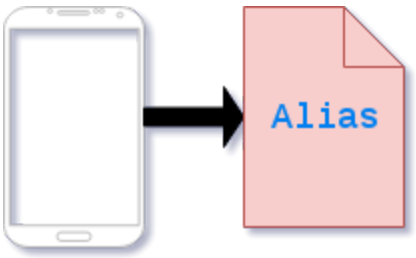
Call by Reference



Call by Reference is when a function makes an alias of the incoming argument. That is, any variable with an **ampersand** will act as if you are using the variable from the function invocation. Therefore, anything you do in the function will alter both variables at the same time. The **ampersand** can be read as **"address of."** This allows for the reference, or alias. The **ampersand** may be placed after the keyword **int** or before the local variable itself (also, whitespace doesn't affect anything). Here is an example to clarify:

```
int function(int &setNum)
{
    setNum = 50; //Pink to denote function's local variable
    return setNum;
}
int main()
{
    int number = 100; //Green to denote main's local variable
    cout << number << endl;
    cout << function(number) << endl;
    cout << number << endl;
}
```

There are three cout statements to examine.

The first cout statement will display 100 as that's what **number** has been assigned.

In the second cout statement, there is a function invocation with the argument **number**. The contents of **number** is 100. This value is referenced, or aliased, and the address of **number** is placed into the variable **setNum**. Now **setNum** is linked (refers) to **number**. In the function, **setNum** is modified. Since both **setNum** and **number** refer to the same thing, **number** has now been officially changed to 50. The cout statement would display 50 as that's what has been returned.

The last cout statement will print the number 50. Again, **number** has been altered in **function** via **setNum**.

Remember, literals cannot be called (passed) by reference as literals are not stored in memory. Therefore, there is no address to be passed to the function.

This method is also known in CSI as pass by reference as you are passing the address of the variable to the function and creating a reference to the argument variable.

See the **Functions** page for a **const** reference function example.