Function:

Introduction:

A function is a group of statements that is executed, when it is called from some point of the program.

A function is self contained block of statement written once in a program but can be used again & again.

A function is called or invoked by a function call. The function call specifies the function name and provides information (arguments) that the called function needs to do its job.

Functions allow the programmer to modularized a program. All variables declared in function definition are local variable. They are known only in the function in the way they are defined.

Function Definition:

A function definition contains name of the function with it return type, the formal arguments to be passed in the brackets if any and the body of the function which contains executable statements.

The syntax of function declaration is—

```
Function return type   function name ( data type arg1, data type arg2, ………………)

{

-------------------------------------------

-------------------------------------------

Body of the function

-------------------------------------------

-------------------------------------------

Return value

}
```

In above function declaration return type should be any fundamental data types like, int, float, char etc. In case the function does not return any value than the type of function will be taken as void.

Types of Functions:

Functions can be defined into two categories.

1. Standard Library Functions
2. User Defined Functions.

1. **Standard Library Functions:** Library functions are pre defined functions whose references are all ready given in C++ compiler. Library functions are also called built- in function. To use library functions in program, programmer need to include a particular header file related to that library function.
   For Example: If you want to use clrscr ( ) or getch ( ) library function in your program, then conio.h header file is required to include before main ( ) function. Otherwise compiler will give the prototype error.
2. **User Defined Functions:** User defined functions are those functions whose definitions are defined by the user itself and it can be called in the main function or any other function which defined by user.

General Form of Function

Generally user defined functions have three parts.

1. Function Prototype or Declaration
2. Function Calling.
3. Function Definition.

1. **Function Prototype:** Prototype of a function tells the compiler that this function will be used later in the program. It is compulsory to declare a function before its use.
   The syntax of function prototype is---

Return type	function name	(data type with parameter list);
any one fundamental data type	any valid identifier name	parameters with valid fundamental data types

   For Example:

   int area (int r);

2. **Function Calling:** To execute the definition of a function, we have to call that function in the main ( ) body of the program, which start with the function main ( ).

A function call consists of the function name followed by the argument or parameters which separated each by (,).

The syntax of function calling is ----------

Variable name = function name (argument name);

For example:

add = sum (a, b);

Here add is an integer type variable which will receive the sum of the variables a & b, which returns by function sum.

3. **<u>Function Definition:</u>** It is a block of statements surrounded by braces { }. That means all the statements are defined and execute when we call the function.

The syntax of function definition is---

```
return type          function name ( arguments with data type)
{
        Statement1;
        Statement2;
        _________;
        _________;
}
```

For example:

```
        int factorial ( int n)
        {
        int i, fact = 1;
        for ( i = n; i>0; i--)
        {
                Fact = fact * n;
        }
        return fact;
        }
```

- Parameters / Arguments: A parameter / argument is a piece of data that is passed into a function.

There are two types of parameter. (a) Formal parameter (b) Actual parameter.

(a) Formal Parameter: Formal parameters declared in the brackets in function definition after the function name.

(b) Actual Parameter: When any function being called from the main program, the parameters used in the calling are called the actual parameters, which are individually declared in the main program.

- **Important points related to Function:**
  1. Every C++ program contains at least one function, that is called main ( ) function.
  2. Function can be called any number of times.
  3. A function cannot be defined another function.
  4. Function will execute in the order they called in the main program.
  5. The default return type of a function is integer.
  6. Function prototype is not compulsory in C++. But if we do not declare any function then we have to write the definition of that function above the main ( ).
  7. Only a single value can be returned from a function using return statement at a time.

<u>Classification of Functions:</u>

Functions are classified in to 4 categories, on the basis of return types and parameters.

1. Return type and parameter.
2. No return type and no parameter.
3. Return type but no parameter.
4. No return type but parameter.

1. **Return type and Parameter:** This category of function is mostly used, because this category of function passes the parameter value and also return the value.

<u>Syntax:</u>

```
return type function_name          (data type of parameter);
```

For example:

```
int area ( int, int)
{   area= l * b;
    return area;    }
```

2. **No return type & No parameter:** No return no parameters, means we cannot pass any arguments to the function and definition of the function can’t return any value.

Syntax:

```
void function_name (void);
```

Definition of the function:

```
void function_name (void)
{   statement1;
    Statement2;
    _________;
    _________;          }
```

For example:

```
Void swap (void)
    { cout<<"A";          }
```

3. **Return type but no parameter:** In this type of function can return any type of value but at the time of calling we can't pass any argument/parameter to function definition.

   Syntax:

   ```
   return type        function_name (void);
   ```

   For Example:

   ```
   int   display (void);
   ```

   Definition:

   ```
   int  display (void)
   {   int  n=10;
       Cout<<"The Value of n="<<n;
   return n;            }
   ```

4. **No return type but parameter:** In no return but parameter type function, we pass arguments to the function definition but definition of function can't return any value.

   Syntax:

   ```
   void          function_name (data type of arguments, ---------);
   ```

   Definition of function:

   ```
   void          function_name(arguments name with data type)
   {   statement1;
       Statement2;
       _________;
       _________;          }
   ```

   For Example:

   ```
   void display (int);
   ```

- **<u>Global and Local Scope:</u>** Any variable which defined outside the main ( ) body of the program is said to be global scope. We can defined variable as well as function as a global scope.

- **For Example:**
```
int a = 10;              // Global variable.
int sum ( int, int);     // Global function.
{   ------------
    -------------        }
```
Global variables are used in the program with the help of scope resolution (: :) operator. Global entities are visible at the program level, so they must be unique at the program level. That means global variable or function may not be defined more than once at the global level.
- **Local Variable:** The local variables are defined inside the function body or in a compound statement other than the formal arguments. The scope of such variables is inside the function where they are defined. They had no identity outside that function identifiers declared as label, constant, type , variables and even function defined inside a block are said to be local variables/ parameters of that particular block.
The example of local variable declaration is------
```
long int fact ( int n )
    { int I, fact, j;        // fact is the local variable.
            ________________;
            ________________;
            Function body;
            ________________;
            ________________;          }
```

- **Recursion:** A recursion function is a function which calls itself either directly or indirectly through another function. The recursive function includes the keyword return which is used to pass back the result to the original caller function possibly the main( ) function. The recursive functions are very useful while constructing the data structures like linked lists, double linked lists and trees.

To find the factorial of a no. is best example of recursion.
<u>For Example:</u>

```
int fact ( int n)
    {       return n = 0 ? 1 : n * fact ( n – 1 );
    }
```

A recursive function must have at least one termination condition which can be satisfied, otherwise the function will call itself indefinitely until the runtime stack overflow.

- **Inline function:** Inline function is used when a programmer required to fined the absolute value of an integer quantity quickly. Inline function execute first compare to normal function.

  Inline function has a number of advantages.

1. It leads to a more readable program.
2. It is reusable.
3. It avoid undesirable side effect.

The general syntax of Inline function is---------

```
inline   return type   function_name ( data type of arguments………)
{        statement1;
         Statement2;
         …………..;
         ……………;          }
```

The calling of inline function is same, as we call the other function. We don't have to include the inline keyword when calling the function.

The example of inline function is------

```
inline int  ABS (int n)
    {       return (n>0 ? n : - n);
            }
```

The effect of this is that, when ABS is called, the compiler, instead of generating code to call ABS function, expands and substitutes the body of ABS in place of the call.