Calling Functions

This lesson teaches how to call functions that we have written in our main function and uses an example to further elaborate

We'll cover the following

Calling

Calling refers to how a function is used from code.

Example:

Let's take a look at how functions that have already been made are *called* in the `main` function.

```
#include <iostream>
using namespace std;

void fctn1();    // declaring a void function
int fctn2(int, int);   //declaring int type function taking two int arguments

int main()
{
    int sum;

    fctn1();        //calling the void function
    sum = fctn2(2,3);    // calling the function fctn and saving the value returned in variable 'su
    cout << "The value of sum is: " << sum << endl;
    return 0;
}

void fctn1()      // writing the function definition here
{
    cout << "This is function 1!" << endl;  // function only couts a string
}

int fctn2(int num1, int num2)    // writing the function definition
{
    //the value 2 has been passed as num1
    //the value 3 has been passed as num2
```

```
    return num1 + num2;      // returning the sum of num1 and num2
}
```

Explanation

- *First* note the use of the **two** declarations that precede the `main` function. They allow main to use `fctn1` and `fctn2` even though they aren't *defined* until after `main`.
- The *forward declarations* ensure that the compiler knows that when it sees the symbols `fctn1` and `fctn2` that those names refer to functions somewhere in the program.
- *Next*, notice that to *call* `fctn1` all we had to do was enter `fctn1()`;. Since it doesn't require any *arguments* and it has a `void` *return value*, this is very simple.
- For `fctn2` however, we see that it not only requires **two** *arguments*, but also *returns* an **integer** as well.
- To pass data to a *function*, you simply list the data in the order it is called for by the function *definition*. To catch the r*eturned data*, we use the assignment operator `=` and *assign* the returned value to a *variable* `sum`.

> **Note:** Functions with a void return type, as `fctn1`, do not require a `return` statement as they do not return anything.

Now that we know how to *call* functions in our code, in the next lesson we will delve into some other details such as the parameters we pass in functions.