= Premium Content Log In | Create a Free Account | Subscribe Now

THE CHRONICLE of Higher Education

Wednesday, November 10, 2010 Subscribe Today!

HOME NEWS OPINION & IDEAS FACTS & FIGURES TOPICS JOBS ADVICE FORUMS EVENTS

The Chronicle Review | Commentary | Brainstorm | Books | Letters to the Editor | Academic Destinations | Campus Viewpoints

Commentary

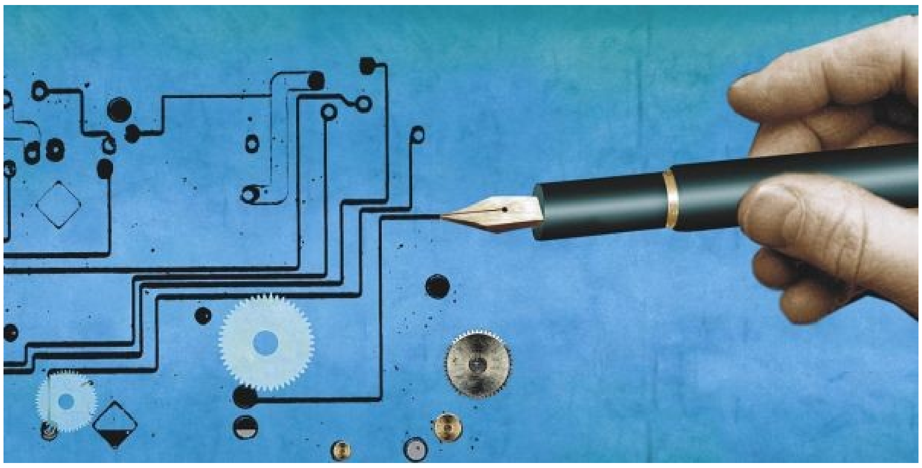
Search The Chronicle Search

Home > Opinion & Ideas > Commentary

E-mail Print Comment Share ARTICLE TOOLS

November 7, 2010

Decoding the Value of Computer Science



Michael Morgenstern for The Chronicle

By Kevin Carey

In *The Social Network,* a computer-programming prodigy goes to Harvard and creates a technology company in his sophomore dorm. Six year later, the company is worth billions and touches one out of every 14 people on earth.

Facebook is a familiar American success story, with its founder, Mark Zuckerberg, following a path blazed by Bill Gates and others like him. But it may also become increasingly rare. Far fewer students are studying computer science in college than once did. This is a problem in more ways than one.

The signs are everywhere. This year, for the first time in decades, the College Board failed to offer high-school students the Advanced Placement AB Computer Science exam. The number of high schools teaching computer science is shrinking, and last year only about 5,000 students sat for the AB test. Two decades ago, I was one of them.

I have never held an information-technology job. Yet the more time passes, the more I understand how important that education was. Something is lost when students no longer study the working of things.

My childhood interest in programming was a product of nature and nurture. My father worked as a computer scientist, first in a university and then as a researcher for General Electric. As a kid, I tagged along to his lab on weekends, watching him connect single-board DEC computers into ring networks and two-dimensional arrays, feeling the ozone hum of closet-sized machines. By my adolescence, in the mid-1980s, we had moved to a well-off suburb whose high school could afford its own mainframe. That plus social awkwardness meant many a night plugged into a 300-baud modem, battling other 15-year-olds in rudimentary deep-space combat and accumulating treasure in Ascii-rendered dungeons without end.

Before long I wanted to understand where those games came from and how, exactly, they worked. So I took to programming, first in Basic and then Pascal. Coding taught me the shape of logic, the value of brevity, and the attention to detail that debugging requires. I learned that a beautiful program with a single misplaced semicolon is like a sports car with one piston out of line. Both are dead machines, functionally indistinguishable from junk. I learned that you are

Most Popular

Most Viewed 1.

Most E-Mailed

Most Commented

Fusion Scientists Report Breakthroughs, Hoping for Bigger Budgets
Spotting Your Faculty Enemies
Nudging Provosts Toward a President's Office
The World Beyond Reach: Why Languages Are Indispensable
Reader Choice, Not Vendor Influence, Reshapes Library Collections
Reader Choice, Not Vendor Influence, Reshapes Library Collections
Students Lack Basic Research Skills, Study Finds
Elvis at the Beach
Nudging Provosts Toward a President's Office
Donations by the Wealthy Dropped Sharply in the Recession
Spotting Your Faculty Enemies
Things We Can Live Without
21st-Century Public Intellectuals
The World Beyond Reach: Why Languages Are Indispensable
Students Lack Basic Research Skills, Study Finds

Recent News

Arbitrator Orders Florida State U. to Rescind Tenured Faculty Layoffs

By Peter Schmidt

The independent arbitrator also slammed the university for how it had decided on what jobs to eliminate.

Want Students to Perform Well? Perhaps Give Them Money for Doing Just That

By Beckie Supiano

A new study shows that using financial aid more strategically could boost achievement.

Election Results Bring Reminders of '94

By Kelly Field

Higher-education cuts promised by the GOP wave 16 years ago didn't all materialize.

Chart: How Much Does Party Matter to Spending on Student Aid and Research?

good enough to build things that do what they are supposed to do, or you are not.

I left for college intending to major in computer science. That lasted until about the fifth 8:30 a.m. Monday class, at which point my enthusiasm for parties and beer got the upper hand, and I switched to the humanities, which offered more-forgiving academic standards and course schedules. I never touched Pascal again.

Fortunately, other students had more fortitude. They helped build and sustain the IT revolution that continues to make America the center of global innovation. But the number of people going this way is in decline. According to the Computing Research Association, the annual number of bachelor's degrees awarded in computer science and computer engineering dropped 12 percent in 2009. When Zuckerberg started Facebook, in 2004, universities awarded over 20,000 computer degrees. The total fell below 10,000 last year.

This "geek shortage," as *Wired* magazine puts it, endangers everything from innovation and economic growth to national defense. And as I learned in my first job after graduate school, perhaps even more.

There, at the Indiana state-budget office, my role was to calculate how proposals for setting local school property-tax rates and distributing funds would play out in the state's 293 school districts. I did this by teaching myself the statistical program SAS. The syntax itself was easy, since the underlying logic wasn't far from Pascal. But the only way to simulate the state's byzantine school-financing law was to understand every inch of it, every historical curiosity and long-embedded political compromise, to the last dollar and cent. To write code about a thing, you have to know the thing itself, absolutely.

Over time I became mildly obsessed with the care and feeding of my SAS code. I wrote and rewrote it with the aim of creating the simplest and most elegant procedure possible, one that would do its job with a minimum of space and time. It wasn't that someone was bothering me about computer resources. It just seemed like the right thing to do.

Eventually I reached the limit of how clean my code could be. Yet I was unsatisfied. Parts still seemed vestigial and wrong. I realized the problem wasn't my modest programming skills but the law itself, which had grown incrementally over the decades, each budget bringing new tweaks and procedures that were layered atop the last.

So I sat down, mostly as an intellectual exercise, to rewrite the formula from first principles. The result yielded a satisfyingly direct SAS procedure. Almost as an afterthought, I showed it to a friend who worked for the state legislature. To my surprise, she reacted with enthusiasm, and a few months later the new financing formula became law. Good public policy and good code, it turned out, go hand in hand. The law has reaccumulated some extraneous procedures in the decade since. But my basic ideas are still there, directing billions of dollars to schoolchildren using language recorded in, as state laws are officially named, the Indiana Code.

A few years later, I switched careers and began writing for magazines and publications like this one. It's difficult work. You have to say a great deal in a small amount of space. Struggling to build a whole world in 3,000 words, I thought back to stories my father would tell of consulting on the Galileo space-probe project. To make it work with 1970s technology and withstand the rigors of deep space, he and his fellow programmers had to fit every procedure—"take pictures," "fire rockets," "radio NASA back on Earth"—into a machine with 64K bytes of what we now call RAM. That's about 1/60,000th of what you can buy in a cheap laptop at Best Buy today. So every program on Galileo was polished to a gleam.

Good editors know there is no Moore's law of human cognition that would double our ability to retain and process information every 18 months. Instead, the technology-driven surfeit of modern information has made the need for clarity and concision more acute.

My editors rarely said a word about words, in the sense of how to phrase an idea. The real work was in structure, in constructing an unbroken chain of logic, evidence, and emotion that would bring readers to precisely where I wanted them to be. I learned to minimize the number of operating variables (people). I also learned that while some talented writers can get away with recursive scene-setting, it is usually best to start at the beginning and finish at the end. I discovered that skilled writers, like programmers, embed subtle pieces of documentation in their prose to remind readers where they are and where they are going to be.

Writing, in other words, is just coding by a different name. It's like constructing a program that runs in the universal human operating system of narrative, because, as Joan Didion once said, we tell ourselves stories in order to live. Authors adhere to a set of principles as structured in their own way as any computer language. Publications worth writing for insist on Galilean standards of quality and concision.

Computer science exposed two generations of young people to the rigors of logic and rhetoric that have disappeared from far too many curricula in the humanities. Those students learned to speak to the machines with which the future of humanity will be increasingly intertwined. They discovered the virtue of understanding the instructions that lie at the heart of things, of realizing the danger of misplaced semicolons, of learning to labor until what you have built is good enough to do what it is supposed to do.

I left computer science when I was 17 years old. Thankfully, it never left me.

Kevin Carey is policy director of Education Sector, an independent think tank in Washington.

Comments

1. 11122741 - November 08, 2010 at 11:35 am

As a cogntive psychologist and someone who teaches cognitive psychology and education, all I can say is great article Ken you hit all of the nails on the head and this article is no kludge. There are so many things that i really cannot teach this generation and that they have little understanding of as they simply have no concept of how things work ...really work ...they do not know things experientially and absolutely; they are slowly becoming the Eloi in Well's novel the Time Machine. It is one of the real downsides of miniturization, automation and "appliance" technology in all areas. programming should be the New Latin.

2. inarchetype - November 08, 2010 at 04:03 pm

Outstanding article. One of the best I've read lately.

Computer programming came easily when I took my first course in it as an undergraduate because I first took a course in the philosophy department called symbolic logic, composed entirely of writing Fisk proofs. It I have long beleived that everyone should be forced to take symbolic logic for all the reasons you discuss, in that it is more general than computer programming.

But computer programming offers the majority of the same benefits, in a package more accessible to those who have less tolerence for abstraction, and less patience with strictly academic pursuits. It is hands-on, useful, fun, and you get to see what you made working. But if you logic is flawed, your program won't work, so it inherently trains rigor and coherance in a way that less structured activities do not.

So I agree with the anonymous first poster. I used to think that formal logic should be the new latin, but am now convinced that programming has the edge.

The only problem is that, if taught with any rigor, it seems to be the sort of thing that some people can't- that there is a strongly bi-modal distribution of performance in programming courses. Typically, those who really aren't cut out for it, whose brains simply aren't wired that way, simply choose other things to do. If training of this sort were made universal, I fear that to give everyone a fair shot at success would require dumbing it down such that it would loose much of its value.

3. fizmath - November 08, 2010 at 08:51 pm

There is no geek shortage. That is industry propaganda to scare Congress into raising the annual cap on H1-B visas. Microsoft and Oracle reject 98% of their applicants. The industry is doing everything it can to offshore IT jobs. Look up the writings of Norm Matloff for a better explanation of this issue.

Other than that, learning how to program is very beneficial, even if you won't get a job with it.

4. robertwgehl - November 09, 2010 at 09:33 pm

I would love to see far more connection between computer science/software engineering and writing. I recently read Fred Brooks's classic The Mythical Man-Month and am struck by how readily his prescription for software design maps onto the ways in which I've been teaching composition. Brooks calls for two stages: architecture and implementation. The architecture is the overall goal of the software - what it does, what it means to the user. The implementation is the nitty-gritty details that bring the architecture to life. It parallels the thesis-support model that so many students seem to struggle with these days.

5. tuxthepenguin - November 10, 2010 at 06:54 am

You studied computer science before Java. Object oriented programming is more interesting to computer scientists, but is not a good introduction to programming. In the old days it was BASIC and Pascal and Fortran that were used in introduction to programming courses. Today it's Java. The object oriented approach is better but just too complicated.

6. 11167997 - November 10, 2010 at 07:06 am

Kevin: stick with writing like this because (a) it touches a lot of us, and (b) in this case, it validates your own journey. I wrote first and discovered programming (and SAS) in my mid-40s. Somewhere in that mid-life interstice, I thought, "haven't I seen this somewhere before?" There it was in a box in the basement: the mimeographed text for my freshman college math course, named something like Math 1: it was finite math, symbolic logic, binary coding. The epiphany enhanced the programming, but combined with the prior writing trajectory, it meant that I could make numbers sing. That doesn't leave you, either, and it's never too late.

7. 3224243 - November 10, 2010 at 07:17 am

I guess I'm a geek. I preferred DOS to Windows, enjoyed compiling a SPSS routine much more in DOS than using the graphical interface. The challenge of writing a routine that work (finally) was a pleasure never matched by the visual versions of the same programs.

It's like childhood was - back in my "day," we had to figure out how to get enjoyment by creating from what was around us. Most of today's kids don't create - they consume (without an understanding of where it came from or how it came to be) and compile (from a variety of sources without taking a moment to evaluate the substance of the content). They've lost the curiosity because they're too occupied with their smart phones. They have no time to just think and wonder because their lives have been scheduled to the minute by Mom and Dad.

How sad.

8. csgirl - November 10, 2010 at 08:52 am

I am a geek - but I learned object oriented programming first, and still find it much easier. The only programming language that ever defeated me was Basic, with its twisting morass of spaghetti. So I fundamentally disagree with the poster who thinks it is a better introductory language.

9. csgirl - November 10, 2010 at 08:56 am

I also want to comment on the sad state of computer science. When I was teaching in the 90's, we used to get lots of students who were hobbyists - they had run their own bulletin boards, written their own little games, opened up their computers and tweaked them. They had a creative, can-do, spirit and were genuinely interested in the inner workings of almost everything. The students I have now are like lumps. If the slightest thing goes wrong, they throw up their hands and cry "Professor! Come fix this!". In their high school "technology" courses, they were taught to make PowerPoint presentations rather than to program or play with logic circuits. They have no curiousity, and cannot think out of the box. It is very sad, because without that tinkering, creative spirit, I don't know what will become of the U.S.

10. fundmaven - November 10, 2010 at 08:59 am

Beautiful article! It's well-written and carries a poignant, if not fearful, portrait of intellectual malaise in our culture.
However, I am not so sure that the phrase "forgiving standards" accurately depicts the rigor of true academics in the humanities.
I submit that a thorough study into the complexities of economics, History (NOT "social

sciences"),language, and literature (including poetry)provides context in the understanding of markets and cultures that influence them. It also promotes the abilities to express one's ideas fluently and cogently whether in code, technical prose, or poetry.

11. ronknott - November 10, 2010 at 09:03 am

In the 70s, I was an English major in the same college where my father was an English professor. After I took an introductory computer programming class (Fortran) to meet a core requirement, I flirted with taking a computer science minor, and signed up for advanced computer programming. Part way through the course I announced to my father at the supper table that his English department ought to require all English majors to take the same course. It was all about logic and rhetoric. I got the lowest grade in that course of any during my college career (I got distracted by other college fun). It blasted my GPA and was the end of my computer science minor. But I admired everything about the discipline of thought it required of me. I need to take a similar class again.

Add Your Comment

You must be logged in to add a comment. Please login now or create a free account.

Teaching

Snooze Through U.

History

The 'Me' Decade

Reaching Out

College as Philanthropy

HOME | NEWS | OPINION & IDEAS | FACTS & FIGURES | TOPICS | JOBS | ADVICE | FORUMS | EVENTS

Subscribe | Newsletters | Advertise | Help | About The Chronicle | Contact Us | Site Map | Permissions | Privacy Policy