Unit 5 Functions

Structure:

5.1 Introduction

The mathematical concept of function expresses dependence between two quantities, one of which is given (the independent variable, argument of the function, or its "input") and the other produced (the dependent variable, value of the function, or "output"). A function associates a single output to each input element drawn from a fixed set, such as the real numbers.

All the instructions of a C program are contained in functions. Each function performs a certain task. A special function name is main( ): the function with this name is the first one to run when the program starts. All other functions are subroutines of the main( ) function (or otherwise dependent procedures, such as call-back functions), and can have any names you wish.

Every function is defined exactly once. A program can declare and call a function as many times as necessary.

Objectives:

At the end of this unit, you will be able to:

- provide an overview of functions.
- explain the library functions of Language C.
- write user defined functions for a specific task.
- explain recursive functions and their operation.

5.2 Functions

Function is a self-contained block of statements that perform a coherent task of some kind. Every 'C' program can be thought of as a collection of functions. In other words, we may say that a function is the collection of expressions that can be called by repeated number of times. There are some conventions to declare and define a function.

1) Any 'C' program contains at least one function, it must be **main ()**.
2) There is no limit in the number of functions that might be present in a 'C' program.
3) A function is defined when function name is followed by a pair of braces in which one or more statement may be statement.
4) A function can be called any number of times.
5) A function can call itself. Such a process is called **'recursion'.**
6) The function should have some return type.
7) A function may have number of arguments but only one return type.

There are basically two types of functions:

a) User defined function
b) Library Function

5.2.1 User Defined Function

The function defined by user is called user defined function. All these functions are usually defined for local purpose within the program. These set of functions can also be included in the own header file for general use.

Before defining a function, the function prototype should be defined. It is compulsory if return type is other than integer and for integer return it may be ignored. The prototype declaration is the prior declaration to the complier so that compiler goes through the function properly.

There are **three (3)** steps that must be written when using any user defined function.

a) Function Prototype Declaration
b) Function Calling
c) Function Body Definition

a) **Function Prototype Declaration:**

It is the prior declaration to the compiler of function. If the Function Body Definition is done first, then Prototype Declaration is optional.

Syntax:

<return type > **function_name** (arguments);

Example: float **addition** (float,float);

Explanation: The function addition has two arguments of floating point data types and return one floating point data type.

If there is no return type then we write **void.**

b) Function Calling:

The function calling means that the function is called by other functions. Depending on the definition of the function, the function may be called with or without argument. The function may or may not return a value. According

to this, the method of calling a function varies. Four different methods of calling a function are given below:

i) No Argument No Return Value

Example: under_line(); /*Say, to draw a line */

ii) No Argument With Return Value

Example: b=pi(); /*pi() returns value to b */

iii) Argument With No Return Value

Example: under_line(60); /* Say, to draw a line for 60 same characters */

iv) Argument With Return Value

Example: c= add(a,b); /*Say, add two numbers and store to c*/

c) Function Body Declaration:

The body of a function contains one or more statements that define the purpose of the function.

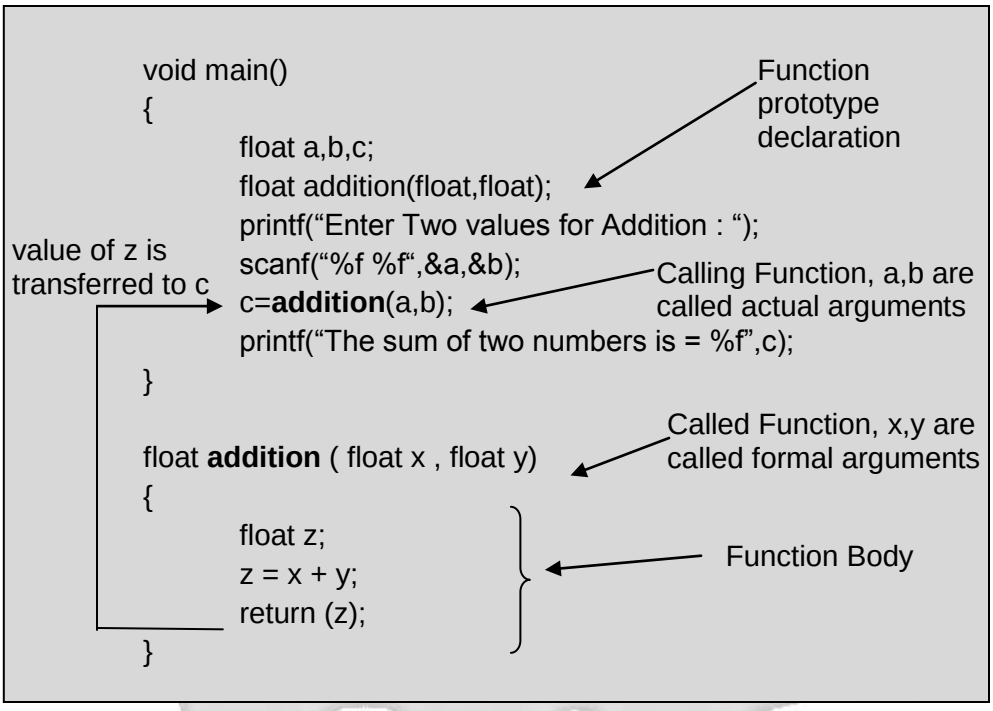
Syntax:

```
<return type> function_name ( arguments)
{
  Body of the function;
}
```

- Function for addition of two numbers

```
float addition( float a, float b)
{
      float c;
         c = a+b;
         return(c);
}
```

Now it is the complete program for addition of two numbers using function.



Example 1:

1) Write a program to print the PASCAL triangle after inputting the limit value.[File Name: u5q1.c]

Solution:

```
/*****************************************************************/
/*      Program to print the PASCAL triangle                     */
/*                                                               */
/*****************************************************************/

void main()
{
int n,i,j,a;
int fact(int);              /* Function Prototype Declaration*/
clrscr();
printf("Enter the value[max 7] ");  scanf("%d",&n);
```

```
for(i=0;i<=n;i++)
{
  for(j=0;j<=i;j++)
   {
    a=fact(i)/(fact(i-j)*fact(j));   /* Function Calling */
    printf("%5d",a);                 /* using formula nCr =n!/(n-r)! r! */
   }
   printf("\n");
}
getch();   /* For halting the output*/
}
/*****************************************************************/
int fact(int a)                                  /*Function Definition */
{
int f=1,k;
for(k=a;k>=1;k--)
   f=f*k;
return(f);
}
```

2) Write a program to check whether the given number is prime or not.

[File Name: u5q2.c]

Solution:

```
/* Program to check prime number */
void main()
{
int i,j,n,count;
prime(int);                    /* Function prototype Declaration */
clrscr();
printf("Enter the limit : ");
```

```
scanf("%d",&n);                /* Enter the limit */
if(prime(n))
 printf("The number  %d is prime ",n);
else
 printf("The number %d is not prime",n);

getch();
}                                  /* end of main() function */
/********************************* ******************************************/
int prime(int n)                   /* The prime() Function Definition*/
{
int i,count=0;
 if(n<=1) count=1;
 for(i=2;i<n;i++)
  {
   if(n%i==0)
     {count++; break;}
  }
  return(!count);
}
```

3) Write a program to show the calendar after inputting the year value. (File Name: u5q3.c)

Solution:

```
/*********************************************************************/
/* Program to Print the Calendar between years (1900 - 3000)  */
/*********************************************************************/
#include<stdio.h>
#include<string.h>
```

```
void main()
{
int year,i,j,k,d_month,yr,x=10,y=5;
long total_days=0,days;
int isleap_year(int);
void heading(void);
clrscr();
printf("Enter the year between ( 1900 - 3000 )  :  "); scanf("%d",&year);
if(year<1900 || year>3000)
{
       printf("Invalid Input");  getch(); main();
 }  /* return to main*/
clrscr();
for(yr=1900;yr<year;yr++)
{
  if(isleap_year(yr))
      days=366;
  else
     days=365;
total_days=total_days+days;
}    /*  calculate for total number of days   */
x=(int)(total_days%7);    /* type casting */
x=18+x*8;
```

```
for(i=1;i<=12;i++)
{
 y=8;clrscr();
  switch(i)
   {
   case 1:
     printf("\n\n\n\t\t\t JANUARY     -    %d   ",year );
     d_month=31;
     break;
   case 2:
     printf("\n\n\n\t\t\t FEBRUARY    -    %d   ",year);
     if(isleap_year(year))
           d_month=29;
      else
           d_month=28;
      break;
    case 3:
      printf("\n\n\n\t\t\t MARCH       -    %d   ",year);
     d_month=31;
     break;
    case 4:
      printf("\n\n\n\t\t\t APRIL       -    %d   ",year);
     d_month=30;
```

```
 break;
case 5:
  printf("\n\n\n\t\t\t MAY       -   %d   ",year);
 d_month=31;
 break;
case 6:
  printf("\n\n\n\t\t\t JUNE      -   %d   ",year);
 d_month=30;
 break;
case 7:
  printf("\n\n\n\t\t\t JULY      -   %d   ",year);
 d_month=31;
 break;
case 8:
  printf("\n\n\n\t\t\t AUGUST    -   %d   ",year);
 d_month=31;
 break;
case 9:
  printf("\n\n\n\t\t\t SEPTEMBER  -   %d   ",year);
 d_month=30;
 break;
case 10:
  printf("\n\n\n\t\t\t OCTOBER    -   %d   ",year);
```

```
    d_month=31;
    break;
   case 11:
     printf("\n\n\n\t\t\t NOVEMBER   -   %d   ",year);
    d_month=30;
    break;
   case 12:
     printf("\n\n\n\t\t\t DECEMBER   -   %d   ",year);
    d_month=31;
    break;
  }
    heading();
     for(k=1;k<=d_month;k++)
    {
     gotoxy(x,y); printf("%2d",k);   x+=8;
          if(x>64) { x=10; y++; }
    }
   printf("\n
==================================================\n");
    getch();
  }
}
```

```
void heading()                          /* For Heading Printing*/
{
printf("\n\n      SUN    MON    TUE    WED    THU    FRI    SAT");
printf("\n
================================================\n");
}
int isleap_year(int year)               /* checking for leap year */
{
return(year%4==0 && year%100 !=0 || year%400==0);
}
```

5.2.2 Library Functions

The library functions are common required functions grouped together and stored in files called library. The library functions are present on disk in the most of the compilers. According to the compiler version the library function varies. Here the discussion is made over the Turbo C compiler version 3.0. There are 101 header files in the compiler. Some header files and some functions are given below which are commonly used. Before we use any library function, we must open the respective header file.

Syntax for file inclusion:

#include<header file name>

OR

#include “header file name”

The basic difference between two syntaxes are that first one can search files from the current directory and its subdirectory but the second one can search file from any location of the disk, if path is given.

Example 2:

(1) #include<stdio.h>

(2) #include “d:\user\myheader.h”

We can also create header files and can include functions into it.

Some Useful Library Functions

1) String Functions: <String.h>

strlen(), strcpy(),strcat(), strcmp(), strcmpi(),strupr(), strlwr(),strrev().

2) Mathematical Functions: <math.h>

abs(), fabs(), atof(), atoi(), atol(), sin(), cos(), tan(), exp(), floor(), ceil(), sqrt(), pow(),log(), log10(),fmod().

3) Character Type Functions: <ctype.h>

isalpha(), isdigit(), islower(), isspace(), toupper(), tolower().

4) Color Input Output Functions: <conio.h>

clrscr(), cprintf(), cputs(), cscanf(), getch(), getche(), gotoxy(), highvideo(), insline(), lowvideo(), normvideo(), putch(), textbackground(), textcolor()

5) Standard Header File Functions: <stdlib.h>

abs(), atof(),atol(), atoi(), exit(), free(), itoa(), malloc(), calloc(), max(), min(), rand(), random(), realloc(), system()

6) Standard Input Output Functions :<stdio.h>

fclose(), feof(), fflush(), fgetc(),fopen(), fputs(),fread(), fscanf(), fseek(), ftell(), fwrite(), getc(), gets(), getw(), puts(), remove(), rename(), rewind(), scanf(), sprintf

7) Disk Operating System Functions: <dos.h>

delay(), getdata(), gettime(), int86(), nosound(), setdate(), settime(), sleep(), sound()

Some functions are described in detail with example.

1. String Functions<string.h>:

The string functions are stored in the library **string.h.** When we want to use any functions of that header file we must include it.

a) Syntax: variable =strlen(string/variable);

The function returns the string length that is the total number of character(s) present in the string.

Example 3: int length;

```
char name[]=”Computer Centre”;
length = strlen(name);
```

output: length = 15

b) Syntax: strcpy (string-1,string-2);

This function copies the string-2 into the string-1. The previous value of the string-1 will be destroyed.

Example 4: char word1[20]=”Computer”,word2[20]=”Center”;

```
strcpy(word1,word2);
output: word1=”Centre”  word2=”Centre”
```

c) Syntax: strcat (string-1, string-2);

This function copies the string-2 into the string-1 after the previous contain of string-1. Note that, the target string that is string-1, must be sufficiently large to hold the whole string.

Example 5: char word1[40]=”Computer”, word2[20]=”Center”;

```
strcat(word1,word2);
Output: word1=”ComputerCentre”  word2=”Centre”
```

d) Syntax variable = strcmp(string-1, string-2);

This function compares the string-1 and string-2 to find out whether they are same or different. The two strings are compared character by character. If two strings are identical function returns zero (0). If they are not matched, it

returns the numeric difference between the ASCII values of the first two non-matching characters.

Example 6: int c;

char word1[]=”Ramachandra”, word2[]=”Ramananda”;

c=strcmp(word1,word2);

Output: c= -11

e) Syntax variable = strcmpi(string-1, string-2);

This function is similar to strcmp() but it ignore (i) the case that is upper case or lower case. The syntax remains same as strcmp().

f) Syntax : strupr(string);

This function converts string into uppercase and stores into itself.

Example 7: char word[]=”Computer”;

strupr(word);

Output: word=”COMPUTER”;

g) Syntax: strlwr(string);

This function converts string into lowercase and stores into itself.

Example 8: char word[]=”Computer”;

strlwr(word);

Output: word=”compute”

h) Syntax: strrev(string);

This function converts string into reverse order and stores into itself.

Example 9: char word[]=”Computer”;

strrev(word);

Output: word=”retupmoC”

1) Mathematical Function <math.h>:

The string functions are stored in the library **math.h.** When we want to use any functions of that header file we must include it.

a) Syntax: variable =abs(variable/value);

This function is used to find the absolute value of integer values.

Example 10: int a=-20,b;

b= abs(a);

Output: b=20

For floating point values we use **fabs()** with same syntax.

b) Syntax: variable =atof(variable/value);

This function is used to convert ASCII to float data type.

Example 11: char a="123.95"; float x;

x=atof(a);

Output: x=123.95

The other function **atoi()** is used to convert ASCII to integer data type, **atol()** is used to convert ASCII to long data type.

c) Syntax: variable =cos(angle);

variable = sin(angle);

variable = tan(angle);

These are trigonometric functions that are used to find the value after giving the angle as argument. The angle value must be given in radian system.

Example 12: float pi=22/7, a= pi/6, b;

b= sin(a);

Output : b=0.5

[cos() and tan() performs similar operation]

d) Syntax: variable=exp(variable/value);

It returns the exponential (e^x) of a given value of x.

Example 13: float x=4,y;

y=exp(x);

Output: y=54.5982

e) Syntax: variable=floor(variable/value);

This function returns nearest integer value which is less or equal to the given argument.

Example 14: float a=12.75,b;

b=floor(a);

Output: b=12

f) Syntax: variable=ceil(variable/value);

This function returns nearest integer value which is larger or equal to the given argument.

Example 15: float a=12.75,b;

b=ceil(a);

Output: b=13

g) Syntax: variable=sqrt(variable/value);

This function returns the square root of the given positive argument value. The argument must not be negative otherwise a domain error occurs

Example 16: float a,b=3;

a=sqrt(b);

Output: a=1.732

h) Syntax: variable=pow(variable/value,variable/value);

The pow() function returns base raised to the expth power. There's a domain error if base is zero and exp is less than or equal to zero. There's

also a domain error if base is negative and exp is not an integer. There's a range error if an overflow occurs.

Example 17: float a=2,b=3,c;

c=pow(a,b); [Equivalent Algebraic Expression $c=a^b$]

Output: c=8

i) Syntax: variable=log(variable/value);

The function log() returns the natural (base e) logarithm of value. There's a domain error if value is negative, a range error if value is zero.

In order to calculate the logarithm of x to an arbitrary base b, you can use:

double answer = log(x) / log(b);

Example 18: float x=3,y;

y=log(x);

Output: y=1.0986

j) Syntax : variable = log10(variable/value);

The log10() function returns the base 10 (or common) logarithm for value. There's a domain error if value is negative, a range error if value is zero.

Example 19: float x=3,y;

y=log10(x);

Output: y=0.4771

k) Syntax: variable = fmod(variable/value, variable/value);

This function returns remainder of two floating point values.

Example 20: double a=7.8, b=4.3, c;

c=fmod(a,b);

Output: c=3.5

Note: To get more detail from Turbo C compiler, open and type the function name and press **Ctrl+F1**.

Example 21:

Q4) Write a program to count the frequency of the characters.

[File Name: u5q4.c]

Solution:

```
/* Program to count the frequency of the character */
#include<stdio.h>
#include<string.h>
void main()
{
char text[80],result[80];
int count[80],i,j,n,c,k=0;
clrscr();
printf("Enter Any Text  ");
fflush(stdin); gets(text);       /* fflush() to clear the garbage input*/
n=strlen(text);
for(i=0;i<n;i++)
{
 if(text[i]==' ')    continue;
 c=1;
  for(j=i+1;j<n;j++)
  {
    if(text[i]==text[j])
    { c++; text[j]=' '; }
  }
  result[k]=text[i];
  count[k++]=c;
  c=1;
}
```

```
printf("\n\n\n");
for(i=0;i<k;i++)
printf("%c   -  %d\n",result[i],count[i]);
getch();
}
```

Q 5) Write a program to check whether the word is palindrome or not. [File Name: u5q5.c]

Solution

```
#include<stdio.h>
#include<string.h>
void main()
{
  char word1[80],word2[80]; clrscr();
 printf("Enter the word");
 fflush(stdin); gets(word1);
 strcpy(word2,word1);     /* use of library function   */
 strrev(word1);           /*  use of library function  */
 if(strcmpi(word1,word2)==0)
    printf("The given word is palindrome");
 else
    printf("the given word is not palindrome ");
 getch();
}
```

5.3 Recursion

Recursion is a tool to allow the programmer to reach a particular point called the recursion point. The recursion wants to know “What to do? “. It executes from bottom to top approach. It copies the function into the memory number of times. The function calling track has been maintained by **System Stack** (Automatically Created by Operating System).

Advantages:

1) The basic advantage is that just we have to mention “what to do?“ But not “How to do?”. So the expression is very simple.
2) For large program is good practice and applicable over function only

Disadvantage:

1) It copies function number of time and takes more memory space
2) Execution speed is slower than iteration.

Example 22: Recursive function to calculate the factorial of a given number.

```
int factorial(int n)
{
if(n==0)
   return 1;
else
   return(n* factorial(n-1));
}
```

Explanation: if call the function from main() as : x = factorial(5);

factorial(5) = 5*factorial(4)
=5*(4*factorial(3))
=5*(4* (3*factorial(2)))
=5*(4* (3*(2*factorial(1))))
=5*(4* (3*(2*(1*factorial(0)))))[Rich to the recursion point]
=5*(4* (3*(2*(1*1))))
=5*(4* (3*(2*1)))
=5*(4* (3*(2)))
=5*(4* (6))
=5*(24)
=120

So, x= 120

The output will be executed as follows:

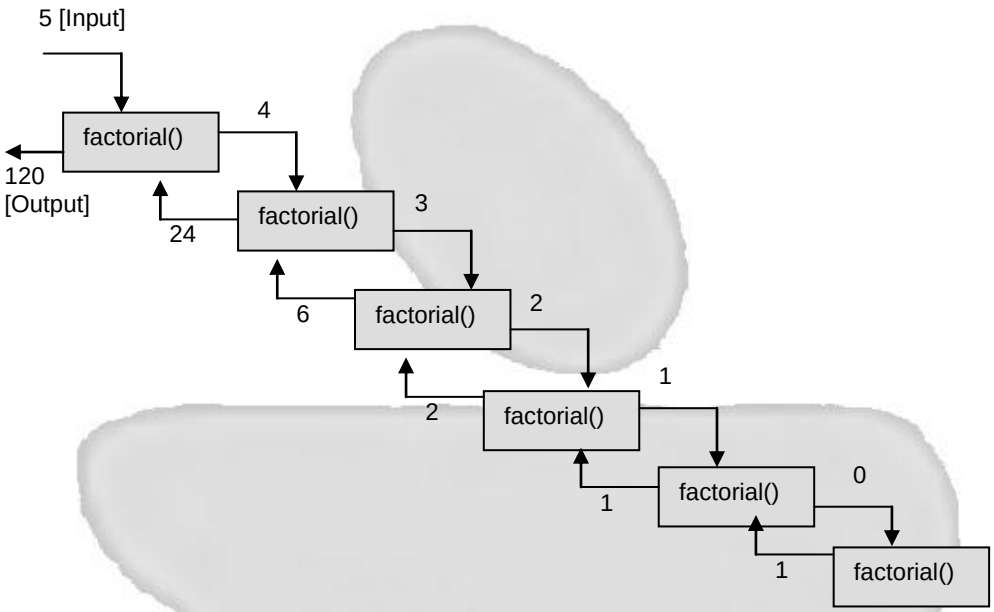


Fig. 5.1

Q. 6) Write a program to print the Fibonacci numbers

[File Name: u5q6.c]

```
void main()
{
 int n=10,i;
 clrscr();
 printf("Fibonacci Numbers using recursion\n");
  for (i=0;i<10;i++)
   {
     n=fibo(i);
     printf("%5d",n);
    }
    getch();
 }
```

```
/* The Fibonacci function  */
Int fibo(int n)
 {
   if(n==0||n==1)
   return 1;
  else
    return( fibo(n-2)+fibo(n-1));
 }
```

5.4 Summary

All the instructions of a C program are contained in functions. Each function performs a certain task. A special function name is main( ): the function with this name is the first one to run when the program starts. All other functions are subroutines of the main( ) function (or otherwise dependent procedures, such as call-back functions), and can have any names you wish. Every function is defined exactly once. A program can declare and call a function as many times as necessary. This unit covers all type of functions like library functions and user defined functions.

Self Assessment Questions

1. Function cannot be called number of times. (True / False)
2. Function prototyping is compulsory if return type other than integer. (True / False)
3. For library function, header file inclusion is a must. (True / False)
4. ceil () is the mathematical function. (True / False)
5. Recursion is faster than iteration. (True / False)

5.5 Terminal Questions

1. What is a function?
2. What is the difference between library function and user defined function?
3. Write a user defined function which is equivalent of strlen().
4. What is recursion? Write a recursion function to print all the non prime numbers.
5. State the difference between iteration and recursion.

5.6 Answer to Self Assessment and Terminal Questions

Self Assessment Questions

1. False
2. True
3. True
4. True
5. False

Terminal Questions

1. Section 5.2
2. Section 5.2.2
3. Section 5.2.2
4. Section 5.3
5. Section 5.3