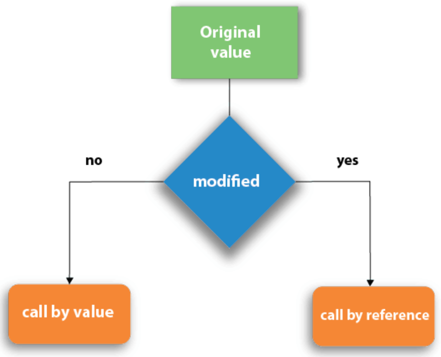
Call by value and Call by reference in C

There are two methods to pass the data into the function in C language, i.e., *call by value* and *call by reference*.



Call by value in C

- In call by value method, the value of the actual parameters is copied into the formal parameters. In other words, we can say that the value of the variable is used in the function call in the call by value method.
- In call by value method, we can not modify the value of the actual parameter by the formal parameter.
- In call by value, different memory is allocated for actual and formal parameters since the value of the actual parameter is copied into the formal parameter.
- The actual parameter is the argument which is used in the function call whereas formal parameter is the argument which is used in the function definition.

```
#include<stdio.h>
void change(int num) {
     printf("Before adding value inside function num=%d \n",num);    // 100
   num=num+100;   // 200
   printf("After adding value inside function num=%d \n", num);     // 200
}
int main() {
   int x=100;
   printf("Before function call x=%d \n", x);    //100
   change(x);
   printf("After function call x=%d \n", x);   // 100
return 0;
}
```

Call by reference in C

- In call by reference, the address of the variable is passed into the function call as the actual parameter.
- The value of the actual parameters can be modified by changing the formal parameters since the address of the actual parameters is passed.
- In call by reference, the memory allocation is similar for both formal parameters and actual parameters. All the operations in the function are performed on the value stored at the address of the actual parameters, and the modified value gets stored at the same address.

```
#include<stdio.h>
  void change(int *num) {
       printf("Before adding value inside function num=%d \n",num);    // 100
     *num=*num+100;   // 200
     printf("After adding value inside function num=%d \n", *num);     // 200
  }
  int main() {
     int x=100;
     printf("Before function call x=%d \n", x);    //100
     change(&x);
     printf("After function call x=%d \n", x);   // 200
  return 0;
  }
```

Recursive function:

Recursive function is the process of defining something in terms of itself. If a function is called by itself is known as a recursive function and this process is known as recursion.

There are two types of recursion functions in c.

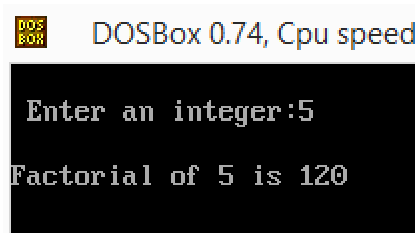
- Direct recursion (Call by itself)
- Indirect recursion (A function is passed as an argument to another function).

Direct Recursion

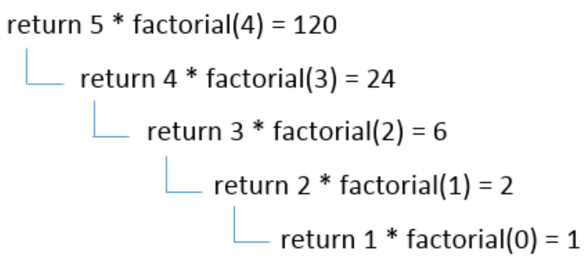
If a function is call by itself then it is known as a direct recursive function.
The following program illustrates the direct recursive function.

```
#include<stdio.h>
int factorial(int);
void main()
{
int n,f;
clrscr();
printf("\n Enter an integer:");
scanf("%d",&n);
f=factorial(n);
printf("\nFactorial of %d is %d",n,f);
getch();
}

int factorial(int n)
{
int f;
if(n==0)
return 1;
else
return n*factorial(n-1);
}
```

Output



```
Enter an integer:5

Factorial of 5 is 120
```



```
return 5 * factorial(4) = 120
    return 4 * factorial(3) = 24
        return 3 * factorial(2) = 6
            return 2 * factorial(1) = 2
                return 1 * factorial(0) = 1
```

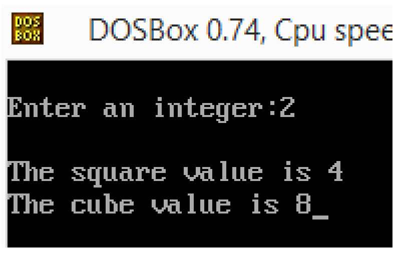
Indirect recursion
If a function is passed as an argument to another function, then it is called as indirect recursion and this process is known as indirect recursion.

We can pass not only the values, arrays and address as an argument to a function but we can also pass a function as an argument to another function.

The following program illustrate the use of indirect recursive function

```
#include<stdio.h>
#include<conio.h>
int square(int);
int cube(int,int);
void main()
{
int n,sv,cv;
clrscr();
printf("\nEnter an integer:");
scanf("%d",&n);
sv=square(n);
cv=cube(square(n),n);
printf("\nThe square value is %d",sv);
printf("\nThe cube value is %d",cv);
getch();
}
int square(int x)
{
return x*x;
}
int cube(int x,int y)
{
return x*y;
}
```

Output



```
Enter an integer:2

The square value is 4
The cube value is 8_
```