Function

Functions is a set of statements. A function is a block of Code, that contains the set of Programming Statements enclosed by { } . C Program consider as Set of functions. We are using one function in every C Program name is main().

Advantages of function

- Easy to debug.
- Easy to manage.
- It removes the repetition.
- It provides the Clarity to the Program.
- To time save.
- Function decrease the complexity of the Program..
- It makes the programming simple. Function has the modular approach.

Defining a function

The General form of a function is a given below

```
<return type><function name>(<parameter list>)
{
        statement 1;
        statement 2;
        statement n;
        return(expression) ;
}
```

A Function Example

Example

```
main()
{
message1();       //Calling the function
message2();
}
message1()       //function defination
{
Printf ("C is a Programming Language");
}
message2()
{
Printf ("C++ is Object Oriented Language");
 }
```

Function Call or Calling a Function

Every function which is defined in the Program needs to be called. A function can be called by Specifying its name, followed by a list of arguments enclosed in Parentheses and separated by commas.

The General form of function call

fun (a,b);

↙ ↘

Name of the function — are the actual arguments with in parentheses

- There will be no return value in this function call and it ends with ;

(Example same as function defining ↑)

Actual and formal Arguments

Actual Arguments which are passed in calling function or while calling a function is called actual arguments (Passed to a function)

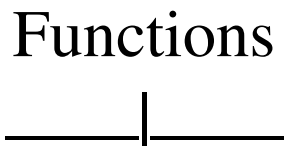
Formal Arguments which are defined in function defined are called formal Arguments (received by function)

Example

```
main ()
{
sum(a,b);                // Actual Parameters
}
int sum (intx, inty)      //Formal parameter
{
return (x+y);
}
```

Function Categories

There are two types of functions in c.



Library Function

User defined Function

Library function : are also called Inbuilt Function or system defined function library function are those functions whose meaning is already defined. The definition of such functions is stored in header files

Examples: printf(), scanf(), main() etc

User-defined functions :which are created by the user to perform the desired task we can design user defined functions according to our need

Examples: sum(), add(), message() etc.

Function Prototype

A function Prototype is simply the declaration of a function that specifies function's name, Parameters and return type. It doesn't contain function body.

Syntax<type><name > (type 1 arguments, type 2 arguments);

↓ ↓ ↓

Return type Name of Function List of Parameters

Example int sum (int a, int b);

- Where int is the return type of function.
- sum() is name of the function.
- Two arguments of type int are passed to the function.

The semicolon marks the end of Prototype declaration.

Passing arguments to function

Arguments are passed from calling function to called the function.
You can call the function or pass the arguments in two different ways

1. Call by Value
2. Call by reference

Call by Value: - In this type of Parameter Passing only the values are passed from the calling function to called function Therefore this technique is accordingly called as Call by value .

Program to show call by value method. (Swap two number)

```
#include <stdio.h>
#include <conio.h>
void main()
{
void swap (int a, int b);          //function declaration
Int a,b;
clrscr();
printf ("Enter two numbers ");
scanf (“%d%d",&a,&b) swap (a,b);
swap(a,b);
printf(“a=%d b=%d”,a,b);
getch();
}
void swap (int a, int b)      // function definition
{
int temp;
temp=a;
a=b;
b=temp;
printf(“a=%d b=%d\n”,a,b);
}
```

2. Call by Reference : - In this method, the address of actual arguments is passed, instead the creating the copy of actual arguments

- When we call a function by passing the address of actual parameters then the way of calling the function is known as call by reference

Program using Call by Reference (swap two Number)

```
#include <stdio.h>
#include<conio.h>
void main()
{
void swap (int *, int *);
int a, b;
clrscr();
printf ("enter two number");
scanf ("%d%d",&a,&b);
printf (" In before swapping a=%d b=%d", a,b);
swap (&a, &b);
printf ("In After swapping a=%d b=%d",a,b);
getch();
}
void swap (int*a, int*b);
{
int temp;
temp=*a;
*a = *b;
*b= temp;
}
```

Return Statement

The return statement is used to terminate the execution a function and transfer Program Control back to the calling function.

A function can return a value of any type, using the return statement

Syntax: return (exp);

→ where exp can be a constant, variable or an expression

→ Parentheses around exp are optional

Example

```
big (inta, intb)
{
if (a>b)
return (a);
else
return (b) ;
}
```

Recursion - Recursion is derived from the word recur which means repetition. Recursion is a process in which a function calls itself again and again is called recursion and function is called recursive function. In C It is Possible for the function to call itself.

Syntax

```
fun ()
{
 ----
fun();
}
```

Requirements for recursion:

1) The function must call itself again and again.
2) The function must have an exit Condition without this Condition, it will work in endless manner.

Program to compute factorial of a number using Recursion.

```
#include <stdio.h>
# include <conio.h>
void main()
{
int fact (int num);
int num;
clrscr();
printf ("enter number");
scanf ("%d" &num);
printf ("In factorial is = %d", fact (num));
getch();
}
fact (int n)
{
if (n==1)
return (1);
else
return (n* fact (n-1));
}
```

Advantages of Recursion

- Some Problems are easier to solve using recursion.
- Reduce unnecessary calling of function.
- Reduce the length of Code.
- It is very useful when applying the same Solution

Example of function

(Write a Program to calculate the sum of three given numbers by using a function.)

```
#include<stdio.h>
#include<conio.h>
void main()
{
int calsum (int, int,int);
int n1, n2, n3, sum;
clrscr();
printf ("In Enter three numbers ");
scanf ("%d%d%d", &n1, &n2, &n3);
sum= calsum (n1, n2, n3);
printf ("In sum is = %d", sum);
getch();
}
int calsum (int a, int b, int c)
{
int s;
s=a+b+c;
return(s);
}
```