C FUNCTION

Dr. Mukti Jadhav

WHAT IS C FUNCTION?

- A large C program is divided into basic building blocks called C function. C function contains set of instructions enclosed by “{ }” which performs specific operation in a C program. Actually, Collection of these functions creates a C program.

USES OF C FUNCTIONS:

- C functions are used to avoid rewriting same logic/code again and again in a program.
- There is no limit in calling C functions to make use of same functionality wherever required.
- We can call functions any number of times in a program and from any place in a program.
- A large C program can easily be tracked when it is divided into functions.
- The core concept of C functions are, re-usability, dividing a big task into small pieces to achieve the functionality and to improve understandability of very large C programs.

C FUNCTION DECLARATION, FUNCTION CALL AND FUNCTION DEFINITION:

- There are 3 aspects in each C function. They are,
- Function declaration or prototype – This informs compiler about the function name, function parameters and return value's data type.
- Function call – This calls the actual function
- Function definition – This contains all the statements to be executed.

C functions aspects	syntax
function definition	Return_type function_name (arguments list) { Body of function; }
function call	function_name (arguments list);
function declaration	return_type function_name (argument list);

SIMPLE EXAMPLE PROGRAM FOR C FUNCTION

- As you know, functions should be declared and defined before calling in a C program.
- In the below program, function “square” is called from main function.
- The value of “m” is passed as argument to the function “square”. This value is multiplied by itself in this function and multiplied value “p” is returned to main function from function “square”.

Conti..

```
#include<stdio.h>
// function prototype, also called function declaration
float square ( float x );
// main function, program starts from here

int main( )
{
   float m, n ;
   printf ( "\nEnter some number for finding square \n");
   scanf ( "%f", &m ) ;
   // function call
   n = square ( m ) ;
   printf ( "\nSquare of the given number %f is %f",m,n );
}

float square ( float x )   // function definition
{
   float p ;
   p = x * x ;
   return ( p ) ;
}
```

Output

Enter some number for finding square

2

Square of the given number 2.000000 is
4.000000

HOW TO CALL C FUNCTIONS IN A PROGRAM

- There are two ways that a C function can be called from a program.
- Call by value
- Call by reference

CALL BY VALUE

- In call by value method, the value of the variable is passed to the function as parameter.
- The value of the actual parameter can not be modified by formal parameter.
- Different Memory is allocated for both actual and formal parameters. Because, value of actual parameter is copied to formal parameter.
- Note:
- Actual parameter – This is the argument which is used in function call.
- Formal parameter – This is the argument which is used in function definition

EXAMPLE PROGRAM FOR C FUNCTION (USING CALL BY VALUE):

- In this program, the values of the variables “m” and “n” are passed to the function “swap”.
- These values are copied to formal parameters “a” and “b” in swap function and used.

Conti..

```
#include<stdio.h>
// function prototype, also called function declaration
void swap(int a, int b);

int main()
{
   int m = 22, n = 44;
   // calling swap function by value
   printf(" values before swap  m = %d \nand n = %d", m, n);
   swap(m, n);
}

void swap(int a, int b)
{
   int tmp;
   tmp = a;
   a = b;
   b = tmp;
   printf(" \nvalues after swap m = %d\n and n = %d", a, b);
}
```

Output

- values before swap m = 22
  and n = 44
  values after swap m = 44
  and n = 22

CALL BY REFERENCE:

- In call by reference method, the address of the variable is passed to the function as parameter.
- The value of the actual parameter can be modified by formal parameter.
- Same memory is used for both actual and formal parameters since only address is used by both parameters.

EXAMPLE PROGRAM FOR C FUNCTION (USING CALL BY REFERENCE)

- In this program, the address of the variables “m” and “n” are passed to the function “swap”.
- These values are not copied to formal parameters “a” and “b” in swap function.
- Because, they are just holding the address of those variables.
- This address is used to access and change the values of the variables.
-

Conti..

```
#include<stdio.h>
// function prototype, also called function declaration
void swap(int *a, int *b);

int main()
{
   int m = 22, n = 44;
   //  calling swap function by reference
   printf("values before swap m = %d \n and n = %d",m,n);
   swap(&m, &n);
}

void swap(int *a, int *b)
{
   int tmp;
   tmp = *a;
   *a = *b;
   *b = tmp;
   printf("\n values after swap a = %d \nand b = %d", *a, *b);
}
```

Output

- values before swap m = 22
- and n = 44
  values after swap a = 44