UNIT-V

USER-DEFINED FUNCTIONS

INTRODUCTION:

C functions can be classified into two categories. They are:

1.Library functions Eg: printf, scanf

2.User – defined Functions Eg: main

FUNCTION:

A function is a self – contained block of code that performs a particular task.

In order to make use of a user – defined function. We need to establish three element that are related to

Functions.

1. Function definition
2. Function call
3. Function declaration

Function Definition :

A function definition also known as function implementation. It contain the following elements.

1.Function name
2.Function Type
3.List of parameters
4.Local variable declarations
5.Function statements
6.A return statement

All the six elements are grouped into two parts namely.

1.Function header
2.Function body

Syntax:

```
Function_type   function_name ( parameter list )
{
local variable declaration;
            executable statement 1 ;
……….
            ……….
            return statement ;
  }
```

The First Line function_type function _ name (parameter list) is known as the function header and the statements within the opening and closing braces is called function body .

2. FUNCTION CALLS:

A function can be called by simply using the function name followed by a list of actual parameters.

Eg:

```
main()
{
int y ;
y = mul(10 , 5);  /* Function Call */
printf( " %d \n", y);
}
```

3. FUNCTION DECLARATION:

Like variables all functions in a C program must be declared, before they are invoked. A function declaration also known as function prototype. It consists of four parts.

1.Function type (return type)
2. Function name
3. Parameter list
4. Terminating semicolon

Syntax:

Function_type function_name(parameter list);

Example:

int mul(int m , int n); /* Function protype */

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
4. Function with no arguments but return a value
5. Function that return multiple values

1. Functions with no arguments but return a value:

This type of function does not receive any data from the calling function and it is going to return a value to the called function.

Syntax:

Return datatype function _name (void)

Program :

```
#include<stdio.h>
int add(void)
 void main()
 {
        int s;
        clrscr();
        s = add();
        printf("The value of s is %d", s);
        getch();
 }
  int add( )
  {
        int x, y ,z;
        scanf("%d %d ", &x ,&y);
        z= x+y;
        return(z);
  }
```

Output :
4 6
The value of s is 11

2. Function that return Multiple value:
In C the argument not only used to receive information but to send back information to the calling function. The arguments that are used to send out information are called output parameters.

Program :

```
#include<stdio.h>
void mathoperation(int x, int y, int *s, int *d);
void main()
{
        int x=20 , y=10 , s ,d;
        mathoperation(x,y,&s,&d);
        printf("s=%d\n d=%d\n", s,d);
}
void mathoperation ( int a ,int b, int *sum , int *diff)
{
        *sum  = a+b;
        *diff  = a-b;
}
```

Output :

```
s = 30
d = 10
```

NESTING OF FUNCTIONS:
C permits nesting of functions freely. The one function call the another function is called nesting of functions.

Program :

```
#include<stdio.h>
int getdata();
int display();
void main()
{
    int c;
    clrscr();
    c=getdata();
    printf("The value of C is is %d\n",c);
    getch();
}
```

```
int   getdata()
{
        int a,b,c;
        printf("Enter the value of a and b:");
        scanf("%d %d ", &a , &b);
        c= display();
        return(c);
}
 int   display()
 {
   int a,b,c;
   c = a + b ;
   return(c);
 }
```

Output :

```
Enter the value of a and b : 2   5
The value of  c is 7
```

RECURSION:

Recursion is a special case of this process where a function calls itself.

Program :

```
#include<stdio.h>
int factorial(int);
void main()
{
        int c;
        clrscr();
        c = factorial(6);
        printf(" The value of c is %d", c);
        getch();
 }
 int factorial (int n)
 {
          int fact;
          if(n= =1)
               return(1);
          else
                fact=n*factorial(n-1);
                return(fact);
  }
```

Output :

```
 The value of c is 720
```

PASSING ARRAY TO FUNCTIONS:

In Function we can use array element and also we can pass the value to that element.

Program :

```
#include<stdio.h>
void small(int a[ ] , int b);
 void main()
 {
        int d[6]={1,2,3,4,5,6};
        clrscr();
        small( d, 6);
        getch();
 }
 void small (int a[] , int b)
  {
       int i;
        for(i=0;i<b;i++)
        {
              Printf("The value of ais %d\n",a[i]);
         }
   }
```

Output :

```
The value of a is 1
The value of a is 2
The value of a is 3
The value of a is 4
The value of a is 5
The value of a is 6
```

UNION:

Union is a derived data type and it is declared like structure . The difference between union and structure is in terms of storage . In structure each member has its own storage location, where all the members of union use the same location , although a union may contain many member of different types.

Syntax:

```
union union_name
{
union member 1;
union member 2;
……..
……..
};
union union_variable;
```

Example:

```
Union result
{
int marks;
float avg;
char grade;
};
```

Program :

```
#include<stdio.h>
union name
{
        int a;
        char b[2];
};
void main()
{
         union name c;
         c.a=256;
         printf("c.a value is %d\n",c.a);
         printf("c.b[0] value is %d\n",c.b[0]);
         printf("c.b[1] value is %d\n",c.b[1]);
 }
```

Output :

```
c.a  value 256
c.b[0] value 0
c.b[1] value 1
```