



Editor's Forum

What Every Programmer Should Know

A few years ago I had the great pleasure of stumbling across an ACM article by David Goldberg entitled "What Every Computer Scientist Should Know About Floating-point Arithmetic." If you're not familiar with this excellent monograph, you might be curious about how long the article was. After all, how much is there to say about floating-point numbers? A lot, evidently, because the article was 48 pages long. It contains many things I've learned the hard way over the years, and a few things that I didn't know but was very happy to learn. I strongly recommend it as required reading by all professional programmers.

That article's catchy title leads me to an even more important question: "What should every programmer know?" This question is of increasing importance to me since I have returned to academia. The institution I am privileged to teach at does not dwell in the "academe shade," but is intent on graduating skilled artisans ready to contribute immediately to industry (that's why I took the job). So I've asked myself: What are the fundamental knowledge, skills, and attributes that the "ideal" CS graduate should possess? What do you think? (I really want your feedback on this one.)

I think near the bottom of the list is which development tools the candidate has used. Yes, I suppose it's nice to be familiar with whatever is popular, but learning to use a tool is about the easiest and least critical skill for productive programming. Having used one, it's easy to adapt to another of similar ilk. A likewise relatively unimportant experience is GUI design. (I can hear the complaints roll in already). Anyone can drag and drop. Let the Human Factors or Graphic Art staff design screens. A programmer can't match their visual skills anyway.

So much for the bottom of the list. At the top I would place, in no particular order, things like:

- inferring important abstractions from the problem at hand and crafting code to implement them
- the inclination to think in terms of invariants and to regularly use assertions and such to enforce the same
- a feel for the complexity of algorithms and the cost of development
- the ability to write code readable by humans as well as compilers
- the desire to improve code through refactoring when the opportunity presents itself, as well as to seek and accept feedback from peers
- the tenacity and ingenuity to find and fix bugs, along with the patience to think through a problem before coding blindly, so the bugs aren't needlessly multiplied *a priori*
- the wisdom to not "reinvent the wheel," but to learn and use available components
- and last, but certainly not least, the ability to communicate verbally and in writing to not only fellow programmers, but "normal people" as well

Add to that Dijkstra's advice to hone one's natural inclination towards mathematics as well as mastery of your native tongue, and I think we have a good start.

Oh, and one more thing. A Good Programmer needs to have some passion for solving problems through code. I'm glad I still have mine.

Chuck Allison
Senior Editor