CHAPTER 7

USER DEFIENED FUNCTIONS

Functions can be classified in two categories

1) library functions (are those which are defined by the compiler)
ex: printf(),scanf().

2) user-defined functions (are those which are defined by the user)

Use of function:-

It is possible to code any programs utilizing only main function, it leads to a number of problems like program may become to large and complex and as a result the task of debugging, testing and maintaining becomes difficult.
So if a program is divided into functional parts, then each part may be independently coded and later combined into a single unit. These programs are called functions.

Advantages:-

1) the length of the program is reduced
2) it leads to a modular programming
3) it is easier to locate faulty function for further processing.
4) Function may be used in other programs also
5) Reusability

Definition:-

A function is a self-contained block that performs a particular task.
Structure of functions:

```
main()
{
        -----
        -----
        func1();
        -----
        func2();
}

func1()
{
        ----
        ----
}
func2()
{
        ------
        ------
}
```

Declaration of a function:

function_name(arg list)

```
    arg declarations
    {
        local variable declarations;
        st1;
        st2;
        return(exp);
    }
```

prog) to calculate product of two numbers

```
main ()
{
        int x = 10 , y = 10 , prod;
        prod = mul(x,y);
        printf("%d",prod);
}

mul(int x , int y)
{
        int p;
        p = x * y;
        return(p);
}
```

Category of functions:

1) functions with no arguments and no return values
2) functions with arguments and no return values
3) functions with arguments and return values.

1) Prog that computes x raised to power y

```
Main ()
{
Int x, y;
Double power ();
Printf ("Enter x & y value");
Scanf ("%d %d", &x, &y);
Printf ("%d to power of %d is %f \n", x, y, power (x, y));
```

```
}
double power (x,y)
{
int x,y;
{
double p;
p=1.0                    /* x to power zero */
if (y>=0)
while (y--)              /* computes the powers */
p*=x;
else
while (y++)              /* computes the +ve powers */
p / = x;
return (p);
}
```

Nesting of functions :

```
main ()
{
fun1 ();
}
fun1 ()
        {
        fun2 ();
        }
fun2 ()
        {
        }
```

Recursion functions :

```
Main ()
{
main ();
}
```

A function calling itself is called a recursive functions.

Example:

Factorial of n numbers.

```
Main ()
{
printf (“Enter the n whose fact is to be known”);
scanf (“%n”,&d);
res = fact(z);
}
fact(int n)
{
int res ;
if (n = = 0) {
printf(“Enter correct”);
return 0;
if (n = = 1) return1;
else
res = n*fact (n-1);
return (res)
}
```