COMBINING PROBLEM-SOLVING INSTRUCTION AND PROGRAMMING INSTRUCTION TO INCREASE THE PROBLEM-SOLVING ABILITY OF HIGH SCHOOL STUDENTS

Abstract

Researchers have identified a number of reasons why many programming students have difficulty acquiring a knowledge of programming, writing programs, and transferring programming knowledge to other problem-solving domains. These reasons include a lack of instruction in basic problem-solving skills and poor programming skills. In the study described here, students were introduced to a generalized problem-solving strategy, and were given instruction in BASIC programming in an attempt to determine whether this combination enhanced student problem-solving ability. (Keywords: problem solving, programming.)

Problem solving and reasoning are critical skills needed in all disciplines and subject areas. If students lack adequate problem-solving skills, they will have a difficult time making inferences, discovering relationships, and competing for jobs. Research indicates that skills in problem solving and reasoning can be enhanced by practice using real problems, organizational skills, pattern recognition, decomposition, planning, experimentation, direct instruction in problem-solving strategies, and domain specific knowledge (Bransford, Sherwood, Vye, & Rieser, 1986; Dudley-Marlin & Owston, 1988; McCoy & Orey, 1988; Pea & Kurland, 1984; Salomon & Perkins, 1987; Vockell & VanDeusen, 1989).

BACKGROUND

Several researchers (Bransford, Sherwood, Vye, & Rieser, 1986; Linn & Dalbey, 1985; Pea & Kurland, 1984; Perkins, Schwart, & Simmons, 1987; Salomon & Perkins, 1989) have identified a number of reasons why many programming students have difficulty acquiring knowledge of programming, writing programs, and transferring programming knowledge to other problem-solving domains. These reasons include inert knowledge, poor programming skills (decomposition, flow control, debugging, planning, and design), the view that programming is an abstract entity unrelated to other subject areas, gender differences, and lack of instruction in basic problem-solving skills. The present study focused on two of these reasons: inadequate problem-solving and programming instruction.

Research on Logo Programming

Many individuals have claimed that computer programming teaches problem solving and reasoning and helps students learn about their thought process. This claim has not been proven, nor is it new. Mathematics instruction (geometry and trigonometry), the Latin School Movement of the 1600s, logic instruction, and writing systems were introduced into classes, also with the claim that they would help students learn problem-solving and reasoning skills.
Papert (1980) claimed that the child-centered environment of Logo allowed children to build their own intellectual models, apply these models to other situations, and think about the entire process. Papert's claim for the value of Logo instruction has not been proven, but programming in Logo and other languages does provide students with a useful arena to practice some cognitive skills.

Pea (1983) found that Logo was "cognitively complex beyond its early steps, and quite difficult to learn without instructional guidance" (p. 2). After extensive studies of Logo with students between the ages of 8 and 12, Pea found that they learned the semantics and syntax of the language easily but did not learn how to structure and organize programming statements that would accomplish a specific goal. Pea suggested that problem-solving or thinking skills should be taught directly; it should not be assumed that these skills will occur spontaneously. In a subsequent study, Pea, Kurland, & Hawkins (1985) again found that Logo programming did not have a measurable influence on the planning abilities of students.

Lieberman (1985) insisted that the effectiveness of Logo instruction was directly related to the teacher's role and the instructional methods used. Other researchers have also stressed the need for teacher mediation and direct, explicit guidance for the successful acquisition of problem-solving, thinking, and transfer skills through the use of Logo (Clement, Kurland, Mawby, & Pea, 1986; Collis, 1988; Emihovich & Miller, 1986; Govier, 1988; Lehrer & Smith, 1986; Seidman, 1981).

Several authors have criticized the discovery learning method advocated by Papert as being too unfocused to promote the development of reasoning or thinking skills that will transfer to other areas. Many researchers have stressed the need for a structured curriculum that emphasizes the skills to be learned and that encourages students to deliberately look for connections with material outside the programming domain (Govier, 1988; Littlefield et al., 1988; Seidman, 1989).

Fay and Meyer (1988) studied 30 fourth-grade students to see whether cognitive changes occurred when these students learned the syntax and semantics of Logo programming. They concluded that "programming can serve as a vehicle for learning general thinking skills, but only if those skills are learned and practiced in the programming environment and their applicability to other domains is recognized" (p. 73).

Swan and Black (1988) found that a combination of general problem-solving strategies, direct instruction, and a mediated learning environment enabled students to develop transferable problem-solving skills from their Logo experience.

According to Mayer, Dyck, and Vilberg (1986), learning a programming language can enhance thinking or problem-solving skills, but these improvements seem to be directly related to the concepts emphasized within the language. They noted improvement in the problem translation and procedural comprehension of the college students participating in their study, but they found no evidence of changes in the students' general abilities (logical reasoning or visual and verbal ability).
Several studies involving Logo instruction have resulted in an improvement in the participants' language fluency, reflective thinking, ability to give directions, problem solving, and creativity. Mann (1986) found that the problem-solving abilities of students, as measured by the Cognitive Abilities Test (Thomdike & Hagen, 1986), were enhanced as a result of learning Logo programming skills. Clements and Gullo (1984) found that their Logo group improved in language fluency, divergent and reflective thinking, and ability to give directions.

**Other Research**

Other programming languages, such as BASIC and Pascal, have been studied in order to determine whether improved problem-solving and thinking skills, or the transfer of these abilities, resulted from instruction in these languages. Several studies have indicated that these languages improved problem-solving ability (Jones, 1988; Mayer et al, 1986; McCoy & Orey, 1988). However, other studies have indicated that studying these languages has no significant impact on student problem-solving ability (Palumbo, 1990; Shaw, 1984).

**METHOD**

The subjects in this study were 11th- and 12th-grade students at a Midwest-est high school. There were 20 students in the treatment group (10 males and 10 females), and 19 students in the control group (10 males and 9 females). These students were introduced to a generalized problem-solving strategy along with BASIC programming instruction. The goal of the project was to determine whether problem-solving instruction combined with programming instruction would improve the problem-solving abilities of these students.

Structured programming instruction was given to all students, acquainting them with the syntax and grammar of BASIC and with the logical constructs needed to write their programming assignments. Problem-solving instruction was administered to one group only, the treatment group. The problem-solving strategy was a generalized tool that allowed students to plan their programming solutions.

A quasi-experimental design was used because the participants were a part of intact groups. Subjects were not randomly assigned to either a treatment or control group; instead, each class was randomly assigned. All students were pretested with the Cognitive Abilities Test (CAT): Non-Verbal Battery to measure problem-solving ability. The test has reliability coefficients of .92 to .93 and a validity coefficient in the .70s.

After the pretest, all students in the treatment group received instruction that combined a general problem-solving strategy with BASIC's syntax and semantics. This treatment condition was designed by the author and administered by the teacher. It encouraged students to systematically approach problem solving by using the commands, structures, and logical constructs of BASIC. Students in the treatment group were required to write programming solutions using the steps of the problem-solving strategy before they entered their solutions into an IBM PS/2 computer. This treatment was administered daily
for three weeks. During this time period, the control group received programming instruction only. The instruction was example-oriented, and was not combined with a problem-solving technique. Students learned BASIC commands, functions, and syntax. After the three-week interval, both the treatment group and control group received a posttest measure (CAT).

**Controls**

Two controls were added to help generalize the results and eliminate confounding variables. To control for individual differences, students had the same general ability (as measured by GPA), and classes of students were placed in either the treatment or control group by random assignment. To control for differences in programming skill and ability, all students participating had little or no prior programming experience. Differences in teacher motivation and skill were controlled by having the same teacher administer the treatment to all students in the treatment and control groups.

**RESULTS**

Did problem-solving instruction combined with programming instruction improve the problem-solving ability of students (as measured by the CAT: NonVerbal Battery)? No statistically significant mean difference was found in the problem-solving ability of the treatment and control groups as compiled in an ANCOVA, which analyzed the main effects (group and sex) and the 2-way interaction of group x sex (see Tables 1, 2, and 3).

In the treatment group, 15 students showed improvement in their CAT posttest scores, while 3 students showed a decrease in their scores and one student showed no change. Of the 15 students with improved scores, 5 raised their pretest scores by 12-18 points, while 10 increased their scores by 3-10 points. Of the students in the control group, 10 showed improvement in their scores, while 8 showed a decrease in their scores and one student showed no change. Of the 10 students who improved their scores, one student gained 14 points and the other 9 gained 2-8 points.

**Qualitative Observations**

For this study, the author and the teacher functioned as observers and gathered qualitative data in an attempt to answer the following questions: Were there commonalities or differences in the approaches to problem solving between the treatment group and the control group? What types of programming solutions did students use? What types of student interaction took place?

Both the treatment group and the control group were observed separately, and all students in both groups were separately interviewed using open-ended questions. The students in the treatment group were asked questions about the problem-solving strategy and its influence on their programming assignments. Most students said that the strategy was helpful in formulating programs. General comments indicated that the strategy made programs easier to write, saved time, and helped students visualize long and complex problem ideas. Several students did not like using the strategy. Their
complaints included the following: "It does not make sense," "It duplicates what I do in my head," and "It slows me down" (Field Notes, 12/11/91).

Students who used the strategy asked different types of questions than students who did not use it. According to the classroom teacher, the types of questions the treatment group asked involved BASIC syntax.

Students in the treatment group produced diverse and creative programming solutions to the problems they received, and they also interacted with each other a great deal. It was noted that "more of the students in the treatment group worked in pairs; they moved around the classroom frequently, and they asked each other questions" (Field Notes, 12/11/91).

Students in the control group produced programs that were very similar to each other. There was little creativity or diversity shown in their work; their programs were carbon copies of the examples they had received in their instruction. Also, there was very little interaction or collaboration among the control group students. It was observed that "these students were quiet; they did not interact with each other at all. They quietly pursued their programming goals" (Field Notes, 12/17/91).

**CONCLUSION**

Although no statistically significant differences in the problem-solving ability of the two groups were found, some interesting qualitative observations were noted. Students who received problem-solving instruction integrated into their programming instruction tended to ask questions involving syntax, while students who did not receive this type of instruction asked more questions involving logic. Students who received the combined instruction interacted with each other more often than their counterparts, and the programming solutions produced by these students were more creative and diverse.

Careful planning and analysis occurred among students in the treatment group because they had a game plan. Before they started to do any programming at a computer terminal, they had to fill out a form listing the steps they planned to take in accomplishing their programming goal. They also had to identify and define what was needed, determine what was given, and decide what programming commands they needed to use. This type of careful planning and analysis required a great deal of thought and attention to detail. Consequently, students spent more of their time thinking critically. They studied their problem carefully, analyzed it, and evaluated all of the information about the given problem. This allowed them to work through the logistics of the problem and ask fewer questions involving logic. This supports research findings that demonstrate explicit instruction in problem solving helps facilitate the learning of problem solving (Govier, 1988; Littlefield et al., 1988; Lehrer & Smith, 1986).

Students in the treatment group interacted with each other more than students in the control group. Treatment group students worked in pairs and moved around their classroom environment freely.
exploring, comparing, and devising plans. Many of the steps in the problem-solving strategy encouraged the problem solver to analyze the problem from many angles, using all available resources. These students used their classmates as resources, and explored others' solutions in an attempt to formulate their own.

The students' programming solutions were also analyzed by a team of three PhD candidates. They evaluated the programs for error content and programming structure. Errors were ranked according to their severity, and notes were made about the form and style of the programs. The team that analyzed the programs found that the programs of the treatment group were more creative than those of the control group. The programs of the students in the treatment group did not follow a standard pattern, and all these programs were not carbon copies. Some students' programs calculated information using computational variables, while other students chose to use the sum function. Variable names were different and uniquely used to represent objects. Syntax errors were found more frequently in these programs. Just the opposite was found in the programs written by the students in the control group: Their programs used virtually the same input and output messages, followed the same format, and used many of the same variable names (e.g., N, A, B, C). They were exact duplicates of the programming examples the control students had observed.

Given these results, this author favors the use of programming instruction combined with problem-solving instruction in high school programming environments because it serves as a catalyst to help students analyze and evaluate information and produce creative, diverse, and correct programming solutions. It encourages interaction among students, fosters team work, encourages more individual expression in programs written, allows students to analyze problems before they begin to code, and helps students make wise use of their resources (i.e., computer time, instructor's time).

**Contributor**

Demetria Ennis is an assistant professor in technology and cognition at the University of North Texas: Her teaching areas include artificial intelligence in education and the development of educational software. Her research areas involve gender equity in computer programming environments and practical applications of neural networks. (Address: 1200 Dallas Drive, No. 928, Denton, TX 76205.)

**Table 1 Analysis of Covariance of CAT Posttests by Group (Treatment or Control) and Sex (M/F)**

Legend for Chart:

A - Source of Variation
B - Sum of Squares
C - df
D - Mean Square
E - F
### Table 2 Pretest and Posttest CAT Means by Group (Treatment or Control) and Sex (M/F)

**Legend for Chart:**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td><strong>Group &amp; Total</strong></td>
<td><strong>B</strong></td>
<td><strong>Sex &amp; Total</strong></td>
<td><strong>C</strong></td>
<td><strong>Pretest Means</strong></td>
<td><strong>D</strong></td>
</tr>
<tr>
<td><strong>A</strong></td>
<td><strong>Number of Cases</strong></td>
<td><strong>B</strong></td>
<td><strong>Number of Cases</strong></td>
<td><strong>C</strong></td>
<td><strong>Number of Cases</strong></td>
<td><strong>D</strong></td>
</tr>
</tbody>
</table>

<p>| <strong>Treatment</strong> | | | | | | | | | | | | | |
| <strong>Female</strong> | 36.40 | 15.09 | 42.90 | 15.53 | 10 |</p>
<table>
<thead>
<tr>
<th></th>
<th>Male</th>
<th>Control</th>
<th>Female</th>
<th>39.55</th>
<th>8.17</th>
<th>41.44</th>
<th>12.27</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>39.70</td>
<td></td>
<td></td>
<td>39.60</td>
<td>13.85</td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>37.89</td>
<td>12.09</td>
<td>42.21</td>
<td>13.72</td>
<td>19</td>
</tr>
<tr>
<td>Male</td>
<td>39.35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>41.85</td>
<td>11.20</td>
<td>20</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>37.70</td>
<td>12.64</td>
<td>43.50</td>
<td>12.00</td>
<td>20</td>
</tr>
</tbody>
</table>

**Table 3 Adjusted Means for CAT Posttest Scores by Group (Treatment or Control) and Sex (M/F)**

Legend for Chart:

A - Group/Total
B - Sex/Group
C - Adjusted Mean
D - Standard Error
E - Number of Cases

<table>
<thead>
<tr>
<th></th>
<th>Treatment</th>
<th>Female</th>
<th>44.93</th>
<th>2.35</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Male</td>
<td>43.77</td>
<td>2.34</td>
<td>10</td>
</tr>
</tbody>
</table>

|       | Control   | Female | 40.61 | 2.47 | 9  |


|       | Control | Female | 40.61 | 2.47 | 9  |


Male                    38.63           2.34             10
Total
Female                  42.77           1.70             19
Male                    41.20           1.66             20
Total
Treatment               44.35           1.66             20
Control                 39.62           1.70             19

References


Collis, B. (1988, June). Research in the application of computers in education: Trends and issues over four years of research windows. Presentation at the National Educational Computing Conference, Dallas, TX.


Mann, R. J. (1986). The effects of Logo computer programming on


