Unit-03
Functions

Functions in C

- A function in C is a set of statements that when called perform some specific tasks.
- It is the basic building block of a C program that provides modularity and code reusability.
- The programming statements of a function are enclosed within {} braces, having certain meanings and performing certain operations.
- They are also called subroutines or procedures in other languages.
- A function is a block of code which only runs when it is called.
- Functions are used to perform certain actions, and they are important for reusing code - Define the code once, and use it many times.
- A function is a self-contained block of codes or sub programs with a set of statements that perform some specific tasks when it is called.
- Takes input, perform some computation and produce output
- There are basically two types of function those are
    1. Standard library functions
    2. User-defined function
- System defined function can't be modified, it can only read and can be used.
- The user defined functions defined by the user according to its requirement.

Standard Library functions

- Standard library functions are built-in functions in C programming.
- These functions are defined in header files.
- For example –
    - The printf() is a standard library function to send formatted output to the screen (display output on the screen).
        - This function is defined in the stdio.h header file.
        - Hence, to use the printf() function, we need to include the stdio.h header file using #include <stdio.h>.
    - The sqrt() function calculates the square root of a number.
        - The function is defined in the math.h header file.

User-defined Functions

- A user-defined function is a type of function in C language that is defined by the user himself to perform some specific task.
- It provides code reusability and modularity to our program.

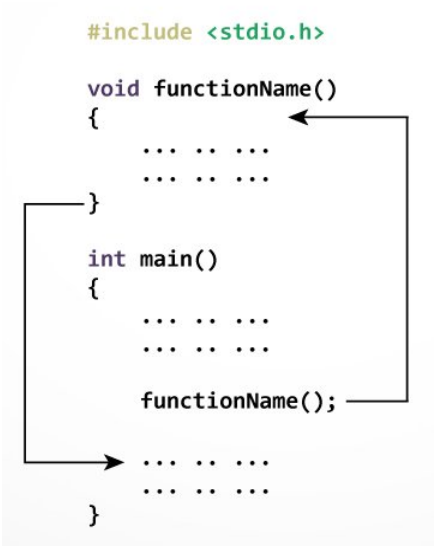
- User-defined functions are different from built-in functions as their working is specified by the user and no header file is required for their usage.

Need for user defined Functions –

- It is possible to code any program utilizing only main function – But, it leads to a number of problems -
- The program may become too large and complex and as a result the task of debugging, testing, and maintaining becomes difficult.
- It is much easier to write a structured program where a large program can be divided into a smaller, simpler task.
- Allowing the code to be called many times.
- Easier to read and update.
- It is easier to debug a structured program where their error is easy to find and fix.

Working of User-defined Functions

- C program doesn't execute the statement in function until the function is called.
- When function is called the program can send the function information in the form of one or more argument.
- When the function is used, it is referred to as the called function
- Functions often use data that is passed to them from the calling function
- Data is passed from the calling function to a called function by specifying the variables in a argument list.
- Argument list cannot be used to send data. Its only copy data/value/variable that pass from the calling function.
- The called function then performs its operation using the copies.
- Functions are classified as one of the derived data types in C



Similarities between functions and variables in C

- Both function name and variable names are considered identifiers and therefore they must adhere to the rules for identifiers.
- Like variables, functions have types (such as int) associated with them.
- Like variables, function names and their types must be declared and defined before they are used in a program.

Elements of User-defined Functions

- Function declaration or prototype –
  - 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.

Sr. No.	C Function aspects	Syntax
1	Function Declaration	return_type function_name (argument list);
2	Function Call	function name (argument list);
3	Function Definition	return_type function_name (argument list) {body of function}

- These three things are represented like –

```
int function (int, int, int); /*function declaration*/
main () /* main function*/
{
        function(arg1,arg2,arg3);        /* function call */
}

int function (type 1 arg 1, type2 arg2, type3 arg3) /*function definition*/
{
        Local variable declaration;
        Statement;
        Return value;
}
```

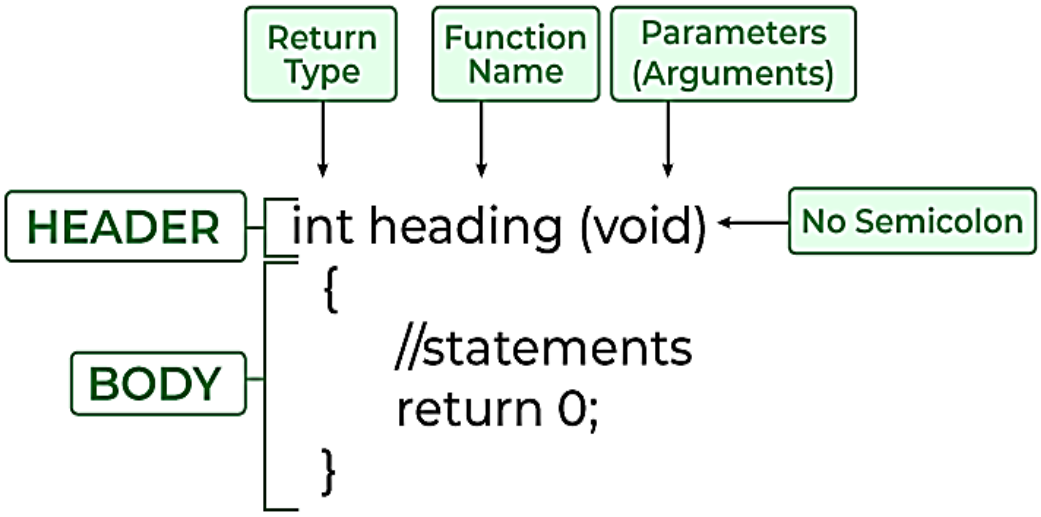
Function Definition

- Function definition contains the block of code to perform a specific task.
- For example -

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

- A function definition, also known as function implementation shall include the following elements –
  - Function name;
  - Function type;
  - List of parameters;
  - Local variable declaration;
  - Function statements;
  - A return statements.
- All six elements are grouped into two parts, namely,
  - Function header (First three elements); and
  - Function body (Second three elements)

Syntax of function definition –



- When a function is called, the control of the program is transferred to the function definition.
- And, the compiler starts executing the codes inside the body of a function.

Function Header

- The function header consists of three parts:
  - Function type (also known as return type)
  - Function name
  - Formal parameter list.
- Semicolon is not used at the end of the function header.
- **Name and Type**
  - The function name is any valid C identifier.
  - The function type –
  - Specifies the type of value (like float or double) that the function id expected to return to the program calling the function.
  - If the function is not returning anything then we need to specify the return type as void.

Function Arguments or Parameters –

- If a function is to use arguments, it must declare variables that accept the values of the arguments.
- These variables are called the parameters of the function.

Parameter –

- A parameter is a special kind of variable, used in a function to refer to one of the pieces of data provided as input to the function to utilize.
  - These pieces of data are called arguments.
  - Parameters are Simply Variables.
- The parameter list declares the variables that will receive the data sent by the calling program.
- They serve as input data to the function to carry out the specified task.
- The parameter is known as arguments.
  - float quadratic (int a, int b, int c) { ….. }
  - double power (double x, int n) { ….. }
- There is no semicolon after the closing parenthesis.
- The declaration parameter variables cannot be combined.
- **Formal Parameter**
  - Parameter Written in Function Definition is Called "Formal Parameter.
  - Formal parameters are always variables, while actual parameters do not have to be variables.
- **Actual Parameter**
  - Parameter Written in Function Call is Called "Actual Parameter".
  - One can use numbers, expressions, or even function calls as actual parameters.
- Example:

```
void display(int digit)
{
        printf( " Number %d " , digit);
}
void main()
{
        int number;
        display(number);
}
```

- digit is called the Formal Parameter
- number is called the Actual Parameter

  - In this, 'digit' is called the Formal Parameter
  - And, 'number' is called the Actual Parameter

Function Body

- The function body contains the declarations and statements necessary for performing the required task.
- The body enclosed in braces, contains three parts –
  - Local declarations - that specify the variables needed by the function
  - Function statements - that perform the task of the function
  - A return statement - that returns the value evaluated by the function
- If a function does not return any value, we can omit the return statement.
  - Its return type should be specified as void
- Function Definition – Example

```
float multiplication (float x, float y)
{
        float result;
        result = x * y;
        return result;
}
```

  - In this –
    - Function name: multiplication
    - It accepts the arguments x and y of type float.
    - Function return float value
    - When this function is called, it will perform its task which is to multiply two numbers and return result of multiplication.
    - If function is returning a value, it needs to use keyword return.
    -

Function Return Types

- A function may or may not send back any value to the calling function.
- If it does, it is done through return statement.
- The called function can only return one value per call.
- It is used to return value to the calling function
- Syntax:

```
return;
or
return (expression);
```

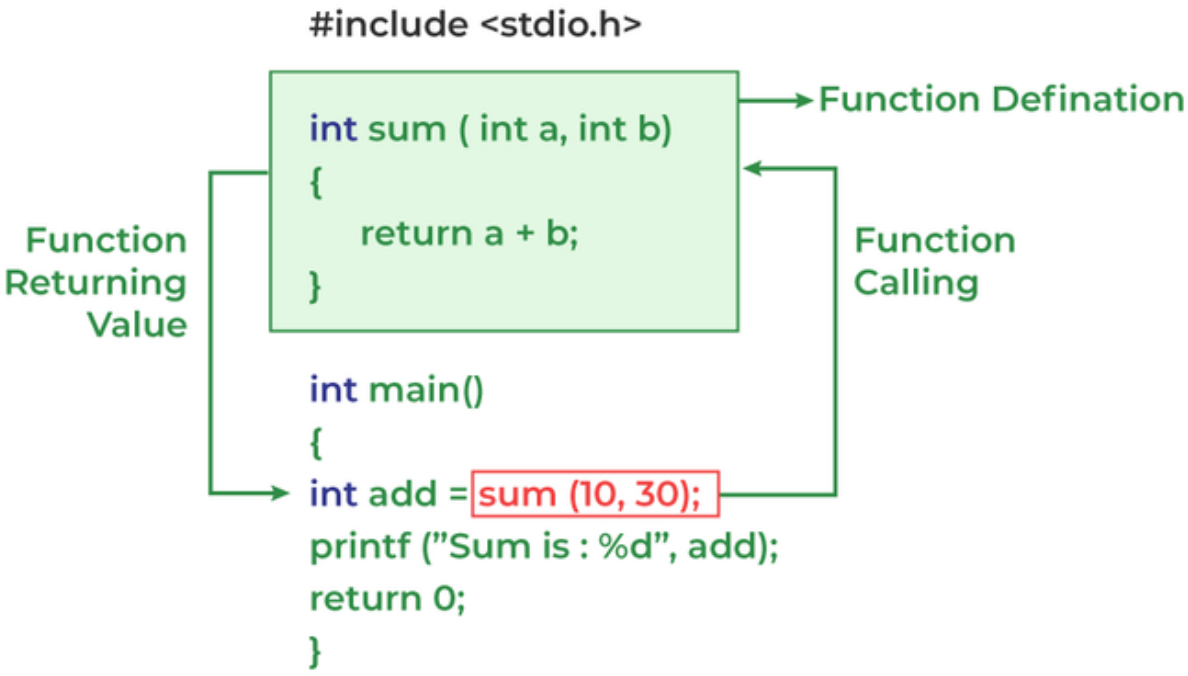
- The function return type specifies the data type that the function returns to the calling program.
- The return types are of C's data types: char, int, float, double…
- One can also define a function that doesn't return a value by using a return type of void.

- Few examples:
  - int func1(...) /* Returns a type int */
  - float func2(...) /* Returns a type float */
  - void func3(...) /* No Returns */

Calling a Function

- When the function gets called by the calling function then that is called, function call.
- Function can be called by using function name followed by actual parameters.
- Example:

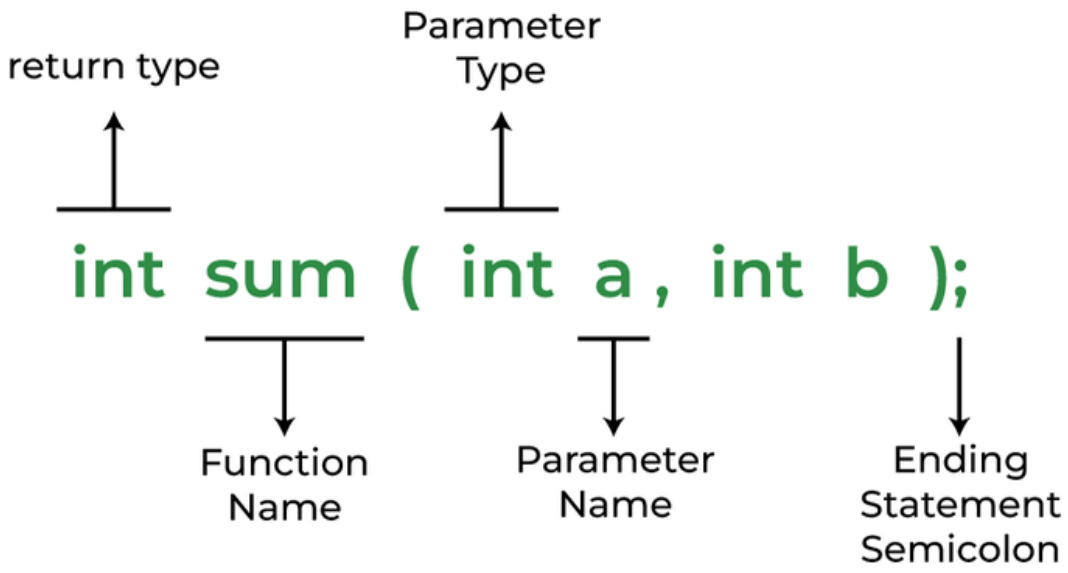
```
function_name(arg1,arg2,arg3);
```



- There are two ways to call a function.
  1. Any function can be called by simply using its name and argument list alone in a statement.
  2. Functions that have a return value –
     - Because these functions evaluate to a value (that is, their return value), they are valid C expressions and can be used anywhere a C expression can be used.
     - An expression with a return value used as the right side of an assignment statement.

Function Declaration / Prototype

- Also known as function prototype: declaring properties of function
- It informs the compiler about three thing –
  - name of the function,
  - number and type of argument received by the function
  - type of value returned by the function.
- It consists of four parts:
  - Function type (return type)
  - Function name
  - Parameter list
  - Terminating semicolon
- Syntax of function prototype –



- Points to note
  - The parameter list must be separated by commas.
  - The parameter names do not need to be the same in the prototype declaration and the function definition,
  - The types must match the types of parameters in the function definition, in number and order.
  - Use of parameter names in the declaration is optional.
  - If the function has no formal parameters, the list is written as (void).
  - The return type is optional, when the function returns int type data.
  - The retype must be void if no value is returned.
  - When the declared types do not match with the types in the function definition, compiler will produce an error.

Categories of Function

- C function with arguments (parameters) and with return value
- C function with arguments (parameters) and without return value
- C function without arguments (parameters) and without return value
- C function without arguments (parameters) and with return value

C function with arguments (parameters) and with return value

- Arguments are passed by calling function to the called function
- Called function return value to the calling function
- Mostly used in programming because it can two-way communication
- Data returned by the function can be used later in our program for further calculation.
- Form:

```
int function ( int );        // function declaration
function ( a );              // function call
int function( int a )        // function definition
{statements;
return a;}
```

C function with arguments (parameters) and without return value

- A function has argument/s
- A calling function can pass values to function called, but calling function not receive any value
- Data is transferred from calling function to the called function but no data is transferred from the called function to the calling function
- Generally, Output is printed in the Called function
- Form:

```
void function ( int );       // function declaration
function( a );               // function call
void function( int a )       // function definition
{statements;}
```

C function without arguments (parameters) and without return value

- Called function does not have any arguments
- Not able to get any value from the calling function
- Calling function Not returning any value
- There is no data transfer between the calling function and called function.
- Form:
    - `void function ( int );` // function declaration
    - `function( a );` // function call
    - `void function( int a )` // function definition
    - `{statements;}`

C function without arguments (parameters) and with return value

- Does not get any value from the calling function
- Give a return value to calling program
- Form:

```
int function ( );      // function declaration
function ( ); // function call
int function( ) // function definition
{statements; return a;}
```

Sno	C function	C function

1	with arguments and with return values	int function (int); // function declaration
		function (a); // function call
		int function(int a) // function definition
		{statements; return a;}

2	with arguments and without return values	void function (int); // function declaration
		function(a); // function call
		void function(int a) // function definition
		{statements;}

3	without arguments and without return values	void function(); // function declaration
		function(); // function call
		void function() // function definition
		{statements;}

4	without arguments and with return values	int function ( ); // function declaration
		function ( ); // function call
		int function( ) // function definition
		{statements; return a;}

Advantages of Functions in C

- The function can reduce the repetition of the same statements in the program.
- The function makes code readable by providing modularity to our program.
- There is no fixed number of calling functions it can be called as many times as you want.
- The function reduces the size of the program.

Disadvantages of Functions in C

- Cannot return multiple values.
- Memory and time overhead due to stack frame allocation and transfer of program control.