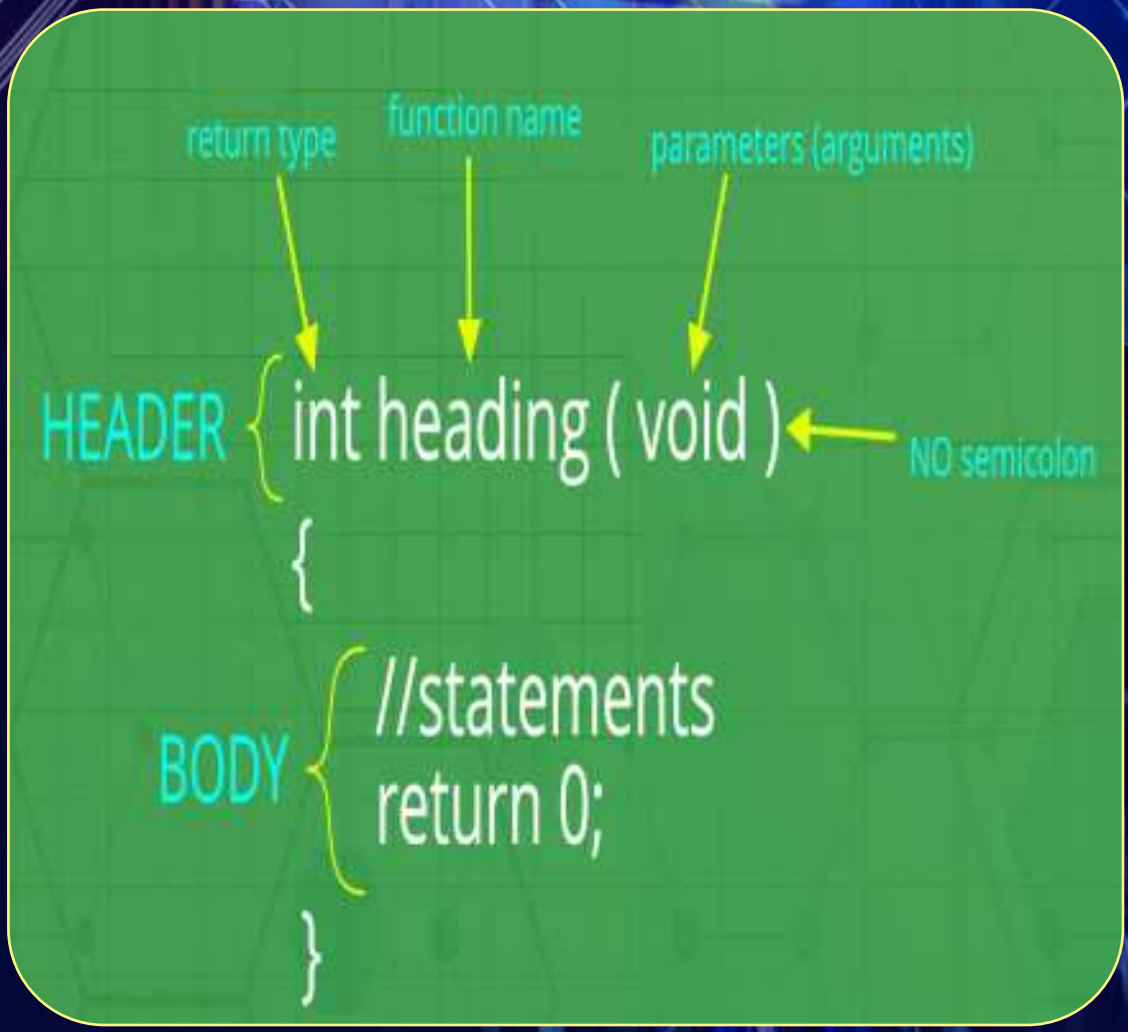
FUNCTIONS

Unit 5



- Define the term function.
  - Function Declaration.
  - Function Definition
  - Function Call
- Differentiate between function call and function definition

5.1 INTORDUCTION TO FUNCTIONS

A set of statements written to perform a specific task and having a unique name is called a function. In structured programming, the complicated and large program coding is broken down into smaller modules which are called subprograms. In C++, subprograms are called functions. Every program has at least one `main( )` function in C++. When the program starts, the `main( )` function is called for execution.

There are two types of functions.

1. Pre-defined Functions
2. User-defined Functions

5.1.1 Pre-define Functions:

The pre-define functions are the part of every high-level programming language. It can be used for different purposes. Predefined functions are also known as System-defined or library functions.

These functions do not need to be declared and defined. Pre-define functions are declared in header file. All predefined functions can be used simply by calling the function like sqrt(), strcpy(), toupper(), pow() etc. Many pre-define functions need proper header file by using #include pre-processor directive. The definitions of many common functions are found in the cmath and cstdlib libraries.

Teachers Note 	Teacher should explain briefly Pre-define function and give some examples of pre-define function for the practice of students.

5.1.2 User – defined-function

Programmer can also write their own functions to perform specific task. They are called user-define-functions. These functions need declaration and definition. When the user-defined function is called from any part of the program, it will execute the code defined inside the body of the function. multiply ( ), sum( ), average( ) may be the example of User–define functions. A user-define function based on two parts:

1. Function declaration or prototype
2. Function definition

1. Function Declaration:

A function without its definition (code block) is known as a function declaration or function prototype. It is declared before the main( ) function. A function declaration tells the compiler about the function's name, return data types, and arguments/parameter data types and it ends with statement terminator (;).

Syntax:

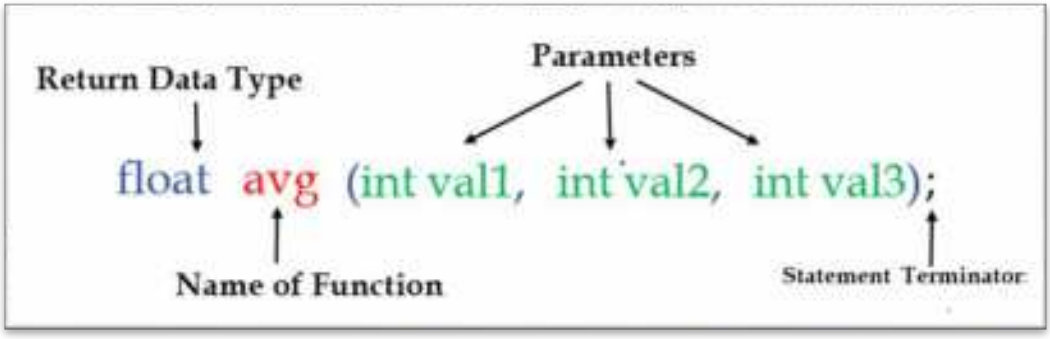


Fig. 5.1 Function Declaration

a. Return Data Type:

It shows the data type of value returned by function. It may be int, float, double and char data type. If no value is returned by the function in that case keyword "void" is used.

b. Function Name:

It specifies the name of function. It is recommended that meaningful and understandable names are given to the function.

c. Parameters:

It define the list of data types of function parameters that are to be passed to the function. Parameters are separated by commas. Parameters are also known as arguments. If there is no parameter in function, programmer uses keyword "void". Variable names are optional in prototype parameters/ arguments.

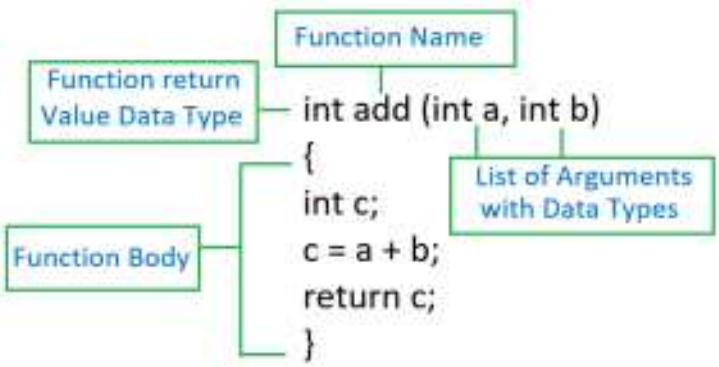
d. Statement Terminator:

In function declaration, statement terminator must be used at the end.

2. Function Definition:

Function definition is function itself. It has a function header and a body or code block. Header has three parts; return value data type, function name and list of augments with datatypes in parenthesis. Body of the function includes statements in braces. Function definition may be defined before or after the main function.

For example:



Function call

To use the function code, we have to call or invoke that function with its name. It is called function call. When function is called for execution, control will transfer to the function definition and all statements of function definition will execute and after executing the statements the control will transfer back to the calling function (statement after function call).

If the function is without return value and no arguments then it is called by its name. It means function's braces will be empty.

Syntax: **function_name ( );**
Example: add( );

If function returns a value, then we can store return value to a variable in the calling function.

Example: x=add (y, z);

Function passing argument or parameters:

An argument is a part of data passed to the function. When function is called for execution, the actual values as parameters are also given with function call statement. Passing actual values to function as arguments with function call statement are known as actual parameters. Actual parameters may be variables or constants. They are placed in paratheses after the function name. These values are received in variables of the header of function definition. These receiving variables are called the formal parameters of the function. It acts as a local variable inside the function in which they are used.

Returning value from Function:

In C++, the return keyword allows a function to return a value. When a function completes its execution, it can return a single value to calling function. Return data type must be specified with the function header in the function definition as well as function declaration. It is written before function name.

Syntax: int function name( );

Differentiate between function definition and function call

Function definition and function call can be differentiated on the basis of following criteria.

Function definition	Function call
The function definition is function itself. It has header and body with statements. Function definition may appears before or after the main ( )function.	Function call is to invoke the code of function by its name. As the function is called, control is transferred to the called function.

Syntax:
`data_type function_name (parameters list)`
`{`
`statements;`
`}`
Example
int sum(int p, int q)
{int z;
z= p + q;
return z;
}
	Syntax:
	`variable_name= function_name (parameter list);`
	Example
	a=sum (x, y);

Different Ways to Use User-Define Function: based on argument / parameters and return type

There are four different methods in C++ based on passing parameters to the function and return values from the functions.

- No return value and No passing arguments/parameters

  void function name**(void);**
- Return value but No passing arguments/parameters

  int/float/char function name**(void);**
- No return value but Passing arguments/parameters

  void function name**(int, float, char);**
- Return value and Passing arguments/parameters

  int/float/char function name**(int, float, char);**

Teachers Note	Teacher should explain how to write a User-define function in four different methods in C++.

Using the Pre – defined Function in C++ Program

```
//Using Square Root formula in program
#include<iostream.h>
#include<cmath.h>
using namespace std;
int main(void)
{

   int a;
   cout << "\n Enter the value of a.....";
   cin >> a;
   cout << "\n The square root of a is....." << sqrt(a); //
Pre-defined function
   return 0;
}
```

OUTPUT
Enter the value of a……4
The square root of a is….16

Applying the User-defined function in the program.

```
/* Program of an average of two numbers by using User-define
function */
#include<iostream.h>
using namespace std;

flaot average(int x, int y);
int main(void)
{
    int x,y;
    average(x,y);

return 0;

}
```

```
float average(int x, int y)
{
  cout << "\n \t Enter the value of x.........:";
  cin  >> x;

  cout << "\n \t Enter the value of y.........:";
  cin  >> y;

  avg = (x+y) / 2;

  cout << "\n The average of two numbers is……" << avg;
  return avg;

}
```

OUTPUT
Enter the value of x……..24
Enter the value of y……..34
The average of two number is…..29

Differentiate between Pre-define and User-define Function.

Pre – define function/System Define	User – define function
These are the library functions.	These functions are created by the programmer.
It cannot be modified.	It can be modified by the programmer.
No need for function definition as that is part of C++ compiler.	Their declaration and definition are needed in the program.
Example: gets(), putchar(), getch(), sqrt(), etc.	Example: add(), int sum(); float avg(float a, float b) etc.

Local variable and global variable

In structured programming, generally two types of variables are used.

1. Local Variables
2. Global Variables

1. Local Variables:

They are declared inside any function. A local variable is only accessible within a specific part of a program.

For Example:

```
#include<iostream.h>
using namespace std;
int main(void)
{
int a=10;  // Local Variable
cout << "\n The value of a……" << a;
return 0;
}
```

Teachers Note 	The variables declared in the header of the function definition are also known as Local variable.

2. Global Variable:

Global variables are declared outside of the main function. It is also known as **external variable.** It holds the value of variable during the entire execution of the program. The value of global variables can be shared with different functions.

For Example:

```
#include<iostream.h>
using namespace std;
void add(int c);
int a=20;  // Global Variable
int main(void)
{
      int x;    //Local Variable
      cin>>x;
      add(x);
      cout << "\n the value of a is……" << a;

 return 0;
 }

      void add (int c)
      {
            int b;   //Local Variable
            b=a+c;
            cout << "\n the addition of two numbers is…..”
      << b;
      }
```

Here 'a' is global variable and 'x' and 'b' are local variables. Receiving arguments are also local variable as 'c' variable in this program.

Teachers Note 	• Teacher are supposed to demonstrate practically about difference between local and global variables in C++. • At this point students should be able identify and remove errors from programs.

- A group of statements written to perform specific task is called Function.
- Function in C++ helps the programmer to manage the code of the large program.
- Functions are divided into three sections:
  1. Function declaration
  2. Function definition
  3. Function calling
- Function declaration tells the compiler about the function name, return types and parameters types.
- Function call is to invoke the code of function by its name.
- Functions are divided into two categories.
  - User – defined-function
  - Pre –defined – function
- A programmer can write his/her own function which is called User – defined function.
- In C++, Pre –defined – function are already declared in header files.

A. ENCIRCLE THE CORRECT ANSWER:

1. The functions that are defined by the programmer are called:
   a. Built-In function b. User-defined-function
   c. Subfunction d. Function
2. A programmer creates a function for a particular task and the programmer wants to include that function in program. Which extension is required to save that function?
   a. .obj b. .h
   c. .cpp d. .exe
3. In C++, int main( ) returns which data type value by default?
   a. float b. int
   c. char d. double
4. The parameters specified is the function header are called:
   a. formal parameters b. actual parameters
   c. default parameters d. command line parameters
5. The word "prototype" means:
   a. Declaration b. Calling
   c. Definition d. Both a & b
6. The function prototype consists of:
   a. Name of function
   b. the parameters are passed to the function
   c. The value return from function
   d. All of these
7. All variables declared in function definition are called
   a. Local variable b. Instance variable
   c. Global variable d. Static variable
8. Which are not the built-In function?
   a. sqrt() b. time()
   c. exp() d. sin()

B. RESPOND THE FOLLOWING:

1. Differentiate between function declaration and function definition.
2. What is the purpose of keyword "void" in function?
3. Why we use header files?
4. Differentiate between passing argument and return the value from function.
5. What is the difference between external variables and function local variables?
6. List the five standard built-In functions with examples.
7. Write down the advantages of User – define functions in C++.
8. Get the output and highlight the errors from the following program.

```
#include<iostream>
void Table(void);
void Table(void)
{
      int m,n;
      cout << "\n The value of m……";
      cin >> m;
      for(n=1; n<=10; ++n)
{
      cout <<"\t "<<m<<"*"<<n<<"="<<m*n<<"\n";
}
}
      void main(void)
{
      Table();
}
```

LAB ACTIVITIES

1. Write a program on the following given series by using For loop.

   0, 5, 10, 15, 20, 25, 30, 35

   Apply the technique (no return value and no pass parameters) in program.

2. Write a program to take input from the keyboard and check whether given number is Even or Odd. Apply the technique (return value and pass parameters) in program
3. Write a program to convert kilogram in grams using function. The function should take value in kilogram as parameter and should return value in grams.
4. Create a function that takes length and height as arguments and print a box of stars accordingly.
   e.g. length =10, height = 4
   ****
   ****
   ****
   ****
   Apply the technique (return value but not pass parameters) in program.
5. Write a function that returns factorial of a given number.
6. Develop programs for manipulating the following formulas in form of function.

Title	Formula	Description
Area of Rectangle	a= l b	area = length x width
Area of Circle	a= π * r^2	area = pi x radius x radius (pi = 3.14)
Pythagorean Theorem	$c^2 = a^2 + b^2$	*****