Functions in C

In this tutorial, you will learn everything about functions in the C language with the help of simple and easy examples. This tutorial will guide you on how to define a function and various ways to call a function.

The function is an elementary and essential part of C programming. In the case of large programs, writing and management of the codes become complicated. Moreover, we have to perform similar jobs again and again for which a common set of instructions are used. To solve this problem, the function has been introduced.

Benefits of using a function in C

Functions play a vital role in all programming languages because of the following fair reasons

A. Function enhances the readability of code.
B. Reusability of the code is the vital achievement of functions in C.
C. Reduces redundancy of code by avoiding rewriting the same code at multiple locations.
D. Functions enable easy debugging as errors are easy to trace due to modularity.
E. The size and complexity of code are lessened since function calls replace the duplicate code block.

Types of function

Functions are basically of two types:

1. Standard library
2. User-defined

The **Standard Library function** means a pre-written set of codes by the compiler to perform specific tasks. These are the built-in functions that are defined already in the header files. For example `scanf` to store the input data and `printf` to display something are the standard library functions defined in the header file `stdio.h`. Normally all input and output operations including mathematical ones are implemented by the library functions. Whereas, **user-defined functions** are written by the programmers and used whenever needed in the course of the program.

Important aspects of a function

A function in a C program becomes complete only when these three important aspects are specified properly. They are:

- Function Declaration
- Function Definition
- Function Call

How to declare functions in c

Function Declaration is always a better practice in C language, as it gives a clue to the compiler a function will be used in the program later. Function declaration gives information like the name of the function, the number of parameters used in the function, and the type of function to the compiler. An important point to be considered while declaring a function is that it must be declared globally, which means outside the main() function.

A function declaration can be perceived as an announcement to the compiler that in the future a function will be used in the program. Hence it doesn't have function body.

The syntax of the function declaration is :

```
return_type function_name ( data_type parameter(s));
```

How to define functions in c:

The basic syntax of the function definition in c language is given below.

```
return_type function_name ( data_type arg(s))
{
   //function body with C statements
}
```

The components of the above function definition are tabulated below.

Components	Description
return_type	defines the data type(int, float, char) of value to be returned. Optional
function_name	Should be a valid identifier. (refer rules of identifiers)
Parameter(s)	Used to pass input values to the functions. Optional
function _body>	Valid set of C statements enclosed in braces {}

Function Definition Example:

You have to write the codes inside the body of any function to define it. Suppose we want to form a simple function 'show', whose job is to print 'hello'. In this case, you are free to define the function 'show' anywhere inside the program like

```
void show()
{
    printf("hello");
}
```

For functions working with variables, the definition is like as below

```
float area(int r)
{
  pi=3.14;
  a = pi*r*r;
  return(a);
}
```

In this example, we have defined a function to calculate the area of a circle. This function remains as a script and does not execute until a function call is made. Hence is called a function definition.

How to call a function in C

Like function definition, the function call is another important aspect of function. After defining the function properly, we need to call the function to do the job, it is designed for. To do this we have to just write the name of the function and () followed by a semicolon. The syntax is :

```
function_name(arg(s));
```

Various ways of Function calling

In c we can make a function call in four different ways which are listed below and its simple example follows.

1. function with no arguments and no return value
2. function with arguments and no return value
3. function with no arguments but with return value
4. function with both arguments and return value

Observe the below example to calculate the area of a circle in these four different aspects :

Example 1: Function with no argument and no return value.

```
#include<stdio.h>

void area() {
    float a, pi;
    int r = 10;
    pi = 3.14;
    a = pi * r * r;
    printf("Area of Circle is : %f\n", a);
}

int main() {
    area();
}
```

Output:

```
Area of Circle is : 314.000000
```

When you observe the above program you can find the following:

- return type provided is void, which is a null data type
- no return statement mentioned in the function definition
- arguments are not specified in the function call

Example 2 : Function without argument and with return value

```
#include<stdio.h>

float area() {
    float a, pi;
    int r = 10;
    pi = 3.14;
    a = pi * r * r;
    return (a);
}

int main() {
    printf("Area of Circle is : %f\n", area());
}
```

Output:

```
Area of Circle is : 314.000000
```

Here in this program, you can find the following:

- The return type provided is float
- A return statement is mentioned in the function definition to return value to the function call.
- Arguments are not specified in the function call

Example 3: Function with argument and without a return value.

```
#include<stdio.h>

area(int r) {
    float a, pi;
    pi = 3.14;
    a = pi * r * r;
    printf("Area of Circle is : %f\n", a);
}

int main() {
    area(10);
}
```

Output:

```
Area of Circle is : 314.000000
```

Here in this program, you can find the following difference from the previous one:

- The return type is not specified
- The return statement is not mentioned in the function definition
- Argument 10 is specified in the function call

Example 4: Function with both argument and return value.

```
#include<stdio.h>

float area() {
    float a, pi;
    int r = 10;
    pi = 3.14;
    a = pi * r * r;
    return (a);
}

int main() {
    float arc;
    arc = area();
    printf("Area of Circle is : %f\n", arc);
}
```

Output:

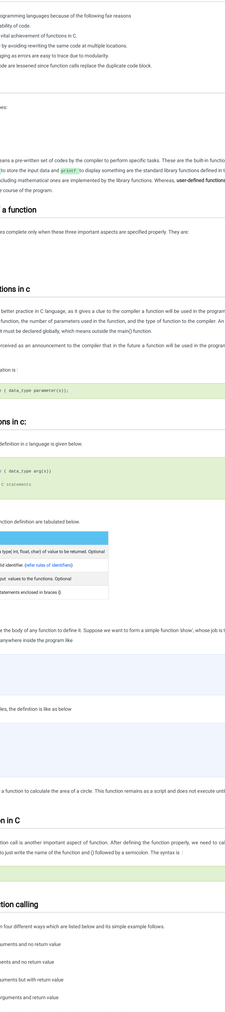
```
Area of Circle is : 314.000000
```

In the above code snippet, you can find the following

- The return type provided is float
- A return statement is mentioned in the function definition to return value to the function call.
- The argument is specified in the function call.

How function works in C

Let's learn the overall working of the function and execution sequence.



How function works in c

In C, the execution of a program starts with the main() function where the function call is residing. When the compiler comes across a function call, the control will shift to the corresponding function definition where it performs some task designed for it. The final result will be returned to the function call from where it encounters a return statement. If not it outputs the result in the function itself and immediately terminates the function to switch the control back to the main() function.

Return statement in C

The return statement in the C language returns a value to the calling function by terminating the execution of its corresponding function. To return a value we must declare a function with any of the valid data types unless can keep it as void. In our previous example, the value of **a** is returned to the main function and is assigned to the variable **arc** which is visualized below.