Functions - Returning data from functions

- In our prior example of multiply two numbers, our multiply function displayed the results of the calculation at the terminal
- You might not always want to have the results of your calculations displayed .
    - ➔ Functions can return data using specific syntax.
    - ➔ Should be familiar from previous experience with the main function
- Let's take another look at the general form of a function.

```
Return_type Function_name(list of Parameters - separated by commas)
{
	//Statements...
}
```

- The Return_type specifies the type of the value returned by the function.
- You can specify the type of value to be returned by a function as any of the legal types in C, includes enumeration types and pointers.
- The return type can also be type void which no value is returned .
- The return statement provides the means of exiting from a function return;
- This form of the return statement is used exclusively in a function where the return type has been declared as void. Does not return a value.
- The more general form of the return statement is:

```
return expression;
```

- This form of return statement must be used when the return value type for the function has been declared as some type other than void.
- The value that is returned to the calling program is the value that results when expression is evaluated.
    - ➔ Should be of the return type specified for the function.
- A Function that has statements in the function body but does not return a value must have the return type as void.
    - ➔ Will get an error message if you compile a program that contains a function with a void return type that tries to return a value.
- If expression results in a value that's a different type from the return type in the function header, the compiler will insert a conversion is not possible then the compiler will produce an error message.
- There can be more than one return statement in a function.
    - ➔ Each return statement must supply a value that is convertible to the type specified in the function header for the return value.

Invoking a function

- You call a function by using the function name followed by the arguments to the function between parentheses.
- When you call the function, the values of the arguments that you specify in the call will be assigned to the parameters in the function.

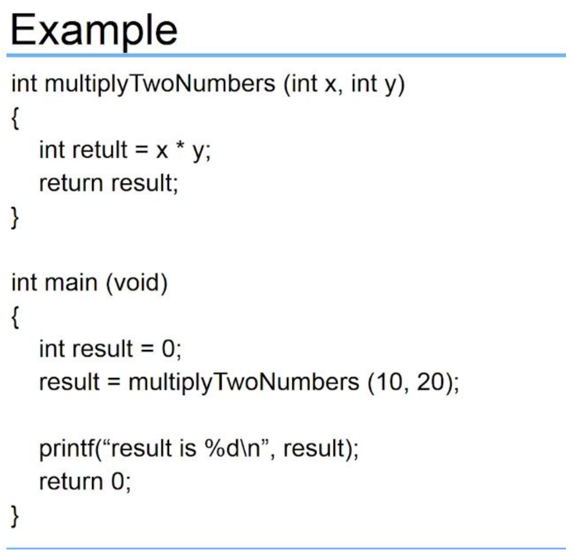
- When the function executes, the computation proceeds using the values you supplied as arguments.
- The arguments you specify when you call a function should agree in type, number, and sequence with the parameters in the function header.

Invoking a function and assigning a data returned.

- If the function is used as the right side of an assignment statement the return value supplied by the function will be substituted for the function
    - ➔ Will also work with an expression

int x = myFunctionCall();

- The calling function doesn't have to recognize or process the value returned from a called function.
    - ➔ Up to you how you use any values returned from function calls.



Example

```
int multiplyTwoNumbers (int x, int y)
{
    int retult = x * y;
    return result;
}

int main (void)
{
    int result = 0;
    result = multiplyTwoNumbers (10, 20);

    printf("result is %d\n", result);
    return 0;
}
```