Function in C

In C, a function is a block of code that performs a specific task. It helps in breaking down a large program into smaller, modular, and reusable pieces, making code more organized and easier to maintain. A function in C consists of a function declaration (or prototype), function definition, and function call.

Structure of a Function in C:

Function Declaration (Prototype):

This is an optional step but a good practice. It tells the compiler about the function's name, return type, and parameters before the function is called.
Syntax: return_type function_name(parameter_list);

Function Definition:

This is where the actual code of the function resides.
Syntax:

```
return_type function_name(parameter_list) {
    // Function body
    return value;  // If the function returns a value
}
```

Function Call:

To execute the function, it is called from the main() function or other functions.
Syntax:

```
function_name(argument_list);
```

Example of Function

```
#include <stdio.h>
// Function declaration (optional but recommended)
int add(int a, int b);
int main() {
    int x = 5, y = 10;
    // Function call
    int sum = add(x, y);
    printf("The sum is: %d\n", sum);
    return 0;
}

// Function definition
int add(int a, int b) {
    return a + b;  // Returns the sum of a and b
}
```

Conditions of Return Types and Arguments

In C programming language, functions can be called either with or without arguments and might return values. They may or might not return values to the calling functions.

- Function with no arguments and no return value
- Function with no arguments and with return value
- Function with argument and with no return value
- Function with arguments and with return value

How Does a C Function Work?

The working of a C function can be divided into the following steps:

- **Declaring a function:** This step involves specifying the function's name, return type, and parameters. It tells the compiler what to expect from the function.
- **Defining a function:** In this step, the actual logic and body of the function are provided.
- **Calling the function:** Here, the function is invoked by passing the required arguments, allowing the program to utilize the function's functionality.
- **Executing the function:** This step involves executing the statements within the function to achieve the desired result.
- **Returning a value:** After the function has executed, the calculated result is returned, if applicable. The function then exits, and the memory allocated to variables and the function itself is released, returning control to the main program.

Types of Functions

There are two types of functions in C:

- Library Functions
- User Defined Functions

Library Function

A library function, commonly known as a "built-in function," is part of a compiler package that includes a set of predefined functions, each with a specific purpose. The benefit of using built-in functions is that they can be utilized directly without needing to be defined first, unlike user-defined functions, which must be declared and defined prior to their usage.
Example: pow(), sqrt(), strcmp(), strcpy() etc.

User Defined Function

Functions that the programmer creates are known as User-Defined functions or "tailor-made functions". User-defined functions can be improved and modified according to the need of the programmer. Whenever we write a function that is case-specific and is not defined in any header file, we need to declare and define our own functions according to the syntax.

Passing Parameters to Functions

Passing Parameters to Functions
Passing parameters to functions involves providing input values to a function when it is called. This allows the function to operate on different data each time it is invoked. There are several ways to pass parameters in C:

- **Pass by Value:** In this method, a copy of the actual value is passed to the function. Changes made to the parameter inside the function do not affect the original value.
- **Pass by Reference:** This approach involves passing the address of the variable instead of the actual value. As a result, any modifications made to the parameter inside the function will directly affect the original variable.
- **Using Pointers:** Parameters can be passed to functions using pointers, which allows the function to modify the variable's value at the memory address being pointed to.