Functions in C

The function is a self contained block of statements which performs a coherent task of a same kind.
C program does not execute the functions directly. It is required to invoke or call that functions. When a function is called in a program then program control goes to the function body. Then, it executes the statements which are involved in a function body. Therefore, it is possible to call fuction whenever we want to process that functions statements.

Advantages :

- It is easy to use.
- Debugging is more suitable for programs.
- It reduces the size of a program.
- It is easy to understand the actual logic of a program.
- Highly suited in case of large programs.
- By using functions in a program, it is possible to construct modular and structured programs.
- It modularizes the software or program

Types of functions :
There are 2(two) types of functions as:

1. Built in Functions
2. User Defined Functions

1. Built in Functions :
These functions are also called as 'library functions'. These functions are provided by system. These functions are stored in library files. e.g.

- scanf()
- printf()
- strcpy
- strlwr
- strcmp
- strlen
- strcat

2. User Defined Functions :
The functions which are created by user for program are known as 'User defined functions'.

Two ways of passing parameter is:
Call by value(without using pointers and Call by reference (using pointers)

Syntax:

```
// Function prototype
        <return_type><function_name>([<argu_list>]);

void main()
{

        // Function Call
        <function_name>([<arguments>]);
}

// Function definition
<return_type><function_name>([<argu_list>]);
```

```
{
        <function_body>;
}
```

```
#include <stdio.h>
#include <conio.h>

void main()
{
        int a, b, c;
        clrscr();
        printf("\n Enter Any 2 Numbers : ");
        scanf("%d %d",&a,&b);
        c = a + b;
        printf("\n Addition is : %d",c);
        getch();
}
```

Program :

```
#include <stdio.h>
#include <conio.h>

void add();

void main()
{
        add();
        getch();
}

void add()
{
        int a, b, c;
        clrscr();
        printf("\n Enter Any 2 Numbers : ");
        scanf("%d %d",&a,&b);
        c = a + b;
        printf("\n Addition is : %d",c);
}
```

Output :

```
 Enter Any 2 Numbers : 23 6
 Addition is : 29_
```

Difference between user defined function and library function

The user-defined functions are defined by a user as per its own requirement and library functions come with compiler.

Library Function	User Defined Functions
• LF(library functions) are Predefined functions.	• UDF(user defined functions) are the function which r created by user as per his own requirements.
• LF are part of header file (such as MATH.h) which is called runtime.	• UDF are part of the program which compile runtime
• in LF it is given by developers.	• In UDF the name of function id decided by user
• LF Name of function can't be changed.	• in UDF name of function can be changed any time
Example : SIN, COS, Power	*Example : fibo, mergeme*

Example:1

```
#include <stdio.h>

int change(int number);              /*Function prototype */

int main(void)
{
  int number = 10;
  int result = 0;

  result = change(number);
  printf("\nIn main, result = %d\tnumber = %d", result, number);
  return 0;
}

int change(int number)
{
  number = 2 * number;
  printf("\nIn function change, number = %d\n", number);
  return number;
}
```

```
In function change, number = 20
        In main, result = 20    number = 10
```

Function calls function

```
#include<stdio.h>

f1 (void)
{
   printf ("f1-1 \n");
   f2 ( );
   printf ("f1-2 \n");
}
f2 ()
{
   printf ("f2 \n");
}
main ( )
{
   printf ("1 \n");
   f1 ( );
   printf ("2 \n");
}
```

```
1
f1-1
f2
f1-2
2
```

Write a c program to define a function which adds two integer numbers and returns the result to the main function

```
#include <stdio.h>

int add (int x, int y) {
   int z;
   z = x + y;
   return (z);
}
main ()
{
   int i, j, k;
   i = 10;
   j = 20;
   k = add(i, j);
   printf ("The value of k is %d\n", k);
}
```

```
The value of k is 30
```

Recursion

Formal definitions of recursion

In mathematics and computer science, a class of objects or methods exhibit recursive behavior when they can be defined by two properties:

1. A simple base case (or cases), and
2. A set of rules which reduce all other cases toward the base case.

For example, the following is a recursive definition of a person's ancestors:

- One's parents are one's ancestors (*base case*).
- The parents of one's ancestors are also one's ancestors (*recursion step*).

The Fibonacci sequence is a classic example of recursion:

- Fib(0) is 0 [base case]
- Fib(1) is 1 [base case]
- For all integers n > 1: Fib(n) is (Fib(n-1) + Fib(n-2)) [recursive definition]

Characteristics of recursion function

1. A recursive function is a function that calls itself.
2. The speed of a recursive program is slower because of stack overheads.
3. In recursive function we need to specify recursive conditions, terminating conditions, and recursive expressions
4. Recursion works on stack i.e, first in last out.
5. Recursion is a process of calling itself with different parameters until a base condition is achieved. Stack overflow occurs when too many recursive calls are performed.

Recursion in a screen recording program, where the smaller window contains a snapshot of the entire screen.

Recursive Function Definition

Recursive function is a special function that contains a call to itself. C supports creating recursive function with ease and effecient.

Why Recursive Function

Recursive function allows you to divide the complex problem into identical single simple cases which can handle easily. This is also a well-known computer programming technique called ***divide and conquer***.

Note of Using Recursive Function

Recursive function must have at least one exit condition that can be satisfied. Otherwise, the recursive function will call itself repeatly until the runtime stack overflows.

Example of Using C Recursive Function

Recursive function is closely related to definitions of functions in mathematics so we can solving factorial problems using recursive function.

All you know in mathematics the factorial of a positive integer N is defined as follows:

N! = N*(N-1)*(N-2)…2*1;

Or in a recursive way:

N! = 1 if N <=1 and N*(N-1)! if N > 1

The C recursive function to calculate factorial:

```
# include<stdio.h>


int factorial(unsigned int number)
{
   if(number <= 1)
      return 1;
   return number * factorial(number - 1);
}
void main()
{
```

```
    int x = 5;
    printf("factorial of %d is %d",x,factorial(x));
 }
```

The output of program
factorial of 5 is 120

Write a recursive function to find sum of digits of any number input through keyboard.

```
main()
{
int s, n;
printf("\nEnter any number:");
scanf("%d",&n);
s =sum(n);
printf("\n Sum of digits = %d",s);
}

sum(int n)
{
if(n<10)
return(n); else
return(n %10 + sum(n/10)) ;
}
```

Program to generate Fibonacci series using recursion in c

```
#include<stdio.h>

void printFibonacci(int);
int main(){
   int k,n;
   long int i=0,j=1,f;
   printf("Enter the range of the Fibonacci series: ");
   scanf("%d",&n);
   printf("Fibonacci Series: ");
   printf("%d %d ",0,1);
   printFibonacci(n);
   return 0;
}

void printFibonacci(int n){
   static long int first=0,second=1,sum;
   if(n>2){  //because two terms of the series are already printed
       sum = first + second;
       first = second;
       second = sum;
       printf("%ld ",sum);
       printFibonacci(n-1);
   }

}
```

Sample output:

Enter the range of the Fibonacci series: 10
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34 55 89

C code to print Fibonacci series without recursion:

```
#include<stdio.h>

void printFibonacci(int);

int main(){

   int k,n;
   long int i=0,j=1,f;

   printf("Enter the range of the Fibonacci series: ");
   scanf("%d %d ",&n);

   printf("Fibonacci Series: ");
   printf("%d ",0);
   printFibonacci(n);

   return 0;
}

void printFibonacci(int n){

   long int first=0,second=1,sum;

   while(n>0){
       sum = first + second;
       first = second;
       second = sum;
       printf("%ld ",sum);
       n--;
   }
}
```

Program to print string Vertically from given string.

```
#include
#include
void main(){
 int i;
 char string[5]="HELLO";
 clrscr();
 for(i=0;i<5;i++)
{
printf("\n %c",string[i]);
}
```

```
getch();
}
```

Working on array of strings

- Here we have two indexes/subscripts. Normally the two indexes refer to the **rows** and **columns**, that is the [6] refers to rows and [10] refers to columns. If we assign initial string values for the 2D array it will look something like the following.
  char Name[6][10] = {"Mr. Bean", "Mr. Bush", "Nicole", "Kidman", "Arnold", "Jodie"};
- Here, we can initialize the array with 6 strings, each with maximum 9 characters long. If depicted in rows and columns it will look something like the following and can be considered as contiguous arrangement in the memory.