# Pass by Reference

May 3, 2020

## 0.1 Passing Values

In the following example, the *value* of `int i` is passed to the function `MultiplyByTwo`. Look carefully at the code and try to guess what the output will be before you execute it. When you are finished executing, click the button for an explanation.

```
In [ ]: #include <iostream>
        using std::cout;

        int MultiplyByTwo(int i) {
            i = 2*i;
            return i;
        }

        int main() {
            int a = 5;
            cout << "The int a equals: " << a << "\n";
            int b = MultiplyByTwo(a);
            cout << "The int b equals: " << b << "\n";
            cout << "The int a still equals: " << a << "\n";
        }
```

Compile & Execute

Explain
Loading terminal (id_i0wc1j2), please wait...
In the code above, `a` is passed by value to the function, so the variable `a` is not affected by what happens inside the function.

## 0.2 Passing References

But what if we wanted to change the value of `a` itself? For example, it might be that the variable you are passing into a function maintains some state in the program, and you want to write the function to update that state.

It turns out, it is possible to modify `a` from within the function. To do this, you must pass a *reference* to the variable `a`, instead of the *value* of `a`. In C++, *a reference is just an alternative name for the same variable*.

To pass by reference, you simply need to add an ampersand `&` before the variable in the function declaration. Try the code below to see how this works:

```
In [ ]: #include <iostream>
        using std::cout;


        int MultiplyByTwo(int &i) {
            i = 2*i;
            return i;
        }

        int main() {
            int a = 5;
            cout << "The int a equals: " << a << "\n";
            int b = MultiplyByTwo(a);
            cout << "The int b equals: " << b << "\n";
            cout << "The int a now equals: " << a << "\n";
        }
```

Compile & Execute

Explain
Loading terminal (id_a91ke5k), please wait...
In the code above, `a` is passed by reference to the function `MultiplyByTwo` since the argument to `MultiplyByTwo` is a reference: `&i`. This means that `i` is becomes another name for whatever variable that is passed into the function. When the function changes the value of `i`, then the value of `a` is changed as well.

### 0.2.1 Practice

Modify the function below to accept a reference so that the passed variable can be directly modified by the function.

```
In [ ]: #include <iostream>
        #include <string>
        using std::cout;
        using std::string;

        void DoubleString(string &value) {
            // Concatentate the string with a space and itself.
            value = value + " " + value;
        }

        int main() {
            string s = "Hello";
            cout << "The string s is: " << s << "\n";
            DoubleString(s);
            cout << "The string s is now: " << s << "\n";
        }
```

Compile & Execute

See Solution
Loading terminal (id_njq15mz), please wait...