Much of today’s class was consumed by questions about programming assignment #1. That is natural (this being the due date of the first assignment). Once we moved on, we reviewed what was discussed on Monday about methods having hidden arguments, and then the difference between pas by value parameters and pass by reference parameters.

If an argument is passed by reference, is it changed? In general you don’t know. At best, you could go look at the implementation (cpp) file, but (a) you don’t want to have to do this, and (2) it might not be easy to know (if that function calls another using the same argument…). Java tries to remedy this problem through documentation (Java Docs), but you don’t really know.

If an argument is passed by value, you are safe: it won’t be changed as a side effect. If it passed by reference, you aren’t sure.

But what about efficiency? Well, for primitive data types (int, double, etc.) it doesn’t matter: passing an address is just as expensive as copying the value (because passing an address is copying a value).

But for complex objects, copying the whole object may waste both memory and cycles. Imagine a large object, like an image or a video or a database. Call by reference is faster for big objects.

There is an efficiency down side to call by reference, however. Let’s say I have a complex object with lots of fields (quaggas have weights, heights, partners, ages, etc.). I may be processing that data, and some of it may already be in registers. Now I make a function call, and pass the quagga by reference. What happens to the values in registers? (Assuming the called function didn’t need all the registers) After the function call, the compiler has to assume that the function call changed them, and that they are no longer valid. As a result, they have to get re-fetched.

So C++ has a better option for big objects: constant call by reference. Pass the argument as a constant reference. This is done by putting const before the argument, and & after it. Now the object doesn’t have to be copied, but user’s know that it isn’t being changed, and so does the compiler. Best of all worlds.

So in summary, there are three ways to pass an argument in C++:

1. Call by value. Should be your default method for primitive data types.
2. Call by reference. Should only be used when you intend to side-effect the argument.