ACTUAL AND FORMAL ARGUMENTS:

The arguments listed in the function calling statement are referred to as *actual arguments.* They are the actual values passed to a function to compute a value or to perform a task.

Rules to call a Function:

The following rules are used to call a function in a C program.

(i) A function has a statement block which is called by the **main()** or any other function.

(ii) When the data type in a function declaration is omitted, the function will return a value of the type integer.

(iii) The data type of the formal argument may be declared in the next line which follows the function declaration statement.

(iv) A function can return a value at any stage of its execution by using more than one **return** statements.

(v) A function can also be called by another function.

(vi) The **return** statement is omitted if it does not return a value directly to the calling program.

FUNCTION PROTOTYPE:

When a C program is compiled, the compiler does not check for data type mismatch of actual arguments in the function call and the formal arguments in the function declaration. To enable the compiler to check the same, a function prototype declaration is used in the main program.

[Function prototype is always declared at the beginning of the **main()** program.]

*Example:- Write a function to find the factorial of a given integer and use it to find nCr.

nCr = n! / r!(n-r)! [2014]

Ans:-

```
/*Program to find nCr */
#include <stdio.h>
main()
{
 int fact (int k);
/*function prototype declaration */
int n,r, nCr;
```

```
    printf("\n Enter value to n and r:");
    scanf(" %d %d", &n, &r);
    nCr = fact(n)/ (fact(r)*fact(n-r));
   printf ("\n Value of nCr = &d", nCr);
   getch();
}
/*function subprogram to find factorial*/
int fact(int k)
 {
   int i, p = 1;
   for(i=1; i <=k; i++)
       p = p*i;
    return(p);
}
```

.