UNIT: 5
Functions

❖ Introduction to Functions:

Definition of Function: Function is a group of statement to perform a specific task.
Types of functions:
1. Library function
2. User-defined Function

❖ Advantages of using Functions:

1. By using functions, we can avoid rewriting same logic/code again and again in a program.
2. We can call C functions any number of times in a program and from any place in a program.
3. We can track a large C program easily when it is divided into multiple functions.

❖ Types of Functions: Built-in and user defined Functions

Library function(Built-in):
- The function which is develop by system itself is known as Library function .
- Example: printf(),scanf(),sqrt(),ceil() etc …
- Library function is all ready exist in system.

User-defined Function:
- The function which is develop by user itself is known as User-defined function.
- Example: main()
- User-defined function is created by the user at the time of writing of program.

Difference between Library function and User-defined Function.

No.	Library function	User-defined Function
1	The function which is develop by system itself is known as Library function.	The function which is develop by user itself is known as User-defined function.
2.	Example: printf(),scanf(),sqrt() etc …	Example: main(),add()
3.	There no elements in Library function.	User-defined function has three elements.

❖ Declaring, Defining and calling user defined Functions

(1) Function declaration

Syntax:

ReturnType FunctionName (Parameter List);

- Function Type is a return type or datatype of a function.
- Function name is the name of the user defined function.

Example:

```
int sum (int, int);
int sum( int a, int b);
```

(2) Function call

Syntax:

Identifier = Function name(Argument value);

Where identifier is the name of variable

Example:

```
a = sum(10,5);
```

(3) Function definition

Syntax:

```
ReturnType  Function Name(Parameter List)
{
Local variable declaration
Function statement
Return Statement
}
```

Example:

```
int sum(int x, int y)
{
int total;
total = x + y;
return total;
}
```

❖ Categories of user-defined Functions

A function may or may not accept any argument. It may or may not return any value. Based on these facts, There are four different categories of function calls.

- function without arguments and without return value
- function without arguments and with return value
- function with arguments and without return value

- function with arguments and with return value

Example for Function without argument and return value

```
#include<stdio.h>
void sum();
void main()
{
   printf("\n sum of two numbers:");
   sum();
}
void sum()
{
   int a,b;
   printf("\nEnter two numbers");
   scanf("%d %d",&a,&b);
   printf("The sum is %d",a+b);
}
```

Example for Function without argument and with return value

```
#include<stdio.h>
int sum();
void main()
{
   int result;
   result = sum();
   printf("%d",result);
}
int sum()
{
   int a,b;
   printf("\nEnter two numbers");
   scanf("%d %d",&a,&b);
   return a+b;
}
```

Example for Function with argument and without return value

```
#include<stdio.h>
void sum(int, int);
void main()
{
   int a,b,result;
   printf("\nEnter two numbers:");
   scanf("%d %d",&a,&b);
   sum(a,b);
}
void sum(int a, int b)
{
   printf("\nThe sum is %d",a+b);
}
```

Example for Function with argument and with return value

```
#include<stdio.h>
int sum(int, int);
void main()
{
   int a,b,result;
   printf("\nEnter two numbers:");
   scanf("%d %d",&a,&b);
   result = sum(a,b);
   printf("\nThe sum is : %d",result);
}
int sum(int a, int b)
{
   return a+b;
}
```

❖ Call by Value and call by Reference

Call By Value with example:

- When a function is called using the value of variables, then it is known as call by value.

- The value of variables, which are to be passed, will be copied from the variables of the calling functions to the variables of the called functions.
- All the process done on the duplicate variables rather then actual variables.

Example:

```
#include <stdio.h>
void swap(int , int);     // function  prototype
int main()
{
   int a = 10;
   int b = 20;
   printf("Before swapping the values in main a = %d, b = %d\n",a,b);
   swap(a,b);    // function  call
printf("After swapping values in main a = %d, b = %d\n",a,b); //actual parameters a=10, b=20
}
}
void swap (int a, int b)
{
   int temp;
   temp = a;
   a=b;
   b=temp;
   printf("After swapping values in function a = %d, b = %d\n",a,b); // Formal parameters, a=20, b=10
}
```

Call By Reference with example:

- When a function is called using the address of variables, then it is known as call by reference.
- Instead of passing the value of variables from calling function to the called function, addresses of the variables are passed.
- All the process done on the actual variables.

Example:

```
#include <stdio.h>
void swap(int *, int *);          // function  prototype
 void  main()
{
   int a = 10;
   int b = 20;
   printf("Before swapping the values in main a = %d, b = %d\n",a,b);
```

```
    swap(&a,&b);          // function  call
     printf("After swapping values in main a = %d, b = %d\n",a,b); // actual parameters a = 20, b = 10
}
void swap (int *a, int *b)
{
   int temp;
   temp = *a;
   *a=*b;
   *b=temp;
   printf("After swapping values in function a = %d, b = %d\n",*a,*b); // Formal parameters a=20, b=10
}
```

❖ Recursion

- Function call itself is called Recursion.
- Recursive call to a function may be conditional or unconditional.
- In conditional recursive call, function is terminated when condition is false.
- In unconditional recursive, function is called infinitely.
- It is used to find the factorial of a given number.
- Example :

```
void main()
{
        printf("Recursion");
        main();
}
```

- Example :

```
void main()
{
        ----------------
        function1();
        ---------------
}
function1()
{
        ------------------
        function1();
        ---------------
}
```

Advantages of Recursion:

- The recursion is very flexible in data structure like stacks.
- Using recursion, the length of the program can be reduced.

Disadvantages of Recursion:

- It requires extra storage space.
- The recursion function is not efficient in execution speed and time.
- Proper termination is required; otherwise it leads to infinite loop.

❖ Built-in Functions: String and Maths

Built-in Functions:

C Maths Function:

The <math.h> header file contains various methods for performing mathematical operations such as sqrt(), pow(), ceil(), floor() etc.

ceil(number)

Rounds up the given number.
It returns the integer value which is greater than or equal to given number.

Ex: ceil(3.6)
It returns 4.

floor(number)

Rounds down the given number. It returns the integer value which is less than or equal to given number.

Ex: floor(3.6)
It returns 3.

sqrt(number)

Returns the square root of given number.

Ex: sqrt(16)
It returns 4.

pow(base, exponent)

Returns the power of given number.

Ex pow(2,4)
It returns 16.

abs(number)
Returns the absolute value of given number.
Ex: abs(-16)
It returns 16.

C String Functions:
There are many important string functions defined in "string.h" library.

No.	Function	Meaning
1)	strlen(string_name)	returns the length of string name.
2)	strcpy(destination, source)	copies the contents of source string to destination string.
3)	strcat(first_string, second_string)	concats or joins first string with second string. The result of the string is stored in first string.
4)	strcmp(first_string, second_string)	compares the first string with second string. If both strings are same, it returns 0.
5)	strrev(string)	returns reverse string.
6)	strlwr(string)	returns string characters in lowercase.
7)	strupr(string)	returns string characters in uppercase.

❖ Storage Classes: -auto, static, register and extern

Storage Classes in C:

Storage classes in C are used to determine the lifetime, visibility, memory location, and initial value of a variable. There are four types of storage classes in C

- Automatic
- External
- Static
- Register

Storage Classes	Storage Place	Default Value	Scope	Lifetime
auto	RAM	Garbage Value	Local	Within function
extern	RAM	Zero	Global	Till the end of the main program Maybe declared anywhere in the program
static	RAM	Zero	Local	Till the end of the main program, Retains value between multiple functions call
register	Register	Garbage Value	Local	Within the function