Journal of The International Council for Computers in Education

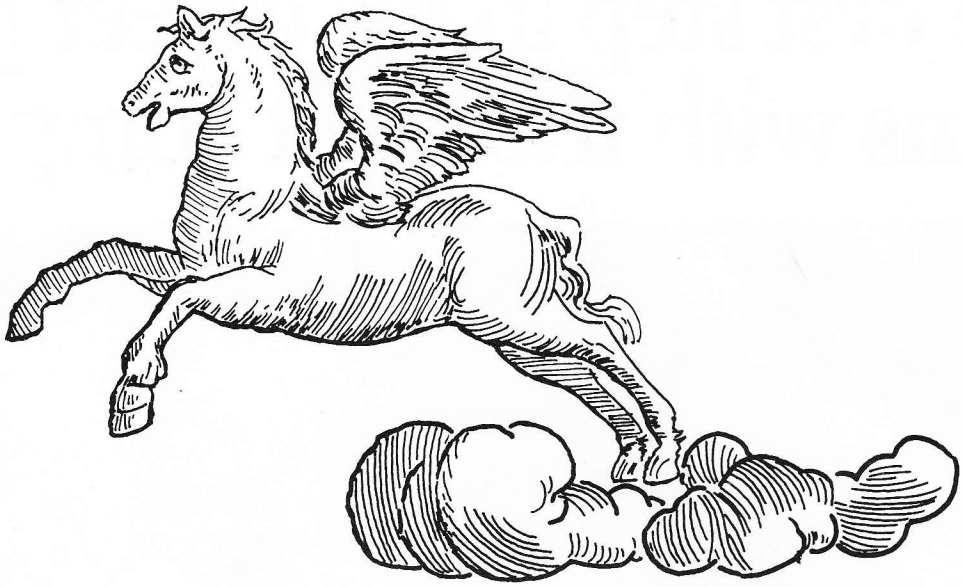
THE COMPUTING TEACHER

May 1987 • *Volume 14 Number 8* • *Price $3.50 U.S.*

Graphic by Percy Franklin

FEATURES

DEPARTMENTS COLUMNS



The Many Myths of Programming

by
Jason Ohler

After hearing numerous arguments calling programming instruction either evil or godsend, I decided to try to put these misconceptions to rest.

Programming should be treated as an art elective; it is as valid a use of time and school resources as drawing or making music. Programming should not be treated as a core skill/content area. Nor should it be presented as a skill students need in order to be computer literate or job ready. Literacy/readiness requirements are:

1. Familiarity with the microcomputer (and system components) as an operator and a consumer;
2. Proficiency in software applications, most notably word processors, spreadsheets, data base managers, and telecommunications and graphics packages;
3. An understanding of programming, but not necessarily fluency in a language; and
4. An appreciation of the opportunities and dangers of a computerized society.

Most important, application software can be used as a tool to facilitate learning in standard academic areas, allowing teachers to combine subject area material and computing skills in an integrated approach to teaching both. Programming languages are not nearly as well suited for this purpose. To find time to teach programming, a teacher usually has to give up a content area or simply squeeze programming into an already tight curriculum.

MYTHS AND MORE MYTHS

Myth #1: "If my children (or students) are going to get ahead in the world, they need to know how to program."

Response: That's simply not true. As application packages have become more powerful and commonplace, the need for and practicality of do-it-yourself programming solutions has dwindled dramatically. The market-driven economy now supports many more positions for those who are skilled application users than programmers. The main reason for this is time. Programming is a lengthy, bug-ridden process that can often be avoided by the artful use of application software.

Myth #2: "But if my child (student) has a knack for programming and has made a career decision to be a programmer, s/he should pursue BASIC in high school."

Response: Not necessarily. I have heard from more than one source that kids who hack in BASIC develop so many bad programming habits that making the transition to the more structured languages that are used in the programming profession (such as C or Pascal) becomes extremely difficult. BASIC is a great language to use in order to get acquainted with programming. But it is a lousy production language. Unfortunately, BASIC is built into both the Apple IIe and the IBM PC and has become very popular for that reason. The overwhelming presence of public domain software, particularly educational software, written in BASIC, makes it useful to know enough BASIC to modify existing software for one's own use.

Myth #3: "My child loves to program. But because market conditions do not favor programmers, it is a good idea to discourage him/her from developing this talent."

Response: I disagree, for the same reason that I don't think it's a good idea to discourage students from sculpting even though market conditions don't offer much for sculpters these days. Students internally motivated to program (or sculpt, or play music, or become an astronaut), should be encouraged to pursue their interests. Edu-

cators need to make sure that students understand how competitive a market it is for programmers (and sculpters, musicians, and astronauts) without killing the student's enthusiasm or implying that taking a risk is something to be avoided. Besides, programming could end up being a very fulfilling avocation, if not a vocation.

Myth #4: "Programming is dead. All we need to know how to do is run application software."

Response: Who do you suppose wrote the programs that rendered programming unnecessary? Elves? This is a lot like saying we don't need auto mechanics, only drivers. Society will value programmers and car mechanics for as long as computers and cars are in use. In the past three years a few projects I worked on benefitted greatly due to my programming abilities. It is a matter of perspective. The vast majority of us will only need to be drivers and software users. However, knowing how to tune up your car or write your own accounting package can be very valuable, whether you plan to be a full time mechanic or programmer or not. These skills can save money and teach us a great deal about the machines in our lives.

A closely related myth is: "Programming is not something people do a lot of after graduating from school and therefore should not be taught in school." Neither do people usually spend much time dissecting frogs, finding logarithms or performing push-ups after leaving school, and yet these activities are staples of any American education. Are we to conclude that the educational community has misinterpreted what to teach? Or are we to conclude that it is wrong to expect a high degree of correlation between what is taught in school and what adults spend their time doing?

Myth #5: "I am disturbed by kids who use valuable computer time to make the computer do useless things like make funny sounds, print numbers all over the screen, and so on. What do these kinds of activities do for their education? Do they make students any more employable? Programming is a waste of time."

Response: "I'll make my point by way of analogy. "I am disturbed by kids who use valuable art class time to make weird drawings, impressionistic paintings, and so on. What do these kinds of activities do for their education? Do they make students any more employable? Art is a waste of time." In either case, who knows what idea, problem, or creative insight the student was working through? Just because adults can't observe immediate, tangible benefits in an activity doesn't mean there aren't any, especially in a discovery learning environment. For this reason adults have to be very careful about limiting student's options. I have observed high risk students in alternative learning settings come alive under no other circumstances than when they taught themselves to program using the discovery method. Programming can empower. But so can application software.

Myth #6: "At least one thing is certain: Programming is an effective way to develop students' logical and critical thinking skills."

Response: "Logical and critical thinking skills" is probably among the most overused, least understood buzz phrases in education. I'll add my own oversimplification purely for the sake of discussion. Logical and critical thinking skills are the skills required to advance from the knowledge and comprehension levels of learning to the application and analysis levels. This shift requires a certain autonomy in the student's learning process. Learning at the knowledge and comprehension levels is largely teacher (whether human or CAI) induced. Application and analysis skills require the student to assume more leadership for his or her own learning. At these levels students need to move beyond facts, making inferences and drawing conclusions, and personalizing what they have learned. There is no application or analysis by rote.

With all of that said, there is little evidence to support the myth that programming develops these skills. Even if it does, educators still need to determine how efficiently and effectively it does so. Software exists just for the purpose of developing logical and critical thinking skills, and some educators claim it does a much better job than programming languages. This discussion begs a few questions: Does learning math and science develop these skills? How do we know? If we are going to demand that programming develop these skills, should we also demand that math and science do the same?

[Jason Ohler, Educational Technology Program Director, School of Education and Liberal Arts, University of Alaksa-Juneau, 11120 Glacier Highway, Juneau, AK 99802.]