USER DEFINED FUNCTIONS

What is a function?

1. It is a small program used to do a particular task.
2. It is a small program with in another program.
3. It is a sub program or sub routine.
4. It is a procedure.
5. It is a self-contained block.
6. It is a reusable code component.
7. It is a module.

Advantages:

1. **Modularity**: Dividing program instructions into small pieces.
2. **Simplicity**: It is easy to understand the program instructions.
3. **Reusability**: Just write once, use several times.
4. **Efficiency**: performance is improved.
5. It is easy to identify the errors.

Entire ‘C’ program is collection of functions. Hence ‘C’ is a function oriented structured programming language.

We are having 2 types of functions.

1. Predefined / library / built-in functions

These are the functions provided with software and ready to use.

Eg: printf(), scanf(),…

2. **User defined functions**: They are created by the user.

Eg: sum(), prime(), factorial(),……

Every user defined function is divided into 3 parts.

1. Function declaration/ proto typing
2. Function calling
3. Function definition

Function declaration/proto typing:

Generally function declaration is conducted before or within the main(), before function calling.

It tells the compiler that we are going to use this kind of function in our program in future.

Syntax:

[return_datatype] function_name ([arguments / parameters]);

Eg:

void sum();

Function calling:

It should be conducted within the main() only.

When a function is invoked (called), then the compiler will search for the matching function definition and it is available then the program execution is jumped from function calling area to the function definition area.

Linking a function call to the function definition is called **binding.** C language supports only static or compile time binding. But C++ supports static and dynamic binding[run time binding].

Without function call, a function never participates in function execution. Hence it is mandatory to call a function in a program.

Syntax:

function_name([arguments]);

Eg: sum();

Function definition:

It contains the function header and body. The function header should be matched with function declaration.

It is conducted outside the main(). If the definition is conducted before the main(), there is no need of function declaration.

Function header consists of function name and the parameters.

Function body consists of statements related to the task of function.

When a function is invoked, the program execution is shifted from function calling area (main()) to function definition area. After executing the function body, the execution is jumped to the next statement after the function call in main().

Function declaration and function definition should be identical.

Syntax:

Function Header

```
[return data type]  Function_name( [parameters] )
{
  Statements;
  [return value;]
}
```

Function Body

}

Here return data type indicates the type of value that the function is returning to main() / called function. It is depended on the return value.

If the sub program is having return statement then it is called function and without return value it is called procedure.

The **default return value of a function is integer**. Use the keyword **void** [nothing] when the function is not having any return value. Otherwise integer will become the return value.

A function may have any number of return statements. But only one return statement is executed at a time. The statements after the return statement are not executed and gives compile time warning.

return data type should be matched with return value data type.

function name is used to identify a function and it should not be predefined.

Arguments / parameters are used to carry the values from function calling area to function definition area. They are optional and of two types.

1. **Actual** arguments/parameters.

2. **Formal** arguments/parameters.

The arguments we are sending in function calling at main() are called actual parameters and the parameters we are using in function definition to receive the actual argument values is called formal parameters.

Actual and formal parameter names may be same or different.

Difference between arguments and parameters is "arguments are nothing but different vehicles which are parked in same parking area and the parking area is the parameters. i.e. arguments are always changed but parameters are fixed.

The calling function arguments should be matched with definition parameters list in quantity, data type and order.

Based on arguments and return values, functions are divided into 4 types.

1. Function without arguments, without return values.
2. Function with arguments and with return values.
3. Function without arguments and with return values.
4. Function with arguments and without return values.

Eg:

1. **function without arguments, without return values**