# **C++ Functions**

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times.

## **Create a Function:**

C++ provides some pre-defined functions, such as main(), which is used to execute code. But you can also create your own functions to perform certain actions.

To create (often referred to as declare) a function, specify the name of the function, followed by parentheses **()**:

A function is a set of statements that take inputs, do some specific computation and produces output.

The idea is to put some commonly or repeatedly done task together and make a function so that instead of writing the same code again and again for different inputs, we can call the function.

**Syntax:**

Return\_type Fun\_name(arg1, arg2,…..)

**Why do we need functions?**

* Functions help us in reducing code redundancy. If functionality is performed at multiple places in software, then rather than writing the same code, again and again, we create a function and call it everywhere. This also helps in maintenance as we have to change at one place if we make future changes to the functionality.
* Functions make code modular. Consider a big file having many lines of codes. It becomes really simple to read and use the code if the code is divided into functions.
* Functions provide abstraction. For example, we can use library functions without worrying about their internal working.

## **Function Declaration and Definition:**

A C++ function consist of two parts:

* **Declaration:** the function's name, return type, and parameters (if any)
* **Definition:** the body of the function (code to be executed)

**1.Function Declaration:**  
A function declaration tells the compiler about the number of parameters function takes, data-types of parameters and return type of function. Putting parameter names in function declaration is optional in the function declaration, but it is necessary to put them in the definition. Below are an example of function declarations. (parameter names are not there in below declarations)

-

-

-

-

-

-

-

-

#### //Code with HARRY

#### Functions in C++:

Functions are the main part of top-down structured programming. We break the code into small pieces and make functions of that code. Functions help us to reuse the code easily. An example program for the function is shown in Code Snippet 1.

int sum(int a, int b){

int c = a+b;

return c;

}

int main(){

int num1, num2;

cout<<"Enter first number"<<endl;

cin>>num1;

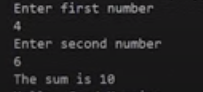
cout<<"Enter second number"<<endl;

cin>>num2;

cout<<"The sum is "<<sum(num1, num2);

return 0;

}



**Figure 1: Function Output**

#### Function Prototype in C++:

The function prototype is the template of the function which tells the details of the function e.g(name, parameters) to the compiler. Function prototypes help us to define a function after the function call. An example of a function prototype is shown in Code Snippet 3.

// Function prototype

int sum(int a, int b);

**Code Snippet 3: Function Prototype**

As shown in Code Snippet 3, we have made a function prototype of the function “sum”, this function prototype will tell the compiler that the function “sum” is declared somewhere in the program which takes two integer parameters and returns an integer value. Some examples of acceptable and not acceptable prototypes are shown below:

* **int sum(int a, int b); //Acceptable**
* **int sum(int a, b); // Not Acceptable**
* **int sum(int, int); //Acceptable**

##### **1.Formal Parameters:**

The variables which are declared in the function are called a formal parameter. For example, as shown in Code Snippet 1, the variables “a” and “b” are the formal parameters.

##### **2.Actual Parameters:**

The values which are passed to the function are called actual parameters. For example, as shown in Code Snippet 2, the variables “num1” and “num2” are the actual parameters.

The function doesn't need to have parameters or it should return some value. An example of the void function is shown in Code Snippet 4.

void g(){

cout<<"\nHello, Good Morning";

}

**Code Snippet 4: Void Function**

As shown in Code Snippet 4, void as a return type means that this function will not return anything, and this function has no parameters. Whenever we will call this function it will print “Hello, Good Morning”