Ekaba Bisong
Programming in C++
University of Calabar

Lesson Note #35
June 09, 2015

Adapted from C++ Tutorial at http://www.cplusplus.com/doc/tutorial/control/
and C++ How To Program edited for our own purposes

References and Reference Parameters

Two ways to pass arguments to functions in many programming languages are pass-by- value and pass-by-reference. When an argument is passed by value, a copy of the argument's value is made and passed (on the function call stack) to the called function. Changes to the copy do not affect the original variable's value in the caller.

With pass-by-reference, the caller gives the called function the ability to access the caller's data directly, and to modify that data.

For example, suppose that we called our first function addition using the following code:

```
int x=5, y=3, z;
z = addition ( x , y );
```

What we did in this case was to call to function addition passing the values of x and y, i.e. 5 and 3 respectively, but not the variables x and y themselves.

```
int addition (int a, int b)
                 ↑     ↑
z = addition (   5   ,   3   );
```

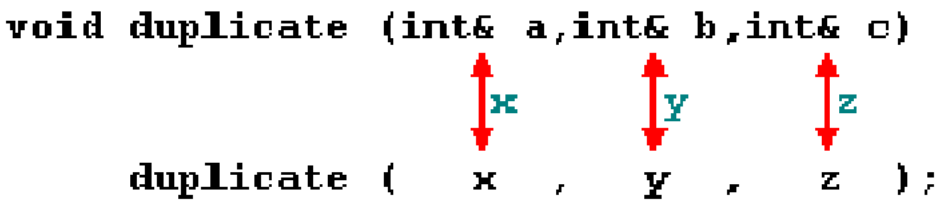
This way, when the function addition is called, the value of its local variables a and b become 5 and 3 respectively, but any modification to either a or b within the function addition will not have any effect in the values of x and y outside it, because variables x and y were not themselves passed to the function, but only copies of their values at the moment the function was called.

But there might be some cases where you need to manipulate from inside a function the value of an external variable. For that purpose we can use arguments passed by reference, as in the function duplicate of the following example:

```
1   // passing parameters by reference
2   #include <iostream>
3   using namespace std;
4   void duplicate (int& a, int& b, int& c)
5   {
6       a*=2;
7       b*=2;
8       c*=2;
9   }
10
11  int main ()
12  {
13      int x=1, y=3, z=7;
14      duplicate (x, y, z);
15      cout << "x=" << x << ", y=" << y << ", z=" << z; return 0;
16  }
```

The first thing that should call your attention is that in the declaration of duplicate the type of each parameter was followed by an ampersand sign (&). This ampersand is what specifies that their corresponding arguments are to be passed by reference instead of by value.

When a variable is passed by reference we are not passing a copy of its value, but we are somehow passing the variable itself to the function and any modification that we do to the local variables will have an effect in their counterpart variables passed as arguments in the call to the function.



To explain it in another way, we associate a, b and c with the arguments passed on the function call (x, y and z) and any change that we do on a within the function will affect the value of x outside it. Any change that we do on b will affect y, and the same with c and z.

That is why our program's output, that shows the values stored in x, y and z after the call to duplicate, shows the values of all the three variables of main doubled.