A SELF-PACED FIRST COURSE IN COMPUTER SCIENCE

Dr. John O. Aikin
Director of Computer Services
The Evergreen State College
Olympia, WA 98502

Abstract

As demand for a first course in computer science increases, more efficient and effective approaches to such a course become increasingly desirable. This paper describes the development and use of a completely self-paced CAI course at The Evergreen State College. Use of behavioral objectives in designing the course is explained, the content of the course is outlined, the process used to develop the course is described, experiences with 256 students are reported, and some general observations on implementing CAI courses are offered.

Student Demand for "The First Course"

Like most colleges today, The Evergreen State College has been experiencing a rapid growth in student interest in computers and programming. In addition to those full-time students who plan a concentration in computer science or who need some knowledge of computers in order to pursue their other studies, an increasing number of community residents seek to understand this new force in our society. This latter category includes owners and employees of small businesses who suspect that (but do not know precisely how) computers can improve their operations, consumers who are intrigued by the idea of having their own computer but don't know how to choose or use one, and many others.

CAI as a Response

We began with the premise that CAI might offer an effective way of presenting an introduction to computers and programming to a large number of students. Many researchers (e.g., Jamison et al. (1974) and Homeyer (1970)) have reported that CAI can be at least as effective as traditional instruction and that its use typically reduces the time required for students to master material.

Pessel (1977) reported on the use of a self-paced (although not CAI) introduction to computers course at the University of Rochester which suggested that demands on faculty time might be reduced and attractiveness to part-time students increased through this approach.

CAI (like the traditional large lecture course) offers a way of multiplying the number of students who can be taught by one person. Further, it offers the student almost complete freedom to choose when instruction will take place. Finally, CAI both requires and makes possible a competency-based approach to instruction that allows each student to proceed at his or her own pace. These all seemed like good objectives for which to strive in designing a new course.

How the Course was Developed

In 1978, the college sought and obtained a grant from the Control Data Corporation to develop such a course using the PLATO system of computer-based education. Control Data provided the equipment and the college provided the staff to develop, implement, and test the course. The course was first used in the fall of 1979. It was extensively tested through the 1979-80 academic year, then published by Control Data.

The remaining sections of this paper describe the course, report our experiences in using it, and comment on some general lessons we have learned about CAI. The reader should bear in mind that the major objective of the paper is not a comparison of CAI and its costs with traditional methods of instruction, but rather a compendium of methods which we believe are helpful in applying CAI.

Methodology and Pedagogy

We began by analyzing our student audience and the content which we felt it was important to convey in a first course in computer science. We tried to keep in mind that this course would also be the last course in computing for many of these students.

The methodology which we followed in designing the course is similar to that advocated by Lynne Baldwin (1978). One proceeds by developing a set of resultant behaviors that one desires students to exhibit, specifying what entering behaviors are expected, and then detailing enabling objectives that make possible the resultant behaviors.

While there has been (and continues to be) much debate about "behaviorist" approaches to education, it is fairly clear that computer-based education requires such an approach to designing courseware because of the nature (some would say limitations) of the medium.

To illustrate how this procedure works, the first module of the course has the objective of introducing the student to computers. We decided that having an introduction to computers <u>means</u> that the student knows something about:

(1) the uses of computers,

(2) the history of computers,

(3) how computers work,

(4) numbering systems,

(5) peripheral devices,

(6) how data is represented and stored, and

(7) the social issues surrounding computers and automation.

To carry this example a bit further, we decided that "knowing something about how computers work" <u>means</u> that the student knows:

(3a) what the major parts of a computer are,

(3b) how these parts work together,

(3c) how instructions are represented and data stored,

(3d) that a program is a detailed sequence of instructions, and

(3e) how a program is executed by the machine.

Such a breakdown analysis is carried out to the level where one knows exactly what content is implied in the innocent statement that one is providing an "introduction to computers." Of course, a way of assessing whether the student has gained such a knowledge is also required. Good objectives ought therefore to be stated in terms of testable behaviors rather than in terms of "having knowledge." For instance, one says that a student "knows the parts of a computer" if s/he can name them, that a student "knows about peripheral devices" if s/he can recognize and identify the functions of several, and so forth.

It is a well known fact that no two teachers will ever agree on the specific choices of objectives and the particular methods used to assess students' mastery of these. The important point here is that the method just described forces certain results, <u>whatever</u> choices one makes. Among these are that one cannot avoid becoming very precise about what one is teaching, why one has chosen to teach this and not that, and how one can tell if one has succeeded. We believe this to be an enormous benefit of the behavioral objectives method over less formal approaches to structuring course content. At least one can tell where one disagrees with a colleague's course design. (For a review of some varied approaches to the "Intro" course, see Rine (1978).)

<u>Content and Organization</u>

The course is divided into five modules. The first module has been largely described. It provides what is believed to be a sufficient introduction to computers, principles of operation, and terminology to permit the student to converse intelligently about computers and to go through the rest of the course.

The remaining four modules provide a thorough introduction to programming using the BASIC language as a vehicle. While a detailed exposition of content would best be given in terms of behavioral objectives (for the reasons previously discussed), for the present purposes a briefer, more conventional outline of course content will suffice:

Module A: Introduction to Computers

1. Uses of Computers
2. History of Computers
3. How Computers Work
4. Number Systems
5. Peripheral Devices
6. Computer Data
7. Social Issues

Module B: Introduction to BASIC

1. High Level Languages
2. Introduction to BASIC
   (line numbers, variables, LET, and END)
3. Input/Output in BASIC
   (PRINT and INPUT)
4. Doing Arithmetic in BASIC
5. Problem Solving: Writing a Complete Calculation Program

Module C: Dealing with Data in BASIC

1. Alphabetic Data (Strings)
2. Storing Data in Programs
   (DATA, READ and REMark)
3. Controlling Program Flow
   (GOTO and IF...THEN)
4. Advanced Input/Output
   (LINPUT, ENTER, and PRINT USING)
5. Solving a Complete Data Handling Problem
   (searching a list for an item)

Module D: Control Structures in BASIC

1. Structure and Organization in Programs
   (Structured Programming, IF...THEN DO*)
2. Program Documentation
3. Iterative and Control Loops
   (FOR...NEXT loops, WHILE...DO*, and

UNTIL...DO*)

4. Arrays
5. A Complete Structured Program

Module E: Advanced BASIC Tools

1. Functions
2. Subroutines
3. Files
4. A Complete Application Using Files

*NOTE: While most versions of BASIC can hardly be said to facilitate good program design and coding practices, the version of BASIC used at Evergreen (and taught in this course) is an exception. Local enhancements provide all of the control structures required for teaching structured methodology. Indeed, the only reason statements such as GOTO are introduced at all is a recognition of the fact that most BASICs that the student will subsequently use do not provide other control structures.

For each topic in the outline, there is a statement of the objective, an assessment test, and at least one learning activity. Typically there are several learning activities, some presenting concepts and others involving application exercises. The majority of the learning activities are CAI, but parts of two conventional texts (Computers in Action, by Donald Spencer and BASIC: A Hands-on Method, by Herbert Peckham) have been integrated as assignments.

To aid the student, we developed a BASIC interpreter available through PLATO which provides exceptionally "friendly" diagnostic messages and a simplified environment in which the student may practice programming. Throughout the course, the student is referred to the simulator for programming exercises that reinforce material presented in the learning activities. Within all of the CAI lessons, there is a dictionary available which allows the student to request a definition of any unfamiliar technical term.

The PLATO system completely coordinates the student's progress through the self-paced course. Following a mandatory group orientation meeting, the student usually begins by going through an introduction to the PLATO keyboard and course organization. The student may then elect to challenge the test for the first module, or begin studying lessons for that module. Whenever the student is ready for a test on one or more objectives, PLATO constructs and administers an objective test, scores the test, and (if required) prescribes learning activities for review prior to the next text. When all of the objectives in the first module have been mastered, the student is able to go on to the second module. When that module has been mastered, the student goes on to Module C, and so on until the last module is mastered.

When the student finishes studying the course materials, s/he picks up a comprehensive take-home examination. In order to receive credit, the student must perform satisfactorily on this examination. The student must also submit specifications, a listing, and a sample run of an individual programming project. These are evaluated by the instructor (along with a written self-evaluation prepared by the student) and used to make a credit determination and construct the narrative evaluation that becomes part of the student's permanent record.

As the student progresses through the course, s/he has the ability to leave a note (via the computer) for an instructor, to which the instructor may respond. Course instructors read and answer their "mail" several times each day. The subjects of such notes may be simple queries about information presented in a lesson, requests for an appointment, or reports of errors in either the system or in one of the lessons.

Experience in Using the Course at Evergreen

Since this course was first offered fall quarter 1979, 256 students have enrolled in the course for credit. An additional 116 students went through the course as part of the Coordinated Studies Program* "Society and the Computer." Since students in this latter group did not take the course in a completely self-paced mode, they will not be included in subsequent analyses, although their comments have been very helpful in refining the course.

The enrollment in the course has thus been about 50 students each quarter and has remained remarkably high for five consecutive quarters. Enrollment has been deliberately limited in order to permit adequate access to equipment.

The first observation that one might make is that this level of enrollment is almost precisely twice as large as the number of students taking the "Intro" course when it was being taught using a traditional approach. While it is likely that every student has a different reason for taking any course, student responses to the questions in an exit survey suggest strongly that the self-paced approach to the course does make it available to many who would otherwise be unable to enroll. Students repeatedly comment favorably on the fact that they are able to easily mesh other class and work schedules with the course, to work at their own pace, and to avoid traditional passive and public classroom learning situations.

About forty-five percent (45%) of the students taking the course have done so as regular full-time students; the majority thus being part-time students for whom one might expect the foregoing factors to be especially significant. About forty percent (40%) of the students taking the course have been women. This is a much higher percentage than one historically finds in a technical course of this type. One might therefore wish to argue

. . . .

* See Aikin (1978) for a discussion of Coordinated Studies.

that the self-paced mode is an effective way of making computer science more accessible to women, although it is unclear just why this might be so.

Obviously a factor of great interest is whether the students who enter the course actually complete it and earn credit, in other words, do they learn? The following table summarizes some of the relevant data:

	Fall, 79	Wtr., 80	Sprg., 80	Smr., 80
Enrolled	51	38	63	50
Actually began	44	35	58	40
Finished PLATO	23 (52%)	20 (57%)	42 (72%)	27 (68%)
Earned Credit*	25 (59%)	18 (51%)	44 (76%)	32 (80%)
Hours of PLATO time to complete	30.7	29.7	32.7	28.2
Mean student age	Not avail.	27.8	28.8	28.2
Percent part-time students	29% (est)	57%	48%	63%

* NOTE: It is possible for students to earn credit without completing all of the PLATO modules if they pass the final exam and submit a satisfactory project. Therefore this number can exceed the number of students finishing the PLATO sequence.

It is clear from these results that not every student is able to successfully complete the course and earn credit. It is also clear, however, that the percentage of students completing the course and earning credit has increased since the course was first offered. In part this is an artifact: During the first two quarters, the number of students exceeded our expectations and we did not have sufficient terminals to permit every student enough access to complete the course. More importantly the effectiveness of the learning activities has been steadily improved and sources of confusion have been gradually eliminated from the course structure.

While evaluations of the 56 students enrolled during fall, 1980 have not yet been completed as of this writing, all but three actually began the course this time and we expect more than 75% to complete the sequence and earn credit.

The number of students currently completing the course and earning credit thus compares favorably both with other courses at the college and, more importantly, with the traditional version of "Introduction to Computers and Programming" taught previously. It also compares favorably with the 40% figure cited by Pessel (1977) for the self-paced course then in use at the University of Rochester.

It is worth noting that the average age of students taking this course (28.4 years) is more than five years greater than the overall mean age (23) of students at the college. This is thus far, however, the only significant demographic observation that can be made. We are now examining in detail the survey data that has been collected and we hope to be able to use this data to predict which students will have difficulty with the self-paced mode. We would like to be able to administer an entry questionnaire that could detect those students who are likely not to complete the course and advise them to register for something else.

Advantages Over a Traditional Course

The principal advantages over a traditional course are the ones already touched upon: greater availability to students and as effective or more effective presentation and mastery of the material. Some other advantages also deserve mention, however. First there is a real reduction in the staff effort required to teach the course. While we would not be prepared to argue that CAI is cheaper than traditional means of instruction,* it is clear that CAI permits one instructor to serve many more students. It is also clear that CAI may make the repeated teaching of an introductory-level course both more tolerable for the instructor and more effective for the student. High demand initial contact courses are among the most difficult to teach well because they usually require enormous amounts of student contact and they often do not stretch the limits of the discipline. The computer does not tire of being repeatedly assigned to teaching the same old course. To the extent that the authors of a CAI course succeeded in putting a joy in the subject into their work, that joy will be there time after time, student after student.

A second result of using CAI to teach this particular course is that students are forced to spend many hours interacting with a computer. As a way of exposing students to both the powers and the frustrations of the machine this probably cannot be improved upon. Students develop an ability to use a terminal (including the typewriter keyboard) which they would not be as likely to develop otherwise. They experience directly how very sophisticated and subtle can be the machine's interaction with the human. They also become painfully aware of how even the most carefully (we like to think) designed large program will display the machine's rigidity and susceptibility to the programmer's error. These are important lessons for students to learn early in a potential computer career and for the citizenry generally to possess.

Finally, although one might expect that using a machine to teach would be viewed by students as impersonal and perhaps even dehumanizing, student comments in the exit survey reveal that many students have found PLATO to be more personal and

. . . .

* For a discussion of CAI costs, see <u>Computers and the Learning Society</u> (1978).

patient than a human teacher. The machine's temperament is constant, students' mistakes are private, and the student can ask a question (through the dictionary) or repeat a lesson without embarrassment. Older students are particularly likely to comment favorably on this aspect of CAI.

Disadvantages

There are, of course, some disadvantages too. Some students simply cannot learn through CAI. This is not meant as a criticism of such students, but rather as a statement of a fact that must be dealt with, just as some students cannot learn from lectures. It appears that students have three major kinds of difficulties in mastering a self-paced course of study. First, self-motivation assumes a critical role in self-paced study. There is no teacher standing there with assignments that must be turned in; there isn't even a regular time when the student has to be somewhere. The entire motivational burden falls on the student. Educational purists will probably not find this a disadvantage at all, but the fact is that many students have been ill prepared to assume such a burden. There is a consistently higher drop-out and failure rate among full-time students entering this course directly from high school than among older students returning to school. More than three fourths of the latter group complete the course and earn credit, while slightly under two thirds of the students entering directly from high school do so. These figures should be viewed with caution, however, since the number of students entering directly from high school is small. Nevertheless, these data suggest that CAI works best with mature students. (Of course, almost all educational methods work best with mature students, but the difference may be more pronounced in courses taught using CAI. Pessel (1977) reports similar experiences with student procrastination.)

Second, there appear to be some students who have physical difficulties in spending the amount of time using a terminal that is required for this course. Some students report eyestrain, for example. Once it is explained to students that they can cope with this by arranging shorter sessions on the computer and by refocusing their eyes periodically, most, but not all, students can overcome this obstacle.

Third, and probably most significant for CAI as a methodology, the machine can never be as flexible and responsive as the best human instructor. Despite the hundreds of hours of time that has been invested by the authors and the extremely flexible system provided by PLATO, most people would learn better and faster from a highly skilled personal human tutor. But that is not the real alternative! For some reason, critics frequently compare CAI with the best human instructor under the best and most individualized conditions. How about the 500 student lecture course taught mainly by an inexperienced TA or by the faculty member who is teaching it for the umpteenth time and has an exciting graduate seminar coming up in the afternoon? Perhaps one should simply acknowledge that the best human teachers under the best circumstances are much more effective than CAI, but that for many students in many circumstances, CAI will be as effective for equal numbers (though perhaps for different individuals) as the real alternatives.

One of the things that can go a long way towards overcoming some of the disadvantages of CAI is to seek frequent feedback from students using the course. Where do they have problems, what do they complain about? Such feedback can help refine a CAI course, but in the end we have found three devices most helpful in "humanizing" the course: First, students are repeatedly urged to use the notes capability of the system and instructors are religious about responding quickly. This provides students with a way to get a quick answer to an individual question.

Second, students are provided three opportunities each quarter to come to a "class" meeting. It is interesting that before we did this, there were fairly frequent comments about how much such lab sessions were needed, but once we started doing it, only 5-10 students out of the group of fifty ever came. Apparently the reassurance that the option is there is needed, but it is rarely exercised.

Finally, students are encouraged to form study groups on their own. Again, not too many students do so, but there is some evidence that participation in such a group can make the difference between success and failure for some of the individuals who join them. It may be that the favorable impact of study groups is more due to their motivational impact than to any direct clarification of the subject matter.

Comments on Preparing CAI Materials

The remainder of this paper discusses the process of designing and constructing "courseware" and presents some observations that may be helpful to others planning to develop CAI materials.

The construction of CAI courseware incorporates elements of programming, teaching, and writing. It is similar to programming in that it obviously does involve programming a computer, but more significantly in that the entire methodology of structured design and implementation is highly appropriate. An outline* of the process which was used to develop the present course will illustrate this parallelism:

I. Develop overall course design
   - A. Statement of what students are to learn
     (overall terminal objectives)
   - B. Analysis of audience
     (entering behaviors, assumptions)

. . . .

* At each major division of the outline there is a review, revision, and approval cycle before proceeding to the next major division.

C. Instructional task analysis (enabling objectives for terminal objectives)

D. Testing strategy (how will student achievement be assessed?)

E. Evaluation and approval process (how will course success be assessed?)

(how will design/implementation be approved?)

F. Design constraints (budget, time frame, delivery system, etc.)

G. Development schedule

II. Develop detailed design documents for each learning activity under each enabling objective including:

A. Description

B. Prerequisites

C. Objectives addressed

D. Content outline

E. Method of presentation (medium)

F. Testing strategy

III. Design utilities and general support systems

A. Key conventions

B. Standard entries, exits, screen formats, etc.

C. Dictionary and other "help" services

D. Title page formats

E. Unit linkages within CAI lessons

F. Graphics/animation utilities

G. Simulators

H. Etc.

IV. Implement computer management structure (CMI)

A. Course management (sequencing, testing procedures, etc.)

B. CAI lesson stubs

C. Statements of objectives

D. Test stubs

V. Implement learning activities

A. Subject matter expert writes "script" with:

1. text

2. graphics

3. student interactions

4. other specifications

B. Script is reviewed and edited by other members of team.

C. Programmer implements script as CAI lesson.

D. Professional editor reviews for language and clarity.

E. Students use lesson with the "comment" feature enabled so that they can point out troublesome portions.

VI. Entire course is tested with students, then revised as needed.

VII. Course is submitted to publisher for review (assures adherence to technical and mechanical standards), then published.

The foregoing process is obviously an ideal, but it clearly illustrates the similarities between programming and developing CAI materials. In general, we would conclude that the application of structured design methodology to constructing CAI materials can be expected to yield the same beneficial results as in programming.

Some other observations grow much more directly out of our own experiences in authoring the course and are not so obvious. For example, there is a learning curve for being able to use the CAI medium effectively. What works in the classroom, or in a text book, is not the most effective use of the CAI medium. Since it takes time to master the instructional medium, we would suggest that new authors write the <u>last</u> lessons in a courseware package first. That way, students will have experience in learning from the medium when they encounter one's worst lessons and will have their introduction to the medium through one's best lessons.

Some basic stylistic guidelines are:

(1) Avoid "page turners;" that is, CAI lessons that simply use the computer as a text display medium. Such lessons have no advantages over text books and may even be less effective because the student's "window" is smaller.

(2) CAI is most effective when it is highly individualized and when it frequently requires the student to <u>do</u> something. Therefore one stylistic objective is to ask frequent questions. Ideally these questions should be designed to allow the student to demonstrate that s/he already knows something, thus allowing branching to material which the student has not mastered. Likewise, questions can be used to verify that a concept which has just been presented has been understood. When it hasn't, the lesson should take appropriate action to reinforce the concept, ideally by presenting it in an alternative way. As a goal, one might strive to write materials such that each student follows a unique path based on what the student already knows and on how rapidly the student can absorb new knowledge.

(3) Graphics and animation can make possible very effective presentation of ideas by drawing attention to key points. Graphics is especially powerful as a way of illustrating processes. But, graphics for graphics' sake can be extremely tedious for students if they are forced to wait repeatedly for the display to complete. What is cute the first time is not cute the 51st time.

(4) Guiding the student through a task, or otherwise simulating a process is a highly effective approach which CAI can accomplish well. For example, several of our best lessons ask the student to write a program to accomplish a task, giving feedback on the student's choice of program logic and statements as s/he develops the program. This is much more effective than simply presenting concepts. Such lessons are not easy to write, however.

(5) It is important to provide a consistent package for a course. When students begin using a CAI system, they have a technical hurdle to overcome before they can concentrate on the material: there is an unfamiliar keyboard and machine to master, a new way of studying material, etc. It is extremely helpful if the author is kind enough to design a unified course where keys always do the same sort of thing, where a given typeface always is used in the same way (for instance using italics every time a new word is first used), where the screen formats consistently locate action directives in a particular place, etc. For this reason, there are advantages to authoring large packages (entire courses) rather than small ones.

(6) Use mixed media, with the computer making assignments, testing, and only sometimes actually teaching. Too much CAI is like too much TV; it's better to intersperse some text materials, or a lab, or an audio or filmstrip activity. This helps to break up the monotony.

(7) Try to have at least two different learning activities that address each objective. This will help to insure that no student gets stuck repeating a learning activity over and over without improving.

Some other practical conclusions which we have reached concern the development process itself. Faculty (unless they are computerphiles) must have technical support to write CAI materials. In our discipline this is less of a problem than in some others, but even so, a college planning to develop a CAI course should provide competent programmers and editors to subject matter experts writing CAI materials.

We have reluctantly concluded that students cannot be trusted to author CAI materials. They can sometimes be trusted to implement scripts if standards are very rigorously enforced. Probably the best use of student help if it is available is in implementing utility routines and in testing.

It is important to test materials early and often. It is simply amazing how frequently something that seems obvious to the author does not come across to others. This happens in lectures and textbooks too, but there is a subtle difference in that within the traditional classroom setting the student has a recourse: s/he can ask for clarification. In a self-paced course there is often nowhere the student can turn if s/he gets lost. By watching people go through lessons while they are being implemented, one can locate most of the problem areas. If the system permits (as PLATO does), students can be given the opportunity to comment on lessons at any point, thus providing an effective means of constant improvement. For example, our course has a dictionary of computer terminology that allows students to enter a word or phrase and receive a definition. If a student enters a word or phrase for which no definition exists, the system records that request, and where it came from. This allows the dictionary to be updated in areas where frequent inquiries have arisen and draws attention to learning activities where there are frequent requests, thus suggesting the need for revision. Tools such as this one are very useful.

One final observation is that unlike textbooks (and even handouts), CAI can be updated or corrected immediately. Thus if a student finds an error, that error can be corrected at once so that no other students need to encounter it. When students are encouraged to report errors and when they see that their reports produce immediate changes, they will do so readily.

Conclusion

The self-paced course described in this paper has proven to be an effective way of presenting the "first course" in computer science at Evergreen. The use of CAI offers both some special advantages and some special problems. The authors have been sufficiently encouraged by the results of this project that they are now working on a similar course based on Pascal.

The BASIC course described herein has been published by the Control Data Corporation and is available through the CDC PLATO network. To use the course one simply needs access to a PLATO terminal and the printed materials used by the course. Access to another computer system would be useful for students to practice programming in BASIC and would reduce the amount of PLATO time required, thus improving cost-effectiveness.

Bibliography

Aikin, John O., "Computer literacy: an interdisciplinary, hands-on approach at The Evergreen State College," SIGCSE Bulletin, Vol. 10, No. 3, August, 1978, pp 8-12.

Baldwin, Lynne J., "Quasi-behavioral objectives for curriculum specification," SIGCSE Bulletin, Vol. 10, No. 3, August, 1978, pp 1-7.

Homeyer, F. C., "Development and evaluation of an automated assembly language teacher," Technical Report No. 3, 1970, The University of Texas at Austin, Computer Assisted Instruction Laboratory.

Jamison, D., Suppes, P., & Wells, S., "The effectiveness of alternative instructional media: a survey," Review of Educational Research, Vol. 44, No. 1, 1974, pp 1-67.

Peckham, H. D., BASIC: A Hands-On Method, McGraw-Hill Book Company, New York, New York, 1978.

Pessel, D., "Introduction to computing self-paced: a novel approach which may be fun but does it work and is it worth it?" Proceedings of the Computer Science and Engineering Curricula Workshop, 1977, IEEE Catalog No. EHO 126-3, pp 60-63.

Rine, D. C., "A course entitled introduction to computing: for computing, engineering, and business majors," College Curricula in Computer Science, Engineering, and Data Processing, 1978, IEEE Catalog No. 78CH1311-OC, pp 34-38.

Spencer, D. D., Computers in Action, Hayden Book Company, Rochelle Park, New Jersey, 1978.

Subcommittee on Domestic and International Scientific Planning, Analysis, and Cooperation of the Committee on Science and Technology, U.S. House of Representatives, 95th Congress, First Session; Hearings: "Computers and the learning society," No. 47, U.S. Government Printing Office, 1977.

Subcommittee on Domestic and International Scientific Planning, Analysis, and Cooperation of the Committee on Science and Technology, U.S. House of Representatives, 95th Congress, Second Session; Report: "Computers and the learning society," Serial JJ, U.S. Government Printing Office, June, 1978.