1-1-1990

Computer programming in the junoir high

Stacey L. Hartsock

NORTHERN ILLINOIS UNIVERSITY

Computer Programming in the Junior High

A [illegible] Submitted to the

University Honors Program

in Partial Fulfillment of the

Requirements of the Baccaluareate Degree

With Upper Division Honors

Department of Learning and Instructional Technology

by

Stacey L. Hartsock

DeKalb, Illinois

May 1990

Comp~tey Programming in t~e Junior High

Computers. They used to be mystical and magical in appearance. We were all amazed at thE:?iy' capabilities. ~~ that was at the infancy of the computEr 2Ya- the students In the ~Lnc~6~ today are growing ~p ~n a computerized society. We bank through computers, we play games with computers, we make reservations through computers, and computers even connect our phone ca'ls. There is no question that we must produce computer literate adults, and our schools have adapted ~ 1(f~5.r curriculum to reflect that need. CJnE'. of the additions t~ the cuyriculum has been computer programming ever though most co not believe that it is necessary for computer literacy. But, if computer programming is going ~o be taught we neec to look at why we are teaching it and what our goals s~oul~ oe.

Although at one time the education profession without a doub0 believed that by teaching compute: programming we were teaching problem solving/thinking skills, this belief has been severely questioned ln the past few years. Numerous studies have been done but the results are often inconclusive, or contradictory to other studies conducted. With no solid ~ in either directions schools pJ,u.ni}E~ along hoping that the students are having a positive experience that will have a lastlng effect on their critical thinking ~~ Now what is more importantly discussed is why the

results of studies have produced conflicting results. One theory is that students can learn problem solving skills through computer programming if it is taught with the correct approach; another is that problem solving skills are only acquired after the student has mastered the language.

Numerous studies have been done to try to determine if students are learning problem solving skills by learning to program a computer. In general, no proof has been presented that would allow us to make a definite decision in either direction. One study, conducted with junior high students, tried to determine if problem solving and thinking skills could be improved through the teaching of [illegible] The results showed some improvement on the New Jersey Test of Reasoning Skills, in the male students only (Abrams). Another study done in 1988 by a different group of researchers found no difference in problem solving skills. They did however note that the students who had received LOGO instruction appeared to use more effective planning, and had an increased understanding of geometry. The method of instruction in this study was "inquiry-based mediation" (Lehrer). Still another study, conducted in 1973, determined that although there was a slight improvement in [illegible] thinking skills there was little evidence that the problem solving and thinking skills were transferred out of the context of the programming language (Crook, 127). In general, there seems to be a definite Lack of proof that [illegible] programming will

enhance students' ability to solve problems, but it appears that more studies are waryanted with greater attention placed on the instructional methods used.

Leah McCoy in her article "Computer Programming Can Develop Problem Solving Skills" suggests that in order for students to galn problem solving skills the programming class must teach metacognition, planning and debugging (46). Students must learn to be aware of how they aye solving a pyoblem while they *aye* solving it. McCoy also suggests that the students should be required to follow the four step problem solving plan that was developed by Polya. In thi':::; model a student first galns an understanding of the problem, secondly devises a plan, thirdly carries aut the plan, and lastly looks back to see if the answer is reasonable and to reflect on the process used to salve the problem (46). In regards to planning McCay suggests that after the students understand the problem they should be required to break the problem dawn into sub-problems and o~gani7e their plan into a flowchart *or* structure chart. The students should develop a program, from their flow chart, written in pseudocode. After this is completed the students can begin to code their program (47).

Inevitably, whenever someone cades a program they must also debug it. The synta~tic bugs are found by the c~mputey or compiler and are fairly easy to correct. Logical errors aye much moye difficult to correct. Students must ~eaYn

tactics that can be used to pinpoint errors and how to use test data to ensure that the program is producing correct results. In addition, students need to learn to reassess their programs to see if there was a more efficient way to solve the problem (McCoy, 47).

This method of instruction can be applied at every level from beginning LOGO to advanced Pascal. By observing students who learned to program by this method, four basic outcomes were noted. First, students were more willing to attempt to solve unfamiliar problems. When solving problems they tended to spend more time trying to understand the problem before looking for a solution. Second, they were able to break complex problems into smaller simpler problems, and they were able to understand the relationship between the parts. Third, the students worked on problems in a more logical manner. They had a better understanding of the cause-effect relationships. Lastly, they accepted the fact that debugging was an inevitable part of programming (McCoy, 49).

Besides focusing on the process instead of the language the teacher can also enhance the acquisition of problem solving skills through the proper use of teacher intervention. According to an article in the November 1987 issue of The Computing Teacher there are three elements a teacher can develop to properly intervene in student learning. The first is activities in which the students

learn or manipulate basic commands. The students are to develop plans, predict outcomes, and then test the plan. If their predictions are wrong the students are to explain where they went wrong and correct their plan (Au, 13). The students are given a problem solving strategy and are encouraged to use it. This general plan would encourage analysis of the problem, planning, prediction, experimentation, monitoring the results, and evaluation of the plan and results. The students would also be encouraged to break down problems into smaller problems and to use structure or flow charts (Au, 13).

The second element of teaching intervention would be the questioning techniques used by the teacher. The teacher must respond to students' questions not by giving answers, but by asking questions that encourage students to reflect upon their own thinking in developing and carrying out their plan (Au, 14).

The final element is that the teacher must encourage a classroom atmosphere that is characterized by student interaction. The students should be encouraged to discuss how they solved problems, and to share with each other the different possible solutions. Most importantly, the students must be encouraged to discuss **how** they derived their solution (Au, 14).

The fact that the language needs to be secondary to the process of programming in order to encourage the development

of thinking skil:s may also indicate that the student must have enough experience in programming in a specific language that the use of the language is not an obstacle ln solving the problem. In this light we may look at programming a computEr as a skill that goes through three distinct phases of learning. The first phase wOAld be ttlat of mastering the language. If a child car read a program that was written by someone else and have a good understanding of the purpose, and procedure used then the student more than likely has a wo~king knowledge of the language. The second phase is the acquisition of design skills. This phase requires that the stUdent develops a repetoir of basic code patterns. These are procedures that a commonly required in programs such as sorting, searching and mathematical computations (Linn, **15)**. Tle students then combine the language skills tley have with the templates they have developed to design new programs. This would also include an understanding of how to test a program and how to adjust a program that is not producing the appropriate results (Linn, 16). The last stage that programmers go through is the stage in which they acquire problem-solving skills. The skills acquired would include "techniques for planning, testing, and reformulating that can be applied to several formal systems." Some other formal systems may include algebra problems, geometry proofs, or other areas of scientific lnqulry (Linn,

In a study done by University of California, Berkeley it was found that after twelve weeks of programming In;truction the students were still at the level of tlying to master the language (Linn, 25). Short term classes such as this are very popular at junior high schools. So, unless the student elects to pursue additional instruction in programming he or she may never have the opportunity to benefit from the instruction in regards to attaining improved problem solving skills. Students who progress to th8 desiqn stage do acquire some problem solving skills, but it is in the very narrow domain of programming a computer (Linn, 26). Generalization of these skills is limited to applying problem solving skills they learned to another computer language (Linn, 28). According to a study done in 1984 by Kurland, Mawbry, and Cahir it takes approximately 500 hundred hours of computer on-line time to become an Gxpert at prcgramming in a specific language (Linn, 29). During a twelve week class most student will have a maximum of twelve to eighty-four hours of on-line computer time available to them (Linn, 27). The average school would be closer to the twelve hour end of the spectrum. But again, this study did not indicate if ~here was a focus on programming as a process or if ~ne initial focus wa~ on the language itself.

If schools are going to turn towards the more process based approach to teaching programming, it is important that the students learn to use structured programming. A

structured program is easier to read, easier to decode, and helps to prevent students from developing bad programming habits (Lockard, 110). Structured programming has built into it some of the problem solving skills that would be used in teaching programming from a process orientation. Structured programs have each task broken down into the simplest task possible. Each task is performed in its own subroutine. One main routine controls all of the individual subroutines~ and each routine has only one entry point and one exit point (Lockard, 107-9). Finally the use of structured program eliminates the use of GOTO commands that can turn a program into complicated spaghetti code. All programs can be written using only the three basic constructs of sequence, selection, and iteration (Lockard, 109-10). By using structured programming the students can write long, involved programs with minimal complications.

The curriculum of computer science in the junior high classyooms vary not only from district to district, but from building to building within a district. Not all schools teach programming as part of their computer curriculum, but most do. The languages of choice are most often LOGO and BASIC. LOGO is generally chosen because of its expandability from a level suitable for elementary students to a level appropriate for high school students. BASIC is generally the second choice because results can be seen from a minimal amount of lines of code, and because of the inexpense of it

as a language choice. BASIC is a language that can be easily used on any personal computer. It requires very little memory space and often comes free with the purchase of a personal computer. Other languages such as PASCAL, FORTRAN, and COBOL require more extensive memory space and are moye expensive to purchase. In addition, it is often difficult to find teachers that have enough knowledge of any of these languages to enable him or her to teach it properly.

There is generally two structural approaches to programming instruction. One method is to teach it as part of the mathematics course. If this approach is taken the teacher may set aside one day a week that is designated for computer instruction. Or, the teacher may elect to have a curriculum that the requires minimal teacher instruction. In this manner the students may work independently on the computer at specific assigned times. If computer science is part of the math curriculum students generally receive from a half to one hour of computer instruction per week. This may or may not be for the entire year. Computers would be used in other content areas for simulations and CAI programs. In general, I found that most schools have only about sixteen weeks of computer instruction at anyone grade level. The othey approach is to have computer science as a subject of its own. The course would be an elective or exploratory course that is nine to eighteen weeks in length. The time in these courses is often divided between programming graphics,

word processing and simulations. The use of the computers in the schools may also be limited due to how the funds for the purchase of the computers was acquired. Many schools acquire additional computer hardware through state and national grants. These grants generally state specifically how the computers may be used for a specific number of years. If this is the case the schools must use the computers as the grant specified. The dictated use varies from grant to grant. While some grants focus on reading development, others focus on math skills or other content areas.

The instructional objective of each school's computer science curriculum is often a combination of ideas. First and foremost, all schools want the children to be computer literate. Definitions of what computer literacy entails vary considerably, but the following four components seem to be standard:

1. Familiarity with computer systems and the proper use of computers

2. Proficiency in running basic software systems (word processing, data bases, telecommunications, and graphics) and the ability to learn other systems

3. An appreciation of how computers can be used in society with an understanding of the dangers of a computerized society (Schwartz, 161).

4. An understanding of how computers are programmed, but not necessarily proficiency in a computer language (Ohler, 32).

A teacher can meet these objectives without ever teaching programming. Therefore, it can be deduced that most schools

do not teach programming ta hf21 P in making the children computer literate. The teaching of programming most definitely makes the students appreciative of the complexity of software progl"amS they use~ but this agaln is not a reason for teaching programming. So w~y do schools teach pyogranmlng.~7 Most schools teach programming foy two reasons. One is that they believe that the students should know how to program a computer in at least one language. The reason for this is generally two-fold- vocational reasons and so that they have a better understanding of how a computer works • The other reason they teach programming is to enhance the students critical thinking skills and problem solving skills. Whether or not they are attaining this goal most schools do not know for sure. One teacher stated that she has noticed a change in the way students attack math story problems. Since ~Le'/ have received programming instruction they tend to be mor0 willing to attack unfamiliar problems, they work at it longer, and they think of more possible plans for solving the p:oblem (Ya;usian). This can be interpreted as an indication that the students are learning some valuable skills.

Whether or not students are learning problem selving skills through computer programming is still a questio n that needs to be answered. As [illegible] we need to begin to look more closely at how we are teaching this ~till very new subject area. We may find in the future that it is beneficial to the students' thinking skills to teach

programming if we t%=-h it in the most appropriate manner.

Then again ... we rna; +ind out that the ~yoblem sG~ving skil_S they 2cqulye aYE le~e~ tYansfer rEd out w+, the context in which they are learned.

BIBLIOGRAPHY

Abrams, Peter D., et al. (1988). The Effect of Learning to Program in LOGO on Reasoning Skills of Junior High School Students. [illegible] Research, 4 (2), 203-213.

Au, Wink Kee, Jane Hartson, & Ken Ryba. ([illegible]). Logo, [illegible]

Beatly, Lamon~ F., & Robert V. Bullough, Sr. (1987). [illegible] Columbus, [illegible]

Carter, [illegible] Englewood 21iffs, NJ: Prentice Hall.

[illegible] Public Schools, Computer Curriculum Department (1990 [illegible]) Telephone Communication.

Crook, Charles, & Julie Rutkowska [illegible]

Hampton, Pat. (1990 April 20). Telephone Communication.

Linn, Marcia C. (1985). The Cognitive Consequences of Programming Instruction in Classrooms. Educational Researcher, 14 (5), 14-29.

Lockard, James. (1985-86). Computer Programming in the Schools: What Should be Taught? Computers in the Schools, 2 (4), [illegible]

McCoy, Leah P. (1989-90). Computer Programming Can Develop Qyoblem Solving Skills .. The Computing Teacher, 17 (4), 46-49.

Nicerson, Raymond S. (1982). Computer Programming as a Vehicle for Teaching Thinking Skills. Thinking, The [illegible] (3 & 4), 42-48.

Normal Community School District. (1990 May 3). Telephone Communication.

Ohler, Jason. (1987). The [illegible] Myths of Programming. The Computing Teacher, 14 (3), [illegible]

Parker, Catherine. (1989 May 2). Telephone Communication ..

Schwartz, Joel B. (1989). Implementing Computer Literacy in

Our Schools: What Have Weeay~eD- '? Computers in the Schools, 6 (1), 159-166.

Wiburg, Karin M. (1989). Does Programming Deserve a Place in t~e Schoo_l Curriculum? The Computing Teacher, 17 (2), 9-11.

Yazusian. (1990 April 17). Personal C;OfTurnU ni c ::t.:L c: n.