

# Re: debate: to get a Master's Degree in CS or Not

---

*Source:* <http://coding.derkeiler.com/Archive/General/comp.programming/2006-01/msg00070.html>

---

- *From:* Randy <joe@xxxxxxxxxxxxxxxxxx>
  - *Date:* Thu, 05 Jan 2006 12:48:56 -0600
- 

Walter Bright wrote:

> "Randy" <joe@xxxxxxxxxxxxxxxxxx> wrote in message  
> [news:dphc78\\$3in\\$1@xxxxxxxxxxxxxxxxxx](mailto:news:dphc78$3in$1@xxxxxxxxxxxxxxxxxx)

....

>>Knowing math has value, but only if it's necessary to what you do.

>>Math skills are largely irrelevant in all but design.

>

>

> I strongly disagree. Math skills give one the skill of a certain pattern of  
> thought which is widely applicable far beyond just calculating numbers.  
> Knowing the math also helps impart that engineering 'feel' for if a design  
> is right or not. Math skills help replace emotional decision making with  
> rational decision making.

Is a mathematical/theoretical mindset valuable? Of course. Is it more valuable than a results-oriented/practical mindset? IMHO no, unless the domain is design or process improvement as in debugging or optimization.

Which mistake is made more often (per capita) -- theorists ignoring practice or practitioners ignoring theory? Probably the former, since professional theorists are seldom rewarded for being practical. In contrast, practitioners are rewarded for doing whatever works, even if it involves employing theory.

We're not disagreeing, I think. I'm just emphasizing the relative values of theory vs. practice, given my experience in and out of the Real World (business vs. gov't vs. academia: I've worked in all three).

>

>>Sure, they are

>>also useful in optimization practices of all kinds, but from what I've  
>>seen in several decades of working in the world, few organizations care  
>>about efficiency much less optimality. Neither ever got a product  
>>funded or a company built. Ergo, math skills take a distant second to  
>>skills of a more social or practical nature.

>

>

> If you try to run a company without knowing any math, you're going to be  
> taken to the cleaners by the finance people who do. On a more personal

## Re: debate: to get a Master's Degree in CS or Not

- > level, credit card companies make fortunes off of math deficient people, as
- > do car finance companies, and every gambling casino. You can't understand
- > money without solid math skills.

Examples abound on the peripheries. Lots of businesses employ folks with degrees in operations research, applied math, applied physics, mathematical economics, and management science. No doubt about it. All these roles seek to maximize returns, to mine the margins of business.

But the vast majority of business operates in the mainstream, where value lies in selling a useful product for a competitive price. Software development (and most engineering) facilitates that work, and as such, deals more in building a product that works and arrives in the hands of the consumer acceptably soon, and less in optimizing that product. It's mostly meat-and-potatoes work, and benefits most from attention to feedback: do more of what works and do less of what doesn't.

IMHO, math is a minor player in implementing the software behind that world. A quick look at the vast majority of software development job ads should confirm that. Among the alphabet soup of skills mentioned in job ads, math skills are almost never requested.

- >
- > I won't denigrate the desirability of social skills. But to say that math
- > skills are somehow not of a "practical" nature is way, way off base.

Math skills are not the same as math practices. If employees don't employ a set of skills often enough for employers to recognize them as being essential to doing business, job ads will emphasize only what is necessary.

By way of example, look at salaries for mathematicians vs. salaries for engineers. If theory were deemed to be of equal value as practice, the wages would be the same. But they aren't.

Likewise, compare the number of jobs for folks with advanced math skills (mathematicians, engineers, physical scientists) with those who lack those skills (computer scientists, trade programmers, technicians). The bulk of the work out there is practical, and results oriented.

Again, sure, math has value. But it serves a minority role in the world, somewhat like philosophy, writing, psychology, etc. Should it? I guess the answer depends on whether you subscribe to Friedman or Galbraith.

- >
- >>And math skills are no measure at all
- >>of whether someone can implement these algorithms well or correctly.
- >
- >
- > What nonsense. I've implemented algorithms I didn't understand. The problem

## Re: debate: to get a Master's Degree in CS or Not

- > was, I had no way to tell if I did it right or not. For algorithms I did
- > understand, I could tell if they're working right or not, I knew how to fix
- > them if they weren't, and I could even improve upon them. Otherwise, it's
- > just shooting blind and deaf.

But it happens all the time. How many programmers understand accounting or business, or even their users? Not many. How many combine output from one spreadsheet with another to drive a database? How many winnow databases using metrics and statistics they don't understand, which drive pie charts and the like? A great many. How many programmers understand well the domain in which they work? ALMOST NONE.

That's the nature of software development. The product (software) serves the needs of others, not the programmer. As such, a really large amount of software sucks, fails, and often never gets fielded. A lot of effort has gone into identifying the causes for this in the past 20 years, and some good ideas have emerged (agile programming, use cases, etc), but most firms don't make good use of these in (re)defining or validating requirements, much less using user feedback (or business feedback) to tune them.

In programming, flying blind is the norm, not the exception.

- >
- >>Not that I believe math skills are unnecessary or valueless in
- >>computing, but I do think that it's a pretty small fraction of software
- >>jobs that can employ even calculus, much less number theory, abstract
- >>algebra, topology, analysis, or math (or logic) proofs of any kind.
- >
- >
- > I just shake my head in disbelief <g>. How on earth do you debug your
- > programs, for example, if you are unable to think in terms of logical
- > proofs? If you can't think logically, how can you possibly construct the
- > logic of a program?

Methinks you mistake semantics for syntax. Most programming and debugging emphasize syntax, or the most basic form of operational semantics — getting the program to run to completion and produce the right amount of output. That's easily done using a knowledge of programming language syntax/semantics, and a bit of compiler and O/S facility.

But knowing the semantics of what the program is supposed to DO is entirely a different matter, and few programmers know their program's domain well enough to verify that the tool they just built does what was intended. Data does indeed flow as it was supposed to. But was the \*purpose\* behind writing the code served? Was the corporate mission served? Served well or served marginally? Served flexibly or narrowly? And how would they know, knowing only syntax or perhaps math?

No. From what I've seen in the real world, there's a GREAT deal more

Re: debate: to get a Master's Degree in CS or Not

## Re: debate: to get a Master's Degree in CS or Not

value in knowing more about your problem than more about your tools, and even less value in knowing the theory behind your tools. (To a limit, of course. I'm still speaking about mainstream business and not those that make their revenues on derivatives or margins.)

BTW, nobody programs using logical proofs when writing or even validating code. You must be over 50 to think in such terms. (I'm just about there myself.) The only folks I know who do that entered the field before 1980, back in the days of flow charts and paper-based programming.

Ever wonder why Z notation or other definitive forms of programming never caught on? Proofs are WAY too slow a way to build software. Worse, they place too much emphasis on the semantics of the programming language, and not where it's most needed, on the application/problem domain.

...

>>

>>But that's true for any rigorous subject. Philosophy, law, any form of science -- all of these teach critical methodical thinking.

>

>

> The first two completely lack rigor. If they had rigor, then there wouldn't be reversals in philosophical thought every few years, and there wouldn't be a Supreme Court trying to figure out what the law says and reversing their own rulings. I don't see anyone deciding that  $2+2=5$ , though there is the (probably apocryphal) story of the state legislature that once attempted to pass a law defining pi to be 3.0.

You need to read a philosopher like Russell, Whitehead, etc. Most successful philosophers in history had excellent logical (and semantic) skills. If they didn't, and they contradicted themselves or failed to make a complete, meaningful, and convincing argument, they would have been ignored during their own lifetime, much less forgotten by history.

Of course, law is based on philosophy and logic. The fact that it's so often used illogically (and successfully) says nothing about whether its fundamentals lack rigor or precision. BTW, the public sees far too much of the worst forms of law (trial law). The bulk of the practice is very effective and sometimes can be quite elegant. However enacting \*legislation\* is entirely another matter.

Now economics... THERE I'll agree with you. Talk about reversals and inconsistencies and a lack of rigor.

>

> As for science, you can't do science without math.

Not so. Most of biology and the best psychology lacks math. That is changing, as it should. But the scientific method is logical, not mathematical. (If I may make that distinction.) And much great science

Re: debate: to get a Master's Degree in CS or Not

Re: debate: to get a Master's Degree in CS or Not

has been done without need for figures.

But in general, I agree. Math is called the queen of science for good reason.

....

>

> Hmm. What is the 'value' of a C++ program?

:-) Depends on who you ask. Language theorists will talk the issue to death. Me? I think its beauty is in the eye of the beholder. CS types are often mesmerized by the number and "power" of a language's features. I think C++ is phenomenally ugly, confused, and counterproductive.

Some day the software world is going to realize that the next step in programming must lie NOT with better programming tools, but with a better method of \*solving problems\* using software. When that happens, software tools should become a LOT simpler again, and more effort will go into the process of development and the verification and validation of the software solution WITHIN THE PROBLEM DOMAIN, as it should.

But for CS types, there's no money in doing that. :-(

Randy

.

---

• **Follow-Ups:**

- ◆ **Re: debate: to get a Master's Degree in CS or Not**  
◇ From: markwh04
- ◆ **Re: debate: to get a Master's Degree in CS or Not**  
◇ From: Morgan
- ◆ **Re: debate: to get a Master's Degree in CS or Not**  
◇ From: Walter Bright

• **References:**

- ◆ **debate: to get a Master's Degree in CS or Not**  
◇ From: xtcsonik
- ◆ **Re: debate: to get a Master's Degree in CS or Not**  
◇ From: Walter Bright
- ◆ **Re: debate: to get a Master's Degree in CS or Not**  
◇ From: slebetman@xxxxxxxxxx
- ◆ **Re: debate: to get a Master's Degree in CS or Not**  
◇ From: Walter Bright
- ◆ **Re: debate: to get a Master's Degree in CS or Not**  
◇ From: Randy
- ◆ **Re: debate: to get a Master's Degree in CS or Not**  
◇ From: Walter Bright

- Prev by Date: **Re: Card dealing and random repetition**

Re: debate: to get a Master's Degree in CS or Not

- Next by Date: ***Re: Card dealing and random repetition***
- Previous by thread: ***Re: debate: to get a Master's Degree in CS or Not***
- Next by thread: ***Re: debate: to get a Master's Degree in CS or Not***
- Index(es):
  - ◆ ***Date***
  - ◆ ***Thread***