Programming and Problem Solving through C Language
O Level / A Level

Chapter - 6 : Functions

Function Arguments

If a function is to use arguments, it must declare variables that accept the values of the arguments. These variables are called the **formal parameters** of the function. Formal parameters behave like other local variables inside the function and are created upon entry into the function and destroyed upon exit.

Parameter

- A parameter is a special kind of variable, used in a function to refer to one of the pieces of data provided as input to the function to utilse.
- These pieces of data are called arguments.
- Parameters are Simply Variables.

Formal Parameter

- Parameter Written in Function Definition is Called "Formal Parameter.
- Formal parameters are always variables, while actual parameters do not have to be variables.

Actual Parameter

- Parameter Written in Function Call is Called "Actual Parameter".
- One can use numbers, expressions, or even function calls as actual parameters.

Example

```
void display(int para1)
{
        printf( " Number %d " , para1);
}

void main()
{       int num1;
        display(num1);
}
```

In above, **para1** is called the Formal Parameter
num1 is called the Actual Parameter.

Parameter list

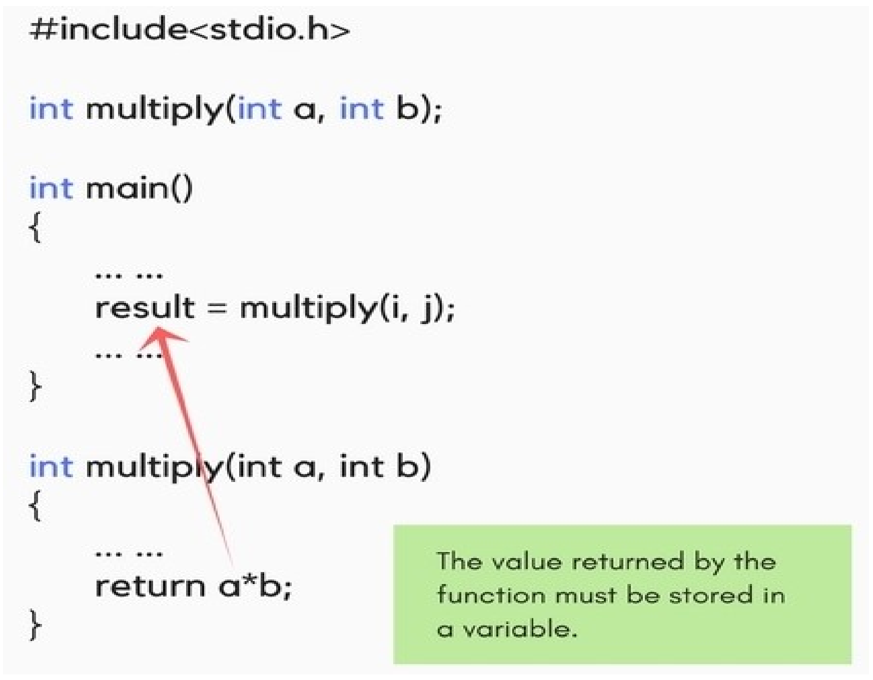
- A function is declared in the following manner:

  return-type function-name (parameter-list,...)
  { body... }

- Return-type is the variable type that the function returns.
- This cannot be an array type or a function type.
- If not given, then int is assumed.
- Function-name is the name of the function.
- Parameter-list is the list of Formal Arguments Variable.
- Given below are few examples:
  - int func1(int a, int b) /* two argument – int , int */
  - float func2(int a, float b) /* two argument – int , float */
  - void func3( ) /* No argument . */

The Function Return Type

- The function return type specifies the data type that the function returns to the calling program.
- The return type can be any of C′s data types: **char, int , long , float , or double** .
- One can also define a function that doesn′t return a value by using a return type of **void**.
- Given below are few examples:
  - int func1(...) /* Returns a type int. */
  - float func2(...) /* Returns a type float. */
  - void func3(...) /* No Returns . */



```
#include<stdio.h>

int multiply(int a, int b);

int main()
{
    ... ...
    result = multiply(i, j);
    ... ...
}

int multiply(int a, int b)
{
    ... ...
    return a*b;
}
```

The value returned by the function must be stored in a variable.

Program – A function that return the maximum between two numbers

```
int max(int num1, int num2)
{       /* local variable declaration */
        int result;
        if (num1>num2)
             result=num1;
        else
           result=num2;

       return result;
}

void  main()
{       int r;
         r=max( 10,15);
        print("Result = %d " , r);
}
```

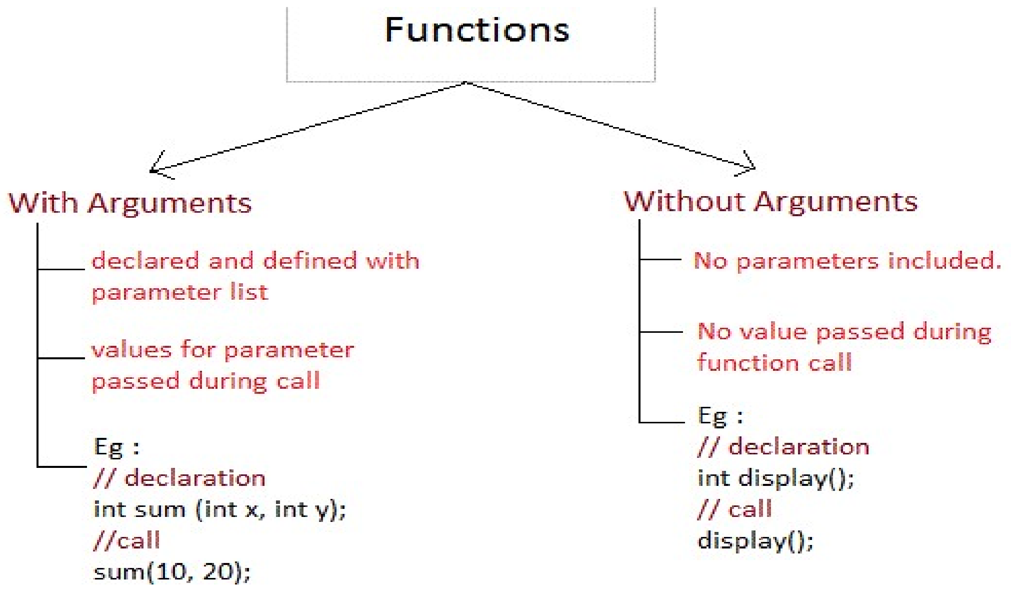
Calling Functions

There are two ways to call a function.

1) Any function can be called by simply using its name and argument list alone in a statement, as i n the following example.
    i) If the function has a return value, it is discarded.

```
wait(12);
```

2) The second method can be used only with functions that have a return value.
    i) Because these functions evaluate to a value (that is, their return value), they are valid C expressions and can be used anywhere a C expression can be used.
    ii) An expression with a return value used as the right side of an assignment statement.


Functions
With Arguments
declared and defined with parameter list
values for parameter passed during call
Eg :
// declaration
int sum (int x, int y);
//call
sum(10, 20);
Without Arguments
No parameters included.
No value passed during function call
Eg :
// declaration
int display();
// call
display();