FUNCTION CALL

Function call is used to invoke the function. So the program control is transferred to that function. A function can be called by using its name & actual parameters.

Function call should be terminated by a semicolon (;).

Syntax:

```
Functionname(argumentlist);
```

Example

```
int abc(int, int, int)  // Function declaration
void main()
{
        int x,y,z;
        abc(x,y,z) // Function Call
        …
        …
}
int abc(int i, int j, int k)  // Function definition
{
        …….
        ….
        return (value);
}
```

Calling function – The function that calls a function is known as a calling function.

Called function – The function that has been called is known as a called function.

Actual arguments – The arguments of the calling function are called as actual arguments.

Formal arguments – The arguments of called function are called as formal arguments.

Steps for function Execution:

1. After the execution of the function call statement, the program control is transferred to the called function.
2. The execution of the calling function is suspended and the called function starts execution.
3. After the execution of the called function, the program control returns to the calling

function and the calling function resumes its execution.

Program:

```
#include<stdio.h>
#include<conio.h>
float circlearea(int);              //function prototype
void main()
{
        int r;
        float area;
        printf(“Enter the radius \n”);
        scanf(“%d”,&r);
        area=circlearea(r); //function call
        printf(“Area of a circle =%f\n”, area);
        getch();
}
 float circlearea(int r1)
{
        return 3.14 * r1 * r1;          //function definition
}
```

Output:

```
Enter the radius
2
Area of circle = 12.000
```

BUILT IN FUNCTIONS (STRING FUNCTIONS, MATH FUNCTIONS):

Library functions are predefined functions. These functions are already developed by someone and are available to the user for use.

MATHEMATICAL FUNCTIONS

These are defined in math.h header file. Math functions perform the mathematical calculations. Some of the built-in math functions are,

S.No.	Function & Description

1	**double acos(double x)** Returns the arc cosine of x in radians.
2	**double asin(double x)** Returns the arc sine of x in radians.
3	**double atan(double x)** Returns the arc tangent of x in radians.
4	**double atan2(double y, double x)** Returns the arc tangent in radians of y/x.
5	**double cos(double x)** Returns the cosine of a radian angle x.
6	**double cosh(double x)** Returns the hyperbolic cosine of x.
7	**double sin(double x)** Returns the sine of a radian angle x.
8	**double sinh(double x)** Returns the hyperbolic sine of x.

9	**double tanh(double x)** Returns the hyperbolic tangent of x.
10	**double exp(double x)** Returns the value of **e** raised to the xth power.
11	**double frexp(double x, int *exponent)** The returned value is the mantissa and the integer pointed to by exponent is the exponent. The resultant value is x = mantissa * 2 ^ exponent.
12	**double ldexp(double x, int exponent)** Returns **x** multiplied by 2 raised to the power of exponent.
13	**double log(double x)** Returns the natural logarithm (base-e logarithm) of **x**.
14	**double log10(double x)** Returns the common logarithm (base-10 logarithm) of **x**.
15	**double modf(double x, double *integer)** The returned value is the fraction component (part after the decimal), and sets integer to the integer component.
16	**double pow(double x, double y)** Returns x raised to the power of **y**.
17	**double sqrt(double x)** Returns the square root of **x**.
18	**double ceil(double x)** Returns the smallest integer value greater than or equal to **x**.
19	**double fabs(double x)** Returns the absolute value of **x**.
20	**double floor(double x)** Returns the largest integer value less than or equal to **x**.
21	**double fmod(double x, double y)** Returns the remainder of x divided by **y**.

Mathematical Functions

Program:

```
#include<stdio.h>
#include <math.h>
#include<conio.h>
void main()
{
    printf("\n%f",ceil(3.6));
    printf("\n%f",ceil(6.3));
    printf("\n%f",floor(3.6));
    printf("\n%f",floor(7.2));
    printf("\n%f",sqrt(16));
    printf("\n%f",sqrt(7));
    printf("\n%f",pow(2,4));
    printf("\n%f",pow(3,3));
    printf("\n%d",abs(-12));
    return 0;
}
```

Output:

```
4.000000
4.000000
3.000000
3.000000
4.000000
2.645751
16.000000
27.000000
12
```