C++ Functions

In this tutorial, we will learn about the C++ function and function expressions with the help of examples.

A function is a block of code that performs a specific task.

Suppose we need to create a program to create a circle and color it. We can create two functions to solve this problem:

- a function to draw the circle
- a function to color the circle

Dividing a complex problem into smaller chunks makes our program easy to understand and reusable.

There are two types of function:

- Standard Library Functions: Predefined in C++
- User-defined Function: Created by users

In this tutorial, we will focus mostly on user-defined functions.

C++ User-defined Function

C++ allows the programmer to define their own function. A user-defined function groups code to perform a specific task and that group of code is given a name (identifier).

When the function is invoked from any part of the program, it all executes the codes defined in the body of the function.

C++ Function Declaration

The syntax to declare a function is:

```
returnType functionName (parameter1, parameter2,...) {
    // function body
}
```

Here's an example of a function declaration.

```
// function declaration
void greet() {
    cout << "Hello World";
}
```

Here,

- the name of the function is greet()
- the return type of the function is void
- the empty parentheses mean it doesn't have any parameters
- the function body is written inside {}

Calling a Function

In the above program, we have declared a function named greet(). To use the greet() function, we need to call it.

Here's how we can call the above greet() function.

```
int main() {

    // calling a function
    greet();

}
```

Function Parameters

As mentioned above, a function can be declared with parameters (arguments). A parameter is a value that is passed when declaring a function.

For example, let us consider the function below:

```
void printNum(int num) {
    cout << num;
}
```

Here, the int variable num is the function parameter.

We pass a value to the function parameter while calling the function.

```
int main() {
    int n = 7;

    // calling the function
    // n is passed to the function as argument
    printNum(n);

    return 0;
}
```

Return Statement

In the above programs, we have used void in the function declaration.

This means the function is not returning any value.

It's also possible to return a value from a function. For this, we need to specify the returnType of the function during function declaration.

Then, the return statement can be used to return a value from a function.

For example,

```
int add (int a, int b) {
   return (a + b);
}
```

Here, we have the data type int instead of void. This means that the function returns an int value.

The code return (a + b); returns the sum of the two parameters as the function value.

The return statement denotes that the function has ended. Any code after return inside the function is not executed.