FUNCTIONS IN C

C functions are basic building blocks in a program. All C programs are written using functions to improve re-usability, understandability and to keep track on them. You can learn below concepts of C functions in this section in detail.

1. 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.

2. 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.

3. 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".

```
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
```

4. How to call C functions in a program?

There are two ways that a C function can be called from a program. They are,

1. Call by value
2. Call by reference

1. 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.

```
#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
```

2. 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.

```
#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
and b = 22
```

All C functions can be called either with arguments or without arguments in a C program. These functions may or may not return values to the calling function. Now, we will see simple example C programs for each one of the below.

1. C function with arguments (parameters) and with return value.
2. C function with arguments (parameters) and without return value.
3. C function without arguments (parameters) and without return value.
4. C function without arguments (parameters) and with return value.

C functions aspects	syntax
1. With arguments and with return values	**function declaration:** int function (int);**function call:** function (a);

	function definition:
	int function(int a)
	{
	statements;
	return a;
	}

2. With arguments and without return values	**function declaration:**
	void function (int);**function call:** function(a);
	
	function definition:
	void function(int a)
	{
	statements;
	}

3. Without arguments and without return values	**function declaration:**
	void function();**function call:** function();
	
	function definition:
	void function()
	{
	statements;
	}

4. Without arguments and with return values	**function declaration:**
	int function ( );**function call:** function ( );
	
	function definition:
	int function( )
	{
	statements;
	return a;
	}

Note:

- If the return data type of a function is "void", then, it can't return any values to the calling function.
- If the return data type of the function is other than void such as "int, float, double etc", then, it can return values to the calling function.

1. Example program for with arguments & with return value:

In this program, integer, array and string are passed as arguments to the function. The return type of this function is "int" and value of the variable "a" is returned from the function. The values for array and string are modified inside the function itself.

```
#include<stdio.h>
#include<string.h>
int function(int, int[], char[]);

int main()
{
    int i, a = 20;
    int arr[5] = {10,20,30,40,50};
    char str[30] = "\"fresh2refresh\"";

    printf("    ***values before modification***\n");
    printf("value of a is %d\n",a);

    for (i=0;i<5;i++)
    {
      // Accessing each variable
      printf("value of arr[%d] is %d\n",i,arr[i]);
    }
    printf("value of str is %s\n",str);
    printf("\n    ***values after modification***\n");
    a= function(a, &arr[0], &str[0]);
    printf("value of a is %d\n",a);
    for (i=0;i<5;i++)
    {
      // Accessing each variable
      printf("value of arr[%d] is %d\n",i,arr[i]);
    }
    printf("value of str is %s\n",str);
    return 0;
}

int function(int a, int *arr, char *str)
{
   int i;

   a = a+20;
   arr[0] = arr[0]+50;
   arr[1] = arr[1]+50;
   arr[2] = arr[2]+50;
   arr[3] = arr[3]+50;
   arr[4] = arr[4]+50;
   strcpy(str,"\"modified string\"");

   return a;

}
```

Output:

```
***values before modification***
value of a is 20
value of arr[0] is 10
value of arr[1] is 20
value of arr[2] is 30
value of arr[3] is 40
value of arr[4] is 50
value of str is “fresh2refresh”***values after modification***
value of a is 40
value of arr[0] is 60
value of arr[1] is 70
value of arr[2] is 80
value of arr[3] is 90
value of arr[4] is 100
value of str is “modified string”
```

2. Example program for with arguments & without return value:

In this program, integer, array and string are passed as arguments to the function. The return type of this function is “void” and no values can be returned from the function. All the values of integer, array and string are manipulated and displayed inside the function itself.

```c
#include<stdio.h>

void function(int, int[], char[]);

int main()
{
    int a = 20;
    int arr[5] = {10,20,30,40,50};
    char str[30] = "\"fresh2refresh\"";

    function(a, &arr[0], &str[0]);
    return 0;
}

void function(int a, int *arr, char *str)
{
   int i;
   printf("value of a is %d\n\n",a);
   for (i=0;i<5;i++)
     {
       // Accessing each variable
       printf("value of arr[%d] is %d\n",i,arr[i]);
     }
   printf("\nvalue of str is %s\n",str);
}
```

Output:

```
value of a is 20

value of arr[0] is 10
value of arr[1] is 20
value of arr[2] is 30
value of arr[3] is 40
value of arr[4] is 50

value of str is “fresh2refresh”
```

3. Example program for without arguments & without return value:

In this program, no values are passed to the function “test” and no values are returned from this function to main function.

```c
#include<stdio.h>
void test();

int main()
{
   test();
   return 0;
}

void test()
{
   int a = 50, b = 80;
   printf("\nvalues : a = %d and b = %d", a, b);
}
```

Output:

```
values : a = 50 and b = 80
```

4. Example program for without arguments & with return value:

In this program, no arguments are passed to the function “sum”. But, values are returned from this function to main function. Values of the variable a and b are summed up in the function “sum” and the sum of these value is returned to the main function.

```
#include<stdio.h>

int sum();

int main()
{
    int addition;
    addition = sum();
    printf("\nSum of two given values = %d", addition);
    return 0;
}

int sum()
{
    int a = 50, b = 80, sum;
    sum = a + b;
    return sum;
}
```

Output:

```
Sum of two given values = 130
```

Do you know how many values can be return from C functions?

- **Always, Only one value can be returned from a function.**
- **If you try to return more than one values from a function, only one value will be returned that appears at the right most place of the return statement.**
- **For example, if you use "return a,b,c" in your function, value for c only will be returned and values a, b won't be returned to the program.**
- **In case, if you want to return more than one values, pointers can be used to directly change the values in address instead of returning those values to the function.**

C – Library functions

- Library functions in C language are inbuilt functions which are grouped together and placed in a common place called library.
- Each library function in C performs specific operation.
- We can make use of these library functions to get the pre-defined output instead of writing our own code to get those outputs.
- These library functions are created by the persons who designed and created C compilers.
- All C standard library functions are declared in many header files which are saved as file_name.h.
- Actually, function declaration, definition for macros are given in all header files.
- We are including these header files in our C program using "#include<file_name.h>" command to make use of the functions those are declared in the header files.
- When we include header files in our C program using "#include<filename.h>" command, all C code of the header files are included in C program. Then, this C program is compiled by compiler and executed.

- If you want to check source code for all header files, you can check inside "include" directory after C compiler is installed in your machine.
- For example, if you install DevC++ compiler in C directory in your machine, "C:\Dev-Cpp\include" is the path where all header files will be available.

List of most used header files in C programming language:

- Check the below table to know all the C library functions and header files in which they are declared.
- Click on the each header file name below to know the list of inbuilt functions declared inside them.

Header file	**Description**
stdio.h	This is standard input/output header file in which Input/Output functions are declared
conio.h	This is console input/output header file
string.h	All string related functions are defined in this header file
stdlib.h	This header file contains general functions used in C programs
math.h	All maths related functions are defined in this header file
time.h	This header file contains time and clock related functions
ctype.h	All character handling functions are defined in this header file

stdarg.h	Variable argument functions are declared in this header file
signal.h	Signal handling functions are declared in this file
setjmp.h	This file contains all jump functions
locale.h	This file contains locale functions
errno.h	Error handling functions are given in this file
assert.h	This contains diagnostics functions