Coding Versus Programming

Steven Zeil

May 24, 2013

Contents

1 So You Want to Be a Programmer?

Your previous studies of C++ have focused on "coding" – rendering step-by-step instructions into C++.

But coding is only a small part of programming. This might come as a surprise. Some of you have faced a struggle in the early assignments just getting the compiler to stop complaining and actually translate your code. But believe me, before much longer you will learn to simply avoid making the same kinds of compilation errors that have plagued you in the past. At that point you will find that relatively little of your time on programming assignments is devoted to just getting things to compile.

So, what else is there? A *programmer* must be able to

- design an algorithm that will solve the problem at hand
- code that algorithm
- test that code to see whether it works, and when, (inevitably :-) it does not work,
- debug that code to find the hidden flaws.

Studies in the real world of large, successful software development projects suggest that

- Programmers spend about half their total time in testing and debugging
- The best programmers spend the bulk of the remaining time analyzing the problem and designing the solution

Coding, however much it may plague you at the moment, is the smallest part of their job.

In the ecology of software development, "coders" are pretty far down in the food chain [1]. Companies large enough to hire people just to be coders won't reward them all that well, and companies too small to have someone who can only code will not hire them at all. Generally, employers are looking for someone with more rounded programming skills.

In the remainder of this course, we will certainly still be learning more C++. But much of our effort will be spent looking at

Design: We'll try to relieve that sinking feeling you get when you first sit down to start a new program and are staring at the most intimidating sight in the whose world - a sheet of blank paper.

Testing: If you have been frustrated by the way I tested your code or if you thought I found some pretty picky bugs in your submitted programs, there's actually nothing magical about what I do. My secrets will be revealed!

[1] Ranking above data entry clerks and slightly below the person who knows how to clear the paper jams from the copier.

Debugging: The real time sink for most programmers. In some ways this may be the hardest skill to teach, which is why so many books and programming courses don't try.

Working on these topics will take us out of the textbook a lot more than does the more straightforward how-to-code-in-C++ material.

You might find it interesting to reflect on the fact that programmers spend half their time doing testing and debugging, but then try looking up those terms in the index of your textbook. You'll see fewer than 2 dozen pages (out of well over a thousand) discussing these topics - a real mismatch with the amount of time and effort programmers spend on them. And I'm not trying to beat up on this particular textbook – almost all programming texts are similarly lopsided.

Some of the techniques that we look at may seem overkill for the size of the programs you will work with in this class. We practice them now, though, on small programs, so you will have these techniques in your personal toolbox when it comes time to work on larger programs in later classes. You'll have to take my word for it right now that these tools will prove useful.