International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 02 Issue 11
November 2015

Functions in C

Ritika Singh

Computer Science and Engineering, MDU

Ritika.17117@ggnindia.dronacharya.info

ABSTRACT

C is a computer programming language. That means that you can use Cto create lists of instructions for a computer to follow. C is one of thousands of programming languages currently in use. A function is a group of statements that together perform a task. Everyc program has at least one function, which is main(), and all the most trivial programs can define additional functions. You can divide up your code into separate functions. A function declaration tells the compiler about a function's name, return type, and parameters.

Keywords: - calling a function, arguments, call by value, call by reference

Introduction

A function is a group of statements that together perform a task. Every C program has at least one function, which is main(), and all the most trivial programs can define additional functions.You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division usually is so each function performs a specific task.The C standard library provides numerous built-in functions that your program can call. For example, function strcat() to concatenate two strings, functionmemcpy() to copy one memory location to another location and many more functions.

The general form of a function definition in C programming language is as follows:

```
Return_type function_name( parameter list )
{
 Body of the function
}
```

Return Type: A function may return a value. The return_type is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword void.

- Function Name: This is the actual name of the function. The function name and the parameter list together constitute the function signature.
- Parameters: A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.

- Function Body: The function body contains a collection of statements that define what the function does.

A function declaration has the following parts:

- Return_type function_name(parameter list);

Function declaration is required when you define a function in one source file and you call that function in another file. In such case you should declare the function at the top of the file calling the function.

Calling a Function:

When a program calls a function, program control is transferred to the called function. A called function performs defined task and when its return statement is executed or when its function-ending closing brace is reached, it returns program control back to the main program.

To call a function, you simply need to pass the required parameters along with function name, and if function returns a value, then you can store returned value.

```
#include <stdio.h>
 /* function declaration */
int max(int num1, int num2);
 int main ()
{
   /* local variable definition */
   int a = 100;
   int b = 200;
   int ret;
 /* calling a function to get max value
*/
   ret = max(a, b);
  printf( "Max value is : %d\n", ret );
  return 0;
}
 /* function returning the max between
two numbers */
int max(int num1, int num2)
{  /* local variable declaration */
   int result;

   if (num1 > num2)
      result = num1;
 else
      result = num2;
return result;
}
```

Function Arguments:

While calling a function, there are two ways that arguments can be passed to a function:

	Description
Call by value	This method copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.
Call by reference	This method copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument.