DOES INSTRUCTION IN COMPUTER PROGRAMMING IMPROVE PROBLEM SOLVING ABILITY?

by Craig A. VanLengen, EdD
Assistant Professor
Computer Information Systems and Accounting
Box 15066 - College of Business Administration
Northern Arizona University
Flagstaff, AZ 86011-5066
Telephone (602) 523-7392

and Cleborne D. Maddux, PhD
Chairman
Department of Curriculum and Instruction
College of Education
University of Nevada-Reno
Reno, NV 89557-0029
Telephone (702) 784-4961

ABSTRACT: Many schools and colleges of business teach computer programming in the introduction to computers course. The rationale for teaching computer programming is that it aids in the development of critical thinking, problem solving, and decision making skills. This contention is not supported by empirical data. An experimental study was conducted to ascertain if instruction in computer programming improved problem solving ability. The results of the study did not show support of improved problem solving ability from instruction in computer programming. Recommendations for changes in curriculum and teaching strategies are made as possible ways to make instruction in computer programming effective as a means of improving problem solving ability.

KEYWORDS: Computer Programming, Problem Solving, Critical Thinking, Introduction to CIS, Computer Literacy, Teaching Methods, Curricula Development: Lower Division.

INTRODUCTION

Many schools and colleges of business include computer programming in introductory computing courses. One reason given for including such instruction is that programming helps students to develop critical thinking, problem solving, and decision making abilities. Many educators support this belief [1, 2, 3, 4]. In addition Papert [5] maintains that working with computers can hasten the development of formal thinking. These controversial views are based on little empirical data, making the issue seem dogmatic [6, 7].

EXAMINATION OF RESEARCH

Studies of computer programming and problem solving have yielded mixed results. Studies that failed to find a relationship between computer programming and problem solving had various weaknesses in experimental design and instructional approach. Specifically, many studies employed small samples [8, 12], did not use random selection and/or assignment of the subjects [8, 9, 10, 11], or lacked control groups [9].

In addition, the main instructional approach in the non-support studies was non-directive (discovery) [8, 9]. Program planning, development, and debugging were not specifically taught [8, 9]. This is a problem, since using a non-directive instructional approach with a limited amount of treatment time does not appear to be effective.

Another difficulty is that mastery of programming was not measured [8, 12, 10, 11]. Without ensuring that programming is mastered, it makes little sense to talk of problem solving transfer [13].

A number of other studies showed some positive relationship between computer programming and problem solving ability. Some of these studies used random selection and/or assignment [14, 15, 16, 12, 17, 18]. Random selection should result in sample groups that are more closely related to the population. Several studies were for longer periods of time (semester or more) [19, 17, 20]. Longer studies should allow for more treatment time. The instructional strategy used in a number of these studies was directed with specific instruction in program planning and development [15, 16, 12, 17, 18].

The results of some of the studies are not conclusive since the programming activities and dependent variables appeared to be highly related [16, 18]. Even though the inferences were not clear-cut, these studies are a beginning of an experimental process directed at investigating a possible link between computer programming instruction and general problem solving ability.

Recommendations included in this study

Based on the review of related literature this study was designed to use strategies from the studies with positive results and to avoid the criticisms of the nonsupport studies.

1. Specifically, the study was conducted over an entire semester to allow for as much treatment time as possible [21, 22, 23].
2. A directed instructional strategy was used [24] with program planning and development explicitly taught for the computer programming group within a knowledge context [24] and linked to other contexts [23]. Many researchers have indicated the importance of making the subjects aware of their cognitive processing during the problem solving process [25, 26, 27, 28]. It is also considered important to teach strategies used by computer experts including templates or analogies [25, 29, 26, 30]. The strategies and principles used in a computer context must then be generalized and linked to other non-computer problem solving contexts [25, 30, 26, 28].
3. At the end of the study the students were given a problem statement similar to those used in the study. The computer programming group used a program logic chart to complete a BASIC program. The software package group specified the commands, formulas, and functions needed to complete a spreadsheet. Only those subjects who achieved mastery of their tool were compared for differences in general problem solving ability [13]. Mastery was defined as a score of seventy-five percent correct.

Research Methodology

The research hypothesis was that instruction in computer programming would not result in significant increases in general problem solving ability for college-level students enrolled in a course on introduction to computers. To test this hypothesis a random sample of sixty-four subjects were chosen from a large daytime lecture section of introduction to computers in the College of Business Administration at Northern Arizona University. The majority of the subjects had no computer experience. The subjects with computer experience had a semester or less of experience from secondary school. Summary data on the subjects is presented in the appendix. The subjects were then randomly assigned to an experimental (computer programming instruction) and a control (no computer programming instruction) group. All subjects were administered the *Watson-Glaser Critical Thinking Appraisal* as a pretest measure of problem solving ability.

For two-thirds of the semester both groups received the same instruction over computer concepts and terminology, hardware, software, and information processing systems. The remaining one-third of the semester was devoted to problem solving activities. The same problem exercises were used for both groups. Both groups received approximately fifty hours of problem development and computer lab work over an entire semester. Class attendance was required to obtain the specifications for the problem exercises.

The experimental group was taught a heuristic strategy for the development of a program from a problem statement. A structured instructional strategy was used to teach a heuristic that breaks down the problem into smaller and smaller problems until they are singular in function. Alternative solutions were formulated, evaluated, and implemented for these single function problem statements. The solutions to the single function problem statements were used on subsequent problems. Program debugging was demonstrated for the subjects to use in program development.

The control group solved the same set of problems using electronic spreadsheet and database management software. The emphasis for the control group was on the functions and commands that were needed to solve the problems, not problem solving strategies.

A section of the final examination for the course was used to measure mastery of the tool used by each subject. Mastery was defined as seventy-five percent correct on this section of the final examination. Subjects were not allowed to use reference materials during the examination.

The experimental group was given an incomplete computer program. Program specifications and a program logic chart were included. The program code was missing statements similar to the templates solved during the semester.

The control group solved the same problem as the experimental group. The subjects were given a narrative description of the problem and an incomplete spreadsheet. The subjects used formulas, commands and functions covered during the semester to solve the problem.

All subjects were administered the *Watson-Glaser Critical Thinking Appraisal* as a posttest measure of problem solving ability. Only those subjects who obtained mastery on their tool were used to test for differences in problem solving ability. Statistical significance was tested with a one-way analysis of covariance. The pretest scores were used as the covariate.

The data was tested and met the assumptions of linearity, homogeneity of variance, homogeneity of regression, and reliability of covariate. Thirty-three of the original sixty-four subjects obtained mastery. Sixteen were in the experimental group and seventeen were in the control group. As shown in Table 1, Analysis of Covariance for Posttest Scores, no statistically significant effect was found for the treatment group.

DISCUSSION OF THE STUDY

Results

The results failed to reject the hypothesis that computer programming instruction would not improve problem solving ability. In addition, no significant differences were noted for increases in problem solving for either the experimental or the control group. The level of significance was virtually the same with or without the seventy-five percent mastery cutoff. Possible reasons for no significant differences in problem solving ability for the experimental group include: (1) short treatment time, (2) no teacher-student interaction in a lab setting, (3) student test apathy, (4) non-use of problem solving strategies, and (5) limited sensitivity and specificity of the test instrument.

Fifty hours of time was available, during the semester, for in-class treatment, assigned readings, and projects. This amount of time may not offer the potential for improvement of problem solving ability with computer programming instruction.

The course does not have a required lab section that would allow for teacher-student interaction while problem solving was taking place. Perhaps guiding the student in the use of problem solving strategies would assist in the development of cognitive processes.

Students appear to be only interested in activities that are part of the course grade. Since pretest and posttest scores were not part of the course grade students may have given less than optimal effort. During testing situations students cannot be forced to use new problem solving techniques and may revert to prior strategies due to the stress of a testing situation.

A last factor may be a lack of sensitivity and specificity of the pre- and posttest instrument. Problem solving strategies taught during the experiment match up with three out of the five areas tested by the *Watson-Glaser Critical Thinking Appraisal*. Evidently the specific strategies taught in the programming context and linked to other problems in the introduction to computers course did not transfer sufficiently to influence total scores on the test instrument. An extremely strong treatment for more than one semester may be necessary to cause a treatment effect [31]. Perhaps an instrument that measures problem solving gains within a specific subject domain with a given knowledge base is needed to show significant results.

TABLE 1: Analysis of Covariance for Posttest Scores

Source of Variance Adjusted	SS	df	MS	F	p
Treatment	20.44	1	20.44	0.89	0.35
Covariate	887.54	1	887.54	38.74	0.00
Error	687.31	30	22.91		
Total	1,576.00	32	49.25		

EDUCATIONAL RELEVANCE

If a link between instruction in computer programming and improved problem solving ability is not proven, then continued instruction in computer programming will have to be justified on some other basis. Other reasons, based on personal beliefs and anecdotal evidence, are that instruction in computer programming: is necessary for computer literacy, allows for a better understanding of computer processing, provides an appreciation for commercial software development, increases social interaction between teachers and students and between students, increases self-confidence from successful programming efforts, provides freedom from repetitive calculations, and provides the ability to simulate complex and/or dangerous situations in experiments and decision making. If instruction in computer programming is not justified an alternative is to replace it with instruction in electronic spreadsheet and database management.

IMPLICATIONS FOR FURTHER RESEARCH

Even though this study failed to support the contention that instruction in computer programming improves general problem solving ability it did point out factors that need to be considered in future studies. Instruction in computer programming may be effective under different instructional conditions and a greater amount of treatment time.

Schools and colleges of business, accreditation agencies, the Data Processing Management Association and the Association for Computing Machinery need to examine the purpose of the introduction to computers course. Current methods of instruction and content may not support improvement of problem solving, critical thinking, and decision making abilities. To accomplish more than an introductory level of computer knowledge and vocational skills a change may be required in content and instructional methods. Perhaps more than one-third of the course needs to be spent on computer problem solving efforts for computer programming to be successful as a cognitive amplifier.

Current instructional strategies of lecture and demonstration, used in this study, might be expanded to include guided instruction in the use of problem solving strategies. The guided instruction would require a teaching lab facility which would allow for teacher-student interaction during the problem solving process.

The guided instruction in a teaching lab environment would allow the teacher to use a questioning dialogue to assist the students in the application of the problem solving strategies. The teaching lab environment would also provide for better monitoring of program planning, documentation, and the actual use by students of the problem solving strategies.

CONCLUSION

The results of this study failed to reject the hypothesis that instruction in computer programming would not improve the general problem solving ability of college-level introductory computer students. Current research efforts in this area have

indicated some potential for using computer programming as a means of improving problem solving ability [28]. Future studies need to increase the strength of the treatment, increase time spent on computer problem solving efforts, use guided discovery in a lab setting to increase teacher-student interaction, and select or develop test instruments that better measure problem solving in a computer context. If future efforts do not provide a link between computer programming instruction and improved problem solving ability it needs to be justified for other reasons or be removed from the introduction to computers course content.

REFERENCES

1. Davis, R. E., & Allen, J. R. (1984). Quality and computer education. Proceedings of the Western Educational Computing Conference. California Educational Computing Consortium, 94-97.

2. Wiechers, G. (1974). Programming as an educational tool. International Journal of Mathematical Education in Science and Technology, 5(4), 699-703.

3. Smart, J. R. (1984). A BASIC programming course to satisfy a general education requirement in quantitative reasoning. Proceedings of the Western Educational Computing Conference. California Educational Computing Consortium, 142-145.

4. Widmer, C. C. & Parker, J. (1985). A study of characteristics of student programmers. Educational Technology, 25(10), 47-50.

5. Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. New York: Basic Books, Inc.

6. Maddux, C. D. (1986). Computer programming in special education: The promise of Logo. Computers in the Schools, 3(3/4), 159-171.

7. Turner, J. A. (1987). Familiarity with new technology breeds changes in computer-literacy courses. The Chronicle of Higher Education, 33(45), 9,12.

8. Pea, R. D., & Kurland, D. M. (1984). Logo programming and the development of planning skills. (Technical Report No. 16.) New York: Bank Street College of Education, Center for Children and Technology. (ERIC Document Reproduction Service No. ED 249 930)

9. Kurland, D. M., & Pea, R. D. (1983). Children's mental models of recursive Logo programs. (Technical Report No. 10.) New York: Bank Street College of Education, Center for Children and Technology. (ERIC Document Reproduction Service No. ED 249 929)

10. Jansson, L. C., Williams, H. D., & Collens, R. J. (1987). Computer programming and logical reasoning. School Science and Mathematics, 87(5), 371-379.

11. Battista, M. T. (1987). The effectiveness of using Logo to teach geometry to preservice elementary teachers. School Science and Mathematics, 87(4), 286-296.

12. Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. Journal of Educational Psychology, 76(6), 1051-1058.

13. Kinzer, C., Littlefield, J., Delclos, V. R., & Bransford, J. D. (1985). Different Logo learning environments and mastery: Relationships between engagement and learning. In C. D. Maddux (Ed.), Logo in the Schools (pp. 33-43). New York: The Haworth Press.

14. Clements, D. H., & Nastasi, B. K. (1985). Effects of computer environments on social-emotional development: Logo and computer-assisted instruction. In C. D. Maddux (Ed.), Logo in the Schools (pp. 11-31). New York: The Haworth Press.

15. Mayer, R. E., & Fay, A. L. (1987). A chain of cognitive changes with learning to program in Logo. Journal of Educational Psychology, 79(3), 269-279.

16. Black, J. B., Swan, K., & Schwartz, D. L. (1988). Developing thinking skills with computers. Teachers College Record, 89(3), 384-407.

17. Clements, D. H. (1987). Longitudinal study of the effects of Logo programming on cognitive abilities and achievement. Journal of Educational Computing Research, 3(1), 73-94.

18. Battista, M. T., & Clements, D. H. (1986). The effects of Logo and CAI problem-solving environments on problem-solving abilities and mathematics achievement. Computers in Human Behavior, 2(3), 183-193.

19. Kurshan, B., & Williams, J. (1985). The effect of the computer on problem solving skills. (ERIC Document Reproduction Service No. ED 259 714)

20. Linn, M. C., Sloane, K. D., & Clancy, M. J. (1987). Ideal and actual outcomes from precollege Pascal instruction. Journal of Research in Science Teaching, 24(5), 467-490.

21. Leron, U. (1985). Logo today: Vision and reality. The Computing Teacher, 12(5), 26-32.

22. Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. New Ideas in Psychology, 2(2), 137-168.

23. Horner, C. M., & Maddux, C. D. (1985). The effect of Logo on attributions toward success. In C. D. Maddux (Ed.), Logo in the Schools (pp. 45-54). New York: The Haworth Press.

24. Gallini, J. K. (1985). Instructional conditions for computer-based problem-solving environments. Educational Technology, 25(2), 7-11.

25. Kantowski, M. G. (1983). The microcomputer and problem solving. Arithmetic Teacher, 30(6), 20-21, 58-59.

26. Soloway, E. (1986). Learning to program = learning to construct mechanisms and explanations. Communications of the ACM, 29(9), 850-858.

27. Shneiderman, B. (1985). When children learn programming: Antecedents, concepts and outcomes. The Computing Teacher, 12(5), 14-17.

28. Salomon, G., & Perkins, D. N. (1987). Transfer of cognitive skills from programming: When and how? Journal of Educational Computing Research, 3(2), 149-169.

29. Dalbey, J., Tourniaire, F., & Linn, M. C. (1986). Making programming instruction cognitively demanding: An intervention study. Journal of Research in Science Teaching, 23(5), 427-436.

30. Pirolli, P. L., & Anderson, J. R. (1984). The role of mental models in learning to program. Arlington, VA: Office of Naval Research, Personnel and Training Research Programs Office. (ERIC Document Reproduction Service No. ED 265 177)

31. McMillan, J. H. (1987). Enhancing college students'critical thinking: A review of studies. Research in Higher Education, 26(1), 3-29.

AUTHORS' BIOGRAPHIES

Craig A. VanLengen is an assistant professor of computer information systems/ accounting in the College of Business Administration at Northern Arizona University. He holds a BS in Business with a major in accounting, an MBA with a major in information sciences, and an Ed.D. in instructional computing. He has six years of industry experience in the areas of accounting and systems analysis. A Certified Public Accountant licensed to practice in Colorado and a Certified Systems Professional, he has been teaching at the college level for eight years. His teaching responsibilities include systems analysis and design, management information systems, introductory programming, and introduction to computer information systems.

Cleborne D. Maddux is a professor and department chairman in the College of Education, University of Nevada-Reno. He is the author of ten books and numerous journal articles on the topics of computer education and special education. He is series editor for an upcoming series of books on educational computing. Within computer education, he has a special interest in the Logo programming language.

APPENDIX : SUMMARY DATA ON SUBJECTS

The initial sample of sixty-four subjects is represented by the letter (I) and the subjects achieving mastery are represented by the letter (M).

TABLE 2: Gender by Group

Group	Male		Female		Total	
I=Initial; M=Mastery	I	M	I	M	I	M
Computer Programming Instruction	19	9	16	7	35	16
No Computer Programming Instruction	13	7	16	10	29	17
Total	32	16	32	17	64	33

TABLE 3: Class Level by Group

Group	Freshman		Sophomore		Junior		Senior		Graduate		Total	
I=Initial; M=Mastery	I	M	I	M	I	M	I	M	I	M	I	M
Computer Programming Instruction	18	6	6	3	6	3	4	3	1	1	35	16
No Computer Programming Instruction	15	10	7	2	1	0	5	4	1	1	29	17
Total	33	16	13	5	7	3	9	7	2	2	64	33

TABLE 4: Major by Group

Group	Business		Nonbusiness		Undeclared		Total	
I=Initial; M=Mastery	I	M	I	M	I	M	I	M
Computer Programming Instruction	8	4	15	7	12	5	35	16
No Computer Programming Instruction	7	3	10	8	12	6	29	17
Total	15	7	25	15	24	11	64	33

TABLE 5: Means and Standard Deviations for Subjects

	Pretest		Posttest	
Group	Mean	s	Mean	s
(N=16) Computer Programming Instruction	56.63	9.93	56.19	7.34
(N=17) No Computer Programming Instruction	53.29	8.16	55.82	6.92
(N=33) Total	54.91	9.08	56.00	7.02

ISSN 1055-3096