSE
    Hanoi(n-1, source, dest, aux)     // Step 1
    Move nth disk from source to dest         // Step 2
    Hanoi(n-1, aux, source, dest)     // Step 3
  END IF

END Procedure
```