Tropp, is that Judge Larson unfairly dismissed Mauchly's work in digital computing prior to his visit to Atanasoff's laboratory, for which Mauchly's only "evidence" is that certain of his Ursinus students could have given supporting testimony if they had been called upon to do so. A judge, of course, must assume that the attorneys on each side have presented their best possible case. It also happens that the machine Mauchly did build in the late 1930s was an analog device, his electrical harmonic analyzer. It was this device, in fact, and the results he had obtained from it, that he presented in his December 1940 talk before the AAAS—a talk to which he here refers without identification. At least one member of his audience was responsive, too: Atanasoff; this is how Mauchly met Atanasoff.

Another theme is that Mauchly did not learn anything about electronic computing from Atanasoff—related with a note of bitterness over insinuations made from his brief contact with the Iowa scientist. We believe the case for John Atanasoff's original work in this area and its influence on the ENIAC *through Mauchly will ultimately be understood and accepted as the true situation. Certainly no one has yet produced any solid counterevidence to the findings of the court in the essentials of this case, or against the facts as we presented them in the Annals.*

Still another of Mauchly's themes is that any connection between the differential analyzer and the ENIAC *was just part of his strategy, his secret weapon for tricking the U.S. Army into funding his idea. It strikes us as ridiculous to imagine that Mauchly could have deceived visitors from Aberdeen or mathematician Goldstine in this way—or that he felt any need to. It occurs to us that Mauchly's portrayal here may be intended to counter the role of the differential analyzer found by Judge Larson, which we explained in our article; namely, that the idea Atanasoff gave Mauchly for converting his (Atanasoff's) electronic computer to do the work of the analyzer contributed significantly to the invention of the* ENIAC. *Mauchly's story also deprives Goldstine of deserved credit for seeing the potential of such a machine as soon as Mauchly told him of it. But whatever its purpose, it is only that—a story, completely undocumented, and, in our opinion, fantasy in hindsight.*

A final, and less important, theme is the one about money—or lack thereof. The depression years were indeed as Mauchly describes them, although it would seem he might have been able to design and build a low-speed counter with vacuum tubes while at Ursinus College. There seems no reason, however, that he could not have put his ideas for high-speed computation to work at the Moore School, by 1941–1942, where a laboratory with oscilloscopes was available to him. We believe he has extended the wherewithal excuse beyond its applicable period, rather than face the fact that the high-speed counter problem was not solved until the ENIAC *was being designed in 1943, at which point it was solved primarily by Eckert.*

Mauchly also commits serious errors of omission in these two works. Most striking is his failure to give the names of people whose roles he describes, or of corporations or projects he mentions. It must be said that, throughout, there is a pattern of insufficient detail, as well as incorrect detail, such that verification of his statements is rendered impossible. The historian is left with a rather grandiloquent picture of events and circumstances, unsupported by concrete evidence, against which must be weighed not just the word of other actors in the drama but an overwhelming amount of evidence and documentation.

The so-called fireside chat is especially strained in this regard. It is of course understandable that, after many years of adulation for one of the most far-reaching inventions in history, a person might tend to exaggerate in the glow of a "fireside" meeting with his corporate associates. We deem it less understandable that he could distort and mislead, with such equanimity, only 25 days after a federal court had ruled "his" invention derived from another, and his (joint) patent accordingly invalid—moreover, that he could do so without once mentioning either the court's decision or the person whose ideas he had wrongly appropriated.

We will close with a brief comment on Mauchly's letter to H. Helm Clayton, since it too is being published, confining ourselves to the paragraph on Mauchly's computing interests at the time (November 15, 1940). The first reference, to a "proposed 27-ordinate analyser," is to an analog device, the electrical harmonic analyzer we mentioned in Section 3 of our Annals article (and referred to in this Afterword). The second reference, to considering "construction of an electrical computing machine," is to a digital device, what we called in that same section a "keyboard-operated digital calculator based on electronic counters." We do not believe that Mauchly built any electronic counter that could be part of such a machine—certainly not before he visited Atanasoff and very probably not before the ENIAC *project began.*

Arthur W. Burks and Alice R. Burks
Department of Computer and Communication Sciences
University of Michigan
221 Angell Hall
Ann Arbor, MI 48109

Biographies

Editor's Note

The following is an informal verbatim interview, a modern form of biography made possible by the technological development of the cheap, portable, reliable tape recorder. It is autobiographical in that the subject tells his story in his own words, and it is biographical in that the interviewers guide the talk with their questions.

The Annals *is reprinting this interview, more a monologue than a conversation, because Donald E. Knuth is one of the giants of computing. We would like to understand how he developed, what he thought, and how he came by this idea or that as he created his many contributions, not the least of which is the brilliant clarity and comprehensiveness with which he has expressed and explained them. Reprinting is also justified because of the very small intersection between the readership of the* Annals *and the* Two-Year College Mathematics Journal, *the original publisher.*

—Eric A. Weiss

A Conversation with Don Knuth

DONALD J. ALBERS AND LYNN ARTHUR STEEN

As a high school senior in Milwaukee, Donald E. Knuth had doubts about his ability to graduate from college. Four years later he received his B.S. in mathematics, *summa cum laude*, from the Case Institute of Technology in 1960. His work had been so distinguished that by a special (unprecedented) vote of the faculty he was simultaneously awarded an M.S. degree. In 1963 he received his Ph.D. in mathematics from the California Institute of Technology. Over the years he has received many prestigious awards. In 1979, at age 41, he was awarded the National Medal of Science by President Carter.

By any measure, Don Knuth is a remarkable man. He is generally regarded as the preeminent scholar of computer science in the world. He also is an accomplished organist, composer, and novelist.

He is a prolific writer on a host of topics, and he has contributed to an unusually large number of publications. His first publication was for *MAD Magazine*. Since then he has written for *Datamation*, the *Journal of Recreational Mathematics*, the *American Mathematical Monthly*, and dozens of other mathematics and computer science journals such as *Acta Arithmetica* and *Acta Informatica*. He is best known for his monumental series of books, *The Art of Computer Programming*, which has been translated into several languages, ranging from Chinese to Russian. Three of a projected seven volumes in the series are now completed, with part of the fourth in preprint form. His progress on the series has been slowed by a current four-year-long diversion on computer-assisted typesetting.

On January 12, 1981, Donald J. Albers, editor of the TYCMJ, and Lynn Arthur Steen, past editor of *Mathematics Magazine*, interviewed him in his Stanford office. What follows are excerpts from that interview.

TYCMJ. *You are a computer scientist, and yet you started out in mathematics. When did your mathematical interest first emerge?*

Knuth. In my freshman year of high school I got very interested in mathematics. In fact, I think I ruined my eyes drawing hundreds of graphs on orange graph paper with dim lighting. I started to get headaches from drawing those graphs, but I was fascinated with them. The typical graph would be some function like $y = \sqrt{ax + b} - \sqrt{cx + d}$, where I would fix b, c, and d and vary a, in order to see that would happen to the shape of the graph. I had hundreds and hundreds of such graphs where I wanted to see the behavior of functions.

Reprinted with the permission of the *Two-Year College Mathematics Journal*, an official publication of the Mathematical Association of America.
Authors' Address: Two-Year College Mathematics Journal, Menlo College, Menlo Park, CA 94025.
Categories and Subject Descriptors: A.0 [General]—*biographies, D. E. Knuth*; K.2 [History of Computing]—*D. E. Knuth.*
General Terms: Algorithms, Human Factors
AFIPS 0164-1239/82/030257-274/00

President Carter presenting the National Medal of Science to Knuth.

TYCMJ. *Did you have an outstanding teacher along the way?*

Knuth. The mathematics teacher that I remember most and who inspired me the most was in college. My high school senior teacher also introduced me to things like binary numbers and encouraged me to do recreational things. During my senior year in high school, I entered the Westinghouse Science Talent Search. I made two entries. One was sort of physics oriented, and the other was a number system based on π. I had thought about imaginary number bases and irrational number bases, and I played around with the kind of logarithm tables that would result from such bases. I didn't win the prize, but I do remember having a lot of fun thinking about number systems. I also played around a lot with absolute-value functions. When I learned about the absolute-value function, I started making another set of graphs. I worked out a system so that if somebody gave me a pattern of connected straight lines, I would be able to write down a function whose graph gave that pattern. I was absolutely fascinated with graphs in mathematics. My physics teacher was my favorite teacher in high school. I was torn between physics and music, but I enrolled in college as a physics major. I had done a lot of piano playing and some orchestrations, so I didn't know whether I should major in music or physics. My choice of majors was due essentially to the different scholarships I got. The college I chose was better in physics than in music. If I had gone to Valparaiso instead of Case, I would have majored in music.

TYCMJ. *Is there any other music or mathematics talent in your family?*

Knuth. My dad was a church organist, and now I am.

TYCMJ. *Do you think there is much to the suggested connection between musical ability and mathematical ability?*

Knuth. There is definitely something to it. Go, for example, to the Mathematics Institute at Oberwolfach in Germany. Every week mathematicians come there for conferences, and music is the main recreation. They have a tremendous music library and many people will come with their instruments. Chamber music fills the halls almost every night as the mathematicians get together. In our department now we also are surrounded by chamber music. I just was talking to the administrative assistant of our department about this. She had previously worked in the law school, and she said in the law school one professor out of twenty might go to a concert once in a while. They weren't that much interested in music. Here in the Computer Science Department she felt that more than half the people were musicians themselves.

TYCMJ. *Do you have a theory as to what you perceive as a real connection between music and mathematics?*

Knuth. No, I really don't understand why. I guess Euler liked music, but I can't say how many great mathematicians of bygone days were really good musicians. I haven't studied that. There is definitely a correlation, certainly at Stanford.

I also found this in my friend, Professor Dahl from Norway, who carries piano duet music all of the time with him in his briefcase. No matter where he goes, he finds someone to play with him. He was the one who introduced me to the beauty of four-hands piano music, and now I have quite a collection of it built up over the last 10 years.

TYCMJ. *I wonder if we could get back to your early schooling experiences from a different angle, namely, writing. When did your writing interests first emerge?*

Knuth. Our grade school was very good in English grammar. I remember one of the most interesting things for me in the seventh and eighth grades was to diagram sentences. A bunch of us would get together after class to try diagramming. We could diagram all of the sentences in the English book, but we couldn't diagram many of the other sentences we saw around us, especially the ones we saw in the hymnal. We couldn't figure out what was going on. They just didn't fit any of the rules we learned. We worked hard on this, and it was a big thing for us at the time. In high school, I found out that everyone from our grade school was whizzing through the English classes because of what we had learned; so it wasn't just me.

TYCMJ. *I was going to ask you if this was a public school.*

Knuth. It was a Lutheran school. My dad was a Lutheran school teacher, and I attended a Lutheran high school in Milwaukee. My grade school education in writing was really good. I'm trying to do that for my kids now. I have them write an essay every week. If they want permission to watch television the next week, they have to turn in their essay the previous week. When I got to college, I found out that writing was almost 50 percent of what I had to do well, and the other half was mathematics.

The other strange thing I remember about grade school occurred when I was in the eighth grade. There was a contest run by the manufacturers of Ziegler's Giant Bar in Milwaukee. The contest consisted of trying to find how many words could be made out of the letters in "Ziegler's Giant Bar." This contest appealed to me very much, and I told my parents I had a stomachache so that I didn't have to go to school for two weeks. I spent all those two weeks with an unabridged dictionary finding all the words I could get from the specified letters. I wound up with about 4500 words, and the judges had only 2500 on their master list. Afterwards I realized that I could have made even more words if I had used the apostrophe! My dad and mom helped type up the answers I had written out when they saw how interested I was in this project. The prize was a television set for the school. So our school got a TV set in the classroom, and we got to watch during class that year (1952) one of the first live transmissions from San Francisco across the country. We also got a Ziegler's Giant Bar for everyone in the class.

Professor Guenther was my freshman calculus teacher at Case, and he first exposed me to higher mathematics. Paul Guenther died about five years ago, but he was a great teacher for me mostly because he was so hard to impress. Every time I made a suggestion, I was put down, but he would grudgingly appreciate it when I finally came up with a good one. I don't know why I got so excited about him. In addition to his unimpressibility, he had a good sense of humor, and he seemed to know as much physics as my physics teacher and as much chemistry as my chemistry teacher. That impressed me: it seemed that mathematics was a little better somehow.

Also, I was scared stiff that I wasn't going to make it in mathematics. My advisors in high school told me that I had done well so far, but they didn't think I could carry it on in college. They said college was really tough, and the dean had told us that one out of three would fail in the first year. In high school, I did have the all-time record for grades. We were graded not on A, B, C, D, but on percentages in every course. And my overall percentage in classes was better than 97.5.

Knuth is an accomplished organist and composer. "I want to write some music for organ with computer help. If I live long enough, I would like to write a rather long work that would be based on the book of Revelation. The musical themes would correspond to the symbolism in the book of Revelation." (Photo by Tom Black)

TYCMJ. *Didn't that instill in you a great deal of confidence?*

Knuth. I always had an inferiority complex—that's why I worked so hard. I was an overachiever probably.

At Case, I spent hours and hours studying the mathematics book we used—*Calculus and Analytic Geometry* by Thomas—and I worked every supplementary problem in the book. We were assigned only the even-numbered problems, but I did every single one together with the extras in the back of the book because I felt so scared. I thought I should do all of them. I found at first that it was very slow going, and I worked late at night to do it. I think the only reason I did this was because I was worried about passing. But then I found out that after a few months I could do all of the problems in the same amount of time that it took the other kids to do just the odd-numbered ones. I had learned enough about problem solving by that time that I could gain speed, so it turned out to be very lucky that I crashed into it real hard at the beginning.

When not playing the organ, computer scientist Knuth occasionally may be found playing in the backyard. (Photo by Tom Black)

I started as a physics major, but my turn toward mathematics came in my sophomore year. I took a course in abstract mathematics from Professor Green, who is still teaching at Case. He had written his own textbook for the class, where he would give axioms for Boolean algebra, logic, etc. All of a sudden I realized that it was something I liked very much. And he gave a special problem without telling us whether it was possible or not. He said that if anyone could solve the problem, they would get an automatic A in the course. So, of course, none of us tried it. It was obviously hopeless, for he had quite a reputation. As far as we were concerned, there was no way to do it.

I was in the marching band that fall. But I missed the bus and had nothing to do, so I decided to kill time by working on that impossible problem. By a stroke of luck I was able to solve it, so I handed a solution in on Monday. He said, "Okay, Knuth, you get an A in the course." I cut class the rest of the quarter, and he lived up to his bargain. I felt guilty about cutting classes afterwards, so I became the grader for the course the next year. Instead of doing the homework, I was grading it.

In physics I was having a terrible time in the welding lab. I never was very good in laboratories, either in chemistry or in physics. My experiments just wouldn't work. I would drop things on the floor, and would always be the last one to finish. Once I had to report an experimental error of 140 percent in chemistry lab. I objected to their formula for experimental error, since I thought nobody could be more than 100 percent wrong, but they wouldn't listen to me.

In welding it was even worse. I was too tall for the welding tables, and my eyes weren't good enough; I couldn't wear my glasses underneath the goggles. Everything would go wrong, and I was terrified by what seemed like hundreds of thousands of volts of electricity! I just wanted nothing to do with it, yet physics majors were required to do this lab work.

On the other hand, I had Professor Green's course in abstract mathematics, which seemed very appealing to me. Just for fun I had made up sort of random axioms for what turned out to be a ternary logic, something that looked a little like Boolean algebra. The idea was to see if those axioms would lead to any theorems. I was working hard, trying to get something to follow from those axioms, so hard that I found my other grades were going down, so I had to stop working on it. I had set up—it's probably pretty trivial now, I suppose—some operation that would be analogous to a truth table, and I finally proved a theorem that went something like this: "*The abstive of the posilute of two cosmoframmics is equal to the posilute of their abstives.*" I just made up words for certain abstract concepts, and it appealed to me that I could prove a theorem that was analogous to de Morgan's law.

All the way through my student work I had been joyfully stuck in Chapter One of my math books, thinking about the definitions of things and trying to make little modifications, seeing what could be discovered and working from there. I really enjoyed that, but the physics labs were killing me: The combination resulted in my switching to a math major at the end of my sophomore year.

TYCMJ. *Somewhere along the line you began to work with computers. Was there some point where it became clear to you that you would work with them for a long time?*

Knuth. Between my freshman and sophomore year. I had a summer job drawing graphs for statisticians at Case. In the room next to where I worked was a computer.

TYCMJ. *So the graphing continued?*

Knuth. Yes, I could draw graphs.

TYCMJ. *Was this compulsive?*

Knuth. Well, yes; to some extent, METAFONT (the system for computer-assisted typography I recently developed) probably reflects my love of graphing.

The Statistics Department at Case was located right next to a new computer, a wonderful machine with flashing lights. Early that summer, someone explained to me how it worked. Pretty soon I was hooked. I spent a lot of nights, all night long, at the console of the computer. Nobody else was there. I discovered girls in my sophomore year. This was before that; I had computers first.

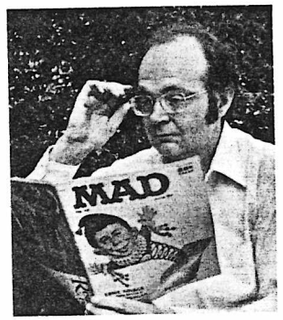
I still have my first computer program. I factored numbers into primes. You would dial a 10-digit number into the console, and it would punch the factors on cards. The program initially was about 70 instructions long, and as I recall, by the time I finished it, I had removed more than 100 errors out of 70 lines. In other words, I made a lot of mistakes, but I always felt I learned from those mistakes. The program wouldn't work, and I kept on fixing it, and finally it worked. My second program was to do base conversions.

My third program was to play tic-tac-toe, and it also would learn *how* to play tic-tac-toe. I worked hard on this one. I developed a learning strategy where every position in a game the computer won would be rated as a little bit better; but if it lost the game, every position would be marked as bad. This was a memory that would adapt itself. Each position in the game had a number from zero to nine representing how good it was thought to be; the neutral value was four, so if it was a drawn game, the position ratings would tend to go toward four. I wrote another program that would play tic-tac-toe perfectly, and then I had these two programs playing each other. After 90 games, the learning program learned how to draw against the good one. In another experiment, after 300 games, two learning programs starting with blank memories learned to draw against each other. They played a very conservative game, not very exciting, but it was interesting to see "the blind leading the blind."

That was my first month of learning to program. Those things were really fun. The next thing I did was somewhat different: I wrote an assembly program for the machine. I started to read the code of other people, and I got especially interested in programming because most of what I read wasn't very well written. I could look at programs and say here I am, only a college freshman, and I can do better than these professionals. I didn't know that lots of people could do better than those professionals. The standards of software at that time were pretty bad. I began to think that I had a special talent for it. Maybe I did, but I don't know now that it was as special as I had once thought. Then I read the code for Stan Poley's assembly program, which I thought was truly beautiful, a masterpiece of elegant programming, so I was inspired to carry his ideas one step further.

TYCMJ. *Feelings of inferiority were certainly not present in that work.*

Knuth. Well, part of me was anxious to prove to the other part that I was okay! So I was always motivated by seeing publications by someone who was apparently an expert, where I thought I could do a little better. Now I always tell my students when I make mistakes in class that I'm just trying to motivate them to do better.



What's this? The world's foremost computer scientist reading MAD magazine! Don Knuth's first publication was in MAD. (Photo by Tom Black)

TYCMJ. *The kinds of experiences with computers you had as a college sophomore, with minor modifications, are going on in junior high school now. What effect do you see that having on the future?*

Knuth. The students aren't learning how to write. That's serious. They don't know how to spell "mnemonic." They're specializing too early. There's a danger that people aren't seeing the other side of the coin, which is writing. That's what I worry about. As long as people keep open to a lot of aspects of life, then it's good, but if they become involved too much with one subculture, then it's going to limit them later.

TYCMJ. *What is it about computers that makes people so compulsive, either pro or con? A lot of people just can't stand programming. There are others who just get consumed by it.*

Knuth. It's partly a strange way of thinking. There are so many different modes of thinking, not really understood yet by psychologists. Teachers of computer science regularly find that 2 percent of the people who enroll in their courses are natural-born computer scientists who really resonate with computer programming. There seems to be a correlation between that and mathematical logic. I said I enjoyed the abstract algebra course where I first really studied axioms and Boolean algebra. If you look in math departments, the faculty who have traditionally been the closest to computer science have been the people in logic and combinatorics. Conversely, the mathematicians who are best at geometrical visualization tend not to enjoy a discrete universe like the computer world.

Another difference is between finite and infinite mathematics. I used to say to Peter Crawley at Cal Tech that he and I intersect at countable infinity, because I never think of anything more than a countable infinity, and he never thinks of anything less than a countable infinity. Higher infinities involve a kind of reasoning and intuition that doesn't apply very much to computers at all.

There are different flavors of mathematics, based on what kinds of peculiar minds we have: we aren't going to change that. People find out what things are best for them. And as for their mentalities, physicists are different from mathematicians, as are lawyers from doctors. Each of these fields seems to have predominant modes of thinking, which people somehow recognize as best for them. Computer science, I am convinced, exists today in universities because it corresponds to a mode of thinking, a peculiar mind-set that is the computer scientist's way of looking at knowledge. One out of fifty people, say, has this peculiarity.

Historically, such people were scattered in many walks of life; we had no home to call our own. When computer science started out it was mostly treated as a tool for the existing disciplines and not of interest in its own right. But being a useful tool is not enough in itself to account for the fact that computer science is now thriving in thousands of places. For example, an electron microscope is a marvelous tool, but "electron microscope science" has not taken the world by storm; something other than the usefulness of computers must account for the rapid spread of computer science. What actually happened was that the people who got interested in computers started to realize that their peculiar way of thinking was shared by others, so they began to congregate in places where they could have people like themselves to work with. This is how computer science came to exist. Now we can look back in old writings and see that certain people were really computer scientists at heart. It was latent back in Babylonian times, and throughout history.

TYCMJ. *Are computer scientists really different from mathematicians, then?*

Knuth. I think you can recognize a difference.

TYCMJ. *But how would I recognize a computer scientist if he walks in the door?*

Knuth. By the thought process. I've been trying to answer exactly that question. In order to get a handle on it, I tried to study several works on mathematics to discover the typical paradigms of good mathematicians, using a random-sampling technique. What I did was to take nine books that would represent mathematics, and I looked at page 100 of each book. I analyzed that page very carefully, until I understood what kinds of things were there on that page. It was interesting to see what aspects of mathematical thought processes were involved. I asked myself: "If I had to write a computer program to discover the mathematics on page 100, what capabilities would I have to put in that program?"

I found that one of the most striking things distinguishing mathematicians from computer scientists was their strong geometric reasoning and reasoning about infinity. The things that were common to both computer science and mathematics were primarily things like the use of abstractions and the manipulation of formulas. The main thing that was prominent in computer science that wasn't in mathematics was an emphasis on the state of a process as it changes, where it changes in time in a discrete way. In computer science when you say n is replaced by $n + 1$, the old value disappears and the new value takes over. We know how to think about an algorithm that is halfway executed; it has a state consisting of the current values of all the variables, and the state also specifies what rule to apply next. In order to formulate this for most mathematicians, it requires putting subscripts on everything. Traditional mathematics doesn't have this notion of a process in highly developed form, but it is vital in computer science.

The other striking difference was that computer scientists are willing to deal with diverse case analyses. The more pure the mathematician, the more he or she instinctively likes to have a clean formula that covers everything in all cases. But computer scientists are able to reason comfortably about things that have different cases, where we do step one, step two, then step three. A mathematician likes to have one step that you can apply over and over again.

TYCMJ. *What you are saying reminds me of an argument by Edsger Dijkstra, that computer scientists have now learned enough about the process of mathematics—in terms of how formulas are manipulated and how things change—that it should feed back into the process of teaching mathematics. Does the computer scientist actually now know enough to develop a science of mathematics that would be useful in teaching?*

Knuth. Since people have different modes of thinking, I doubt if any one way of teaching will be simultaneously the best way to reach different types of students; and I also doubt if many people can design educational plans that work for students having a different mind-set from the educational planners. So I can't be confident that a method best for me would be best for the world. But certainly Dijkstra's proposal would be the best way to teach mathematics to a natural-born computer scientist. From my own perspective, I feel that I have really learned some subject of mathematics at the point when I understand how it works in an algorithmic formulation.

For example, consider Volume 2 of my books. I think every theorem of elementary number theory is in there somewhere, but it's in the context of an algorithm that somebody needed because of a computational problem that had to be solved. I believe that the original discovery of these ideas was because of the need for such algorithms, so I presented it that way. It's a different aesthetic from mathematics, you see, from what is mathematically "clean" to what is not elegant in the same way. It's a different way of thinking, and I can't argue that one is better than the other.

Dijkstra and I are natural-born computer scientists. We found that out after we got older. Such ways of organizing knowledge are not going to be for everybody, but for our subset of the population, an algorithmic approach works best.

The knowledge that we have a computer-scientist mentality is also a challenge because we have to do our best for the other 49 out of 50 people who don't think as computer scientists; computers are affecting everybody's lives. We have to find a way to make it comfortable for other people to use computers, even though we don't really understand the way they think any more than they understand the way we think. You need people who are halfway between the different modes of thinking to bridge these gaps.

Of course, there is no definite boundary that you cross in going from computer scientist to mathematician to physicist, etc. There tends to be, in a multidimensional space of different kinds of abilities, a focus around the place that is most representative of computer scientists, and another one that is more typical of mathematicians. Maybe musicians are also close. Who knows? But it is a continuous thing.

Computer science departments thrive because there are a lot of people near our focal point in "thought space." In past ages, some of the people we now would call computer scientists were called mathematicians, physicists, chemists, doctors, businessmen. Now they have a home. That's what is holding the field together. But to develop our field well, if we want other people to make use of computer science, we have to realize that we can't do it all ourselves. We need others who can understand the other modes of thinking.

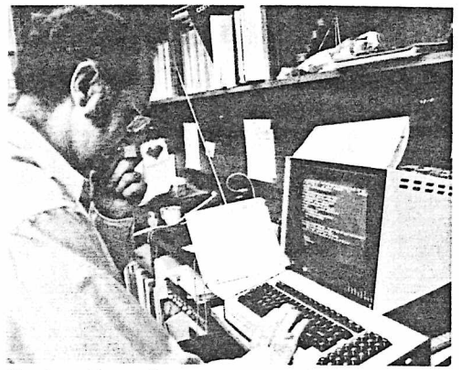
Maybe Dijkstra would argue that our mode of thinking is more powerful somehow, that it includes the other ones. I'm not quite so bold yet to do that, but the reason I raise this possibility is because I once met someone who had been a computer science major in graduate school and then went on to become one of the main advisors to President Lopez in Mexico. He told me that his training in computer science was of great value to him in working with all the people he had to deal with. Even though he wasn't programming computers any more, he felt that his approach to knowledge enabled him to understand lots of different people who were talking to him but who couldn't understand each other very well. The computer-science view seemed to be more powerful in its models of reality. This might be true because computer scientists are accustomed to models that can handle a variety of cases. The real world breaks down into cases, and mathematical models are better or worse depending on the uniformity of what they're modeling. The computer scientist dealing with less uniform models is perhaps able to cope with more general things. On the other hand, when it gets to something that is truly uniform, a computer scientist will not be able to go as deeply as a mathematician. Uniformity might really be the most striking difference between the fields.

TYCMJ. *A few have suggested that mathematics may be a part of computer science. Others say that computer science is a part of mathematics. What do you think?*

Knuth. I really think that they are two things, although they are related. There is a lot of overlap, but also I think I can tell when I'm in my mathematics mode, wearing my mathematician's cap, and that I can almost feel the changes when I go into a computer-science mode. I can't exactly say why; maybe I'll never know. But I can definitely feel when I'm behaving as a computer scientist.

I remember once giving a lecture about a number-theoretic problem that seemed to straddle the two fields. I started with the mathematical definitions and took things as far as I could using traditional mathematical tools. Then I said, "Now let's tackle this problem as a computer scientist would." And I carried on for 15 minutes with an algorithmic viewpoint, after which I said, "This is as far as the computer scientist is going to get. Now let's be mathematicians again for a while." And I could feel strongly that this was true, that I was sometimes doing what a mathematician would definitely do, while at other times I knew I was doing something that a mathematician wouldn't do.

Neither the mathematician nor the computer scientist is bound by a study of nature. With a pencil and paper we can control exactly what we are working on. A theorem is true or it's false. The fact that we deal with man-made things is common to both mathemat-



Knuth working at the computer terminal in his study. He refers to his study as his "book factory." (Photo by Tom Black)

ics and computer science, but the nature of the thought process is sufficiently different that there probably is a justification for considering them to be two different views of the world—two different ways of organizing knowledge abstractly that have some points in common, but in a way they have their own domains.

TYCMJ. *In addition to these intellectual contrasts between mathematics and computer science, there is getting to be a lot of social and educational concern about their interaction. The growth of the computer field, for example, is drawing so many people into bachelor's level computing that there aren't very many people going on into advanced work in mathematics or computer science or into high school mathematics teaching. Computer science and mathematics appeal to the same limited group of people, and at the moment, the momentum seems to be running pretty well in favor of computer science.*

Knuth. But that's an economic consideration. Right now bachelors in computer science are getting better salaries than anybody else. This is our problem because people are going into it who aren't natural-born computer scientists; they're just going into it for the money, and not for the love of it. This trend can actually be to some advantage to mathematicians because they've now got motivated students in their classes instead of people who are just there for some external reason.

I don't agree with your statement that "computer science and mathematics appeal to the same limited group of people," but our disagreement is probably due to the fact that I have been thinking mostly of extreme cases, the differences between the best computer scientists and the best mathematicians. According to the law of large numbers, there will of course be very few people who rate a 10 on a scale of 1 to 10, under almost any ranking criteria, and there aren't many 9's either. So let's consider somebody who is an 8 at mathematics and a 6, say, at computer science. That person is somewhat likely to go into computer science instead of mathematics, nowadays, because the salary is better; and perhaps society will be better off since there is a pressing need for computer scientists. I agree that economic pressure is causing mathematics to lose a lot of its 7's and 8's (if you'll excuse my callous use of numbers in place of human beings); but I hope that the differences between computer science and mathematics will be well enough understood that you don't lose the 9's and 10's. Their mathematical abilities are vital to people in all other fields, including computer science.

TYCMJ. *One of the things that we keep hearing today is that good instruction in mathematics is reversing dramatically and perhaps in other subjects as well. Many say the reversal is due to the influence of computers and calculators.*

Knuth. I have been quite disappointed in the mathematics textbook my son has as a sophomore in high school. But his teacher is very good and compensates for it. The worst excess of this book is the pedantry. The second worst is that its three authors seem to have written three different kinds of chapters without much awareness of what the other authors had done. But the book has some good points, too, like its emphasis on graphs. My suggestion for teaching mathematics at the young levels is to draw graphs! I was glad to see there was much more use of graphs here than in other books I had seen.

I read an article a week ago where someone said he had started to employ older textbooks, and the older the textbook the better the students would do on exams. He went back to about 1900, and he said that was the Golden Era.

Clearly the Hungarian educational system has been the most successful for pure mathematics; it's a model that ought to be studied very carefully because it works. It produces so many good mathematicians per capita.

As I see my children learning mathematics now, I find that the books have too much emphasis on flashy things and on memorizing formulas, rather than on what an idea is good for as a general tool, or on how to reconstruct a formula from a few basic principles instead of from memory.

After reading these books, you don't remember what the degree of a polynomial is, or how to answer various precise questions. My son had learned algebra, but it was not clear to him how to add fractions; that had been glossed over and needed in only one homework assignment. He would remember for a few weeks what the distributive law was, but that would come and go. The high-level wording for things wasn't being put to any use. So it disappeared from his mind. In my own case, I got along fine without knowing the name of the distributive law until my sophomore year in college; meanwhile, I had drawn lots of graphs.

TYCMJ. *Do you think students should be introduced to computing earlier in the formal school curriculum?*

Knuth. I would like to see an approach by means of algorithms, but I really hesitate to say much about it for fear somebody will believe me when I really haven't thought it through. The only experience I had personally with doing something on an algorithmic basis was at a higher level. At Cal Tech I taught an introductory abstract-algebra course. We were studying matrices, and there was a question on canonical forms of matrices. The book was very obscure on that point. We wanted to know when one matrix was similar to another. How can you tell? Well, you have a canonical form that says that if two matrices are similar, they have the same canonical form. Books rarely even say that. They just say here is the name of the form, and here is the definition of it.

So we said then, how can we determine if two matrices are similar? There is a little operation you can do on a matrix, something like this: "Subtract a multiple of the third row from the second row, and then add the same multiple of the second column to the third column." That operation preserves similarity; although it is very simple, it can be used to get another matrix similar to it. So let's take a look at what it does. We start with a matrix and try to zero out most of the first row and the first column, and we see that we can almost always do this by simple transformations that preserve similarity. But every once in awhile we get stuck.

The point is that we are solving a problem. We are trying to do it step by step with an algorithm; pretty soon we find this canonical form. Of course, we might fail because some special case might happen. Well, in that case, we won't be able to make so many elements zero. This, I believe, is the way these canonical forms were first found. But later on there was a tendency for people who had learned about them in a concrete way to present them in a high-level, abstract fashion. For example, they would look at the way linear operators behave on subspaces. This is elegant, but pedagogically unsatisfactory because it conceals the method of discovery.

At the age of three, Don Knuth was already attracted to keyboards.

I think you need both views, especially when you are learning a subject at first. The algorithmic point of view tends to be at a more intuitive level. It helps very much in the early stages of teaching. But as I say I haven't been teaching these courses myself for a long time, and my own intuition might not correlate well with that of the majority of students. Careful experiments should be tried, since an algorithmic approach might well be very successful.

TYCMJ. *It was my impression that you were perhaps very excited by your tic-tac-toe learning program. Now that's using the computer for playing games. What do you think of games as an introduction to computing?*

Knuth. When children are young, they get to a stage where they like to make up rules to games. They enjoy arguing with each other. They argue incessantly about the rules because they like to make up rules to games. I think this might be related to making up computer programs. There is probably in children a tendency that can be well exploited to encapsulate these kinds of rules.

I'm way out of my expertise, of course, when thinking about elementary education. But I would like to see an approach where the students learning some concept learn not only how to follow the rules but also how to explain the rules. Suppose you are teaching somebody to add. We carry this out by giving a bunch

of examples and tell children that you go from right to left and you carry, and so on. If there were only a simple enough computer language for these second and third graders, it might be good to teach addition by having them write little programs in this language. The teacher would say: "Here's the way to add numbers and we're going to teach it to this goofy machine." I think it's definitely worth a try. Students learn quickly to make up computer programs that will draw pictures. It might be that similar skills would work on things like arithmetic.

I like to see the rules and their exceptions. Other people are very comfortable without the rules: maybe people who are good with language just absorb dictionaries very quickly and don't even make rules out of things. To them, everything is an exception. So I suppose an algorithmic approach will be of no help to them; we need a variety of teaching methods.

TYCMJ. *I have been impressed with a toy called Big Truck that uses a language like* TURTLE, *which comes out of Papert's laboratory. The rules are very explicit and very tight.*

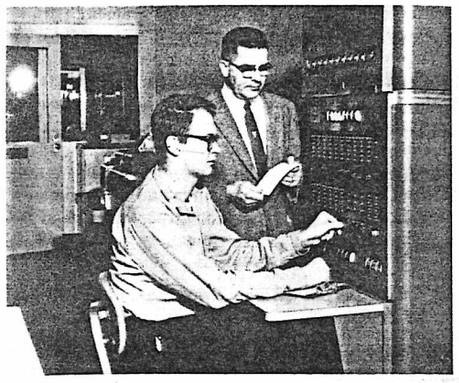
Knuth. That's very good for teaching algorithms. I think arithmetic is another thing that people should still keep learning even though we have calculators. It's nice to be able to add and to multiply numbers by hand, in a pinch. Not only nice—it introduces important patterns of thought.

TYCMJ. *Are you very enthusiastic about calculators?*

Knuth. Well, no. Computer scientists were the last to get enthusiastic about calculators. One of my colleagues remarked the other day that this is probably the only department at the university where nobody has a pocket calculator at the faculty meeting. This may be because we have all these supercomputers here in our offices. I can do my symbolic calculations at MIT, too, three thousand miles away, using a computer network.

TYCMJ. *Can we turn to some other games for a minute? One of your former teachers at Case tells a good story about you. I don't know whether it is true or not, but it is a good story. He says you were absolutely instrumental in the success of the Case basketball team in 1960. Is that true?*

Knuth. Well, it would be nice to say that. We did win the league title that year, and I'll be glad to take all of the credit for it. But here's what really happened. As the manager of the team, I worked out a formula, a rather complicated jumble of symbols that I really don't believe in any more. It was a system that would rate each player with a magic number. The magic number would tell how much each player really contributed to the game, not just the points he scored. If he missed a shot, there was a certain chance that our team would not get the rebound, and the other team would get the ball. Thus, he would lose something for each shot missed. Conversely, when he stole the ball, that was very good because our team got possession.



Student Knuth and Coach Heim putting player ratings into an IBM 650 computer. In 1960 Don Knuth was a student manager of the basketball team at Case Institute of Technology. Knuth developed a formula for rating each player. Heim used Knuth's formula, and Case's Rough Riders went on to win the league championship.

In fact, it is interesting to watch a basketball game and imagine that possession of the ball counts one point. Such an assumption isn't true in the last seconds, but during a good part of the game it isn't out of the question to say that if your team has possession of the ball, it's worth a point. In other words, you look at the scoreboard, and it it's 90 to 85, you add one to whichever team has the ball then. It tends to give you a better feeling of the current score. If you look at basketball this way, then when someone makes a field goal, the score hasn't really changed because his team has gained two points but lost possession of the ball, while the other team has gained possession. The two points sort of cancel each other out. If the other team just goes back and makes another field goal, we're back to where we started. But if somebody steals the ball, he is really making the previous field goal count.

According to my magic formula, possession of the ball turned out in most games to be worth about 0.6 of a point, so you got some credit for field goals too. As I said, I don't really believe in that formula any more, but it did include all the statistics like steals, fumbles, etc., and you could plug into it and get a number. The coach liked these numbers. He said it did correlate with what he felt the players had contributed; and the players on the team, instead of competing for the most points, tried to get a good score by this number. The coach thought it was a good thing. It was written up in *Newsweek* magazine, and it was on Walter Cronkite's Sunday news. They sent a cameraman out to take a picture of me taking the statistics and feeding them into a computer.

TYCMJ. *Did you receive offers to become a consultant for professional teams?*

Knuth. There was some talk about the Cleveland Browns wanting to use computers already in those days, but I never was involved with that.

TYCMJ. *What was the inspiration to launch your series* (The Art of Computer Programming) *in the first place?*

Knuth. *The Art of Computer Programming* developed when I was a second-year grad student at Cal Tech. I had been working as a private consultant writing compilers for different machines. In those days a software firm would ask for hundreds of thousands of dollars to write a compiler. But I didn't know that, and I had written one for $5000. I guess the word got around that I knew how to write compilers. A little later, Richard Varga, who was an advisor to Addison-Wesley, suggested that they ask me to write a book about compilers. They came to me in January 1962 and said: "How would you like to write a book on writing compilers?" It occurred to me that I really did like to write. It sounded good; I decided that it would indeed be nice to write such a text.

I had gotten married in the summer of 1961, and I wonder now how I broke the news to my wife that I was suddenly planning to write a book. Surely neither of us knew how much this was going to change our lives. Anyway that day I went home and sketched out 12 chapter titles that I thought would be nice, and then a little while later I signed a contract for a book about compilers. I got a chance to teach a course at Cal Tech during the fall of 1962, while still a graduate student, with the idea that the class notes would develop into the first three chapters.

Right after getting my Ph.D. in 1963, I started to work hard preparing the chapter on sorting. I knew hardly anything about sorting, but I thought it would be nice to read up on the subject and to toss a chapter about sorting into a book about compilers, especially because the LARC scientific compiler had just come out, and it was reputedly based on the idea of sorting the data in unusual ways. I found that sorting was really interesting and pretty soon I found myself digging into lots of technical articles.

The main thing that struck me was that the literature was so spotty. Computer science was a very new field, without an identity of its own, and standards of publication were not terribly high, especially when quantitative aspects of algorithm performance were concerned. A lot of the published articles were just wrong, so you had three possibilities: the wrong answer by the wrong method, the right answer by the wrong method, and the right answer by the right method. You had about a one-third chance on any of these possibilities. The literature on computing was already large but very unreliable, so it was clear even in 1962–1963 that it would be nice to have a summary of the right parts of the literature. Publications were so bad, in fact, that people didn't even bother to read them, and the good ideas were being rediscovered because people found it easier to do this than to sort them out from the bad ones. Thus, one of my big motivations was to clean up the story that had been presented badly in the literature.

I guess I have an instinct for trying to organize things. At that time, everybody I knew who could write such a book summarizing what was known about computer programming had discovered quite a lot of the ideas themselves. It seemed to me that they would slant it all to their own perspective, which would present only one side of the coin, their own particular part. By contrast, I really hadn't discovered anything new by myself at that point. I was just a good writer. That was very prominent in my mind. For example, I was the only computer scientist I knew who hadn't discovered how to compile arithmetic expressions by the precedence method. Ten people independently discovered it, yet the problem had baffled me; I hadn't seen my way through it. So I felt not only that the story needed to be presented, but also that I could present it from a less biased viewpoint than these other people who seemed to have done their work more in isolation. I had this half-conceited and half-unconceited view that I could explain it more satisfactorily than the others because of my lack of bias. I didn't have any axes to grind but my own. (Then, of course, as I started to write things I naturally discovered one or two new things as I went, and now I am just as biased as anybody.)

But you asked about my original motivation. My original motivation was to write a text about how to write compilers, so I began drafting chapters. I was seriously planning to finish the book before my son was born. (He's a sophomore in high school now, so I'm currently trying to finish before he starts writing

his own books!) I recently found copies of letters I wrote in 1965 saying, "I wish I could come to visit your university this summer, but I can't because I just have to finish this book-writing project." It was going to be just one book.

As I said, however, I started to get interested in sorting, after I had been writing for a while. There were so many interesting things on sorting. So the book was growing rapidly.

I eventually wrote a letter to Addison-Wesley saying: "Do you mind if this book is a little long? I would like to give a fairly complete presentation of the material." They said: "Don't worry, go right ahead, write whatever you want." So I kept gathering more and more stuff. In June 1965, I had finally finished the first draft of the 12 chapters. It amounted to 3000 handwritten pages of manuscript.

To me this was something of a longish book, yet only one volume. I thought I knew about books, and the printed letters in books seemed to be a lot smaller than the ones I write by hand. So I figured that about five pages of my handwriting would be about one page of book. Then I went ahead and did chapter one, typing it from the handwritten manuscript, and sent it off to Addison-Wesley, just to see if it was okay. This was October 1965. They hadn't heard from me for quite a while, and were wondering what was going on. I felt good, for at least I had finished chapter one. Incidentally, that chapter one was pretty much the same as the chapter that was eventually published.

Immediately I got back a letter from Addison-Wesley, from a person very high up in the company. It turned out he was one of the people who had originally talked to me in 1962, but meanwhile he had been promoted three times. From the length of chapter one, the book was now estimated to be almost 2000 printed pages, and they said: "Don, you said you might be writing a longish book, but do you realize that one and one-half pages of typing is one page of book?" I thought to myself that it can't be so. "I've read a lot of books; these guys don't know what they're talking about." So I took one of their books, Thomas's *Calculus*, and I sat down at my typewriter and typed up a page of it. Lo and behold, they were absolutely right! Then I knew why it had taken me so long to get that first chapter finished—three and a half years to write the first chapter is not too good.

It gradually dawned on me how large a project this was going to be. If I had realized that at the beginning, I wouldn't have been foolish enough to start; I wouldn't have dared to tackle such a thing. But by 1965, of course, I was hooked because I still felt the need for these books, and I still felt the project ought to be done. I still believed that I was fairly unbiased and could try to be a spokesman for the people who were making the discoveries. I had collected so much material that I felt it was my duty to continue with the project even though it would take a lot longer than I had originally expected. I had done all of the background work and it would have been very hard to transfer it to anybody else.

Addison-Wesley took another look at the chapters. At first it appeared that the material would fill up two volumes, then three volumes. Publishers have horror stories about ponderous tomes like that. So my guess is that they showed these things to some consultants, and the consultants recommended that seven of the twelve chapters would sell pretty well as individual volumes. So they suggested combining the remaining five chapters with the good seven, hoping to sell seven books this way. I think this was probably the motivation for the present plan. At any rate I saw that it was possible to reorganize the chapters so that they would fit together reasonably well in seven separate volumes.

Volume Two, for example, was the combination of my original idea for chapter two on random numbers and chapter six, which was originally called "Miscellaneous Utility Routines." Suddenly I noticed that all but one of these miscellaneous utility routines were really about arithmetic. So I decided to call that chapter "Arithmetic." That little change in title suggested one or two sections that I ought to add; as I added them, I came to a marvelous realization that there was this book out there waiting to be written, bringing together what is known about arithmetic from a computer scientist's point of view. As a result, that chapter almost wrote itself, and led me to fascinating things in a variety of journals that had just never been put together in book form. Having the title "Arithmetic," and having it bound in the second volume, is what turned out to add a lot of unity to the subject, a unity that probably hadn't been realized before. Most of the articles I found in the literature were written by people who were not aware of many of the other journals, nor of the relation between their ideas and others.

I got so excited about writing Volume Two that I started working day and night on it. As a result, I got a serious attack of ulcers, and had to change my whole life-style. By the middle of Volume Two I kept thinking I was going to finish it soon, and I had something of a breakdown of my health in the summer of 1967. About the middle of Euclid's algorithm is where I broke down. That happened on what is now page 333 of Volume Two out of 688 pages. So I still had a lot to go at the time. I knew it, but I wouldn't admit it to myself.

I always underestimate time, too; otherwise, I never would have started writing these books. If only I had a better estimator of time! Because of my writing, I have now resolved not to give any lectures away from Stanford until 1990. All of book writing comes out of spare time. When I go on a trip to give a lecture, one day wipes out at least five days of book writing because when I come back I still have to do everything else that I was supposed to have done when I was gone. So the spare time disappears.

TYCMJ. *Your field, computer science, has been growing at great speed. Is it still possible to capture it? Is it growing faster than you can write?*

Knuth. Yes, perhaps. I'm thinking of the novel *Tristram Shandy*, a fictional autobiography whose supposed author goes through Volume One of his memoirs covering just the first year of his life. But I still believe it will be possible for me to finish. Volume Four, *Combinatorial Algorithms*, has exploded the most. Volume Four is the one that I'll return to immediately after I finish my typographical research. I think it is going to become Volumes 4A and 4B.

TYCMJ. *So it really is exploding?*

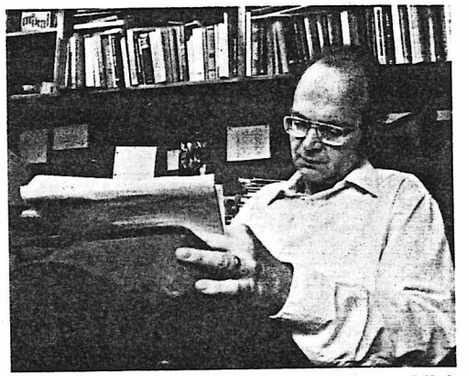
Knuth. In fact, the chapter called "Combinatorial Algorithms" that I planned on that first day in 1962 was thrown in almost as an afterthought. There were very few combinatorial algorithms at the time, but I liked that sort of programming so I thought it would be nifty to have a chapter about it. Almost none of the present material in Volume Four was known then. So when people talk about "combinatorial explosion," the words have a significant double meaning for me. At one point, three years ago, I think 70 percent of the journal articles being published in computer science were about combinatorial algorithms. Volume Four will surely be the hardest, because of the explosion.

TYCMJ. *Is the organization you worked out 20 years ago still pretty well holding its shape?*

Knuth. There are new topics, but I won't live long enough to include those. For example, I never promised to write about operating systems. Therefore, I'm very happy whenever I get a journal in the mail, if most of its articles are about operating systems.

TYCMJ. *How do you combine the discipline of writing with spontaneous creativity?*

Knuth. When I'm writing a book, I surround myself with that subject and nothing else. I read exhaustively on one area, and then after I finish that section, it goes out of my head, and I bring in another. That's what computer science calls "batch processing," as opposed to continual "swapping in and out" or "thrashing."



At work in his "book factory." Although Knuth is a skilled typist and attended secretarial school while in high school, he always prepares his books in longhand. (Photo by Tom Black)

I don't read the literature as it happens. I only read the titles and abstracts to know where I can put articles on the agenda of things to read later. I do the same thing with quotes. I file the quote. Here is a wonderful one from the Beatles' songs: "There's nobody in my tree." That's just perfect for branch-and-bound methods. I hope to live long enough to finish just because I have so many quotes on file that are great.

When I'm working on a topic, I may have to read 60 papers on one subject. The first two I'll read slowly, but with the next 58 I know what to do already. When I read the first two, I use the strategy of trying to figure out the problems before I look at the answers. Then I am ready for the vocabulary and ideas that are going to be occurring in the other papers.

I keep a little notebook, too; every day I write a summary of what I've worked on that day. It helps me to schedule myself a little bit and it helps me to realize how hard things are so that I can plan ahead. If too many days go by where I said I was just too tired and went to bed, or if it reads, "Today I goofed off," then it helps me to make a little more rational schedule.

For the seven-volume book project, I have to cross a threshold every day to get started when I am writing. I have to get psyched up for it. It is a long, ongoing process; I know that even after the end of the day I won't have finished. Every morning I wake up and say: "Another day, and the book isn't finished." I still feel a strong need for the book in the world, and that it is filling a necessary role, but all these logical argu-

ments aren't going to make me get started. On the other hand, once I've started, then I'm excited about it, and it's hard for me to stop again. I have to force myself to stop and not just stay up all night. So I always read a variety of things—detective stories, or more serious works of fiction, or history, sort of rotating between them—at bedtime.

TYCMJ. *Do you do most of your writing at home?*

Knuth. Yes, always; it is not part of my Stanford job. It's all spare time.

TYCMJ. *Do you do it in longhand?*

Knuth. I can't compose at a typewriter. I can't even compose a letter to my relatives on a typewriter, even though I'm a good typist. I went to secretarial school during the summers of my high school years, and I learned to type 80 words a minute. I learned machine shorthand, and I learned Gregg shorthand. But I can't compose in any of those modes.

TYCMJ. *Why would a prospective physics major take those courses in high school?*

Knuth. I had summer jobs doing secretarial work, and I thought it would help me in college taking notes. But all I learned were the abbreviations for Dear Sir and Yours Very Truly, and that didn't help very much in my chemistry class. I kept making up new abbreviations, sitting in the back of the class with my stenograph machine; afterwards I couldn't figure out what I had put down, so I gave it up.

TYCMJ. *How did you come to write* Surreal Numbers?[1]

Knuth. I wrote it in one week, while on sabbatical in Oslo, Norway. It hasn't been a bestseller, but it's been steady and translated into lots of languages. I'm glad for that. Writing *Surreal Numbers* was probably a once-in-a-lifetime experience for me. I got inspired to do it, and I guess there was a muse sitting behind me telling me what to write. The book just fell together, and I don't think I could do it again. That week was one of the most exciting periods of my life.

It was December 1972, and I was in the midst of writing *The Art of Computer Programming*, when suddenly I got the idea for Surreal Numbers in the middle of the night. I woke my wife and said, "Jill, you know how this series of seven volumes—the books I started on after we had been married for only six months—is affecting our lives? Well, it turns out that there's another book I would like to write too. But I don't think it will take me very long to finish this new one." I said that I thought I could write this other one in about a week, if I just worked on it and nothing else. To my great pleasure, she was also delighted by the idea. She said, "This is the best time in your life for you to do such a project."

We planned it so that after the new year I would get a hotel room in downtown Oslo near where Ibsen wrote his plays, so that I might be able to pick up some of the nuances of his art. Then I could work on this book and also she would come to meet me twice. (We always wanted to have an affair in a hotel room.)

During the three weeks or so before I started *Surreal Numbers*, as I would be walking along or skiing, I would be going through the first page or two of the book in my mind. But I didn't go any further than that, at the time, because I wanted the rest of the book to be fresh as it was being written. I wanted it to be a faithful recording of mathematical discoveries, so I didn't want to do any of the mathematics in advance. I only vaguely remembered what John Conway had told met at lunch a year before.

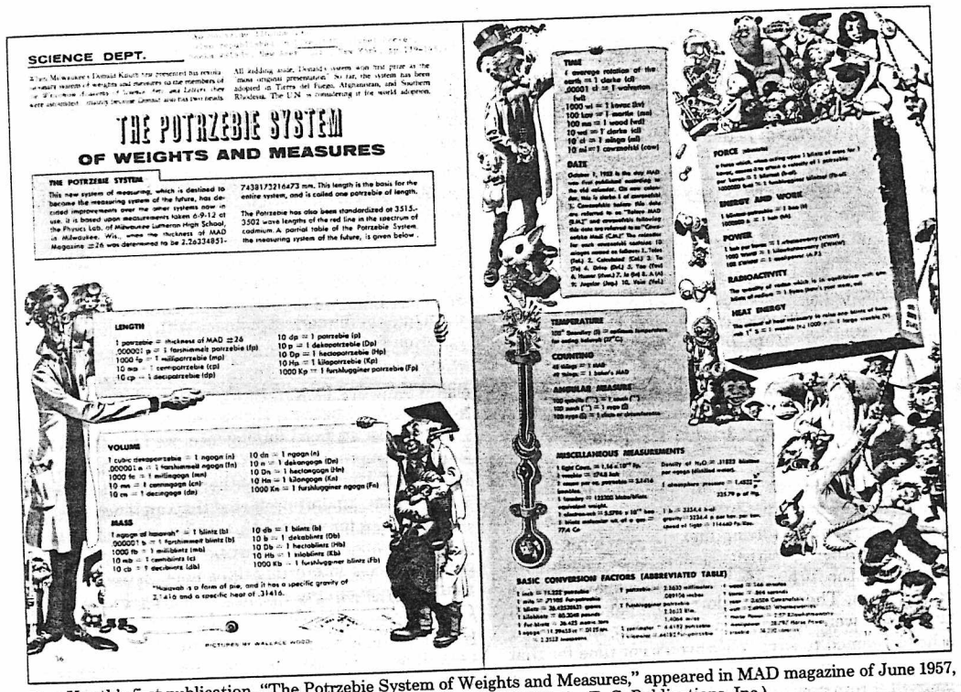
I got to the hotel and started to work. Fortunately, I didn't go through the scene you frequently see in the movies where the guy types the title of the book, stares at the pages for awhile, and tears it up. I didn't have to go through that. I could write out the first page and most of the second, since I had that memorized.

Every morning of that week had pretty much the same scenario. I would start out in the morning with a very leisurely breakfast. Students from Saint Olaf College happened to be staying at this same hotel, and I eavesdropped on their conversations to see what phrases they were using. Then I would go to my room and work for about three or four hours. Then I would get to something I wouldn't know how to handle, and I wouldn't have any idea what to do next. So I would go for a walk around Oslo for about two hours. Maybe I'd go to the library, but usually I'd just walk around watching people. Then the solution to the problem would present itself. I would go back to the hotel, and after two more hours of work, I would get over the hump, and I would magically be able to move a little further. Then I would have a nice relaxing dinner, watch Norwegian television for about an hour, go back to the room, write some more, and put out the light.

The reason I think I had this muse was that the book would seem to write itself and fall into place. Things seemed to work out too nicely. I am, of course, very biased. But after I turned out the light, the next page would flash into my head, and I would have to get up quickly and write it down. The thoughts would come so fast that I would only have time to write the first letter of every word. Then I could turn out the light and sleep like a log. The next morning I would have to figure out what the first sentences were from the first letters of all the words. Every day the same pattern repeated.

The day I finished was the happiest day of my life. Oslo was so beautiful; there was a hoarfrost on all of the trees, more than an inch thick. I walked around in the gardens of the king's palace after having been to a movie with my wife. The frost-encrusted trees by the palace were magnificent. The midnight sky was a perfect, deep blue. I spent an hour gaping upward in the park, marvelling at the patterns of trees against the sky, and then went back to the hotel. That was one of the greatest times I can remember. I knew that I was just one or two pages away from the end of the book. Then I finished the final chapter, except for a few unimportant mathematical details that I knew I could work out, and I relaxed into sleep.

The writing of *Surreal Numbers* had taken six days; so on the seventh day I rested. In fact, I still had the seventh day to tidy up the last page, which I did. Then I wrote "The End."



Don Knuth's first publication, "The Potrzebie System of Weights and Measures," appeared in MAD magazine of June 1957, when he was a high school senior. (Permission MAD Magazine © 1957 by E. C. Publications, Inc.)

[1] In 1974, Addison-Wesley published a novelette by Don Knuth. Its title is *Surreal Numbers: How Two Ex-Students Turned on to Pure Mathematics and Found Total Happiness*. It contains a development of a remarkable new way to construct numbers. The new construction had been found by John Horton Conway of Cambridge University. One day over lunch in 1972, Conway briefly explained his system to Knuth. Knuth was so taken by this revolutionary approach that he was motivated to write a book about it. Martin Gardner says: "I believe it is the only time a major mathematical discovery has been published first in a work of fiction."

In a postscript to *Surreal Numbers*, Knuth explains his purpose in writing the book: "My primary aim is not really to teach Conway's theory; it is to teach how one might go about developing such a theory. Therefore, as the two characters in this book explore and build up Conway's number system, I have recorded their false starts and frustrations as well as their good ideas. I wanted to give a reasonably faithful portrayal of the important principles, techniques, joys, passions, and philosophy of mathematics, so I wrote the story as I was actually doing the research myself."

I couldn't write a word after that. I tried to compose a letter to Phyllis, my secretary at Stanford, telling her how I wanted this book to be typed. I would get into the middle of a sentence, and I could not figure out what verb to use. Suddenly, I couldn't even put simple things onto the page. I had just gone through a week where everything was sort of flowing out, and all of a sudden it was gone! That's why I love this book. It was a part of me that had to be expressed. I wish everyone could have a chance like this—some inspiration that could touch them.

TYCMJ. *At the conclusion of* Surreal Numbers *there is a "dear teacher" letter, in which you suggest that students using the book for a course should do a project and write it up. In your last paragraph, you say that two of the major problems in teaching mathematics are a lack of experience in writing and also a lack of experience with creative thinking. Do you feel the same way today, several years later?*

Knuth. Oh, absolutely. I try to do that with our graduate students in computer science. We try to minimize the competition. We encourage working together on problem solving and discussing problems with each other. Creativity seems to be encouraged until about the fifth grade, at least in the education of my children. There were a lot of creative things that they once were asked to do; but after fifth grade, schools seemed to say: "We haven't got time for that anymore, no time for you to discover anything for yourself. From now on you will have to absorb all this information the world has been developing."

That's wrong. We should teach people things with an emphasis on how they were discovered. I've always told my students to try to do the same when reading technical materials; that is, don't turn the page until you have thought a while about what's probably going to be on the next page, because then you will be able to read faster when you do turn the page. Before you see how to solve the problem, think about it. How would you solve it? Why do you want to solve the problem? All these questions should be asked before you read the solution. Nine times out of ten you won't solve it yourself, but you'll be ready to better appreciate the solutions and you'll learn a lot more about the process of developing mathematics as you go.

I think that's why I got stuck in chapter one all of the time when reading textbooks in college. I always liked the idea of "why?" Why is it that way? How did anybody ever think of that from the very beginning? Everyone should continue asking these questions. It enhances your ability to absorb. You can reconstruct so much of mathematics from a small part when you know how the parts are put together. We teach students to derive things in geometry, but a lot of times the exercises test if they know the theorem, not the proof. To do well in mathematics, you should learn methods and not results. And you should learn how the methods were invented.

TYCMJ. *Occasionally today we have been getting close to the burgeoning area called artificial intelligence. Has your attention been attracted to that area at all?*

Knuth. I enjoy reading about it. Many of the algorithms in Volume Four are used in artificial intelligence to solve interesting problems. They turn out to be in one-to-one correspondence with things that electrical engineers use for other purposes, and again I enjoy bringing together two literatures that are talking about the same thing. The shortest-path problem, for example, is something that arises in many different disguises. In artificial intelligence, we find algorithms for theorem proving and problem solving, expressed in a different language than other people will be using for the same kind of problems they are encountering in electronics for wire routing, or something like that.

I'm not a specialist in artificial intelligence, but I think the most interesting thing about it is something that I can paraphrase from a book by Pamela McCorduck *(Machines Who Think)*. She points out that the question used to be "Can computers think?" By now, however, everything that has been associated with thinking has been done by computers; the only human accomplishments that computers *can't* do well are things that people do *without* thinking! This is so true. The things we do without thinking are the things that computers have never done or hardly done, like walking. To control a robot to walk like an ant walks, or to program a computer so that it will recognize a face when someone has grown a beard, are extremely difficult. Children can talk languages; computers can't even translate languages very well. All the things we do subconsciously are the things that artificial intelligence hasn't been able to do. That's the most striking thing about the subject now. The big mystery is what goes on when we are not thinking. How do ants do such complicated things with no leader telling them what to do? How do their small brains come to the decision of how they are going to communicate with each other and solve problems? That's way beyond what we know now. I am fascinated by that, but I never promised to write a book on it.

TYCMJ. *You're not terribly optimistic then?*

Knuth. I believe the study of artificial intelligence is really important in that we learn much more trying to find out how these things are done than we actually do by having a computer system do them. Trying to automate something is a great scientific achievement. After you automate something, the important thing is what you learned in the process, not really that the computer can now do a complex job.

While you're trying to explain something to a computer, you have to understand it so well that it's even better than the understanding you get by teaching it to somebody else. The old saying is: "You don't learn something until you have taught it to someone else." Today's saying is: "You don't really know something until you have taught it to a computer." That's the secret of learning. The computer is really a good test of understanding. It doesn't allow you to wave your hands and say: "Now you use some common sense." You have got to understand it clearly—there's no room for wishy-washiness. That's why I believe computer science impinges on education. If students can teach something to a computer, then you know they have got it in their heads.

TYCMJ. *How long have you been working on TEX and METAFONT?*

Knuth. Spring of 1977 is when I started, so it will be four or five years by the time I'm done. I thought it would be a one-year project.

TYCMJ. *Do you think it was time well spent?*

Knuth. Yes, I think the things I've learned are really exciting, and they are causing a lot of good waves in

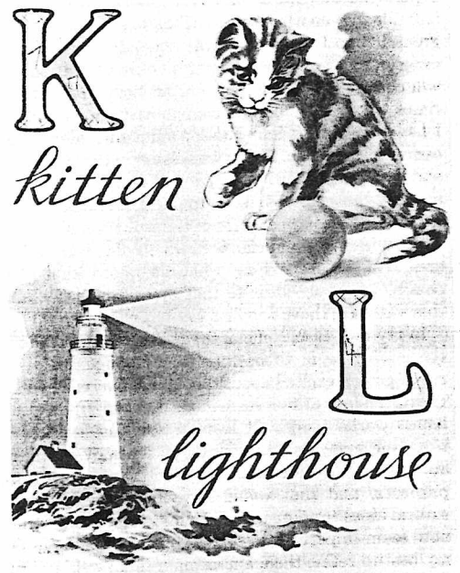
The Roots of METAFONT

"Mathematics books and journals do not look as beautiful as they used to." With those words, Don Knuth introduced his 1979 article, "Mathematical Typography," to readers of the *Bulletin of the American Mathematical Society*. His statement clearly reveals an aesthetic concern about the physical appearance of mathematics.

This concern has resulted in his inventing TEX and METAFONT. TEX is described by C. Gordon Bell, vice president of engineering at Digital Equipment Corporation, as follows: "Don Knuth's Tau Epsilon Chi (TEX) is potentially the most significant invention in typesetting in this century. It introduces a standard language for computer typography and in terms of importance could rank near the introduction of the Gutenberg Press."

METAFONT is a system that makes use of classical mathematics to design alphabets. Knuth's aesthetic concern is clear when he says: "Of course it is necessary that the mathematically defined letters be beautiful according to traditional notions of aesthetics. Given a sequence of points in the plane, what is the most pleasing curve that connects them? This question leads to interesting mathematics, and one solution based on a novel family of spline curves has produced excellent fonts of type in the author's preliminary experiments. We may conclude that a mathematical approach to the design of alphabets does not eliminate the artists who have been doing the job for so many years; on the contrary, it gives them an exciting new medium to work with."

Four years after he had started his research on METAFONT, Don's mother sent to him the alphabet book that he had enjoyed as a boy of 2 or 3. A page from that book is reproduced here. Note the *x*'s that Don placed by each serif in the K and L. The 7 inside the K is a count of the serifs of that letter. Clearly, his aesthetic and mathematical interests in alphabets go back to his early childhood.



Page from Don Knuth's childhood alphabet book.

the printing world. I think it happens fairly often that a person from one field, say a mathematician, will stumble into another field. He'll have a different background than the people in the other field, and so he can contribute new insights. Sometimes such people will change fields and change their life's work. Sometimes they just make the contribution. For example, I have heard that Larry Shepp's daughter developed cancer of the brain. He got interested in that problem, and worked on a technique for locating cancers that's actually turned out to be an important breakthrough. You'll see this happen a lot of times for one reason or another: a mathematician will wander into some other area, and he'll see from what he knows that it is possible to apply ideas to that area right away, and that will help those people a lot.

In my own case, some of the things I learned about typesetting seem to be important enough that I've really gotten excited about them, and that is why it is taking me four or five years instead of one. If the ideas hadn't worked very well, I would have just kept them to myself and not bothered telling anybody anything about them; I would just have used them for my own purposes, and that would have been enough. But several ideas from mathematics and computer science now seem important to typography, so I want to do my best to refine them and explain them well. At the same time I am anxious to return to Volume Four before it becomes too big to write.

Meetings in Retrospect

CIPS '82—History Session

The Canadian Information Processing Society (CIPS) held its National Conference in Saskatoon on May 17–19, 1982. The conference included the first CIPS session devoted to the history of computing, organized and chaired by Mike Williams of the University of Calgary.

Four speakers had been invited so as to present an overview of the history of computing from its earliest beginnings to the advent of the microcomputer, instead of a deep and scholarly inquiry into one specific topic. The speakers were Arnold A. Cohen, senior fellow of the Charles Babbage Institute, Keith W. Smillie, professor of computer science at the University of Alberta, Martin Campbell-Kelly, lecturer in computer science at the University of Warwick, and A. D. Booth, a British computer pioneer who was responsible for some of the early computer science work in western Canada.

The session started with Arnold Cohen describing the history (short though it is) and current status of the Charles Babbage Foundation and the Charles Babbage Institute. He presented a very clear picture of CBI's birth and development along with a discussion of the goals of the organization. After describing why Minneapolis was selected as the home of CBI, and the organizational changes this required in order that it fit into the University of Minnesota pattern, he highlighted the achievements of some of the people who had received support from CBI to pursue historical research.

Keith Smillie took the audience back to ancient Greece in order to introduce his topic of "Logic Machines: From Ancient Greece to Silicon Valley." He surveyed the devices, both notational and mechanical, that have been used to deal with logical problems from the first development of Aristotelian syllogisms all the way to the problems of Boolean algebra and its use in describing electrical and electronic circuits. Smillie's talk, interspersed with amusing examples, covered the attempts of the thirteenth-century Spanish mystic Ramon Lull to convert the Moslem world to Christianity by means of his simple machine "Ars Magna," several nineteenth-century systems—Stanhope's "Demonstrator," Jervons's "Logical Piano," and Lewis Carroll's "Game of Logic"—and the works of Claude Shannon and Jan Łukasiewicz, now being used in the developments of integrated circuits and microprocessors. The treatment of this line of development is, I believe, unique to Keith Smillie, for I know of no other historian interested in the development of machines designed for solving syllogisms.

The third paper of the afternoon, "The EDSAC: Getting Programs Right," was presented by Martin Campbell-Kelly. He described how the EDSAC was completed at Cambridge University, England, in May 1949 and how a programming system was set up in early 1950. He was able to illustrate the debugging problems faced by the first programmers with a sample EDSAC program tape, which was a very early undebugged version of a program written by Maurice V. Wilkes. The mixture of simple clerical and logical errors present in the program was a graphic illustration of the difficulties faced by people attempting to write real programs for one of the world's first working machines. Campbell-Kelly also showed how the techniques developed for "getting programs right" on the EDSAC had a major effect on the programming of other contemporary computers. Some of Campbell-Kelly's material has already appeared in his four papers in the *Annals*, and he indicated that he was currently working on a description of the recently discovered early EDSAC program.

The final session of the afternoon was a descriptive slide show by A. D. Booth. He started by describing the early mechanical calculating equipment devised by Pascal and Morland and ended by comparing the calculating abilities of some of the early electromechanical and electronic computers with the PET microcomputer available today. In between was a fascinating series of stories about how he became involved in computing in the late 1940s, went to the United States to spend some time working at the Institute for Advanced Study with von Neumann, constructed several different early computers at Birkbeck College (London), and eventually constructed several more while he was dean of engineering at the University of Saskatchewan. Booth's tales were combined with personal reminiscences about von Neumann, Aiken, Turing, Wilkinson, and other pioneers he met and worked with in the early years.

Although there were several parallel sessions at the conference, the history of computing talks were well attended—about a quarter of all the conference participants were at this session—which illustrates the ever-growing interest in the history of our subject. All of the talks were recorded, and it is hoped that CIPS will make copies available at some later date. The papers by Cohen and Smillie were printed in the CIPS '82 conference proceedings.

M. R. Williams
Department of Computer Science
University of Calgary
Calgary, Alberta T2N 1N4

Anecdotes

Notes from the Mathematical Centre

The following items are adapted from "MG 25 Informatics Symposium," published by the Mathematical Centre in Amsterdam (see our review on pages 289–290 of this issue). The first excerpt is from W. L. van der Poel's "Some Notes on the History of ALGOL," which the author describes as an anecdotal history of the period from 1962 to 1971. The second is a summary from Maurice V. Wilkes's "The Changing Computer Scene, 1947–1967."

□ At the beginning of the St. Pierre de Chartreuse meeting [in October 1965] we saw three volunteers who had done their job. The first report was by van Wijngaarden ("Orthogonal Design and Description of a Formal Language," Report MR76, Mathematisch Centrum). It says on its cover: "Premature and preliminary edition, intended for use by IFIP WG 2.1 only." Only some 30 copies were produced, so it certainly is a collector's item nowadays. The second was from Niklaus Wirth ("A Proposal for a Report on a Successor of ALGOL 60," Report MR75). This was also produced at the Mathematical Centre, where Wirth had been working for some time. The third was from Gerhard Seegmüller ("A Proposal for a Basis for a Report on a Successor to ALGOL 60," Bavarian Academy of Science, Munich).

Furthermore, there was an important paper on the topic of record handling by C. A. R. Hoare. This paper included many ideas on how to bring in what we now call "structured values." I cite the following from Hoare's "Record Handling."

> For example the question of "who is mother of" is answered by asking the value of the reference field "mother" which is allocated for this purpose in the declaration of the record class for cows. In real life, most relationships are confined to holding between members of given classes; for example a house cannot be the mother of a cow, nor can a cow contain a table.

As a reaction to this, van Wijngaarden circulated the accompanying rebuke around the table. It illustrated how a cow can contain a table and how an object can be both a cow and a table. He then asked, "Can't a house be the mother of a cow if even a mountain can be the mother of a mouse?"

This St. Pierre meeting can be considered as the true beginning of the new language. The first two formal resolutions taken at that meeting ran as follows.