# ***Using Functions in C++***

## Function

Function is the basic building block of C++ language. Functions are those parts of program, which perform certain specific task. Different function can be used in order to fulfill different task.

There are four types of functions depending on the return type and arguments:

- Functions that take nothing as argument and return nothing.
  **void add(void)**
- Functions that take arguments but return nothing.
  **void add(int ,int)**
- Functions that do not take arguments but return something.
  **int add(void)**
- Functions that take arguments and return something.
  **int void(int,int)**

## Use of Functions

There are many programs in which an operation is performed several times in the same way. This increases the amount of programming code, thus increasing the size of program. In order to remove this repetition, a function is defined which perform the operation whenever the main program need the operation. The code of the function is written once and utilized number of time, thus the size of the program is reduced.

**Structure of the function**
There are three basic elements of a function. These elements are:

- Function Declaration
- Function Definition
- Function Call

### a) Function Declaration

A function is declared by name before it is used in the main program. The function declaration tells the compiler the name of the function, the data type of the function returns and the number and data type of the function arguments.

### b) Function Definition:

Function definition is the function itself. It is the programming code of the operation the function is performing. Function definition includes the function name, number and data type of the values the function is returning. Function definition tells the compiler that a function is being defined which can perform certain specific task.

### c) Function Call:

Function call defines that a function is called to perform the task it is designed to do. Function call makes the compiler to transfer control to the programming code in the function.

## Example-1

```
#include<iostream>
using namespace std;

void sum(void);
void main(void)
{
  printf("\nProgram to print sum of two numbers\n");
  sum();
}

 void sum(void)
       {
         int num1,num2,sum;
         printf("Enter 1stnumber:");
         scanf("%d",&num1);
         printf("Enter 2ndnumber:");
         scanf("%d",&num2);
         sum=num1+num2;
         printf("Sum of %d+%d=%d",num1,num2,sum);
        }
```

## Function That Return Value

### General form

Function may return any type of data. The general form of such C++ language function is as:

```
Return_type function_name (parameters list)

{

     Statements;

}
```

Return_Type describe the return type of the function.

## Returning data type

A function can return any type of data except an array. If no data type specifier is present, then the C++ language compiler automatically assumes that the function is returning an integer value. In other words the default data type a function returns is an integer.

Before you can use a function that returns a non-integer value, you need to inform the compiler about its return type. If you do not, the compiler will generate code with the assumption that an integer is being returned.

## The return ( ) statement

If a function is desired to return value in a program then 'return ( )' statement is used to fulfill this task. The 'return ( )' statement has two purposes. First, executing it immediately transfers control from the function back to the calling program. Secondly, whatever is inside the parentheses following the „return" is returned as a value to the calling program.

The „return" statement does not need to be at the end of the function. It can occur anywhere in the function, as soon as it is encountered, control will be transferred to the calling program.

**Example-2**

```
#include <iostream>
using namespace std;

int addition (int, int)         // Function prototype

int main ()
{
  int z;
  z = addition (5,3); // Function calling printf("
  The result is : %d " , z);
}

int addition (int a, int b)        // Function Definition
{
  int r;
  r=a+b;
  return r;
}
```

**Output**
The result is: 8

## Built-in Functions

There are various header files which contain built-in functions. The programmer can include those header files in any program and then use the built-in function by just calling them.

**include<math.h>**

Math library functions allow the programmer to perform a number of common mathematical calculations:

| Function | Description |
|---|---|
| sqrt(x) | square root |
| sin(x) | trigonometric sine of x (in radians) |
| cos(x) | trigonometric cosine of x (in radians) |
| tan(x) | trigonometric tangent of x (in radians) |
| exp(x) | exponential function |
| log(x) | natural logarithm of x (base e) |
| log10(x) | logarithm of x to base 10 |
| fabs(x) | absolute value (unsigned) |
| ceil(x) | rounds x up to nearest integer |
| floor(x) | rounds x down to nearest integer |
| pow(x,y) | x raised to power y |

**include<stdio.h>**

**Formatted input/output:**

| | |
|---|---|
| Fprintf | Write formatted data to stream (function ) |
| Fscanf | Read formatted data from stream (function ) |
| Printf | Print formatted data to stdout (function ) |
| Scanf | Read formatted data from stdin (function ) |
| snprintf | Write formatted output to sized buffer (function ) |
| sprint | Write formatted data to string (function ) |
| sscanf | Read formatted data from string (function ) |
| vfprintf | Write formatted data from variable argument list to stream (function ) |
| vfscanf | Read formatted data from stream into variable argument list (function ) |
| vprintf | Print formatted data from variable argument list to stdout (function ) |
| vscanf | Read formatted data into variable argument list (function ) |
| vsnprintf | Write formatted data from variable argument list to sized buffer (function ) |
| vsprintf | Write formatted data from variable argument list to string (function ) |
| vsscanf | Read formatted data from string into variable argument list (function ) |