DOI: 10.1145/1364782.1364794

Len Shustek

Interview

The 'Art' of Being Donald Knuth

In this first of a two-part talk, the renowned scholar and computer scientist reflects on the influences that set the course for his extraordinary career.

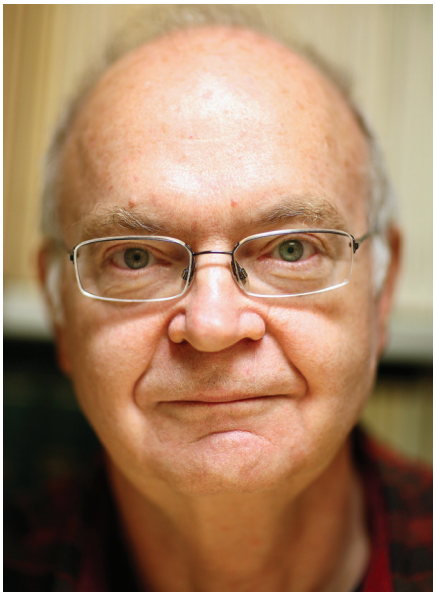
THE COMPUTER HISTORY Museum has an active program to gather videotaped histories from people who have done pioneering work in this first century of the information age. These tapes are a rich aggregation of stories that are preserved in the collection, transcribed, and made available on the Web to researchers, students, and anyone curious about how invention happens.

The oral histories are conversations about people's lives. We want to know about their upbringing, their families, their education, and their jobs. But above all, we want to know how they came to the passion and creativity that leads to innovation.

Presented here in two installments (concluding next month) are excerpts[a] from an interview conducted by Edward Feigenbaum in March 2007 of Donald E. Knuth, Professor Emeritus of The Art of Computer Programming at Stanford University. — L. S.

Don talks about his family background.

My father was the first person among all his ancestors who had gone to college. My mother was the first person in all of her ancestors who had gone to a year of school to learn how to be a typist. My great-grandfather was a blacksmith. There was no tradition in our family of higher education at all. These people were pretty smart, but they didn't have an academic background.



PHOTOGRAPH BY TIMOTHY ARCHIBALD

Some people know from an early age what they want to do. Don didn't, but he knew he wanted to work hard.

My main interest in those days was music. But at the college where I had been admitted, people emphasized how easy it was going to be there as a music major. When I got the chance to go to Case Institute of Technology in Ohio instead, I was intrigued by the idea that Case was going to make me work hard. I was scared that I was going to flunk out, but still I was ready to work.

He initially aspired to be a physicist, but something happened along the way.

In my sophomore year in physics I had to take a required class of welding. Welding was so scary and I was a miserable failure at it, so I decided maybe I can't be a physicist. On the other hand—mathematics! In the sophomore year for mathematicians, they give you courses on what we now call discrete mathematics, where you study logic and things that are integers instead of continuous quantities. I was drawn to that. That was something, somehow, that had great appeal to me.

I think that there is something strange inside my head. It's clear that I have much better intuition about discrete things than continuous things. In physics, for example, I could pass the exams and I could do the problems in quantum mechanics, but I couldn't intuit what I was doing. But on the other hand, in my discrete math class, these were things that really seemed a part of me. There's definitely something in how I had developed by the time I was a teenager that made me understand discrete objects, like zeros and ones of course, or things that are made out of alphabetical letters, much better than things like Fourier transforms or waves.

I'm visualizing the symbols. To me, the symbols are reality, in a way. I take

a Oral histories are not scripted, and a transcript of casual speech is very different from what one would write. I have taken the liberty of editing and reordering freely for presentation. For the original transcript, see http://archive.computerhistory.org/search/oh/

My first program taught me a lot about the errors that I was going to be making in the future, and also about how to find errors. That's sort of the story of my life, making errors and trying to recover from them. I try to get things correct. I probably obsess about not making too many mistakes.

a mathematical problem, I translate it into formulas, and then the formulas are the reality.

He discovers computers, and how hard programming is.
I wrote my first program for the IBM 650 [a vacuum tube magnetic drum computer from the 1950s], probably in the spring of my freshman year, and debugged it at night. The first time I wrote the program, to find the prime factors of a number, it was about 60 instructions long in machine language. They were almost all wrong. When I finished, it was about 120 or 130 instructions. I made more errors in this program than there were lines of code!

My first program taught me a lot about the errors that I was going to be making in the future, and also about how to find errors. That's sort of the story of my life, making errors and trying to recover from them. I try to get things correct. I probably obsess about not making too many mistakes.

At Case he learns about early compilers
For the IT ("Internal Translator") program for the 650 you would punch an algebraic formula on cards and feed the cards into the machine. The lights spin around for a few seconds and then out come machine language instructions that set X1 equal to X2 + X4. Automatic programming coming out of an algebraic formula! Well, this blew my mind. I couldn't understand how it was possible to do this miracle. I could understand how to write a program to factor numbers, but I couldn't understand how to write a program that would convert algebra into machine instructions.

It hadn't yet occurred to him that the computer was a general symbol-manipulating device?
No. That occurred to Lady [Ada] Lovelace, but it didn't occur to me. I'm slow to pick up on these things, but then I persevere.

I got hold of the source code for IT. I went through every line of that program. During the summer we typically had a family get-together on a beach on Lake Erie where we spent time playing cards and playing tennis. But that summer, I spent most of the time going through this listing, trying to find out the miracle of how IT worked. Okay, it wasn't impossible after all. In fact, I thought of better ways to do it than were in that program.

The code, once I saw how it happened, was inspiring to me. Also, the discipline of reading other people's programs was something good to learn early. Throughout my life I've had a love of reading source materials—reading something that pioneers had written and trying to understand their thought processes, especially when they're solving a problem I don't know how to solve. This is the best way for me to get my own brain past the stumbling blocks. At Case I remember looking at papers that [Pierre de] Fermat had written in Latin in the 17th century, in order to understand how that great number theorist approached problems.

But it's been hard to communicate the love of reading historical programs.
I would say that's my major disappointment with my teaching career. I was not able to get across to any of my students this love for that kind of

scholarship—reading source material. I was a complete failure at passing this on to the people that I worked with the most closely.

He graduates from Case and becomes a professional compiler writer while traveling to the California Institute of Technology for graduate school.
I had learned about the Burroughs 205 machine language, and it was kind of appealing to me. So I made my own

PHOTOGRAPH BY TIMOTHY ARCHIBALD

proposal to Burroughs. I said, "I'll write you an ALGOL compiler for $5,000. But I can't implement all of ALGOL for this; I am just one guy. Let's leave out procedures." Well, this is a big hole in the language! Burroughs said, "No, you've got to put in procedures." I said, "Okay, I will put in procedures, but you've got to pay me $5,500." That's what happened. They paid me $5,500, which was a fairly good salary in those days. So between graduating from Case and going to Caltech, I worked on this compiler. Heading out to California, I drove 100 miles each day and then sat in a motel and wrote code.

But he rejects "compiler writer" as a career, and decides what is important in life.

Then a startup company came to me and said, "Don, write compilers for us and we will take care of finding computers to debug them. Name your price." I said, "Oh, okay, $100,000," assuming that this was [outrageous]. The guy didn't blink. He agreed. I didn't blink either. I said, "I'm not going to do it. I just thought that was an impossible number." At that point I made the decision in my life that I wasn't going to optimize my income.

I spent a day that summer looking at the mathematics of how fast linear probing works. I got lucky, and I solved the problem. I figured out some math, and I kept two or three sheets of paper with me and I typed

it up.[b] This became the genesis of my main research work, which developed not to be working on compilers, but to be working on the analysis of algorithms. It dawned on me that this was just one of many algorithms that would be important, and each one would lead to a fascinating mathematical problem. This was easily a good lifetime source of rich problems to work on.

If you ask me what makes me most happy, number one would be somebody saying "I learned something from you." Number two would be somebody saying "I used your software."

At Caltech he finds a mentor, but can't talk to him.

I went to Caltech because they had [strength] in combinatorics, although their computing system was incredibly arcane and terrible. Marshall Hall was my thesis advisor. He was a world-class mathematician, and for a long time had done pioneering work in combinatorics. He was my mentor. But it was a funny thing, because I was in such awe of him that when I was in the same room with him I could not think straight. I wouldn't remember my name. I would write down what he was saying, and then I would go back to my office so that I could figure it out. We couldn't do joint research together in the same room. We could do it back and forth.

He also was an extremely good advisor, in better ways than I later was with my students. He would keep track of me to make sure I was not slipping. When I was working with my own graduate students, I was pretty much in a mode where they would bug me instead of me bugging them. But he would actually write me notes and say, "Don, why don't you do such and such?"

The research for his Ph.D. thesis takes an hour.

I got a listing from a guy at Princeton who had just computed 32 solutions to a problem that I had been looking at for a homework problem in my combinatorics class. I was riding up on the elevator with Olga Todd, one of our professors, and I said, "Mrs. Todd, I think I'm going to have a theorem in an hour. I am going to psyche out the rule that explains why there happen to be 32 of each kind." Sure enough, an hour later I had seen how to get from each solution on the first page to the solution on the second page. I showed this to Marshall Hall. He said, "Don, that's your thesis. Don't worry about this block design with $\lambda=2$ business. Write this up instead and get out of here." So that became my thesis. And it is a good thing, because since then only one more design with $\lambda=2$ has been discovered in the history of the world. I might still be working on my thesis if I had stuck to that problem. But I felt a little guilty that I had solved my Ph.D. problem in one hour, so I dressed it up with a few other chapters of stuff.

If you ask me what makes me most happy, number one would be somebody saying "I learned something from you." Number two would be somebody saying "I used your software."

He's never had trouble finding problems to work on.

The way I work it's a blessing and a curse that I don't have difficulty thinking of questions. I have to actively suppress stimulation so that I'm not working on too many things at once. The hard thing for me is not to find a problem, but to find a good problem. One that will not just be isolated to something that happens to be true, but also will be something that will have spin-offs, so that once you've solved the problem, the techniques are going to apply to many other things.

He starts *The Art of Computer Programming*.

A man from Addison-Wesley came to visit me and said "Don, we would like you to write a book about how to write compilers." I thought about it and decided "Yes, I've got this book inside of me." That day I sketched out—I still have that sheet of tablet paper—12 chapters that I thought should be in such a book. I told my new wife, Jill, "I think I'm going to write a book." Well, we had just four months of bliss, because the rest of our marriage has all been devoted to this book. We still have had happiness, but really, I wake up every morning and I still haven't finished the book. So I try to organize the rest of my life around this, as one main unifying theme.

George Forsythe [founder of the Computer Science Department at Stanford] came down to southern California for a talk, and he said, "Come up to Stanford. How about joining our faculty?" I said "Oh no, I can't do that. I just got married, and I've got to finish this book first. I think I'll finish the book next year, and then I can come up [and] start thinking about the rest of my life. But I want to get my book done before my son is born." Well, John is now 40-some years old and I'm not done with the book.

This is really the story of my life, because I hope to live long enough to finish it. But I may not because it's turned out to be such a huge project.

1967 was a big year.

It was certainly a pivotal year in my life. You can see in retrospect why I think things were building up to a crisis, because I was just working at high pitch all the time. I was on the editorial board of *Communications of the ACM* and *Journal of the ACM*—working on their programming languages sections—and I took the editorial duties very seriously. I was a consultant to Burroughs on innovative machines. I was consumed with getting *The Art of Computer Programming* done. And I was a father and husband. I would start out every day saying "Well, what am I going to accomplish today?" Then I would stay up until I finished it.

It was time for me to make a career decision. The question was where should I spend the rest of my life?

b "Notes on Open Addressing." Unpublished memorandum, July 22, 1963; but see http://algo.inria.fr/AofA/Research/11-97.html

Should I be a mathematician? Should I be a computer scientist? By this time I had learned that it was actually possible to do mathematical work as a computer scientist. I had analysis of algorithms to do. What would be a permanent home? My model of my life was going to be that I was going to make one move in my lifetime to a place where I had tenure, and I would stay there forever.

The crisis comes.
At Caltech, I was preparing my class lectures, or typing my book. I didn't have time to do research. If I had a new idea, if I said "Here's a problem that ought to be solved," when was I going to solve it? Maybe on the airplane. We were doing a lot of experiments but I didn't have time to sit down at home and work out the theory for it. I had attribute grammars coming up in February, and these reductions systems coming up in March, and I was supposed to be grinding out Volume Two of *The Art of Computer Programming*. I was scheduled in June to lecture at a summer school in Copenhagen about how to parse, what's called top-down parsing.

I told my new wife, Jill, "I think I'm going to write a book." Well, we had just four months of bliss, because the rest of our marriage has all been devoted to this book. We still have had happiness, but really, I wake up every morning and I still haven't finished the book. So I try to organize the rest of my life around this, as one main unifying theme.

What happened then, in May, is I had a massive bleeding ulcer, and I was hospitalized. My body gave out. I was just doing all this stuff, and it couldn't take it.

I learned about myself. The doctor showed me his textbook that described the typical ulcer patient: what people call the "Type A" personality. It described me to a T. All of the signs were there. I was an automaton, I think, basically. I saw a goal and I put myself to it, and I worked on it and pushed it through. I didn't say no to people when they asked, "Don, can you do this for me?" At this point I saw I had this problem. I shouldn't try to do the impossible.

He changes his lifestyle, and moves to Stanford.
I wrote a letter to my publisher, framed in black, saying, "I'm not going to be able to get the manuscript of Volume Two to you this year. I'm sorry." I resigned from 10 editorial boards. No more JACM, no more CACM. I gave up all of the editorships in order to cut down my workload. I started working on Volume Two where I left off at the time of the ulcer, but I would be careful to go to sleep and keep a regular schedule. I went to a conference in Santa Barbara on combinatorial mathematics and had three days to sit on the beach and develop the theory of attribute grammars, this idea of top-down and bottom-up parsing.

In February of 1968 I finally got the offer from Stanford. The committees were saying, "This guy is just 30 years old." But when they looked at the book, they said, "Oh, there's some credibility here." That helped me.

Why he writes his books with a pencil.
I love keyboards, but my manuscripts are always handwritten. The reason is that I type faster than I think. There's a synchronization problem. I can think of ideas at about the rate I can write them down with a pencil. But with typing I'm going faster, so I have to sync, and my thoughts have to start up and stop again in a way that involves more of my brain.

Three volumes of "The Art" are done, but it's time for a pause.
Volume Four is about combinatorial algorithms. Combinatorial algorithms were such a small topic in 1962, when I made that Chapter Seven of my outline, that Johan Dahl asked me, "How did you ever think of putting in a chapter about combinatorial algorithms in 1962?" I said, "Well, the only reason was that it was the part I thought was most fun." But there was almost nothing known about it at the time.

The way I look at it, this is where you've got to use some art. You've got to be really skillful, because one good idea can save you six orders of magnitude and make your program run a million times faster. People are coming up with these ideas all the time. For me, the combinatorial explosion was the explosion of research in combinatorics. Not the problems exploding, but the ideas were exploding. There's that much more to cover now.

It's true that in the back of my mind I was scared stiff that I can't write Volume Four anymore. So maybe I was waiting for it to simmer down. Somebody did say to me once, after I solved the problem of typesetting, maybe I would start to look at binding or something, because I had to have some other reason [to delay]. I've certainly seen enough graduate student procrastinators in my life. Maybe I was in denial. C

He solves the problem of typesetting? Stay tuned for Part II of this interview in the August issue and learn how Knuth interrupted his life's work on The Art of Computer Programming *to create a system that makes digitally produced books beautiful.*

Edited by **Len Shustek**, Chair, Computer History Museum