FUNCTIONS

Functions are created when the same process or an algorithm to be repeated several times in various places in the program.

Function has a self-contained block of code that executes certain task.

A function has a name, a list of arguments which it takes when called and the block of code if executes when called.

Functions are two types,

1. Built-in / Library function Example: printf(), scanf(), getch(),exit(),etc.,
2. User defined functions

User defined Functions

Functions defined by the user according to their requirements are called User defined functions.

These functions are used to break down a large program into a small functions.

Advantage of User Defined Function:

1. Reduce the source code
2. Easy to maintain and modify
3. It can be called anywhere in the program.

Categories of User defined function / Function Prototypes:

A function prototype declaration consists of the function return type, name and argument list. Always terminated with semicolon. It always terminated with semicolon.

The following are the function prototypes:

1. Function without return value and without argument
2. Function without return value and with argument.
3. Function with return value and with argument.
4. Function with return value and without argument.

1. Function without return value and without argument

In this prototype, no data transfer takes place between the calling function and the called function. They read data values and print result in the same block.

Syntax:

```
void f_name();      //Function Declaration
```

```
void f_name(void) //Function definition
{
local variable;
statements;
}
void main()
{
f_name();     //function call
}
```

Example

```
void f_mult();        //Function definition
void f_mult(void)
{
int x,y,z;
scanf("%d%d",&x,&y);
z=x*y
printf("The result is%d",z);
}
oid main()
{
f_mult();
}
```

2. Function without return value and with argument:

In this prototype, data is transferred from the calling function to called function. The called function receives some data from the calling function and does not send back any value to the calling function.

Syntax

```
void f_name(int x,int y);  //Function Declaration
void f_name(int x, int y)  //Function definition
{
local variable;
```

```
statements;
}
void main()
{
//variable decalation;
//input statement;
f_name(c,d);//function call        //Actual Parameters
}
```

Example

```
void f_mult(int x,int y);
void f_mult(int x, int y)
{
int z;
z=x*y
printf("The result is%d",z);
}
void main()
{
int c, d;
printf("Enter any two numbers");
scanf("%d%d,&c,&d);
f_mult(c,d); //Function call
}
```

3. Function with return value and without argument

The calling function cannot pass any arguments to the called function but the called function may send some return value to the calling function.

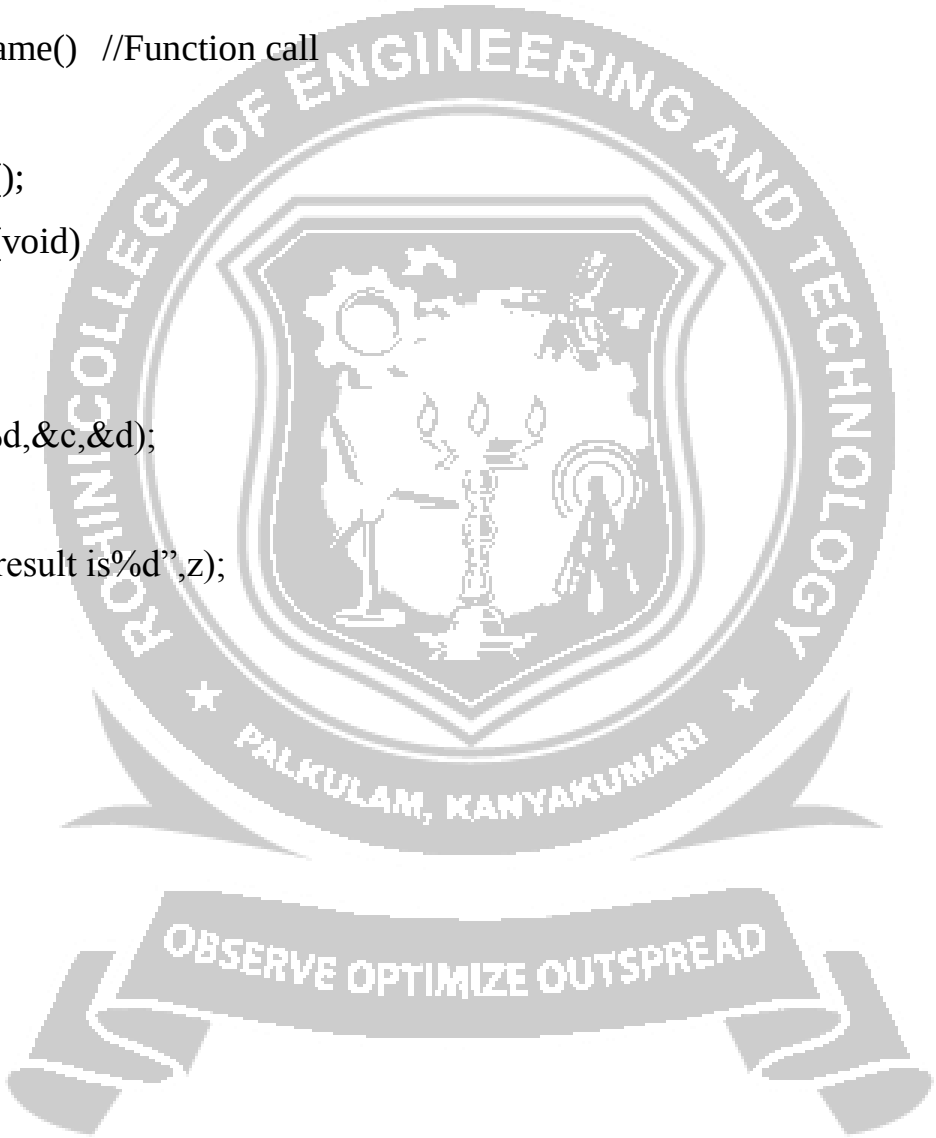
Syntax:

```
int f_name();          //Function declaration
int f_name(void)    //Function definition
{
```

```
local variables;
statements;
return int;
}
void main()
{
//variables declaration
ret_var=f_name()   //Function call
```

Example

```
void f_mult();
void f_mult(void)
{
int x,y,z;
scanf("%d%d,&c,&d);
z=x*y;
printf("The result is%d",z);
return(z);
}
void main()
{
int c;
c=f_mult();
getch();
}
```



4. Function with return value and with argument:

In this prototype, the data is transferred between the calling function and call function. The called function receives some data from the calling function and send back a value return to the calling function.

Syntax:

```
int f_name(int x,int y);     //Function Declaration
int f_name(int x, int y)     //Function definition        //Formal Parameter
{
local variable;
statements;
return int;
}
void main()
{
//variable decalation;
//input statement;
ret_value=f_mult(a,b);;    //function call        //Actual Parameters
}
```

Example

```
int f_mult(int x,int y);
int f_mult(int x, int y)
{
int z;
z=x*y
printf("The result is%d",z);
return(z);
}
void main()
{
int a,b,c;
printf("Enter any two value:");
scanf("%d%d,&a,&b);
c=f_mult(a,b);        //Function call
}
```