


# Unit : 4

## Functions

# Functions

- A function is a group of statements that together perform a task. Every C program has at least one function, which is **main()**, and all the most trivial programs can define additional functions.
- You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division is such that each function performs a specific task

- return type function\_name([parameters])
- {  
    body of function  
    return values
- }

- 
- A function **declaration** tells the compiler about a function's name, return type, and parameters. A function **definition** provides the actual body of the function.
  - A function can also be referred as a method or a subroutine or a procedure, etc.

# Type of Functions

- User Define Functions ( UDF )
- In Built Functions

# User Define Functions

- When a programmer want to create new own block of code its known as user define functions.
- Types of UDF is
  - No parameters no return value
  - No parameters with return value
  - With parameters no return
  - With parameters with return

# Syntax of function

```
return_type function_name( parameter list )  
{  
    body of the function  
}
```

- **Return Type** – A function may return a value. The **return\_type** is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the **return\_type** is the keyword **void**.
- **Function Name** – This is the actual name of the function. The function name and the parameter list together constitute the function signature.



- **Parameters** – A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.
- **Function Body** – The function body contains a collection of statements that define what the function does.

# Main parts of functions

- **Function Declarations**
- **Calling a Function**
- **Function definition**

# Function declaration

- A function **declaration** tells the compiler about a function name and how to call the function. The actual body of the function can be defined separately.
- `return_type function_name( parameter list );`
- Ex `void max(int l, int j);`

# Function calling

- While creating a C function, you give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task.
- Ex `max(10,16);`

# Function definition

```
Void max(int I, int j)
{
    If(i>j)
    {
        printf("%d is greater ",i);
    }
    else
    {
        printf("%d is greater ",j);
    }
}
```

# Built in functions

- There is a rich collection of built in functions in c environment.
- This function divided in several header files like. string, ctype, math, dos, stdlib and etc..

# String functions

- `strlen()`      Calculates the length of string
- `strcpy()`      Copies a string to another string
- `strcat()`      Concatenates(joins) two strings
- `strcmp()`      Compares two string
- `strlwr()`      Converts string to lowercase
- `strupr()`      Converts string to uppercase

# Character functions

- `int isalpha(c);`  
`c` is a letter.
- `int isupper(c);`  
`c` is an upper case letter.
- `int islower(c);`  
`c` is a lower case letter.
- `int isdigit(c);`  
`c` is a digit [0-9].
- `int isalnum(c);`  
`c` is an alphanumeric character  
(`c` is a letter or a digit);
- `int isspace(c);`  
`c` is a SPACE,
- `int ispunct(c);`  
`c` is a punctuation character  
(neither control nor alphanumeric).



- `getch()` get single character from keyboard but not print it.
- `getche()` get single character from key board and print while scanning.

# Math functions

- `ceil()` Returns nearest integer greater than argument passed.
- `exp()` Computes the e raised to given power.
- `floor()` Returns nearest integer lower than the argument passed.
- `pow()` Computes the number raised to given power.
- `sqrt()` Computes square root of the argument.

# User defined functions

- **Benefits of Using Functions**

1. It provides modularity to the program.
2. Easy code Re-useability. You just have to call the function by its name to use it.
3. In case of large programs with thousands of code lines, debugging and editing becomes easier if you use functions.

# Function syntax

```
return-type function-name (parameter-list)  
{  
    function-body ;  
}
```