ED 287 467 IR 012 863

AUTHOR	Baumlin, James S.; Cone, Dennis
TITLE	Teaching Teachers To Compute: A Revised Syllabus.
PUB DATE	[85]
NOTE	9p.
PUB TYPE	Viewpoints (120)
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	*Authoring Aids (Programing); *Computer Literacy; Course Content; Courseware; Higher Education; Humanities; Instructional Design; *Preservice Teacher Education; *Programing Languages

ABSTRACT

Education courses in BASIC remain a foundation of teacher training for computer use. Such courses do little more than re-invent programming structures that are already available in such educational programming languages (EPLs) as PILOT and TEACH. While BASIC is the only general programming language that is both simple and powerful enough to enable beginning programmers like teachers to develop software in their own content area, it typically limits them to writing simple programs with multiple-choice format answers. An EPL can generate more powerful, more flexible software in less time without requiring knowledge of a more difficult lower-level programming language. Education courses emphasizing BASIC are obsolete, and educational computing classes should be restructured toward the more versatile EPLs. Teachers, particularly in the humanities, would be better served by mastering one of these languages. BASIC remains useful to know since so many programs have been written in the language, but it should be offered at a reading-knowledge-only level in computer literacy courses. (Author/RP)

James S. Baumlin
Dennis Cone
Texas Christian University
Fort Worth, Texas 76129

Teaching Teachers to Compute: A Revised Syllabus

The distinction between computer literacy and what could be termed "computer fluency" is considerable, and, as Sue Kay Otto and James P. Pusack have observed, the "tasks teachers may be able to envision doing with the help of a computer require a high degree of expertise."[1] Most teachers, they admit, have neither the time nor the resources to acquire this expertise; yet, like many involved in teacher training, they continue to argue in favor of a general-purpose programming language like BASIC. The logic here is inherently flawed: if we know that courses in BASIC generally fail to make expert programmers, why require or even encourage such coursework for Education majors? Outside of Computer Science, teachers--certainly teachers in the Humanities--are not programmers. Nor do they need to be programmers to make effective use of computers in their instruction. Why, then, does programming in BASIC remain a focus of computer literacy courses for teachers, regardless of level and field?

There is another kind of computer course for teacher training, one which de-emphasizes programming skills and stresses application or, as Roy Agee more ambitiously describes it, the "extensive, systematic use of the published courseware with heavy hands-on experience."[2] This, we agree, would be the ideal literacy course for teachers in the Humanities; the acute problem is that "published courseware," particularly in the Language Arts, remains thoroughly inadequate. And although educational authoring systems (programs like Conduit's DASHER and AIDS) offer some possibilities for modifying the content of such courseware, the systems presently available are so limiting that they hardly begin to exploit the potential of computer-assisted instruction. Teachers in the Humanities <u>do need</u> the ability to translate their own specific materials into courseware, and neither BASIC nor authoring systems have proven fully satisfactory.

There exists, fortunately, a middle ground between the complexities of general-purpose programming languages and the structural limitations of authoring systems: the educational programming language (EPL), of which PILOT and TEACH are examples.[3] While Agee does not incorporate such educational programming languages into his revised syllabus for computer literacy (and we suspect that few courses presently do), we suggest that EPL offers the best solution to teacher training in programming. EPL should, in fact, be the central focus in teaching teachers to compute--and we are surprised, frankly, that EPL has received so little attention both in journals of education and in practical application. We present below some arguments against the present emphases in teacher training, particularly instruction in BASIC, and describe briefly some advantages of TEACH, an educational programming language we have used to design software for instruction in both college-level composition and English as a second language.

****

The weekend seminar in educational computing is now a staple of many Departments of Continuing Education, and few Schools of Education neglect to offer--indeed, some even require of their majors--a course in educational computing. What is the goal of such courses? A quick comparison of the typical Education Department course to a Computer Science course in computing will demonstrate at once that the former is <u>not</u> designed to fashion a programmer. The BASIC language--which "from a pedagogic point of view," Joseph Weizenbaum writes, is "an intellectual monstrosity"[4]--is now the sole province of hobbyists and, it seems, Departments of Education; most college Departments of Computer Science no longer even offer courses in BASIC, and high schools, too, have recognized its inadequacy (the ETS Advanced Placement Test now emphasizes PASCAL for college credit). So why do so many courses and weekend workshops teach it?

One answer, doubtless, is that BASIC is the only language simple yet powerful enough to give "amateur" programmers like teachers the ability to develop software in their own content area, software based on their own individualized teaching. Yet a teacher mastering the typical textbook on educational computing--Culp's and Nickles' <u>An Apple for the Teacher</u>,[5] for example--will be able, after some twenty-four separate lesson programs and 150 pages of instruction, to do little more than ask, in multiple-choice format, "WHICH PLANET IS EARTH FROM THE SUN?" or "WHAT IS THE STATE FLOWER OF TEXAS?" Teachers successfully completing such a weekend workshop or even semester-long course in general-purpose programming will be able to write useful though, surely, simple programs within their content area. They will have also done no more than reinvent the wheel, since educational programming languages would have generated more powerful, more flexible software in less time--and without requiring knowledge of a more difficult lower-level programming language. Education courses emphasizing BASIC programming are, simply put, obsolete, and teachers, particularly in the Humanities, would be better served by mastering an EPL.

The concern of teachers should not be with the nuts and bolts of a program. It should rest with the instructional content. And for the educator who is not a professional programmer, every hour spent constructing the program itself is an hour away from the content--an hour wasted. Once again, the wheel--the presentational structure of an educational program--has already been invented and is available in the structure of an EPL. We would argue, therefore, that the typical graduate of a course in BASIC programming can do few things better than a person using such a higher-level language; usually (depending on his or her expertise) that person can only do less than one using an EPL, and it will take considerably more time.

BASIC remains useful to know. Since so many educational programs have been written in this language, a knowledge of BASIC allows one to read and modify many extant programs. But courses teaching computer literacy to teachers should stress no more than a reading-knowledge of the language, and then get down to business: the translation, through an EPL, of content into courseware. This has been the emphasis in three-week courses we have ourselves taught to students in the graduate program in English at Texas Christian University. Using TEACH on the university's IBM mainframe computer, participants have learned, in little more than a week's time, to write software more sophisticated than a semester's worth of BASIC would have enabled. The subjects of these programs have also been diverse--excitingly so, since our students have explored content areas totally neglected by software publishers. We have learned that CAI need not be restricted to grammar drills and reading comprehension (seemingly the sole interests of publishers): using an EPL like TEACH, our students have designed programs in poetry scansion, in the analysis of stylistic features of a text, in logical analysis, in paragraph organization, in sentence-combining, even in the interpretation of literature. So why, we repeat, teach BASIC? We recommend instead that computer literacy courses stress application and the development of content-specific software by means of an EPL.

What then can teachers in the Language Arts, for example, expect to do if they learn an EPL like TEACH? To begin with, a handful of simple commands enables the novice teacher-programmer to design a tutorial session on any subject, from vocabulary to sentence mechanics, logic, and methods of literary analysis. In addition to its display of information the tutorial session receives and stores student responses (which can in turn determine what information is presented next). The presentation can be personalized by references to the student's name or the instructor's. Modifications of these few commands make it possible to branch to various points within the program: to a menu of instructions, say, to a review section, or to more challenging material. Supplementing this more sophisticated tutorial mode is TEACH's simple yet flexible drill-and-practice mode. To use TEACH for drill the teacher simply enters the questions and correct answers in a file. The program will then automatically number the questions; present them in random order; tell the student if the answer is right or wrong; give a second chance for incorrect responses; display the correct response if necessary; count the number wrong and the number right; calculate the percentage correct; and flag the items presented so that the same questions will not be presented again if the student chooses to do more. It would take literally hundreds of lines of code to do this in BASIC, while TEACH handles it with perhaps a half-dozen one-letter commands. And the teacher retains the option to intervene in various ways, adapting the drill to the content. For example, the number of chances for each question can be increased or decreased; the items can be presented in sequential rather than random order; the correct answer does not have to be displayed; and hints may be provided to those who initially answer incorrectly.

After only a few hours of orientation and on-line experience most teachers are able to begin running simple lessons they have designed and programmed themselvcs. And almost immediately they can begin trying out unique approaches and strategies; our experience has shown that, even though the presentational structure of an EFL is difficult to "rewrite " or modify, one's creativity with an EFL is limited only by the time one spends "playing" with the language and exploiting it to its full potential. Far more useful than a course in BASIC, the EFL demands more consideration than Departments of Education presently give it.

NOTES

[1] Sue K. Otto and James P. Pusack, "Stringing Us Along: Programming for Foreign Language CAI," Calico Journal (September 1983), 26.

[2] Roy Agee, "Are We Really Training Computer Teachers?" T.H.E. Journal (March 1985), 97.

[3] David H. Wyatt describes the differences among general-purpose programming languages, educational programming languages, and authoring systems: "Three Major Approaches to Developing Computer-Assisted Language Learning Materials for Microcomputers," Calico Journal (September 1983), 34-38.

[4] Joseph Weizenbaum, "Another View," a response to Stephan L. Chorover, "Cautions on Computers in Education," Byte (June 1984), 225. Alan Neibauer argues similarly that "BASIC, the language most of the current crop of computer literacy teachers are familiar with, may not be the most suitable language for program development or programming instruction." "The Computer Literacy Myth," T.H.E. Journal (February 1985), 90.

[5] George Culp and Herbert Nickles, An Apple for the Teacher: Fundamentals of Instructional Computing (Monterey, California: Brooks/Cole, 1983).