Robert W Floyd, In Memoriam

by Donald E. Knuth, Stanford University

Nobody has influenced my scientific life more than Bob Floyd. Indeed, were it not for him, I might well have never become a computer scientist. In this note I'll try to explain some of the reasons behind these statements, and to capture some of the spirit of old-time computer science.

Instead of trying to reconstruct the past using only incidents that I think I remember, I will quote extensively from actual documents that were written at the time things happened. The remarks below are extracted from a one-hour keynote speech I gave to the Stanford Computer Forum on 20 March 2002; many further details, including images of the original documents, can be seen in a video recording of that lecture, which has been permanently archived on the Internet by Stanford's Center for Professional Development [`scpd.stanford.edu`]. As in that lecture, I won't attempt to give a traditional biography, with balanced accounts of Bob's childhood, education, family life, career, and outside interests; I believe that the intriguing task of preparing such an account will be undertaken before long by professional historians who are much more qualified than I. My aim here is rather to present a personal perspective.

My first encounter with Floyd's work goes back to 1962, when I was asked by *Computing Reviews* to assess his article "A descriptive language for symbol manipulation" [*Journal of the Association for Computing Machinery* **8** (1961), 579–584]. At that time I was studying mathematics as a second-year grad student at Caltech; he was working as a programmer-analyst at Armour Research Foundation in Chicago. Since I had recently completed a compiler for a subset of ALGOL, and had read the writeups and source listings of several other compilers, I was immediately impressed by what he had written [see *Computing Reviews* **3** (1962), 148, review #2140]: "This paper is a significant step forward in the field of automatic programming. Over the past few years, simple algorithms for analyzing arithmetic expressions have been discovered independently by many people. But conventional methods for explaining such algorithms obscured the essential facts. Floyd has developed a new notation which lets the trees be distinguished from the forest, and which admirably points out what is really going on in a translation process. An algebraic compiler can be described very precisely and compactly in this notation, and one can design such a compiler in Floyd's form in a few hours." In essence, Bob had introduced the notion of *productions* as an organizing principle for programming, anticipating to a certain extent the future development of so-called expert systems.

My own work on programming was merely a sideline by which I could pay for my college education and prepare to start a family. During the summer of 1962 I wrote a FORTRAN compiler for a small UNIVAC computer; this work had almost no connection with what I viewed as my future career as a teacher of mathematics, except that I did spend one fascinating day studying the efficiency of "linear probing" (the simple hash table algorithm by which my compiler maintained its dictionary of symbols). I had never heard of "computer science." My whole attitude changed, however, when I met Bob for the first time in person at the ACM conference in Syracuse at the end of that summer.

We became fast friends, perhaps because we had both learned programming in the late 1950s by sitting at the consoles of IBM 650 computers. Bob showed me some work he had been doing about mathematical techniques for verifying that a program is correct—a completely unheard-of idea in those days as far as I knew. The accepted methodology for program construction was quite the opposite: People would write code and make test runs, then find bugs and make patches, then find more bugs and make more patches, and so on until not being able to discover any further errors, yet always living in dread for fear that a new case would turn up on the next day and lead to a new type of failure. We never realized that there might be a way to construct a rigorous proof of validity; at least, I'm sure that such thoughts never crossed my own mind when I was writing programs, even though I was doing nothing but proofs when I was in a classroom. I considered programming to be an entirely different category of human activity. The early treatises of Goldstine and von Neumann, which provided a glimpse of mathematical program development, had long been forgotten. I was also unaware of John McCarthy's paper, "Towards a mathematical science of computation," presented at the IFIP Congress in Munich that summer, nor did I associate McCarthy-style recursive functions with "real" programming at that time. But Bob taught me how to wear my programmer's cap and my mathematician's cloak at the same time.

Computer programs were traditionally "explained" by drawing flowcharts to illustrate the possible sequences of basic steps. Bob's proof method was based on decorating each branch in the flowchart with an assertion such as "$R \geq Y > 0$, $X \geq 0$, $Q \geq 0$, $X = R + QY$", which captures the essential relations that hold between variables at that point of the calculation. If we can show that the assertions immediately following each step are consequences of the assertions immediately preceding, we can be sure that the assertions at the end of the program will hold if the appropriate assertions were true at the beginning. Of course this is a simple principle, once it has been formulated, but it dramatically opened my eyes. When Bob published it later ["Assigning meanings to programs," *Proceedings of Symposia in Applied Mathematics* **19** (1967), 19–32], he gave credit to unpublished ideas of Alan Perlis and Saul Gorn, but I'm sure that he had developed everything independently. His paper presented a formal grammar for flowcharts together with rigorous methods for verifying the effects of basic actions like assignments and tests; thus it was a direct precursor of the "preconditions" and "postconditions" subsequently developed by Tony Hoare.

We began writing letters back and forth. In one letter, for example, I mentioned among other things that I'd been trying without success to find a systematic way to decide whether a given context-free grammar is ambiguous, even in the simple case

$$\langle A \rangle \ ::= \ \langle x \rangle \ \mid \ \langle A \rangle \, \langle x \rangle$$

where $\langle x \rangle$ is a finite set of strings. He replied on 16 October 1963—using the stationery of his current employers, Computer Associates of Wakefield, Massachusetts—as follows: "I applaud your results on TTL's [whatever those were ... I've forgotten]; but see Greibach, 'The Undecidability of the Ambiguity Problem for Minimal Linear Grammars', Information and Control, June 1963, pg. 119. The paper by Landweber in the same issue is also interesting." Then he proceeded to present an algorithm that solves the simple case I had mentioned. We both learned later that he had thereby rediscovered a method of Sardinas and Patterson that was well known in coding theory [August A. Sardinas and George W. Patterson, "A necessary and sufficient condition for unique decomposition of coded messages," *Convention Record of the I.R.E., 1953 National Convention*, Part 8: Information Theory (1953), 104–108].

Near the end of 1963, Bob came to visit me at Caltech, bringing fresh Maine lobsters with him on the plane. We spent several days hiking in Joshua Tree National Monument, talking about algorithms and languages as we went. (He loved the outdoors, and we hiked together in Coe State Park several years later.) At the time I was getting ready to draft the chapter on sorting for *The Art of Computer Programming* (*TAOCP*). Soon afterwards I had occasion to travel to Boston, where I visited him at a beautiful new home that he and his first wife had built in Topsfield, Massachusetts. We talked about some new ideas in sorting that I had just learned, in particular the "heapsort" algorithm of J. W. J. Williams [soon to appear in *Communications of the ACM* **7** (1964), 347–348; I had been the referee]. Bob responded by introducing an improvement to the initialization phase of that procedure [*Communications of the ACM* **7** (1964), 701]. I also introduced him at that time to the notion of sorting networks, namely to the oblivious methods of sorting that had been introduced by R. C. Bose and R. J. Nelson ["A sorting problem," *Journal of the Association for Computing Machinery* **9** (1962), 282–296]. Shortly after my visit, Bob wrote me a letter dated 5 February 1964, which began by discussing the expected length of the longest decreasing subsequence of a random permutation. Then he said "About the sorting system you showed us, I find that any sorting procedure by interchanges of adjacent lines will correctly sort all inputs if it will correctly sort an input which is in reverse (decreasing) order." This elegant result, and his lemma that proved it, eventually became exercise 5.3.4–36 of *TAOCP*. He ended his letter by saying, "Was this the theorem you wanted me to find? Ask me another."

Bose and Nelson had conjectured that the sorting networks they constructed were optimal, having the fewest possible comparators. Support for their conjecture was obtained by Thomas N. Hibbard ["A simple sorting algorithm," *Journal of the Association for Computing Machinery* **10** (1963), 142–150]. But Bob found improved methods soon after he had learned of the problem, first reducing the number of comparison-exchange modules needed to sort 21 elements from 118 to 117, and then (on 27 March) showing that 9 elements can be sorted with only 25 modules instead of 27.

This startling breakthrough was the beginning of an exciting exchange of letters. On 4 April I wrote back, "Dear Bob, I was quite impressed, indeed awe-struck, etc., by the example you sent me last week showing an improvement of two steps in the fixed-comparison sort for nine elements. ... Therefore I lost another day from book-writing as I pondered this question anew. Here are the few results I obtained today; I hope you check them for accuracy and I also hope you are inspired to find bigger and better improvements. ... I think I can break that $n^{\log_2 3} - n$ barrier, in the following algorithm for $n = 16$. (This is a little startling since powers of two are the best cases for the Bose-Nelson sort.)..." Bob replied, on 10 April: "Very pretty! Now generalize to three-dimensional diagrams...." And we continued to exchange letters dozens of times with respect to this question of efficient networks for sorting. Whenever I had mail from Bob, I'd learn that he had gotten ahead in our friendly competition; then it was my turn to put *TAOCP* on hold for another day, trying to trump his latest discovery. Our informal game of pure-research-at-a-distance gave us a taste of the thrills that mathematicians of old must have felt in similar circumstances, as when Leibniz corresponded with the Bernoullis or when Euler and Goldbach exchanged letters. However, by the time we finally got around to publishing the fruits of this work ["The Bose–Nelson sorting problem," in *A Survey of Combinatorial Theory*, edited by J. N. Srivastava (Amsterdam: North-Holland, 1973), 163–172], we had learned that Kenneth E. Batcher had blown our main theorem away by finding a much better construction.

All of this was incidental to our main research, which at the time was focussed on the translation of artificial languages like ALGOL into machine language. Indeed, all computer science research in those days was pretty much carved up into three parts: either (1) numerical analysis or (2) artificial intelligence or (3) programming languages. In 1963 Bob had written a masterful paper, "Syntactic analysis and operator precedence" [*Journal of the Association for Computing Machinery* **10** (1963), 316–333], in which he launched an important new way to approach the parsing problem, the first syntax-directed algorithm of practical importance. And he followed that up in 1964 with an even more wonderful work, "The syntax of programming languages—A survey" [*IEEE Transactions on Electronic Computers* **EC–13** (1964), 346–353], probably the best paper ever written about that subject. In this survey he masterfully brought order out of the chaos of various approaches that people had been using in the input phases of compilers, but even more important was his introduction of a completely new algorithm with a brand new control structure. He presented this novel method "with a metaphor. Suppose a man is assigned the goal of analyzing a sentence in a PSL [phrase-structure language, aka context-free language] of known grammar. He has the power to hire subordinates, assign them tasks, and fire them if they fail; they in turn have the same power. ... Each man will be told only once 'try to find a G' where G is a symbol of the language, and may thereafter be repeatedly told 'try again' if the particular instance of a G which he finds proves unsatisfactory to his superiors." I think the algorithm he presented in this paper can be justly regarded as the birth of what we now call object-oriented programming.

At the end of 1964 I had nearly finished drafting Chapter 10 of *TAOCP*, the chapter on syntax analysis, and I wrote Bob a long letter attempting to explain a general approach that had emerged from this work (now known as LR(k) parsing). "I must apologize for the complexity of my construction (indeed, it is too complicated to put in my book), but this seems to be inherent in the problem. I know of at least three Ph.D. theses which were entirely concerned with only the most simple cases of parts of this problem! As I go further into chapter 10 I become more convinced that only five really worthwhile papers on scanning techniques have ever been written, and you were the author of all five of them!"

Bob became an Associate Professor of Computer Science at the Carnegie Institute of Technology in the fall of 1965, introducing among other things a course on "the great algorithms," and supervising the Ph.D. theses of Zohar Manna (1968), Jay Earley (1968), and Jim King (1969). He also wrote another major paper at this time, "Nondeterministic algorithms" [*Journal of the Association for Computing Machinery* **14** (1967), 636–655], setting out the general principles of exhaustive search in a novel and perspicuous way that has led to many practical implementations. I found in my files a letter that I'd written to Myrtle Kellington in June, 1967, urging her to have the illustrations in this paper typeset by the printer instead of using the much cheaper alternative of "Leroy lettering." I argued that "Floyd's article, perhaps more than any other article I have ever seen, is based almost entirely on illustrations coordinated with text. ... Saying he should prepare his own diagrams is, in this case, like telling our authors never to use any mathematical formulas unless they submit hand-lettered copy. ... Professor Floyd has been one of our best and most faithful referees in the ACM publications for many years, and he has volunteered much of his valuable time to this often thankless job. Now he is an associate editor of JACM. We certainly owe him a decent treatment of his article." I'm happy to report that she agreed, even though she had received my letter more than two weeks after the publication deadline.

Meanwhile my publishers and I had asked Bob for a detailed critique of *TAOCP* Volume 1, which was being prepared for the press in 1967. Needless to say, his comments proved to be invaluable to me, although I didn't agree with every single one of his remarks. Here are some excerpts from what he wrote: "Chapter I: Overall opinion. I like the chapter, but I think it could be improved by chopping most of the humor and anecdotes, retaining the historical material. ... The system of rating problems underestimates their difficulty for, say, college seniors, and designates too many with the '▸'. The author's personal notes of advice, etc., are often valuable; at times, though, the non-technical material gets a little thick. The technical content meets very high standards of scholarship, and is a credit to the author." Then he gave hundreds of detailed suggestions for improvements to the text.

Our correspondence was not entirely technical. On 22 February 1967 I wrote, "Bob, I have the feeling that this is going to be a somewhat extraordinary letter. During the last year or so I have been getting lots of offers of employment from other colleges. I think I told you that I plan (and have planned for a long time) to decide on a permanent home where I will want to live the rest of my life, and to move there after the year I spend in Princeton working for the government. (Namely, to move in September 1969.) Due to the present supply and demand in computer science, I am fortunate enough to be able to pick just about any place I want to go; but there are several good places and it's quite a dilemma to decide what I should do. I believe the four places that are now uppermost in my mind are Stanford, Cornell, Harvard, and Caltech (in that order). I expect to take about a year before I make up my mind, with Jill's help. It occurs to me that I would very much like to be located at the same place you are, if this is feasible; at any rate your plans have a non-trivial place in the non-linear function I am trying to optimize! ... So I would like to explore some of these possibilities with you. ..." Bob responded with his own perspective on the current state of affairs at various universities; the bottom line of his reply was, "I'd say if you want to make the move, I don't have any plans that would conflict, and I will be very tempted to go to Stanford myself; I probably will go, in fact."

I received and accepted Stanford's offer a year later, and George Forsythe (the chair of Stanford's department) asked me in March of 1968 to write a letter of recommendation on Bob's behalf. I replied that "I don't know anyone I could recommend more highly. He is the most gifted man in his 'age bracket' that I have ever met. Several of his published papers have been significant mileposts in the development of computer science, notably his introduction of precedence grammars, tree-sort algorithms, and methods of 'assigning meanings to programs.' I have also had the pleasure of carrying on frequent correspondence with him for five years, so I am quite familiar with his unpublished work too; this correspondence covers a wide variety of topics, for example, graph theory, word problems in semigroups, mathematical notations, algorithms for syntactic analysis, theorems about languages, algorithms for manipulating data structures, optimal sorting schemes, etc., etc. While I was editing the ACM *Communications* and *Journal*, I asked him to serve as referee for several papers, and it was not uncommon for him to submit a four- or five-page review containing important suggestions for improvements. He also has a good record of working with students at Carnegie Tech on both the undergraduate and graduate level: He has supervised some excellent theses and he has kept several student projects going. He is one of the rare people who have considerable experience and expertise both in writing computer programs and in developing useful theories related to programming. ... He is a true Computer Scientist! His special talents seem to be (a) the ability to devise ingenious algorithms and combinatorial constructions; (b) the ability to develop nontrivial new theories which are of both practical and mathematical interest; (c) the ability to organize a large body of loosely connected material and to perceive the important ideas; (d) a talent for good exposition and for finding just the right words to express an idea. His only fault known to me is that he is sometimes a little too sensitive (too much the perfectionist); for example, although he has lived in the East nearly all his life, he has already decided that no California wine is worth drinking except B. V. Beaujolais. ... One further remark is perhaps necessary, considering contemporary 'standards of society'. Floyd has never gone through the formalities of obtaining a Ph.D. degree. I believe this was due primarily to the fact that he entered graduate school at the University of Chicago when he was only 16 or 17 yours old, as part of an experimental accelerated education program; this was not a mature enough age to do graduate work. [Bob was born 8 June 1936, and he began graduate school after receiving a B.A. degree in 1953 at age 17—about five years earlier than usual for American students at the time.] Certainly he has written at least a dozen papers by now each of which is superior to any Ph.D. thesis I have ever seen in computer science, so the mere fact that he has never formally received the degree should be quite irrelevant."

(Bob used to say that he was planning to get a Ph.D. by the "green stamp method," namely by saving envelopes addressed to him as 'Dr. Floyd'. After collecting 500 such letters, he mused, a university somewhere in Arizona would probably grant him a degree.)

To my delight, Bob did receive and accept an offer from Stanford, and he arrived during the summer of 1968. While I'm quoting from letters of recommendation, I might as well continue with two more that I was asked to write later. The first of these was addressed to the American Academy of Arts and Sciences on 12 February 1974: "… it is difficult to rank [computer scientists] with respect to mathematicians, physicists, etc., since computer science is so young. A mathematician like Lehmer or Polya has been producing high quality work consistently for 50 years or more, and this is much more than a computer scientist could do … perhaps it's too easy [for computer scientists like myself] to become a fellow. On the other hand there are outstanding mathematicians like Bellman and Thompson whose work spans only 20 years or so, and that is closer to what a leading computer scientist (say 15 years) would have to his credit. I will list the two candidates who are generally recognized as leading pioneers and whose names are 'household words', and whose contributions span a broad range of topics as well as a long span of time (consistent quality): 1. Edsger Dijkstra, who is responsible for more landmark contributions in nontheoretical aspects of computer science than any other man. Namely, the current revolution in programming methodology, the fundamental principles of synchronizing cooperating processes, the method for implementing recursive processes, as well as important algorithms. Such aspects of computer science are the hardest in which to make fundamental breakthroughs. He is also an able theoretician. 2. Robert Floyd, who is responsible for many of the leading theoretical ideas of computer science as well as chairman of what I think is the leading academic department (mine! but I'm trying to be unbiased). His work is cited more than twice as much as any other person's in the series of books I am writing (summarizing what is known about programming). His fundamental contributions to the syntax and semantics of programming languages, to the study of computational complexity, and to proofs of program correctness, have been a great influence for many years, and he has also invented dozens of important techniques which are now in common use."

Second, to the John Simon Guggenheim Memorial Foundation on 3 December 1975: "Professor Floyd is one of the most outstanding computer scientists in the world; in my mind he is one of the top three. During his career he has been a leading light in the development of many of the key concepts of our discipline: (a) A production language to describe algebraic parsing techniques (now called Floyd productions). (b) The notion of precedence grammars. (c) Semantics of programming languages. (c) Automatic techniques for constructing computer programs and proving their correctness. Each of these contributions has essentially created an important subfield of research later pursued by hundreds of people. Besides this he has invented many algorithms of practical importance (e.g. to find all shortest paths in a network, to sort numbers into order, to find the median of a set of data), and he has several landmark technical papers which show that certain problems are intrinsically hard. As an example of this, I can cite his recent discovery of the fastest possible way to add numbers: This result meant that he had to invent an addition procedure which was faster than any others heretofore known, and to prove that no faster method will ever be possible. Both of these were nontrivial innovations. In my recent book which reviews what is known about sorting and searching, his work is cited 20 times, far more than any other person. His published papers show an amazing breadth, especially when one realizes that so many of them have been pioneering ventures that became milestones in computer science. I am sure that whatever he proposes to do during his sabbatical year will be of great future value to science, and so I heartily recommend support by the Guggenheim Foundation."

Bob was elected to the American Academy in 1974 and awarded a Guggenheim Fellowship for 1976–1977.

Upon Bob's arrival at Stanford he immediately became a popular teacher. Indeed, students frequently rated his problem-solving seminar, CS204, as the best course of their entire college career. He was promoted to Full Professor at Stanford in 1970, one of extremely few people to achieve this rank after having served only five years as Associate Professor (three of which were at Carnegie).

One of the greatest honors available to mathematicians and computer scientists in those days was to be invited to give a plenary lecture at an international congress. Floyd was one of only eight people in computer or system science to receive such an honor from the International Congress of Mathematicians held in Nice, 1970; the eight invitees were Eilenberg, Floyd, Knuth, and Winograd (USA); Schützenberger (France); Lavrov, Lupanov, and Kolmogorov (USSR). A year later, Bob was the only speaker to be invited to the IFIP Congress in Ljubljana by two *different* technical area committees.

I can't help mentioning also the fact that he helped me shape up my writing style. For example, I was on leave of absence at the University of Oslo when I received the following life-changing letter from his hand:

September 21, 1972

Prof. Donald Knuth

Dear Don:

Please quit using so many exclamation points! I can't stand it!! I'm going out of my mind!!! (Don't get alarmed, I'm only kidding!!!!)

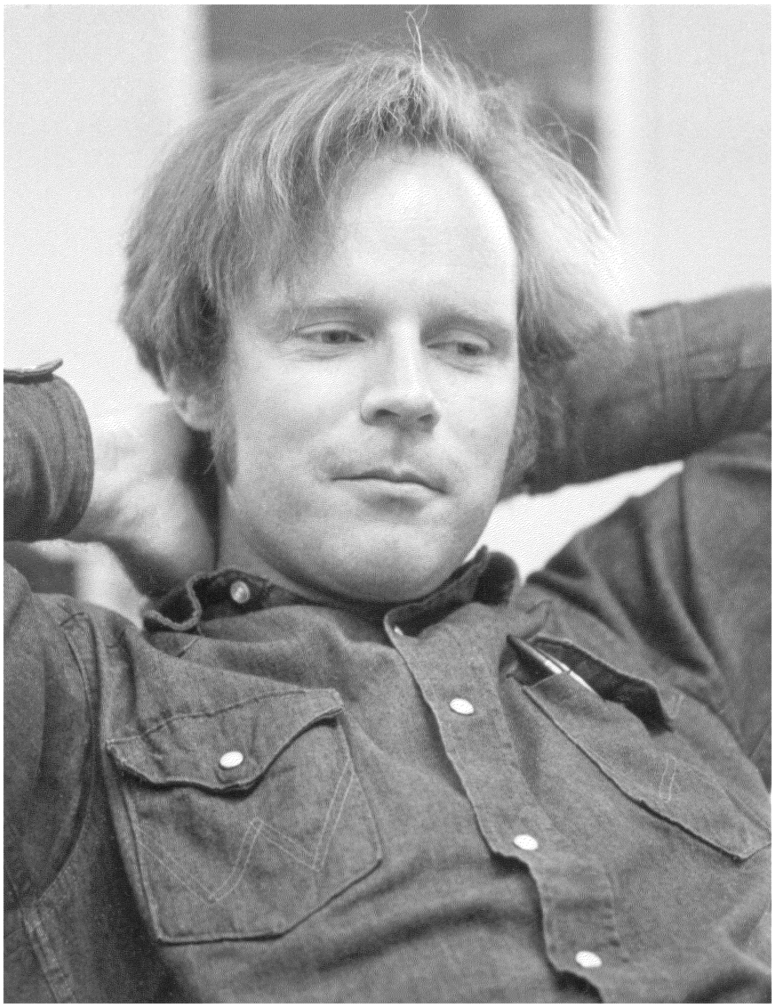
Sincerely yours (!!!!!),

Robert W. Floyd

Of course I took this advice to heart—and wrote the following reply in April of 1973 after learning that Bob was our dean's choice to succeed George Forsythe as department chair: "Bob, Congratulations, to you and to the Dean for his fine decision. I hope you will accept, since I think you will be able to accomplish important things for our department. Please be our Chairman. Sincerely, Don. P.S. If I were allowed to use exclamation points I would be more emphatic."

Bob served as department chair for three years, devoting considerable energy to the task. Above all he worked tirelessly with architects on the design of a new home for the department (Margaret Jacks Hall), in which our entire faculty would be together for the first time. For example, in December 1975 I sent him the following memo: "Thanks for all the fine work you've evidently done together with the architects for the new building. This will have a lasting payoff and we all owe you a huge debt of gratitude." And on 27 August 1976: "(Thank you)^n for the outstanding services you gave our department as its chairperson these last years. You focussed on and solved the critical problems facing us, and the present and future strength of our department is due in large measure to your efforts; the effects will last for a long time. Not having the energy to be chairman myself, I am doubly grateful that you sacrificed so much of your time to these important tasks." Finally in November, 1978, when he was enjoying a well-deserved second year of sabbatical at MIT, I sent the following message: "Dear Bob, The faculty had its first chance to walk through Margaret Jacks Hall yesterday. The building has taken shape nicely. The roof is finished, so the winter rains (if we get any) won't be a problem for the workmen doing the finishing. The carpentry work is super quality, and the spaces are nice to walk through and be in. So I'm writing to say 'thanks for all the energy you contributed towards the success of the project'. Thanks from all of us! Best regards, Don. P.S. Am enjoying the telephone poker game that Rivest told me about." [Okay, I let an exclamation point creep in, but that one was legitimate. Note also that the article "Mental poker" by Adi Shamir, Ronald L. Rivest, and Leonard M. Adleman in *The Mathematical Gardner*, edited by David A. Klarner (Belmont, California: Wadsworth, 1981), 37–43, credits Bob with proposing the problem of playing a fair game of poker without cards.]

Bob received the ultimate honor in our field, the ACM Turing Award, at the end of 1978—for "helping to found the following important subfields of computer science: the theory of parsing, the semantics of programming languages, automatic program verification, automatic program synthesis,



This picture of Bob Floyd was taken by a Stanford student and posted on our department's photo board about 1972. To simulate gray scale with binary pixels, I've rendered it here using the Floyd–Steinberg "error diffusion" algorithm, implementing that algorithm exactly as suggested in the famous article that Bob published with Louis Steinberg in 1976 (diffusing errors from bottom to top and right to left); the resolution is 600 dots per inch. Caution: Software for printing might have mangled the bits that you are now seeing. Furthermore, the physical model in Floyd and Steinberg's paper matches the technology of inkjet printers better than that of laserjet printers, so you may not be seeing this image at its best. Error diffusion does, however, give beautiful results when it has been tuned to work with a typical inkjet device.

and the analysis of algorithms. Your work has had a clear influence on methodologies for the creation of efficient and reliable software." I was surprised to notice, when rereading his Turing lecture "The paradigms of programming" [*Communications of the ACM* **22** (1979), 455–460, with a nice picture on page 455], that he had recommended already in 1978 many of the things that I've been promoting since 1984 under the banner of "literate programming." [See page 458 of his Turing lecture; and see Donald E. Knuth, "Literate programming," *The Computer Journal* **27** (1984), 97–111.]

At the time Bob was receiving this award, and for many years afterwards, I was immersed in problems of digital typography. Thus I wasn't able to collaborate very much with him in the latter part of his career, although we did have fun with one project ["Addition machines," *SIAM Journal on Computing* **19** (1990), 329–340]. He was destined to be disappointed that his dream of a new, near-ideal programming language called Chiron was never to be realized—possibly because he was reluctant to make the sorts of compromises that he saw me making with respect to TeX. Chiron was "an attempt to provide a programming environment in which, to the largest extent possible, one designs a program by designing the process which the program carries out." His plans for the Chiron compiler included novel methods of compile-time error recovery and type matching that have never been published.

I know that when he retired in 1994, the publication of his book *The Language of Machines* with Richard Beigel brought him enormous satisfaction, especially when he received a copy of the fine translation of that book into German.

Then, alas, a rare ailment called Pick's disease began to attack his mind and his body. His scientific life came sadly to a premature end. Yet dozens of the things he did in his heyday will surely live forever.

I'd like to close with a few anecdotes. First, people often assume that my books are in error or incomplete when I refer to Bob as "Robert W Floyd," since the indexes to my books give a full middle name for almost everybody else. The truth is that he was indeed born with another middle name, but he had it legally changed to "W"—just as President Truman's middle name was simply "S". Bob liked to point out that "W." is a valid abbreviation for "W".

Second, he loved his BMW, which was nicknamed Tarzan. He told me that it would be nice to own two of them, so that both cars could have license plate holders that said "my other car is a BMW".

Third, he had a strong social conscience. For example, he spent a significant amount of time and energy to help free Fernando Flores from prison in Chile. Flores, a former vice-rector of the Catholic University of Chile who had developed a computer-based information system for the entire Chilean economy and become a cabinet minister in the government of Salvador Allende, came to Stanford as a research associate in 1976 largely because of Bob's efforts, after having been held without charges for three years by the military junta headed by Augusto Pinochet.

Fourth, he was a connoisseur of fine food. Some of the most delicious meals I've ever experienced were eaten in his presence, either as a guest in his house or in a restaurant of his choice. I particularly remember an unforgettable duckling with flaming cherry sauce, consumed during an ACM conference in Denver.

Fifth, I remember 1 May 1970, the day after Nixon invaded Cambodia. Tension was high on campus; Bob and I decided that such escalation by our president was the last straw, and we could no longer do "business as usual." So we joined the students and picketed Pine Hall (Stanford's Computation Center), preventing anyone from going inside to get work done that day. (I must admit, however, that we sat there talking about sorting algorithms the whole time.)

Finally, one last quotation—this one from the future instead of the past. The seventh volume of my collected works, to be entitled *Selected Papers on the Design of Algorithms*, is scheduled to be published about two years from now. For a long time I've planned to dedicate this book to Bob Floyd; indeed, the dedication page is the only page of the book that has been typeset so far, and it has been in my computer for several years. That page currently reads as follows, using an old word for algorithmics that the Oxford English Dictionary traces back to Chaucer and even earlier: "To Robert W Floyd (1936–2001) / my partner in augrime."

I'm grateful to Christiane Floyd, Hal Gabow, Greg Gibbons, Gio Wiederhold, and Voy Wiederhold for their help in preparing these reminiscences.

Publications of Robert W Floyd

(The following list may well be incomplete; for example, I've heard that Bob published at least one article in a short-lived magazine about backgammon. Please send me any additions or corrections to this bibliography that you may know about.)

(with B. Ebstein) "A formal representation of the interference between several pulse trains," *Conference on Radio Interference and Electronic Compatibility* (1958).

"Remarks on a recent paper," *Communications of the ACM* **2**, 6 (June 1959), 21.

"An algorithm defining ALGOL assignment statements," *Communications of the ACM* **3** (1960), 170–171, 346.

(with B. Kallick, C. J. Moore, and E. S. Schwartz) *Advanced Studies of Computer Programming*, ARF Project E121 (Chicago, Illinois: Armour Research Foundation of Illinois Institute of Technology, 15 July 1960), vi + 152 pages. [A description and user manual for the MOBIDIC Program Debugging System, including detailed flowcharts and program listings.]

"A note on rational approximation," *Mathematics of Computation* **14** (1960), 72–73.

"Algorithm 18: Rational interpolation by continued fractions," *Communications of the ACM* **3** (1960), 508.

"An algorithm for coding efficient arithmetic operations," *Communications of the ACM* **4** (1961), 42–51.

"A descriptive language for symbol manipulation," *Journal of the Association for Computing Machinery* **8** (1961), 579–584.

"A note on mathematical induction on phrase structure grammars," *Information and Control* **4** (1961), 353–358.

"Algorithm 96: Ancestor," *Communications of the ACM* **5** (1962), 344–345.

"Algorithm 97: Shortest path," *Communications of the ACM* **5** (1962), 345.

"Algorithm 113: Treesort," *Communications of the ACM* **5** (1962), 434.

"On the nonexistence of a phrase structure grammar for ALGOL 60," *Communications of the ACM* **5** (1962), 483–484.

"On ambiguity in phrase structure languages," *Communications of the ACM* **5** (1962), 526, 534.

"Syntactic analysis and operator precedence," *Journal of the Association for Computing Machinery* **10** (1963), 316–333.

"Bounded context syntactic analysis," *Communications of the ACM* **7** (1964), 62–67.

"The syntax of programming languages—A survey," *IEEE Transactions on Electronic Computers* **EC–13** (1964), 346–353. Reprinted in Saul Rosen, *Programming Languages and Systems* (New York: McGraw–Hill, 1967), 342–358.

"Algorithm 245: Treesort 3," *Communications of the ACM* **7** (1964), 701.

New Proofs of Old Theorems in Logic and Formal Linguistics (Pittsburgh, Pennsylvania: Carnegie Institute of Technology, November 1966), ii + 13 pages.

(with Donald E. Knuth) "Advanced problem H-94," *Fibonacci Quarterly* **4** (1966), 258.

"Assigning meanings to programs," *Proceedings of Symposia in Applied Mathematics* **19** (1967), 19–32.

(with D. E. Knuth) "Improved constructions for the Bose–Nelson sorting problem," *Notices of the American Mathematical Society* **14** (February 1967), 283.

"The verifying compiler," *Computer Science Research Review* (Pittsburgh, Pennsylvania: Carnegie–Mellon University, December 1967), 18–19.

"Nondeterministic algorithms," *Journal of the Association for Computing Machinery* **14** (1967), 636–644.

(with Donald E. Knuth) "Notes on avoiding 'go to' statements," *Information Processing Letters* **1** (1971), 23–31, 177. Reprinted in *Writings of the Revolution*, edited by E. Yourdon (New York: Yourdon Press, 1982), 153–162.

"Toward interactive design of correct programs," *Information Processing 71*, Proceedings of IFIP Congress 1971, **1** (Amsterdam: North-Holland, 1972), 7–10.

(with James C. King) "An interpretation-oriented theorem prover over integers," *Journal of Computer and System Sciences* **6** (1972), 305–323.

"Permuting information in idealized two-level storage," in *Complexity of Computer Computations*, edited by Raymond E. Miller and James W. Thatcher (New York: Plenum, 1972), 105–109.

(with Donald E. Knuth) "The Bose–Nelson sorting problem," in *A Survey of Combinatorial Theory*, edited by Jagdish N. Srivastava (Amsterdam: North-Holland, 1973), 163–172.

(with Manuel Blum, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan) "Time bounds for selection," *Journal of Computer and System Sciences* **7** (1973), 448–461.

(with Alan J. Smith) "A linear time two tape merge," *Information Processing Letters* **2** (1974), 123–125.

(with Ronald L. Rivest) "Expected time bounds for selection," *Communications of the ACM* **18** (1975), 165–172.

(with Ronald L. Rivest) "Algorithm 489: The algorithm SELECT—for finding the ith smallest of n elements," *Communications of the ACM* **18** (1975), 173.

"The exact time required to perform generalized addition," *16th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, 1975), 3–5.

(with Louis Steinberg) "An adaptive algorithm for spatial greyscale," *Proceedings of the Society for Information Display* **17** (1976), 75–77. An earlier version appeared in *SID 75 Digest* (1975), 36–37.

(with Larry Carter, John Gill, George Markowsky, and Mark Wegman) "Exact and approximate membership testers," *10th Annual ACM Symposium on Theory of Computing* (1978), 59–65.

"The paradigms of programming," *Communications of the ACM* **22** (1979), 455–460. Reprinted in *ACM Turing Award Lectures: The First Twenty Years* (New York: ACM Press, 1987), 131–142. Russian translation in *Lektsii Laureatov Premii T'i͡uringa* (Moscow: Mir, 1993), 159–173.

(with Jeffrey D. Ullman) "The compilation of regular expressions into integrated circuits," *Journal of the Association for Computing Machinery* **29** (1982), 603–622.

(with Jon Bentley) "Programming pearls: A sample of brilliance," *Communications of the ACM* **30** (1987), 754–757.

(with Donald E. Knuth) "Addition machines," *SIAM Journal on Computing* **19** (1990), 329–340.

(with Richard Beigel) *The Language of Machines* (New York: Computer Science Press, 1994), xvii + 706 pages. French translation, *Le Langage des Machines* (Paris: International Thomson, 1995). German translation by Philip Zeitz and Carsten Grefe, *Die Sprache der Maschinen* (Bonn: International Thomson, 1996), xxvii + 652 pages.

(Floyd also wrote many unpublished notes on a wide variety of subjects, often revising and polishing them into gems of exposition that I hope will some day appear on the Internet for all to enjoy. More than 17 boxes of his papers are on deposit in the Stanford University Archives, with catalog number SC 625.)