FUNCTIONS

Function in C++ help the programmer to make his code more compact, readable and easier to understand. It also allows the program to reuse the already define code in modules known as function, with the use of the functions, the typing effort of the user is minimized.

C++ also has a rich collection of functions in its library for various purposes such as mathematical functions, graphical functions etc.

A function is c++, has three main components

i. Function prototype
ii. Function call
iii. Function definition

Function prototype : Function prototype is used to inform the compiler that a function with the name as in prototype will be encountered. In C++ function prototyping syntax is as :

Return_Type function_name (type formal_argument(s));

- Return type indicate the type of the value a function will return after executing the body of the function.
- Function_name is any user-defined name for the function by which it will be called. It is always advisable to use the function name that resembles its use.
- Formal arguments are the list of arguments which will assume the values of the actual parameters from the function calling.

Function Call

A function is called by its name by passing the actual arguments as defined in the function prototype. On receiving a function call, the compiler interprets as a jump to the place where the definition of the function is written. If the prototype has no argument, then arguments to the function call is not passed, else the number of argument and their type is same as given in the prototype. The syntax of the function call is given below:

[identifier=] Function_name(actual arguments);

Here, identifier is optional and is written only when the return type of the function is other than void. When the function does not return any value, then the call to the function is made as :

Function_name(actual argument);

Or

Function_name();

Function Definition:

The function definition is a place where the body of the function is defined. It is the container of the statements that must be executed when a call to the function is made.

Function call by value

In this method of calling the actual argument are replaced into the formal parameter in function definition. As such, a separate copy of the actual are created, so that the value of actual in main function remain the same and are unaffected by the operation in function body.

Example:

```
#include<iostream.h>
void swap(int x, int y);          //function prototype
Void main()
{
        int a,b;
        cout<<"enter two numbers";
        cin>>a>>b;
        swap(a,b);                //actual argument passed to swap function
        getch();
}
void swap(int x, int y)           // x, y are formal arguments
{
        int temp;
        temp=a;
        a=b;
        b=temp;
        cout << "the swaped values are a="<<a<<" and b =  "<< b
}
```

Call by reference

In this method of call, the address of the actual are passed into the formal parameters. So instead of the actual values their reference goes in the function. As such, the changes inside the function also affects the value in the main. See the following example.

```
#include<iostream.h>
#include <conio.h>
void increment_xy(int &, int &); //function prototype
void main()
{
        int a,b;
        cout<<"enter two numbers";
        cin>>a>>b;
        cout<<"\n"<<"the values before function call are:"<<a<<" and "<<b;
        increment_xy (a,b);       //actual argument passed to swap function
        cout<<"\n"<<"the values after  function call are:"<<a<<" and "<<b;
        getch();
}
void increment_xy(int &x, int &y)                    // x, y are formal arguments
{
        X++;
        Y++;
        cout << "the incremented values inside the function are a="<<x<<" and b =  "<< y
```

```
}
```

Function that return value by reference

This type function returns a value from the function body into another variable of type given by return type. The example below illustrates this.

```
#include<iostream.h>

#include <conio.h>

int   largest_xy(int &, int &);  //function prototype

void main()

{

        int a,b;

        cout<<"enter two numbers";

        cin>>a>>b;

        cout<<"\n"<<"the values before function call are:"<<a<<" and "<<b;

        int z  =  largest_xy (a,b);             //actual argument passed to swap function

        cout<<"\n"<<"the largest value return is :"<< z;

        getch();

}

int  largest_xy(int &x, int &y)      // x, y are formal arguments

{

        If (x > y)

        return x;

        else

        return y;

}
```

Function that return value by reference

This is done with the help of a reference variable. The reference variable takes the reference(address) of another variable. Thus any change in one will be reflected at other and vice versa.

e.g.

int x;

int &y = x; y get the reference of x;

so, whether we write y++ or x++ will mean same. We use this property to return the reference from a function.

WAP to find the largest of two numbers

```
#include<iostream.h>
#include <conio.h>
int &  largest_xy(int &, int &);     //function prototype
void main()
{
        int a,b;
        cout<<"enter two numbers";
        cin>>a>>b;
        cout<<"\n"<<"the values before function call are:"<<a<<" and "<<b;
        int &t  =  largest_xy (a,b);                    //actual argument passed to swap function
        cout<<"\n"<<"the largest value return is :"<< t;
        getch();
}
int & largest_xy(int &x, int &y)              // x, y are formal arguments
```

```
{
        If (x > y)
        return x;
        else
        return y;
}
```