CHARLES F. VAN LOAN

Computer Science and the Liberal Arts Student

As we edge into the "computer age" there is a new and important question confronting college educators: What should the liberally educated student know about computer science? Interest in this question is growing in the liberal arts setting as word gets around that nonexperts are successfully using the computer to solve diverse problems in the humanities, arts, and social sciences. Many colleges now offer "computer appreciation" courses for nontechnical students which involve a little computer usage to be "practical," a little coverage of computer applications to be "relevant," and a little discussion of the computer's societal impact to be "humanistic." By touching all bases in this way it is assumed that significant contributions can be made to the student's general education.

Nevertheless, there is room for improvement. The computer science education of liberal arts students appears to have a superficial quality when compared to the instruction that these same students receive in the more classical sciences. In particular, the computer appreciation course is more often a course about how the computer is used than a genuine course in computer science itself. Part of the problem is that the success of computer science as a practical discipline has discouraged interest in the subject as an agent of liberal education; it is easier to demonstrate that the computer is a "useful tool" than it is to demonstrate that the student's intellectual life is as enriched by studying computer science as it is by studying literature, art, and physics. Although the practical value of computer science in-

Charles F. Van Loan is associate professor, Department of Computer Science, Cornell University, Ithaca, New York.

struction should not be underplayed, it makes more sense in the liberal arts environment for computer scientists to promote and teach their subject in a way that is more in harmony with the aims of liberal education. The object of this article is to suggest how this can be accomplished.

To teach computer science in a liberal arts college, it is necessary to keep in mind two problems. First, there is the problem of "computer anxiety," an affliction that is similar in both cause and effect to the much studied phenomena of "math anxiety."[1] Victims of computer anxiety have a feeling of uneasy incompetence in all matters that concern the computer. They cope by trying to avoid those activities which involve the computer, even if it jeopardizes their educational and employment prospects. Convincing the computer-anxious student to take a computer science course is a major undertaking.

The second problem is a problem of time. Unlike the technical student, the liberal arts student usually has a schedule that can accommodate, at most, one computer course. Consequently, the objectives of computer science instruction for liberal arts students should be attainable in a single course if at all possible.

Bearing these two constraints in mind, let us begin our probe into the original question, "What should a liberally educated student know about computer science?"

Literacy and Computer Literacy

It has become fashionable to use the term "computer literacy" when the subject of competency in computer matters is discussed. The term invites a comparison with "ordinary" literacy because the computer, like the inventions of writing and the printing press, is radically changing the way information is accessed and transmitted. Those who fail to understand the computer as a medium for expression will be disadvantaged in the same way as those who cannot read or write. Moreover, if society as a whole is illiterate about the computer, then it must let the process of computerization be largely determined by technical specialists, an unhealthy situation. Our society is in need of liberally educated people who can also express themselves with clarity and precision on issues concerning the computer.

However, it is important to stress in the liberal arts setting that computer literacy is much more than a deterrent against computer misuse; it is a state of knowledge that can have deep intellectual significance for the individual. Consider once again the literacy metaphor:

> After reading material became available. . .the most profound changes were brought about in the literate. They did not necessarily become better people or better members of society, but they came to view the world in a way quite different from the way

> they had viewed it before, with consequences that were difficult to predict or control.[2]

If the analogy holds, what are the "profound changes" that computer literacy might induce? To answer this question it is profitable to consider first the more general notion of scientific literacy.

Scientific Literacy

Many authors have written about the value of a general scientific knowledge for the layperson. There seem to be two ways to argue for the importance of having a scientifically literate society: one is practical, and the other is "appreciative."

The nuclear accident at Three Mile Island underscores the practical value of scientific literacy. Writing in *The New York Times,* Malcolm Browne reiterated the sorry and disturbing fact that "most Americans, including many government leaders, politicians, and newsgatherers, lacked the basic scientific insight to make an intelligent assessment of what was happening."[3] In this incident, near-panic and ineffectual, irrational leadership were the consequences of scientific illiteracy.

In general, a practical scientific literacy is increasingly necessary for the day-to-day running of our lives. But for a liberally educated person this means more than a knowledge of scientific "facts"; it means a knowledge of how scientists think. As Woody writes,

> . . .probably the most general of all values that a liberal education should afford is an acquaintance with the scientific method and enough experience of it to be a judge of its use in the hands of others.[4]

Thus, the practical importance of scientific literacy is that it makes the individual an "intelligent consumer" of science and technology.

There is an equally valuable "appreciative" dimension to scientific literacy which is based upon the idea that science is a very human enterprise. A citizen's knowledge of the triumphs and tragedies of science makes the individual skeptical of those who pretend to have absolute knowledge. This is the philosophy that Bronowski articulates in his now classic book, *The Ascent of Man*. Bronowski's main contention is that intellectual tolerance is established with an appreciation of science. Perhaps better than any other discipline, science delineates what we know and what we do not know, because

> . . .we are always at the brink of the known, we always feel forward for what is to be hoped. Every

> judgment in science stands on the edge of error, and is personal. Science is a tribute to what we know although we are fallible.[5]

The importance of scientific literacy from the appreciative point of view is that it reminds us that no one person or school of thought can have a monopoly on truth—an attitude of open-mindedness consistent with the aims of liberal education.

Practical Value

Like scientific literacy, computer literacy has both a practical and an appreciative value. The second of these will be discussed later. What constitutes the practical value of computer literacy is best motivated by an analogy to the automobile, another complicated machine that can be operated by the layperson in "black box" fashion. The automobile driver who has some knowledge of cars is less likely to be fleeced at the repair shop and more likely to avoid trouble on the road than the driver who is ignorant about the workings beneath the hood. A similar comment applies to the layperson who uses the computer either directly or indirectly. Indeed, an individual is subject to "digital fleecing" unless he or she knows:

(a) the difference between a computer error and a programmer error,
(b) enough computer jargon to make dialogue with a computer specialist a reasonable possibility,
(c) the approximate level of precision and detail required to write a computer program, and
(d) how to assess the legitimacy of a computer based study.

These four tenets of practical computer literacy may be regarded as guidelines for anyone wishing to become an intelligent consumer of what the computer has to offer. Because the computer is increasingly used in all academic disciplines, the last two tenets have a special significance for the liberal arts student. In particular, "c" suggests that the student have a feel for the complexities of computerization. As Mosmann says:

> They need to know what is possible. They need some gross understanding of the kind of things that are easy and the kind that are very difficult, and how to tell them apart. They need an understanding of what "complexity" means when applied to machine based processes. Unless they know what is possible, they will not even be able to think of things they may want to use computers for.[6]

Moreover, with so many existing computer applications in liberal arts areas, it would be disastrous for the student to have the typical person's reverence for computer output ("It must be right, it came out of an electronic computer"). The intention of the last tenet is simply to encourage in the student a scholar's skepticism—not paranoia—for results that have been obtained with the computer.

The importance of practical computer literacy to liberal education is nicely summarized by paraphrasing an earlier quote:

> A liberal education should include enough experience with the computer so that the individual becomes a good judge of its use in the hands of others.

Our next concern is to ascertain the form of experience that best facilitates the reaching of this goal.

Approaches and Shortcomings

Dartmouth College, with its 300 computer terminals servicing 4000 undergraduates, is one of the leading advocates of computer usage in liberal education. According to the chief architects of the Dartmouth system, Thomas Kurtz and the current president of the college, John Kemeny, the use of the computer among students is important because

> An institution like Dartmouth College that trains many of the leaders of the nation in business, in government, and in research, has an obligation to make sure that our liberal arts graduates understand one of the most significant factors that will influence their lives.[7]

The idea behind the Dartmouth system is simple: encourage computer use by making the computer easy to use. Since 90 percent of the students at Dartmouth graduate with some form of computing experience, this appears to be a successful philosophy.

It is reasonable to extrapolate from this success and conclude that society will overcome its computer anxiety as cheap, easy-to-use computing equipment becomes more prevalent. This point is well argued by Sagan who maintains that the "adjustment to intelligent machines is, in part, a matter of acclimatization."[8] As a result of the proliferation of computer technology,

> . . .we have a generation of youngsters who are growing up with pocket computers, machine languages, computer graphics, electronic music,

automated instruction, and computer games. They are unlikely to find anything alien about machine intelligence.[9]

Children may be overcoming computer anxiety faster than adults. Perhaps Shakespeare anticipated this when he said, "old men fool and children calculate."[10] In any case, there is some reason to believe that in a few years liberal arts students will be arriving on campus with hardly a trace of computer anxiety and with considerable computing experience to boot.

However, we must not confuse the overcoming of computer anxiety with the attainment of computer literacy. The situation is analogous to the process of learning how to read. Placing books in a baby's playpen may encourage a familiarity with printed matter, but reading skills must still be taught. Equivalently, the mere exposure to computing equipment does not automatically make a person computer literate if that term implies, as it should for liberal arts students, an ability to read and write perceptively about computer matters. Some type of instruction is required.

One possibility is the "computers and society" course in which the sociological implications of the computer are stressed. These are valuable courses, and a great deal has been written about their content and administration.[11] However, in assessing their value, it is important to bear in mind a remark made by Kemeny about "science and society" courses in general. He argues that they are "a good kind of humanities course, but [they are] almost more important for the scientist than the nonscientist."[12] The problem is that a "science and society" course requires some knowledge about the science in question to be appreciated fully. As a consequence, "computers and society" courses are less than ideal for those liberal arts students who have no knowledge about computer science.

A far more serious complaint against such courses is that they are frequently used as a substitute for bona fide computer science instruction, a circumstance that diminishes the student's liberal education. This point is stressed by Rosovsky, dean of the Harvard Faculty and a prime mover of that university's new policy on core curriculum. In an interview in *The New Yorker,* Rosovsky complains that at Harvard, "too many science courses were about science instead of being basic science."[13] Liberal education cannot be based upon vicarious experience; engaging in science, even in a limited way, is the only means whereby the nonscientist can gain what Rosovsky calls an "intellectual sympathy" for the scientist. This is central to the appreciative value of scientific literacy discussed earlier.

From these remarks we conclude that, for a computer science course to have full liberal education value, it must compel the student to engage in the same type of thinking that the computer scientist engages in, and it should do this by confronting the student with a representative class of elementary problems taken

from the field. In order to be more specific, it is necessary to have a clear understanding about what constitutes computer science.

The Study of Algorithms

There are many ways to describe the multi-faceted field of computer science.[14, 15] However, to highlight the role that the subject can play in liberal education, it is useful to characterize it as the study of algorithms. "An algorithm is a precisely defined sequence of rules telling how to produce specified output information from given input information in a finite number of steps."[16] We are all familiar with the algorithm for long division that produces a quotient and remainder (the output) from the dividend and divisor (the input). In fact, the word "algorithm" is a corruption of "al-Khorwarizmi," the name of a ninth century Persian who wrote a book about certain methods for performing arithmetic.[17]

The notion of an algorithm is not so alien to everyday life as some would believe. For example, a recipe for pumpkin pie can be regarded as an algorithm telling how to go from the ingredients (the "inputs") to the finished pie (the "output"). Likewise, a book about etiquette may be thought of as a collection of algorithms each of which is designed to produce a specified "output behavior." Computer scientists are primarily interested in those algorithms whose individual instructions are so elemental that they can be executed by a machine. Nevertheless, I am reminded of an article by a computer scientist who studied cooking recipes in search of what makes an algorithm readable.[18]

Specialists in any field, computer science included, are often skeptical of one-line characterizations of their specialty such as the one offered above. "After all," they argue, "what we do is too complicated and too diverse to admit a simple description." While I am sympathetic to the difficulties involved, it is important to distinguish between trying to settle an in-house argument about whose research niche is most important and trying to give a clear view of a subject to the outside world. In the case of computer science this is particularly important because all too often it is portrayed as a purely technical subject whose sole mission is the inculcation of programming skills. As we shall see, by emphasizing algorithms and "algorithmic thinking" rather than computers and particular programming skills, a rich, appreciative dimension to computer literacy is made possible.

The Appreciative Value

Beyond helping the student acquire a practical computer literacy as described earlier, the primary objective of computer science instruction in the liberal arts setting is to give the student a quality exposure to algorithmic thinking. By "algorithmic thinking" I mean the mental activity required to express the solu-

tion to a complicated problem in the form of an algorithm. Usually this means an algorithm that can be executed by a computer, i.e., a computer program. As pointed out by Levien and Mosmann, acquaintance with this type of thinking is an important component of liberal education because computer programs are a new and major way that man has to express what he knows.[19]

Equally important, however, computer programs are a new and major way that man has to discover what he does not know, and therein lies the appreciative value of computer literacy. Algorithmic thinking does not always culminate in a completely successful computer program. However, even when this occurs it is a valuable intellectual exercise, because the act of making implicit thoughts explicit during the programming process is likely to reveal inconsistencies in our logic and unforeseen difficulties associated with the problem at hand. What Bronowski says about science can also be said about the computer—it provides a setting where the boundaries of our knowledge are brought into sharp focus. This is because, when we convey what we know to the computer, we are usually required to be painfully detailed and precise.

In science this phenomena has had a well known effect. When a scientist expresses a theory in the form of a computer program, its weaknesses and strengths are exposed, and the execution of the program may very well suggest further experiments and model refinements. It is this role of the computer as a "suggestive aid" to human reason rather than its supplanting role as a "number cruncher" that makes the study of computer power in science inseparable from a study of our own mental capacities.

The computer can similarly complement the human intellect in nonscientific areas. Moreover, the existence of machines that display limited aspects of learning, intelligence, and creativity adds a new twist to the age old question, "What makes human beings so unique?" Perhaps the overriding aim of liberal education is to instill a lifelong interest in this question and to acquaint the student with the various types of thinking that can be applied to answer it. If we accept this, then appreciations of literary thinking, religious thinking, artistic thinking, scientific thinking, and even algorithmic thinking are each to be valued for the complementary insights that they provide.

Form and Content

Having argued that computer science can contribute to the student's liberal education, I now feel obliged to give some concrete suggestions as to how this might be accomplished. Recalling the assumption that the liberal arts student has time for at most one computer course, I will describe the structure of a single introductory course which I will nickname "CS for Poets." It will not be possible to go into much detail because many final syllabus decisions depend upon local factors such as student

background, faculty competence, alternative course offerings, and computing facilities.

To many nontechnically inclined students, algorithmic thinking, however important, is just too boring and alien to consider in depth. ("I can't think like a computer and I don't want to learn.") In order to attract students with attitudes of this sort, it is important that "CS for Poets" project the image of an appreciation course rather than a course for potential specialists. In music appreciation, for example, the aim is not to produce musicians but

> . . .to remove the prejudices and biases against serious music and to help students become sensitive to the less obvious qualities of music which only training can bring into awareness. . . .If education communicates the more refined portions of musical culture to one who would not be able to find them otherwise, then his power to control what happens to him musically and aesthetically becomes less a matter of accident.[20]

Similarly, the aim of "CS for Poets" is not to produce programmers but to help the student overcome his or her "prejudices and biases" against the computer. However, one of the best ways to accomplish this is to require the student to do as the natives do, i.e., program.

Indeed, a "CS for Poets" without computer programming, like a "Chem for Poets" without lab, can be criticized as a vicarious course *about* science. As the philosopher Whitehead said, "first hand knowledge is the ultimate basis of intellectual life."[21] In computer science, computer programming is a primary way of gaining first hand knowledge.

My experience has been to peg the level of programming high enough so that the essence of algorithmic thinking is conveyed, but low enough so as not to swamp the student with details. At Cornell we use the "Poet's Subset" of the programming language PL/I.[22] Other programming languages such as BASIC and FORTRAN can also be used with success. Regardless of the language chosen, it is imperative to remember that the teaching of saleable programming skills is not one of the aims of "CS for Poets." On the other hand, students who discover that they actually have an aptitude and liking for computer programming have that modicum of experience to take a rigorous introductory programming course with newfound confidence.

"CS for Poets" must not be a course solely devoted to computer programming, for computer literacy involves more than technical expertise. It is necessary, therefore, to fill out the nonprogramming portion of the course with some form of activity to reveal further the nature and consequences of algorithmic thinking. One vehicle for doing this is to discuss applications, although it is important to bear in mind that "computer science is

not a union of applications."[23] In order to prevent this aspect of the course from becoming a potpourri of examples, the course must emphasize the core algorithmic concepts. Rather than giving a sequence of colloquial reports on how computers are used in X, Y, and Z, it is more important to stress the algorithms which underlie X, Y, and Z. This approach imparts a unity to the discussion of applications and saves valuable class time because a solid understanding of a few basic algorithms goes a long way towards understanding many seemingly diverse examples. Of course, the discussion of actual computer applications does have an important role to play in "CS for Poets," but only in a *supporting* way. As in a mathematics course, examples serve to enliven and motivate the underlying theoretical concepts.

For purposes of organizing the applications portion of "CS for Poets," it is useful to identify four units of instruction:

1. *Information Processing*
   What is information? How is is stored, retrieved, manipulated, encrypted, transmitted, and checked for errors?

2. *Scientific Computation*
   What is the effect of finite precision arithmetic? How is computer simulation and statistical computation changing the character of physical, biological, and social science?

3. *Artistic Computation*
   How can audio and visual data be manipulated? What is the creative process, and can it be accelerated by the computer?

4. *The Display of Intelligence*
   What constitutes intelligence? What are the computational problems associated with pattern recognition, game playing, and natural language processing?

Again, the emphasis is on algorithmic thinking and the viewpoint it gives to interesting philosophical questions. An adequate complement of problems and programs must be assigned to insure that the student emerges with a "first hand" knowledge of the concepts involved. It is also important that the student leave the course with a rough idea about what computer scientists do who study programming languages, complexity theory, numerical analysis, information retrieval, and other important areas within the subject. Further topics consistent with the aim of "CS for Poets" may be found in Hamming's proposal for a computer appreciation course.[24] Although this proposal is somewhat dated, it is very good for its emphasis on scientific topics.

There is one topic that Hamming mentions that I would like to

discuss since it is underplayed by most authors—the history of computation. My experience has been that several lectures on the historical aspects of computer science at the beginning of "CS for Poets" pay big dividends in terms of rendering an overall perspective and as a means for helping the student overcome his or her computer anxiety.

The Importance of History

Andrew D. White, the first president of Cornell University, remarked in his autobiography that "the best of all methods for presenting every subject bearing on political and social life is historical."[25] Computer science surely has far-reaching political and social impact, but is its history long enough to afford worthwhile perspectives? This would seem unlikely if we accept the prevailing view that the subject "really began" in 1946 with the completion of ENIAC, the first fully electronic digital computer. This date is important because it was only with the advent of sophisticated computers that the important theoretical underpinnings of the subject such as the model of computation proposed by Alan Turing in 1936 could be adequately pursued.[26]

However, I think it is important *not* to regard computer science as a post-World War II development. In fact, I would argue that, just as the ancients could practice astronomy without the benefit of the telescope and Kepler's laws, so could they "practice" computer science without the benefit of the IBM 370 and Turing's results. For example, the Babylonians expressed the solutions to complicated algebraic problems in algorithmic form over 3500 years ago.[27] Just as it took thousands of years for man to recognize that Mercury, Venus, and Earth are all instances of a general object called "planet," so has it also taken thousands of years for him to abstract the notion of an algorithm from countless special cases.

Advocates of computer science as a "new" discipline tend to overlook the very episodes in its history which have contemporary relevance. For example, insight into the pedagogic value of calculators in elementary school can be derived by considering Plato's argument that the good of geometry is undermined when geometers resort to more than the compass and straightedge in their computations. Likewise, an understanding of Leibniz's proposal to compute solutions to social and philosophical problems via his "calculus of reason" buys us a rightful skepticism of those who maintain that the computer can be used for the same purpose.

These examples suggest that computer science has a rich cultural tradition, if we consider the long history of ideas about computation and not just the computing devices themselves. Ignoring this tradition makes us more susceptible to fads and exaggerations. Man has always had an inflated view of his computational aids, as evidenced by what the English philosopher Hobbes

wrote about the Pascal Adder, which was built in 1645. Although this device could only add two eight-digit numbers, Hobbes claimed that

> . . .brass and iron have been invested with the function of brain, and instructed to perform some of the most difficult operations of mind. . . .In what manner so ever there is a place for addition and subtraction, there is also place for reason, and where these have no place, there reason has nothing at all to do; for reason is nothing but reckoning the consequences. . . .When a man reasoneth, he does nothing else but conceive a sum total from addition of partials.[28]

Against a history of such exaggerations, the wild predictions about computer intelligence and computer power made by over-enthusiastic computer specialists would tend to be placed in their proper perspective.

The history of computation has also a deep pedagogical value, as it enables key technical concepts to be explained in situations more pleasing to the computer-anxious student. Thus, it is more attractive to introduce the notion of binary control in the context of the Jacquard Loom (1801) than the CRAY I. Likewise, the modern computer is more intimidating in its complexity than Babbage's Analytical Engine (1843) and hence, less valuable as a machine for illustrating such concepts as machine language and assembly language.

In summary, presenting computer science as a consequence of World War II shortchanges the student and harmfully identifies the subject as an "upstart" discipline. It is important to recall that computer science is directly concerned with three of the seven original liberal arts: logic, grammar, and arithmetic.[29] The appreciation of computer science by those concerned with liberal education is facilitated when the subject is perceived as a classical one of ideas rather than as a transitory one of equipment.

Conclusion

There are several aspects of the computer literacy problem in higher education that I have not pursued such as the education of faculty, the role of computer-aided instruction, and the question of the overly narrow education of the computer specialists themselves. These are difficult problems, but ones which are attracting more and more attention.[30, 31, 32] In any case, the role of computer science in higher education is an exciting one insofar as the subject can serve as an "academic clubhouse where interdisciplinary stimulation can counteract the well known effects of specialization."[33]

In all this hoopla about algorithmic thinking, however, we must

not get carried away and let computer science run roughshod over the other constituents of a liberal education. FORTRAN is not a substitute for French, and contrary to a claim made in an unnamed computer science journal, the subject will not become "the modern *equivalent* of liberal arts education." To believe otherwise is to reveal an unbecoming narrowness of mind. George Boole, a nineteenth century logician whose work is of paramount importance to computer science, expressed this very well when he said that

> . . .[although] the cultivation of mathematical or deductive faculty is a part of intellectual discipline, so truly is it only a part. The prejudice which would either banish or make supreme any one department of knowledge or faculty of mind, betrays not only error of judgment, but a defect of that intellectual modesty which is inseparable from a pure devotion to truth.[34]

All too often, the computer tempts us into believing that the only "valid" knowledge is that which can be expressed in algorithmic form. It is this tendency to imitate the computer's preference for machine readable data that alarms some of those who contemplate the direction of the computer revolution. [35, 36] But it is not enough to be alarmed. We must act. I hope I have shown that a good course of action in higher education involves teaching liberal arts students about computer science.

Notes

1. Sheila Tobias, *Overcoming Math Anxiety* (New York: W.W. Norton & Company Inc., 1978).
2. Alan Kay, "Microelectronics and the Personal Computer," *Scientific American* 237 (September 1977):231-44.
3. Malcolm Browne, "The Untutored Public," *The New York Times,* April 22, 1979, Section 12, p. 1.
4. Thomas Woody, *Liberal Education for Free Men* (Philadelphia: University of Pennsylvania Press, 1951), p. 250.
5. Jacob Bronowski, *The Ascent of Man* (Boston: Little, Brown and Company, 1973), p. 374.
6. C.J. Mosmann, "Computers and the Liberal Education," *The Educational Forum* 36 (November 1971):85-91.
7. General Information Catalog, Dartmouth College, Hanover, New Hampshire, August 1978, p. 22.
8. Carl Sagan, "In Praise of Robots," *Natural History Magazine* 84 (January 1975):8-20.
9. Ibid.
10. Shakespeare, *Julius Caesar* (I, iii, 65).
11. E. Horowitz and M.C. Horowitz, "Computers and Society: An Interdisciplinary Approach," *SIGCSE Bulletin* 5 (September 1973):134-37.

12. Edward B. Fiske, "3 Ways to Science Literacy," *The New York Times,* April 22, 1979, Section 12, p. 14.
13. *The New Yorker,* December 4, 1978, p. 42.
14. Frederick A. Hosch, "Some Comments on the Role of Computer Science Education," *SIGCSE Bulletin* 5 (September 1973):13-17.
15. Donald Knuth, "Computer Science and Mathematics," *American Mathematical Monthly* 81 (1974):323-43.
16. Ibid.
17. Donald Knuth, *The Art of Computer Programming,* Vol. I (Reading, Ma.: Addison-Wesley Publishing Company), p. 1.
18. Lance Miller, "Natural Language Procedures: Guides for Programming Language Design," IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 1976.
19. Roger Levien and Charles Mosmann, "Instructional Uses of Computers," in *The Emerging Technology—A Carnegie Commission Report on Higher Education and Rand Corporation Study,* ed. Roger Levien (New York: McGraw-Hill, 1972), p. 57.
20. Foster MacMurray, "Pragmatism in Music Education," in *Basic Concepts in Music Education,* ed. Nelson B. Henry (Chicago: University of Chicago Press, 1958), pp. 30-61.
21. A.N. Whitehead, *The Aims of Education* (New York: The Free Press, 1929), p. 51.
22. R.W. Conway, *Programming for Poets* (Cambridge, Ma.: Wintrop Publishers, Inc., 1978).
23. G.E. Forsythe, "A University's Educational Program in Computer Science," *Communications of the ACM* 10 (1967):3-11.
24. G.E. Forsythe, "Engineering Students Must Learn Both Computing and Mathematics," *Journal of Engineering Education* 52 (December 1961):177-88.
25. Andrew D. White, *Autobiography of Andrew Dickson White,* Vol. I (New York: The Century Co., 1905), p. 381.
26. Alan Turing, "On Computable Numbers with an Application to Entscheidungsproblem," (Proceedings of the London Mathematical Society, November 17, 1936), pp. 230-65.
27. Donald Knuth, "Ancient Babylonian Algorithms," *Communications of the ACM* 15 (1972):671-77.
28. D.E. Halacy, *Charles Babbage, Father of the Computer* (New York: Crowell-Collier Press, 1970), p. 42.
29. Donald Knuth, "Computer Programming as Art," *Communications of the ACM* 17 (1974):667-73.
30. Richard H. Austing, Bruce H. Barnes, and Gerald L. Engel, "A Survey of the Literature in Computer Science Education Since Curriculum '68," *Communications of the ACM* 20 (January 1977):13-21.
31. Andrew R. Molnar, "The Next Great Crisis in American Education: Computer Literacy," *Technical Horizons in Education* 5 (July/August 1978):35-38.
32. U.S. Congress, Subcommittee on Domestic and International Scientific Planning, Analysis, and Cooperation of the Committee on Science and Technology, *Computers and the Learning Society,* 95th Congress, 2nd Session, 1978.
33. G.E. Forsythe, "Educational Implications of the Computer Revolution," in *Applications of Digital Computers,* ed. W. Freiberger and W. Prager (Waltham, Ma.: Blaisdell, 1963), pp. 166-78.
34. George Boole, *The Laws of Thought* (London: Walton and Maberly, 1854), p. 444.
35. Norman Cousins, "The Computer and the Poet," *Saturday Review* 49 (July 23, 1966):42.
36. Joseph Weizenbaum, *Computer Power and Human Reason* (San Francisco: W.H. Freeman and Company, 1976).