

FUNCTIONS

MONOLITHIC PROGRAMMING

- If we write one program, and if we put everything inside the main function, then such type of programming approach is called Monolithic programming.
- If your main function contains thousands of lines of code, then it is becoming very difficult to manage.
- This is actually not a good programming approach.
- We can break the program into smaller tasks, i.e. into smaller functions and each function performs a specific task, then such type of programming is called “**modular programming**” or “**procedural programming**” and this approach is good for development.

```

main()
{
    line1
    line2
    line3
    line4
    -----
    -----
    -----
    line978
    line979
    -----
    -----
}

```

Monolithic programming:
It means everything inside
a single method

```

function1()
{
    //function1 task
}
function2()
{
    //function2 task
}
function3()
{
    //function3 task
}

main()
{
    function1();
    function2();
    function3();
}

```

**Modular programming or
Procedural Programming**

FUNCTION

- Function is a group of related instructions(or piece of code) that perform a specific task.
- Functions are also called modules or procedures.
- A function is a
 - block of instructions (here we can write n number of valid instructions)
 - having identity (the name is mandatory)
 - taking inputs (technically called as argument list) and processing the input
 - produces the output (technically we called it as return type).
- Argument (or parameters) and return type of a function is optional.

return_type identify (arguments_list)

{

.....

.....

Processing input,

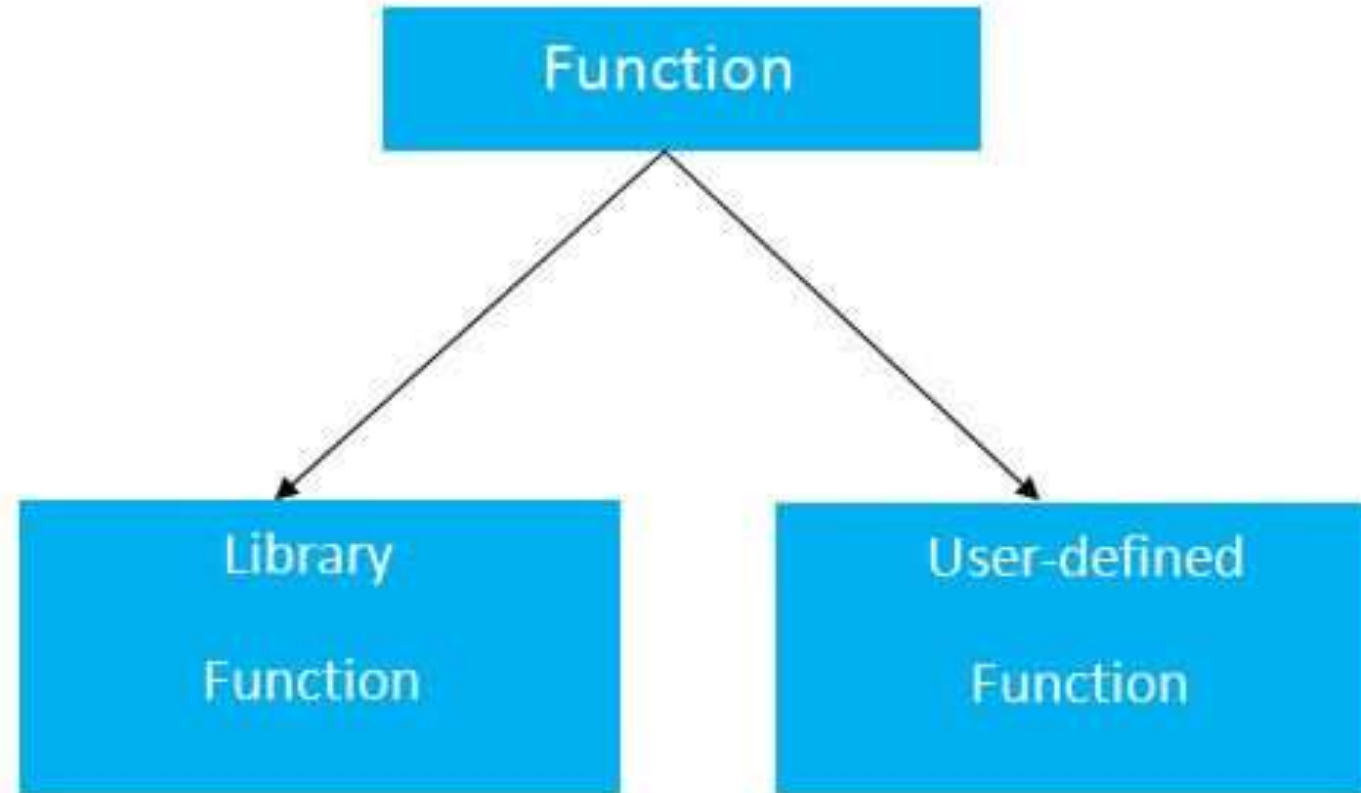
.....

}

OUTPUT

INPUT

TYPES OF FUNCTION



LIBRARY FUNCTIONS

- A header file consists of the function declarations, constant and macros.
- These declarations tell the compiler about function name, return type and parameters.
- Library file contains the actual implementation of the function declared the in header file.
- Therefore, the key difference between header file and library file is that header file contains the function declarations to be shared between several source files while library file is a file that contains the function definition for the declared functions in the header file.

PARTS OF A FUNCTION

- **Return Type:** Defines the type of value function will return
- **Name of the function:** It is the identifier for the function name.
- **Arguments (Parameters):** It is the variables that the function will use.
- **Body:** This contains set of instructions of the function.
- ***return* Statement:** It is used to return data from function.
- Function does not have access to use the variables of other functions (even the main function). So if you want the function to use some variable of another function then you have passed that variable as an argument to the function.

ASPECTS OF FUNCTION


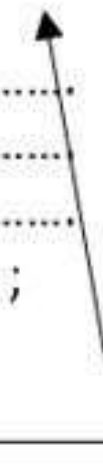
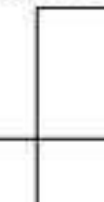
- **FUNCTION DECLARATION OR PROTOTYPE:** It tell the compiler about the function name, function parameters, and return type.
- **FUNCTION CALL:** This calls the actual function.
- **FUNCTION DEFINITION:** It contains the actual statements which are to be executed.

C functions aspects	syntax
Function Declaration or Prototype	Return type function_name (argument list);
Function Definition	Return type function_name (arguments list) { /*Body of function; */ }
Function Call	function_name (arguments list);

```
#include <stdio.h>
void functionName()
{
    ....
    ....
}
int main()
{
    ....

    ....
    functionName();
    ....
    ....
}
```

- The execution of a program begins from the main() function.
- When the compiler encounters **functionName();** the control of the program jumps to **void functionName()** definition and the compiler starts executing the codes inside functionName().
- The control of the program jumps back to the main() function once the code inside the function definition is executed.

NO ARGS & NO RETRUN	WITH ARGS & NO RETURN	WITH ARGS & WITH RETURN	NO ARGS & WITH RETURN
<pre>void fun (void) { }</pre>	<pre>void fun (int 10) { }</pre> 	<pre>void fun (char x) { return 13 ; }</pre> 	<pre>float fun (void) { return 34.56 ; }</pre> 
<pre>fun () ;</pre>	<pre>fun (10) ;</pre>	<pre>int x = fun ('g') ;</pre>	<pre>float x = fun () ;</pre>

ADVANTAGES OF FUNCTIONS

- 1.Module Approach:** By using the function we can develop the application in module format i.e. procedure-oriented language concept.
- 2.Reusability:** By using functions we can create re-usability blocks i.e. develop once and use multiple times.
- 3.Code Maintenance:** When we are developing the application by using functions, then it is easy to maintain code for future enhancement.
- 4.Code Sharing:** A function may be used by many other programs.
- 5.Flexible Debugging:** It is easy to locate and isolate a faulty function for further investigations.
- 6.Data Protection:** Functions can be used to protect data and local data. Local data is available only within a function when the function is being executed.
- 7.Code Reduced:** Reduces the size of the code, duplicate statements are replaced by function calls.