The effects of learning a programming language on logical thinking skills

Margaret Gradwell Mains

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
Mains, Margaret Gradwell, "The effects of learning a programming language on logical thinking skills" (1997). UNLV Retrospective Theses & Dissertations. 3336.
https://digitalscholarship.unlv.edu/rtds/3336
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
THE EFFECTS OF LEARNING A PROGRAMMING LANGUAGE ON LOGICAL THINKING SKILLS

by

Margaret G. Mains

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science
in
Educational Psychology

Department of Educational Psychology
University of Nevada, Las Vegas
August 1997
This Thesis of Margaret G. Mains for the degree of Master of Science in Educational Psychology is approved:

Chairperson, Dr. Eunsook Hong, Ph.D.

Kevin Crehan
Examining Committee Member, Dr. Kevin Crehan, Ph.D.

Peggy G. Perkins
Examining Committee Member, Dr. Peggy G. Perkins, Ph.D.

Randall Boone
Graduate Faculty Representative, Dr. Randall A. Boone, Ph.D.

Dean of the Graduate College, Dr. Ronald W. Smith, Ph.D.

University of Nevada, Las Vegas
May 1997
ABSTRACT

The study examined the effects of learning a computer programming language on logical thinking skills, by comparing the logical reasoning skills of students who took a QBasic class with those who took a continuing education computer graphics/desktop publishing class.

The study took place during a 14-week period in the fall of 1996. The sample was selected from students enrolled in a QBasic programming language from a community college in Las Vegas, Nevada (n = 15). A desktop publishing/graphics class was the comparison group, selected from the same community college (n = 15). Pretest and posttest scores on the students' logical thinking skills were collected using Logical Reasoning (Hertzka & Guilford, 1955, 1993). Syllogistic statements were used because they are closely related to IF-THEN statements required in learning computer programming language structure.

An analysis of covariance indicated that there was no statistically significant difference between programming and graphic groups on posttest scores of logical thinking skills. That is, learning a programming language did not enhance logical reasoning skills, specifically the use of syllogistic
reasoning skills. However, observations of students' classroom behaviors indicate that some students increased the use of logical thinking or problem-solving skills as the programming instruction progressed.
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iv</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>vii</td>
</tr>
<tr>
<td><strong>CHAPTER 1</strong> INTRODUCTION.</td>
<td>1</td>
</tr>
<tr>
<td>Rationale</td>
<td>4</td>
</tr>
<tr>
<td>Research Hypothesis</td>
<td>6</td>
</tr>
<tr>
<td>Assumptions</td>
<td>7</td>
</tr>
<tr>
<td>Delimitations of the Study</td>
<td>7</td>
</tr>
<tr>
<td>Definitions of Terms</td>
<td>8</td>
</tr>
<tr>
<td>Organization of the Study</td>
<td>9</td>
</tr>
<tr>
<td><strong>CHAPTER 2</strong> LITERATURE REVIEW.</td>
<td>11</td>
</tr>
<tr>
<td>Cognitive Skills</td>
<td>13</td>
</tr>
<tr>
<td>Cognitive Effects of Learning Programming Languages</td>
<td>20</td>
</tr>
<tr>
<td><strong>CHAPTER 3</strong> METHOD.</td>
<td>27</td>
</tr>
<tr>
<td>Participants</td>
<td>27</td>
</tr>
<tr>
<td>Setting</td>
<td>29</td>
</tr>
<tr>
<td>Characteristics of the Programming Class</td>
<td>30</td>
</tr>
<tr>
<td>Materials</td>
<td>32</td>
</tr>
<tr>
<td>Procedure</td>
<td>33</td>
</tr>
<tr>
<td><strong>CHAPTER 4</strong> RESULTS, DISCUSSION, LIMITATIONS, AND RECOMMENDATIONS.</td>
<td>36</td>
</tr>
<tr>
<td>Findings</td>
<td>36</td>
</tr>
<tr>
<td>Observations</td>
<td>37</td>
</tr>
<tr>
<td>Discussion</td>
<td>43</td>
</tr>
<tr>
<td>Limitations and Recommendations</td>
<td>45</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>48</td>
</tr>
<tr>
<td><strong>APPENDIX I</strong> FORM LETTERS SENT TO INSTRUCTORS FOR SITE.</td>
<td>57</td>
</tr>
<tr>
<td><strong>APPENDIX II</strong> LETTER OF INTRODUCTION AND STUDENT CONSENT FORM.</td>
<td>60</td>
</tr>
<tr>
<td><strong>APPENDIX III</strong> LETTERS OF APPRECIATION SENT TO THE TWO PARTICIPATING INSTRUCTORS.</td>
<td>62</td>
</tr>
<tr>
<td><strong>APPENDIX IV</strong> SURVEY QUESTIONNAIRES</td>
<td>65</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

Many people have been influential in the completion of this research endeavor, but there is a group of people who deserve special recognition. The professors from the department of Education Psychology and Instructional and Curricular Studies in the College of Education at the University of Nevada, Las Vegas, should be acknowledged. It was through their instruction that I first thought about enhancing thinking processes through computer education.

My most humble thanks go to the instructors at the site who allowed me to conduct this research. They graciously gave their time and talents to further my education.

My most important source of support was Dr. Eunsook Hong. Every corner I intended to turn seemed to have a stumbling block--life's little barriers--but she stood steadfast, supporting my climb. She expected much, but that only made me work more diligently, and I have the utmost respect for her.

My family and friends deserve a portion of the success of this project. The lesson learned was that no project is an island.
CHAPTER 1

INTRODUCTION

The computer was created as a tool to process a variety of information at a faster pace than the human mind. Since then, almost every facet of life has been and is being touched by the computer in some way or another. Scientists, engineers, designers, and programmers still seek a friendlier, more intelligent, more powerful use for this tool that interacts with and affects our daily existence.

As few as ten years ago most students were not exposed to computers in classrooms. Even fewer students were exposed to learning a computer programming language as a tool to enhance thinking skills or solve problems. Today, many students use a computer in one form or another. In the formal educational environment, students are learning keyboarding skills, working on application programs that will help them build job skills, exploring the world wide web for information, and experiencing a variety of other uses for the computer. Students are discovering that computer languages, which make the computer perform various functions, can be powerful and exciting. Schools began to provide computer
programming classes which led educators and researchers to question the effects of learning a computer programming language on cognitive development of their students.

The effects of computer programming on cognitive development has been studied in various domains. The majority of literature first generated indicated that teaching Logo, as a computer programming language, would increase children’s logical thinking skills (Papert, 1980). However, some studies on Logo did not report the same results, and other studies reported mixed findings. Singh (1993), in a literature review of studies testing cognitive effects of programming in Logo, stated that the empirical evidence offered mixed results. Studies using different programming languages offered different results, and the results were related to the language and skill being tested. Pea and Kurland (1984) and Dalbey and Linn (1985) noted some positive effects of learning a programming language but indicated that considerable time, experience, and the right style of instruction were needed in order to produce cognitive changes.

Learning Basic programming language resulted in some cognitive benefits such as improvement in logical reasoning skills (Many, Lockard, Abrams, & Friker, 1988). However, like the results of the studies of Logo, the empirical evidence of the studies again yielded mixed results. Some of the mixed results were due to the language and skill being tested.
Tobin and Capie (1981) and Ahlawat and Billeh (1987) offered mixed theories on whether tests actually measured logical thinking. Some studies (Hillel, Kieran, & Gurtner, 1989) examined logical thinking as it related to a formation of a geometric shape in the Logo program. Others (Palumbo, 1990) tested logical thinking as it related to a mathematical formula because some programming language structures are closely related to the formation of algorithms. Still other studies (Jansson, Williams, & Collins, 1987; Seidman, 1981) tested for cognitive changes in logical thinking as it related to the structure and syntax of English.

A small number of studies investigated the effects of learning a computer programming language on the logical reasoning abilities of students in the area of syllogistic reasoning (Jansson, et. al., 1987; Seidman, 1981). Syllogistic reasoning is closely related to the IF-THEN statement. Syllogistic reasoning follows a given premise to its logical conclusion. Learning a programming language requires not only acquiring the skill for inputting the programming language structure, but following the problem to its logical conclusion.

This study tested whether learning QBASIC programming language would enhance logical thinking skills as measured by the syllogistic reasoning test. Participants were given a premise and responded by following it to its logical
conclusion. To answer the question correctly, they should use some form of IF-THEN reasoning.

Rationale

One of the most important goals in education is to teach students critical thinking skills. Passing along information is not enough. The mega-information storage capabilities and information retrieval possibilities will give us the very information we seek in milliseconds. In our ever changing technological age, students need the reasoning skills, logical thinking skills, and problem-solving skills, that will guide them in their lifes' choice.

Research on educational computing has yielded a variety of studies involving the relationship between problem-solving skills and learning a computer programming language (McCoy, 1990; Palumbo, 1990). Although there is some evidence on the cognitive benefits of learning a computer programming language (Papert, 1980; Dalbey & Linn, 1985), links between computer programming languages and such skills have not clearly been established. Studies have not provided enough empirical evidence to promote the benefits of learning a computer programming language across the curriculum (Maddux, 1985-1986; Burton & Magliaro, 1988).
The purpose of this study was to isolate one component of a precursor to problem-solving skills, logical thinking. It examined whether logical thinking skills would be enhanced as a result of learning a programming language.

Although the relationship of computer programming to problem-solving abilities has been investigated, few studies have been conducted in the area of logical thinking. Even fewer studies have specifically addressed syllogistic reasoning skill as an indicator of logical thinking skills. Jansson, Williams, and Collins (1987) and Seidman (1981) stated that syllogistic reasoning skills are part of the chain of steps necessary when using problem-solving skills and learning a programming language.

Studies on the effects of programming experiences on cognitive growth have shown conflicting results. Even when studies found an increase in thinking skills after exposure to a programming language, students did not transfer their learning to other content areas. More importantly, learners had great difficulties in "debugging" computer programs. This may indicate that learners have not used deductive thinking, problem-solving, or syllogistic reasoning skills in developing their understanding when learning a programming language.

The current study examined the logical reasoning skills of students using a syllogistic reasoning test prior to and after a computer programming experience. Computer
programming languages require learners to utilize syllogistic reasoning or a form of IF-THEN statement. Over time and under the right learning conditions, it can be expected that students' skills in this area would improve.

Research Hypothesis

The purpose of this study was to investigate the effects of learning a computer programming language on logical thinking, specifically syllogistic reasoning skills.

Syllogistic reasoning skill was measured by Logical Reasoning (Hertzka & Guilford, 1955, 1993). A higher test score indicated higher syllogistic reasoning skills. To determine whether learning a computer programming language would increase logical thinking skills, students from two courses were compared—Introduction to Programming, QBASIC, and a Graphics/Desktop Publishing course.

The research hypothesis was that there would be a significant difference in the syllogistic reasoning skill scores between students who learned a programming language (Introduction to Programming, QBASIC) and those who learned Graphics/Desktop publishing.

In addition, observations were made on the students in the programming group to note any changes in their cognitive behavior, in terms of the use of syllogistic language skills, logical thinking skills, or problem-solving skills.
Assumptions

1. An increase in logical thinking skills, as indicated by the test results of syllogistic reasoning tasks, will result from learning QBasic, a computer programming language, because the structure of a programming language follows a logically structured pattern.

2. Observations will indicate that students will demonstrate an increase in their use of conditional logic, IF-THEN logic, in their interaction in class as a result of learning QBASIC.

3. A small increase will be demonstrated between pretest and posttest scores on syllogistic reasoning skills as a result of intellectual maturation by students.

4. There will be a small increase between pretest and posttest scores on syllogistic reasoning skills due to familiarity with test items or test format.

5. The Graphics/Desktop publishing class would not require logical reasoning due to the nature of the computer applications package the class uses.

Delimitations and Limitations

1. The participants were delimited to the students who were enrolled in the Introduction to Programming, QBASIC course and the Computer Graphics/Desktop publishing course.
2. This study was limited by the number of students enrolled in the class. The study was further limited by the high drop-out rate consistent in the community college population.

3. Observations were limited by the participants' irregular attendance.

4. The strength of the study was further limited by the number of students who were available for the posttest.

5. The strength of the results was limited by the fact that random sampling was not possible for this study.

6. The study was limited by the individual differences in participants' experiences with the computer within and between the two groups in the study.

Definition of Terms

1. Logical Thinking—Reasoning in a clear and consistent manner based on earlier or otherwise known statements, events, or conditions (Mayer, 1983).

2. Deductive Reasoning—The sort of reasoning in which the conclusion is deduced following what is given (Thurstone, 1938).

3. Conditional Syllogism—A sentence reasoning involving conditions, which may or may not be introduced by the word "if" and follow with "then" (Mayer, 1983).
4. Syllogistic Reasoning Task--A task that requires using a form of deductive reasoning, consisting of a major premise, a minor premise, and a conclusion: For example, all human beings are mortal (the major premise), I am a human being (the minor premise), therefore, I am mortal (the conclusion) (Seidman, 1981).

5. QBASIC--A modern version of the Basic computer programming language. This language is used for teaching novice programmers (Trombetta, 1994).

6. Top-Down Approach--A method of developing an algorithm by breaking up a complex problem into a set of less complex problems. Any tasks that are still complicated are further divided into their own tasks. This continues until all the tasks are so simple that they can be easily coded (Trombetta, 1994).

Organization of the Study

Chapter 1 includes the introduction, rationale, research hypothesis, assumptions, delimitations and limitations, definition of terms, and organization of the study.

Chapter 2 contains the review of literature as it related to cognitive skills including higher order thinking skills, problem-solving skills, and logical thinking skills, and the cognitive effects of learning computer programming languages.
Chapter 3 describes the research method including participants, instrument, and procedure.

Chapter 4 presents the results of data analysis, observations, and discussion of the findings. It includes limitations and recommendations for future research.
Logical reasoning is a necessary precursor to problem solving (Linn & Dalbey, 1985; Johanson, 1987). Learning a computer programming language requires students to use conditional reasoning. It is one of the components of logical thinking, part of the language construction processes in formulating programming statements. Successful programmers use logical thinking as part of their problem-solving abilities when working with a computer programming language.

Conditional (i.e., IF-THEN) reasoning is one of the major elements of formal logic. Thurstone (1938) called one type of this reasoning in logical thinking "deduction" which was in line with traditional terminology on types of reasoning or thinking. It is the ability to draw logical conclusions when given statements using "some," "all," and "therefore." This type of reasoning has also been specifically labeled syllogistic reasoning. For example, given the following statements and three alternatives, one would choose sentence A.
No birds are insects.
All swallows are birds.
Therefore:
A. No swallows are insects.
B. Some birds are not swallows.
C. No insects are birds.

Answer A is selected by mentally verbalizing, "IF" all swallows are birds, and no birds are insects, "THEN" no swallows are insects. This logical thinking skill requires an understanding of language constructs, syntax, and syllogisms. It is expected that students' success in this skill relies on their use of the "IF-THEN" thought process. With practice, one should observe improvement over time.

Learning a computer programming language requires such practice and application of this skill. King (1976), working extensively with students learning to program in Basic, found that students have often developed faulty understandings of the operation of language constructs. Logical reasoning and direction following were strongly related to program comprehension, composition, debugging, and modification. Successfully designing a computer program or debugging an erroneous computer program requires learners to organize their cognitive activities carefully and systematically. Computer programming is said to provide a rich and
opportunistic environment for metacognitive development (Thomas & Upah, 1996).

Cognitive Skills

Educators have questioned whether programming instruction could be used as a device to teach higher level cognitive skills. Some researchers indicated that programming instruction could be used as a tool to teach such skills and quickly produced studies to provide empirical data to support their assertions.

Maddux (1985-1986) noted his concern:

Singing uncritical praises may be a necessary first step in stimulating curriculum change. If such behavior was ever appropriate, however, it no longer is. Promising more than we can deliver and document is dangerous. We, as computer-enthusiastic educators, have been guilty of this in educational computing in general. As a result, we are beginning to see the start of a great backlash of reaction against computers in education. We are being told to justify our claims about the benefits of educational computing. The only convincing way for us to respond is evidence.
Agreeing with Maddux, Ginther and Williamson (1985) indicated that there was a lack of empirical evidence produced to substantiate the promises of computer education. Palumbo (1990) added that early research in computer programming education was not based on documented educational theory. Studies overstated educational gains, did not test the appropriate learner task, and compiled research with too few subjects. Most studies could not be replicated. Palumbo (1990) further stated that in an attempt to conduct research as fast as technology has changed in the last decade, researchers tested for educational gains in categories too general for thorough insights into the effects of learning a computer programming language.

Studies of the cognitive gains conducted after students experience various areas in computer education would provide educators with essential empirical data. Investigation of the factors that may contribute to increases in higher order thinking skills should be one of the components in basic research.

Higher Order Thinking Skills

Meirovitz and Jacobs (1984) evaluated a program for improving thinking by teaching specific skills. This program taught learners how to put together separate but related facts, eliminate irrelevant information, and reach a conclusion using deductive logic. Computer software was available to give the students extra practice using the
strategies. Meirovitz and Jacobs (1984) indicated that although this type of program achieved some success, they questioned whether transfer of the skills learned had been adequately documented. There is an inherent problem with the concept of "teaching thinking," specifically logical thinking. A curricular truism, according to W. Michael Reed (1988), says that we don't teach because we really think the content is important for students to know in the future, but rather, we seek to have them use the content to develop thinking skills at all levels.

Improving student thinking at the college level studied by McCormick (1987) indicated that educators expected students at this developmental level to have mastered critical thinking. Critical thinking is viewed as something which is mastered over time at a developmental stage (Piaget & Inhelder, 1958). The best students do seem to acquire critical thinking skills. However, most learners require more coaching, more explanation, and more practice. Even the best students experience difficulty in transferring problem-solving skills (Riley, Greeno, & Heller, 1981; Palumbo, 1990). Resnick (1989) indicated that students who achieve success at critical thinking did demonstrate a systematic, albeit sometimes erroneous, approach to completion of the task rather than haphazard errors.

**Problem Solving Skills.** Problem-solving is a higher order thinking activity that requires complex mental
activities. These activities include understanding a problem, devising a plan, carrying out the plan, and evaluating the outcome. The mental activities in computer programming include such skills as debugging, building blocks, formal procedures, variable awareness, function organization, generalizability, precise computation, making assumptions, and the use of creativity.

Expert problem solvers use critical thinking and logical reasoning. In a study by Chi, Feltowich, and Glaser (1981), experts used a top-down approach more often and solved problems with greater success. Additionally, they placed more emphasis on identifying principles in outlining a solution. In contrast, novices worked very close to the data or textbook, and achieved a more superficial solution (Dalbey & Linn, 1985). Even at the graduate school level, instruction sometimes failed to lift students to a higher order thinking level, but rather kept them context-bound by the nature of expertise in their own area (Perkins, 1985). Perkins found a smaller than expected impact of computer education on informal reasoning skills of high school, college, and graduate students.

Consistent with criticism of other computer programming research, studies have not yielded significant results in increasing general problem-solving abilities but have shown positive results when testing for a specific skill. Swan and Black (1987) discussed what they considered the three
attributes critical in succeeding in computer programming. This included (a) focusing instruction on a particular aspect of problem-solving ability, (b) using direct instructional techniques, and (c) using mediational learning techniques in dealing with student-teacher interactions. If educators expect gains in problem-solving skills along with achievement in learning a programming language, the choice of programming language, the attention to design and debugging methods, and the types of programming task assignments all become critical factors. Further, it may be unreasonable to expect progress in general problem-solving abilities from a first course in programming (Dalbey & Lynn, 1985; Palumbo, 1990).

**Logical Thinking Skills.** Computer programming is widespread in our school districts and college curriculum. The belief is that some cognitive benefit will result after learning a programming language. The belief is largely based on the nature of the intellectual activity involved in programming. Computer programming requires utilizing deductive thinking processes which are considered to be content free. Learning a computer programming language frees the learner from memorizing needless facts.

One skill required in learning to program is that the learner processes through a statement to the expected outcome of the syllogistic reasoning task. For example, given a premise, the learner is expected to produce the desired outcome through programming code. Writing a computer program
requires the formulation of an algorithm by stating and analyzing a problem, defining a specific sequence of operations to solve a problem, and following the rules of syntax and semantics of a computer language.

Reasoning logically means to apply logical principles correctly. Conditional logic, or syllogistic tasks, uses the logical connectives, "IF", "IF-THEN", "only IF", and "IF and only IF" to connect ordinary language propositions. "IF a THEN b" (conditional chain) is used not only in a mathematical or scientific description but is descriptive of ordinary communication. Linn (1985) stated that thinking critically involves the need to reason logically, IF-THEN reasoning. This logical construction is part of computer programming. Seidman (1981) indicated that it was not clear whether the ability to handle the conditional logic must precede the programming experience or may develop concurrently with it.

Markovits (1986) indicated that familiarity effects on conditional reasoning should permit subjects to generate specific examples more easily and thus, perform better on conditional reasoning problems. Students could approach reasoning problems either concretely by attempting to generate specific examples, or formally by considering the possibilities inherent in the problem. Learners can be expected to utilize a more abstract capacity or creative thinking skills to generate specific examples in unfamiliar
territory. It is not clear which form of reasoning is promoted by the choice of strategies.

Using formal syllogisms to explore reasoning may not correspond to the processes of thinking. Mayer (1983) thought that subjects tested in his study may be giving correct answers because they have memorized an algorithm, have a partial or enough understanding to give a correct answer, or understand the logic underlying the argument.

Utilizing learning a programming language to improve problem-solving abilities and logical thinking presents a curriculum problem. Educators need to know the programming language best suited to the curriculum, what language provides the greatest cognitive benefits, and what computer programming language is age appropriate.

Cognitive Effects of Learning Programming Languages

Most studies have investigated the effects of Logo on cognitive skills for students at the elementary school level. Despite the claims concerning the cognitive benefits of learning a computer programming language, the findings have been conflicting. Clements and Gullo (1984), Ehrlich, Abbott, Salter, and Soloway (1984), and Statz (1973) all reported significant gains in various aspects of cognitive abilities following programming experiences.
In contrast, Liao (1993) in a meta-analysis of the effects of computer programming on students' cognitive performance indicated that there was little, if any, transfer of learning from the programming language instruction to similar non-programming tasks (Cheshire, 1981; Horner & Maddux, 1985; Kurland, Pea, Clement, & Mawby, 1986; Pea & Kurland, 1984). The studies had been conducted at various instructional levels (kindergarten through college) and with a variety of programming languages (e.g., Logo, Basic, and some mixtures of programming languages). The following sections present the cognitive effects of different programming languages.

**Logo Programming**

The majority of studies on programming languages involved children and the Logo programming language. Seymour Papert (1980), who developed Logo, believes that programming allows children to create their own learning environment. One of his claims was that learning Logo enhances problem-solving skills through concrete experiences. He further described these concrete experiences as promoting thinking at a formal operational level. Piaget and Inhelder (1958) described formal operational thinking as the ability to hypothesize, construct relationships, and make inferences.

The Logo experience significantly improved the fifth graders performance on the inversion principle of conditional
logic (Jansson, Williams & Collens, 1987). No attempt was made in the study to ascertain students' interpretations of the logic conditional statement. Although Jansson et al.'s study did not conclude that computer programming is necessary to enhance the curriculum, it did demonstrate that under certain specific conditions, Logo programming did have a statistically significant effect upon a logical thinking ability.

Modest support for the effect of Logo in developing reasoning skills was found by Many, Lockard, Abrams, and Friker (1988). Interestingly, benefits appeared mostly to males and junior high students but not to ninth graders.

According to Mayer and Fay (1987), a chain of cognitive changes can occur when a child learns to program in Logo. One should observe changes in the child's knowledge of the specific features of the Logo language, changes in the child's thinking within the domain of programming, and changes in the child's thinking in domains beyond programming. However, they indicated that there is not enough evidence to assure that after learners experience computer programming, they actually utilized steps or think logically to construct the language. Much of the experiences in programming Logo with young children have been discovery learning, but they were not necessarily transferred to problem-solving situations or to another domain.

A Logo test and a test of spatial cognition, given before and after three sessions of Logo instruction,
indicated that novice programmers (fourth-grade students) showed a general increase in their knowledge of Logo, including a reduction of their misconceptions of the programming commands and gains on a test of spatial cognition (Mayer & Fay, 1987). This study acknowledged that although strong claims for cognitive consequences of learning to program have not always been demonstrated, there are enough promising results from preliminary studies concerning increases in reflective thinking, procedural thinking, and rule learning to warrant further study. It appears that under the appropriate conditions, students do exhibit moderate cognitive changes. Mayer and Fay (1987) suggested that combining studies of the way in which students learn a programming language, of the way students think, and of what the prerequisites are for learning a programming language, should produce significant empirical evidence in favor of learning a programming language to increase thinking skills.

Although many studies (Papert, 1980; Jansson, et. al., 1987; Many, et. al., 1988) made claims that higher order thinking skills were being taught by computer programming instruction and that transfer of skills could be accomplished when learning in this manner, many of the studies, according to Singh (1993), could not or have not been replicated in their original research form, and when replicated, the same significant results were not obtained. Enkenberg (1994) also
summarized that Logo does not naturally support the problem-solving method based on planning that experts exploit.

Seidman (1981) investigated the effects of learning Logo on the conditional statements in the logical reasoning skills of school children. No significant difference was found between students taking a programming language and those not receiving any programming language instruction when test items were scored in the traditional manner (e.g., material conditional). However, when the test items were rescored under a biconditional interpretation, it was found that the treatment group did significantly better in one area of logical reasoning skills. Specific to a logical conditional statement and the Logo conditional branch statement, the study demonstrated that under certain specific conditions, learning Logo programming does have an effect upon logical ability.

In a summary of the literature on the cognitive effects of programming in Logo, Singh (1993) found very few instances of substantial cognitive benefits of Logo. Additional empirical and anecdotal research has been undertaken with mixed results: some studies have shown improvements in problem-solving performance on rule-learning tasks and in metacognition, whereas others have shown no substantial effects of Logo on planning activities. The majority of the studies concluded that if given enough time, performance and conceptual understanding should improve. Singh (1993)
summarized that planning skills have received extensive focus in teaching and research, but research should focus on processes such as reasoning skills, discovery learning, metacognition, and other programming languages.

Other Programming Languages

Fewer studies have been conducted on the effects of learning other programming languages. Fogg (1983) found a small but statistically significant difference in mastery of conditional logic between eighth-grade students who learned Basic programming language and those students who did not. Linn (1985) concluded that learning programming provides a potential "chain of cognitive accomplishments," especially for higher ability students.

Johanson (1987), in his presentation of an in-depth analysis of educational computing and cognitive skills, stated that most research that asserted increases in higher order cognitive processes as a result of learning a programming language was unsophisticated and done at the wrong age level.

Several other important findings were presented in Johanson's report. There was a natural chain of cognitive events in students as a result of the consequences of programming instruction, but students are not processing to the end of the chain. He indicated that it was difficult to test whether or not students learning a computer program
actually improve their logical reasoning abilities, and that it was questionable as to whether students taught the structure of logical reasoning in a verbal format generalize those skills to other areas of thinking behaviors. It may be unrealistic to expect students learning a computer programming language to apply higher order thinking skills in other thought processes (Johanson, 1987).

The effects of Pascal and Fortran have been studied on college students' problem-solving skills. Choi and Repman (1993) found significant increases in their problem-solving skills after learning one of the programming languages. A higher increase was found in students learning the Pascal computer language than those learning the Fortran computer language. This study also provided important findings supporting previous researchers' concerns of the developmental stage in learning (Piaget & Inhelder, 1958; Papert, 1980). Novice learners and expert learners of programming languages will show significant differences in cognitive effects due to the different stages of learners' cognitive development.

Another important component in improving students' problem-solving ability through programming language instruction is how a particular language is taught (Choi & Repman, 1993; Palumbo, 1990). Linn and Dalbey (1985) and Salomon and Perkins (1987) concluded in their studies of students in formal operational (college level), learning the
Pascal or Fortran programming languages, that systematic exposure and interaction during instruction can increase problem-solving skills.

In the light of these conflicting findings, the current study attempted to determine whether learning a programming language, QBASIC, enhances logical thinking skills, thus contributing to the knowledge base in the area.
CHAPTER 3

METHOD

Participants

A site to obtain participants was confirmed through letters (Appendix I) sent to two instructors who had agreed to allow their classes to participate in the research study. The participants were college students in two courses at a community college in Las Vegas, Nevada. The two courses were Introduction to Programming, QBasic, and Computer Graphics/Desktop Publishing.

Permission to conduct research involving human subjects was granted by the Director, Office of Sponsored Programs, University of Nevada, Las Vegas, on September 9, 1996. Permission to test and observe students at the community college was obtained from the individual instructors and participants.

Students enrolled in an introductory programming class, taken for academic credit, served as the experimental group. A graphics/desktop publishing class, receiving continuing education credit, was the control group. A consent form (Appendix II), explaining the study and requesting the
students' participation, by signature acknowledgement, was developed for the pretest date.

Students in Introduction to Programming class took this course as the first one in part of a series of programming classes to complete an associates degree program or to explore an interest in programming. They met in 1 hour and 20 minute sessions, twice a week, and received 32 hours of instruction during the treatment period.

Of the participants who took both the pretest and posttest on syllogistic reasoning tasks (n = 15), the age range of the participants in the programming class was 15 to 62 years with a mean and standard deviation (in parentheses) of 29.00 (16.37). The class was comprised of 5 (33%) females and 10 (67%) males. The students had taken 1 to 6 years of math classes in high school and college with a mean and standard deviation of 3.47 (1.81). Their academic level ranged from tenth-grade to four years of college credits. Twenty-seven students took the pretest, but only 15 students were present on the day when the posttest was given. This was due to the drop-out and absent students.

The graphics/desktop publishing class was offered as a continuing education class. Participants in this group reported no programming experiences. These students met in four hour time blocks, twice a week, and received 32 hours of instruction during the study. The age range of the participants who took both the pretest and posttest (n = 15)
in the graphics/desktop publishing class was 18 to 54 years with a mean and standard deviation of 31.75 (12.25). The class was comprised of 7 (47%) females and 8 (53%) males. The students in this group had taken an average of zero to 6 math classes in high school and college with a mean and standard deviation of 2.93 (1.64). Students in this group were taking this course for different reasons, generally for business and self improvement, and had no programming language experiences. Among the 20 students who took the pretest, only 15 students took the posttest (Five dropped from the class).

A letter of thanks was sent to both instructors after the completion of the study and after the semester (Appendix III). At the request of both instructors, this letter included the names of the students who had participated in the study.

Setting

The entire facility was two years old. Hallway noise was nonexistent, and classroom acoustics were appropriate. Classrooms provided students with non-glare lighting at the work stations, ergonomically comfortable chairs, and comfortable surroundings.

The classrooms for both groups were designed in the same manner. Each classroom consisted of thirty-two individual computer work stations. The instructors' station, located at
the front of the classroom, gave him the ability to control the information on the computer screen of each individual station at any given moment. Both instructors allowed a great deal of freedom when students were at their stations.

The computer programming language group used a personal computer platform. The lab had two printers, available at anytime, for students to obtain a printout of lessons, instructions, or copies of classroom demonstrations. The graphics/desktop publishing group used a Macintosh platform. This lab had approximately ten peripheral devices for printing or scanning documents. Both classrooms had computers and other apparatus that were up-to-date and connected to the school's mainframe. A variety of application and tutorial programs were available to the students throughout the semester.

Characteristics of the Programming Class

No prerequisites were required for this class, but students were expected to be able to handle simple computer tasks. These tasks included copying files, printing files, and using an editor to create and maintain files. Upon completion of the course, the students were expected to understand the basic components of computer programming and be able to specify design, code, test, document, repair, and enhance simple computer programs. This course was preparatory to advanced programming courses and was intended to provide
the foundation of knowledge leading to development of skills in business, scientific, and systems programming.

During the instructional treatment period, students were guided by lecture and a textbook (QBASIC FOR STUDENTS, Michael Trombetta, 1994). Response statements, structure (including input, strings, and looping), syntax, and multiple decisions were emphasized as students learned programming concepts. In addition to in-class participation, eight microcomputer assignments were required. According to the instructor, these assignments are one of the best indications of the students' grasp of the course concepts.

Eight quizzes were scheduled to be given during the course. A midterm and final examination tested students' ability in program error correction and program writing.

The instructor's teaching strategies included a demonstration of the QBasic top-down instructional approach of computer programming, hands-on practice activities, and question-discussion sessions. He was available for student-initiated questions at any time during or after class. The instructor utilized a top-down approach--devising a program by identifying major tasks and then subtasks--until a task can be coded in its' simplest form. He used the textbook as a reference for the students, rather than a requirement of the course. When some students needed a reference point for the programming code being demonstrated, they utilized the textbook.
Materials

Surveys. Two demographic and computer programming experience questionnaires (student profile surveys), were used before and after the treatment (Appendix IV). The survey was designed to collect demographic information and their academic background in math, science, and computer programming experiences. Students were also surveyed after the posttest, providing more demographic information, indicating their age, years in college, number of programming experiences, and computer or graphic application experiences.

Instrument. Logical Reasoning (Hertzka & Guilford, 1955, 1993), a 30-minute test of logical reasoning skills utilizing syllogistic statements, was used as pretest and posttest for both experimental and control groups. This test was designed to measure a primary mental ability in the general area of intelligence. Thurstone (1938) described this measure as "deduction," also known as logical evaluation, or a test of syllogistic reasoning tasks. Tests evaluating this ability are generally of multiple choice category, where the examinee must decide which of two or more conclusions is correct based on stated premises.

The test was designed in two equal parts of 10 minutes each. Each part consisted of 20 multiple choice questions where examinees were required to choose the most logical answer to a syllogistic reasoning statement. The reported reliability of either half of the test was .80. Two kinds of
validity information were given (factorial and practical validity). Validity coefficients ranged from .40 to .60.

Procedure

Pretest. The study took place during the fall of 1996. In the programming class, after the instructor disseminated the introductory information, the researcher met with the students to explain the purpose of the research and requested their voluntary participation. All students present volunteered to participate in the study, read and signed a consent form, and completed the survey form.

Prior to testing, any questions regarding the instructions, the test, the survey, or research study were answered. For the pretest of Logical Reasoning, test instructions were given according to the test manual, including practice with four sample test items. The test was administered in two ten-minute parts with a two-minute break between each part. Students marked a scantron type answer sheet which was then scored manually. According to the instructions in the manual, one point was assigned for each correct response, and one-quarter point was added to each answer left blank.

Participants in the graphics/desktop publishing continuing education class were also asked to volunteer for the study. This class was taught by a different instructor than the programming class. After consent forms were
obtained, students completed the survey form, and instructions regarding the test were given. The *Logical Reasoning* test was administered in the same manner as was done to the Introduction to Programming class.

No adjustments were made in the continuing education classroom for students not volunteering to participate in the study. Students, who chose not to participate, sat quietly until the completion of the testing instrument.

Students in this group met twice a week in four hour blocks, and did not receive any computer programming instruction. This group received no special instruction in logical thinking or problem-solving skills during the period of this study. This class was designed to introduce the computer as a graphic tool and produced computer page design, page assembly, and topography from software. Students were expected to generate and import computer art work, clip-art, and scanned images.

*Posttest.* After completion of approximately 32 hours of instruction, the investigator met with each class and administered the *Logical Reasoning* posttest to students who participated in the study. Students were given instructions in the same manner used during the pretest instruction. Sample test items were reviewed according to the test manual instructions. The students completed the test in two ten-minute sessions with a two-minute break between each part.
of the test. After the posttest, the students completed a second demographic survey.

Each class was debriefed regarding information about the test items, research study, and surveys. Participants asked questions or gave comments regarding their thoughts about the project. Several students stated that they had never had a test like the one administered. Others thought it was somewhat hard but stated that they utilized some strategy in answering the test items. Several students discussed what they thought would be the outcome of the research. One student asked the researcher if she would be disappointed if the research hypothesis would not be supported. Several asked about the outcome of grades and participation in the study.

Observation. The experimental group was observed on a weekly basis by the researcher. Participants' questions and responses were noted when they were asked by either the students or the instructor. Observations were conducted to gather data regarding any change in logical thinking or the use of problem-solving skills. In addition, notes were taken regarding attendance, students' responses to tests given by the instructor, and personal interviews.
CHAPTER 4

RESULTS, DISCUSSION, LIMITATIONS, AND RECOMMENDATIONS

Findings

The purpose of this study was to investigate the effects of learning a computer programming language on logical thinking, specifically syllogistic reasoning skills.

To test whether there was a group difference on syllogistic reasoning scores, an analysis of covariance (ANCOVA) was performed on the posttest scores of syllogistic reasoning with two groups of the community college classes—Introduction to Programming and Computer Graphics/Desktop Publishing—as an independent variable. The test of Homogeneity of regression coefficients indicated that ANCOVA was an appropriate statistical analysis for this study. The pretest scores were used as a covariate, as it was highly related to the posttest scores, \( r (28) = .80, p < .001 \). Other variables, such as academic backgrounds or computer programming experiences, did not have significant relationships with posttest scores on logical thinking. Thus, none of these variables were used as covariates.
The results of ANCOVA did not demonstrate a significant difference between the two groups on posttest syllogistic reasoning skill scores, $F(1,27) = .40$, $p > .50$ with an adjusted means of 26.61 and 25.41 for the computer programming class and graphics class, respectively. The means and standard deviations (in parentheses) of the pretest scores for the computer programming group and the graphic group were 27.50 (6.68) and 24.30 (7.50), respectively. This indicates that pretest scores of students taking the computer programming language class were higher than those of the graphic group, although statistical significance was not found.

The means and standard deviations of the posttest scores for the computer programming group and the graphics group were 28.05 (6.20) and 23.98 (9.61), respectively. No pretest-posttest change in the graphic/desktop publishing class was expected. However, the nonsignificant pretest-posttest change found in the computer programming class may be due to a short treatment period, or because the measure used in this study did not adequately detect the change between the pretest and posttest.

Observations

The purpose of the observations was to explore the change in students' logical thinking and problem solving behaviors. Students in the treatment group were observed
during their 80-minute class sessions in a 14-week period. Class attendance ranged from 5 to 12 during the treatment/observation sessions.

Student-Instructor Interaction

Novice learning. During the first three weeks of class, students developed skills in writing detailed "remark statements", part of the style in QBASIC programming text that would tell them exactly what they were trying to achieve. During this period, the researcher noticed that the students were bound by only informational questions. These included: "How do I correct the code?"; "Where do I insert this code?"; and "How much information do I include in the remark statement?"

During this same three week observation period, students, even those with some programming experiences in high school, produced the exact replication of the codes the instructor demonstrated on classroom programming tasks. When the instructor reviewed homework assignments with the class, the majority of students stated that they referred to the textbook to complete their programming code.

Three students, who completed major portions of their assignments but had difficulty with parts of the code, stated that they copied a previously demonstrated code when they had a coding problem. Others gave no response or stated that they did not know how they got the program to work finally.
By the fourteenth week of treatment two students who had consistently remained textbook-bound said that they completed the assignments in a specific way because the program was written in a particular style in the textbook. Novice learners, according to Chi, Feltowich, and Glaser (1981) do work very close to the data or textbook.

During the fourth week of observations, some students tried new combinations of the programming code to complete their classroom assignments. When the instructor asked the class how they had arrived at a solution for a programming problem, a few students stated that they guessed or used the textbook. Other students indicated that they randomly changed their work until something worked. One student stated that he used the code from the textbook but in a different order. This student stated that sometimes the new code combinations worked, and other times, he abandoned the assignment and wrote the code again.

At this time, the instructor offered his views on the progress of the students with the researcher. He was asked if he observed any changes in logical thinking patterns attributable to learning the programming language.

According to the instructor, students who were achieving completion of the assignments, readily responded to questions about classroom work or homework assignments. They demonstrated that they understood the top-down approach and understood the objective of a thorough remark statement. On
the other hand, students who were not submitting work or offering solutions to classroom programming tasks, would probably remain textbook-bound.

Near and far transfer. Close to halfway through the treatment period a few students were asking questions about concepts ahead of the skill level that they had learned. They had not covered subroutines, yet one student's question was, "How do I put this in a program to make it work as a code to control another program?" This topic would be covered in the subtasks programming routines later in the class.

At this time the instructor reviewed programming samples with coding errors to prepare the students for the midterm test. When the students were reviewing the programming errors, some students were using previously learned information to offer a solution to the programming error samples, "The order of your arguments is crucial, not the variable names, so you need to change the order to make the program work" (e.g., mathematical computation, mathematical relationships, grammatical structure).

Approximately eight weeks through the treatment period, half of the students asked more technical questions, such as, "Why can't you use zero as a tangent function?" or "Isn't there a problem in a loop if it goes on forever?" This style of question indicated that some students were transferring knowledge of one subject (e.g., mathematics) to the current

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
subject. They were also applying what they had learned toward future programming instruction.

**Logical thinking skill.** During approximately the fifth week of the session, two students who were succeeding to a certain degree at writing programming code demonstrated a desire to write simple programs that answered their questions of the QBasic programming language capability. They used "what if and then" style questions rather than "what do I do to?" or simple informational questions.

When the instructor elicited responses regarding how they completed the homework assignment, the previously described students responded with some form of logical reasoning when explaining their code. For example, one student stated that he asked himself questions like, "if it works in this situation or in this program, then shouldn’t it work here" to complete the programming assignment.

The researcher discussed progress in programming with one of the students after class on several occasions. She stated that her approach to learning a programming language was similar to the skills she had used in her previous career, but she still had difficulty making some of the code work when she was away from the classroom. On another occasion, the student offered insights to her approach to correcting erroneously written code. She stated that she did not always use the same approach when correcting coding errors. Sometimes she randomly changed what she thought she
remembered regarding the programming language structure or syntax. On other occasions, the student explained that noting each step she used in the program helped her with another assignment. She described her notes which included logical reasoning ("IF-THEN") skills.

Problem-solving strategies. At approximately the tenth week of the treatment, none of the students mentioned utilizing the textbook to find a solution to the computer programming problem they were experiencing. They informed the instructor, when asked how they resolved a problem, "I used debugging to fix the problem" or "I used the deleting to fix the program." Some students were attempting to use some problem-solving strategy, and the students were becoming more comfortable with their skills. Two students in the class did refer to the textbook during this period.

At this time the instructor initiated some examples of programming problems and asked how the students might resolve the problems. Students offered some problem-solving responses other than debugging and deleting. One student described using a problem-solving strategy stating that he looked for coding errors in a systematic way. He demonstrated some planning behaviors, first by reviewing the response statement, which described the desired results, and then by organizing the variables and writing the respective code.
At about the tenth week the student, who had previously discussed her progress with the interviewer, described her problem-solving approach. First, she reviewed the code she had written for obvious errors (e.g., spelling, commas, spacing). Then, she asked herself if the code actually achieved what the remark statement described. Then she attempted to make corrections to the program, utilizing a debugging feature inherent in the program and ran her work. She said a systematic approach rather than a haphazard approach seemed to be less frustrating when correcting an erroneous program.

Discussion

Logical reasoning skills did not improve after receiving instruction in an introductory computer programming language course. However, as the treatment period progressed, some students did ask questions and give responses in a manner which indicated that they were processing some form of logical thinking or using problem-solving skills. Students with an understanding of math concepts, as they relate to computer programming, appeared to be more comfortable with the programming code structure than other students. This is consistent with findings in the studies that found evidence of mathematical problem-solving and computer problem-solving.
similarities (Choi & Repman, 1993; McCoy, 1990; Palumbo, et. al., 1988, Dalbey & Linn, 1985). Students in the programming class had taken about .54 more mathematics courses on the average than the students in the graphics class. This may partly indicate the reason for the higher syllogistic reasoning skills in the computer programming class ($p = .07$). However, how the students in the computer programming class acquired higher reasoning skills can not be explained based on the current study. As the treatment progressed, students were gradually using syllogistic reasoning skills or the IF-THEN statement as it relates to solving a programming problem.

It appeared that some students could look at an assignment from the larger picture and compose the detail. This top-down approach was part of the instructor's teaching paradigm. Due to the limited treatment period, it is not known whether other students could have reached this top-down approach to programming. Pea and Kurland (1984) and Dalbey and Linn (1985) indicated that considerable time (experience) and the right style of instruction was needed to produce cognitive changes. It is also suspected that the finding of nonsignificant class difference in this study might have resulted from either/or or both the duration and content/style of instruction that might not have exerted a strong influence on cognitive change in logical thinking. Future studies should address whether different teaching
strategies in computer programming would have differential effects on cognitive learning.

This class appeared to be a typical heterogeneous group. Some students were interested in acquiring programming skills and learning new concepts. These students transferred their understanding or previous learning to this subject, and as the session progressed they demonstrated their use of logical thinking skills. Other students remained textbook-bound and achieved only the objectives of the immediate lesson. A small number of students displayed disinterested behaviors. Students who only reproduced programming code or demonstrated disinterested behavior may be part of the reasons that the effect of learning a programming language did not show in this study. More importantly, it is questionable whether the instrument used in this study was the appropriate measure for the current participants.

Limitations and Recommendations

As discussed in the review of literature, studies investigating the cognitive behavior change after learning computer programming languages have shown conflicting findings. Problems of finding the appropriate instrument to measure the logical thinking skills or problem-solving skills, inadequate research designs, and sample selections, were among those reasons offered for these inconsistent findings.
Studies have indicated that some tests have not actually measured cognitive gains in the domains in which students were learning (Tobin & Capie, 1981; Ahlawat & Billeh, 1987). Valid testing instruments that actually measure the reasoning skills are still to be found or developed. It is possible that the instrument used in the current study was not the proper test for college students in this study. Using or developing a test designed to measure different components of logical thinking and problem-solving skills might help future investigations in finding the cognitive behavior change after learning a computer programming language.

Furthermore, various types of tests may be compared to determine if using different types of test would result in detecting different types of cognitive behavior change. For example, findings from using a multiple-choice logical thinking test can be compared to findings from using a think-aloud protocol. By using a variety of methods for collecting data, researchers not only can determine the cognitive gains, but also can see how the students process information to reach a conclusion or to solve problems.

Syllogistic reasoning skills require conditional logic. However, an introductory class in Basic programming does not necessarily provide students with enough practice in this type of logic to apply it to their programming. This may be one of the reasons for not finding the significant differences between the two groups.
One of the methods found to be useful in this study was the observations of student-teacher interaction. It would be also advantageous to have students keep a personal journal during the course of a treatment period. Students' writing or journal entries may reveal whether and when they understood how to approach and solve problems.

With the current nonsignificant finding and previous conflicting findings on this topic, it is only logical to continue research and find the way to help students improve their logical thinking skills.
REFERENCES


McCormick, K. *Improving student thinking at the community college*. Mid-Career Fellowship Program, Princeton University, NJ (May 1987).


APPENDIX I

FORM LETTERS SENT TO INSTRUCTORS OF COMPUTER PROGRAMMING AND GRAPHICS/DESKTOP PUBLISHING CLASSES TO OBTAIN A RESEARCH SITE
August 1, 1996

Dear

Thank you for speaking to me regarding conducting research for my thesis in your Introduction to Programming, QBasic, classroom during the fall semester 1996. I appreciate your assistance and am looking forward to discussing the final plans for testing and collecting data. Students who volunteer from your class will be the experimental group in this study.

I have enclosed information regarding the test, sample survey forms, a consent form, and a copy of the University Protocol approval information. I am also enclosing two of the research studies, from which I am obtaining some of my information for my thesis.

If there are any questions or further information you require, please contact me at your earliest convenience. I would like to meet with you prior to the beginning of the semester to review the process with your students. Please contact me if this would work into your schedule. Again, thank you for your assistance.

Very truly yours,

Margaret G. Mains
 Graduate Student

Dr. Eunsook Hong
 Faculty Advisor

Enclosures
August 1, 1996

Dear

Thank you for speaking to me regarding conducting research for my thesis in your Graphics/Desktop Publishing classroom during the fall semester 1996. I appreciate your assistance and am looking forward to discussing the final plans for testing and collecting data. Students who volunteer from your class will be considered the control group in this study.

I have enclosed information regarding the test, sample survey forms, a consent form, and a copy of the University Protocol approval information. I am also enclosing two of the research studies, from which I am obtaining some of my information for my thesis.

If there are any questions or further information you require, please contact me at your earliest convenience. I would like to meet with you prior to the beginning of the semester to review the process with your students. Please contact me if this would work into your schedule. Again, thank you for your assistance.

Very truly yours,

Margaret G. Mains
Graduate Student

Dr. Eunsook Hong
Faculty Advisor

Enclosures
APPENDIX II

LETTER OF INTRODUCTION AND
STUDENT CONSENT FORM
September 1, 1996

Dear Student,

My name is Margaret Mains, and I am a graduate student at the University of Nevada, Las Vegas. I am conducting research for my thesis as part of my Masters Degree program. This research will investigate the effects of learning a computer programming language on syllogistic reasoning tasks. You are invited to participate in this study.

You will be asked to take a 30 minute pretest which requires that you mark the best answer to logical statements. You will also be asked to complete a student profile survey which describes you, classes you have taken, your computer experience, and computer availability. Prior to the completion of the semester instruction, you will take a posttest and complete another survey form. These tests and data collection have no bearing on your academic standing, the information is collected for research purposes only.

Your name will not be linked to the information in this study. All data will be in summary form. Studies of this type contribute to the overall effectiveness of education. They may affect future decisions regarding curriculum, program development, testing, and improved educational instruction.

Participation is strictly voluntary, and you must sign a consent/release form prior to administration of the pretest. You may withdraw from the study at any time. I will be happy to answer any questions that may concern you. You may contact me at your convenience. Thank you.

Margaret G. Mains
Graduate Student

Dr. Eunsook Hong
Faculty Advisor

I have read and understand the consent form and I agree to participate.

________________________________________________________________________________________

Student Signature                                      Date
APPENDIX III

LETTERS OF APPRECIATION SENT TO THE TWO PARTICIPATING INSTRUCTORS

62
December 1, 1996

Dear

Thank you very much for allowing me to conduct research for my thesis in your classroom this semester. I appreciated your cooperation and the cooperation of your students. Your assistance was invaluable.

I will deliver a completed copy of my thesis after May 1997. Per your request, I am enclosing a listing of the students from your class who volunteered as part of the experimental group for the study. Please thank your students again, and remind them that they may contact me at any time with any questions regarding their participation.

Again, thank you for your time and assistance. If there are any questions or concerns, please contact me at your convenience.

Very truly yours,

Margaret G. Mains
Graduate Student

Dr. Eunsook Hong
Faculty Advisor

Enclosures
December 1, 1996

Dear

Thank you very much for allowing me to conduct research for my thesis in your classroom this semester. I appreciated your cooperation and the cooperation of your students. Your assistance was invaluable.

I will deliver a completed copy of my thesis after May 1997. Per your request, I am enclosing a listing of the students who volunteered from your class to be part of the control group for the study. Please thank your students again, and remind them that they may contact me at any time with any questions regarding their participation.

Again, thank you for your time and assistance. If there are any concerns or questions, please contact me at your convenience.

Very truly yours,

Margaret G. Mains
Graduate Student

Dr. Eunsook Hong
Faculty Advisor

Enclosures
STUDENT PROFILE SURVEY 1

Please complete the following information as directed. Your cooperation is greatly appreciated.

Name (Print)______________________________________________________
Gender (Circle One) Female Male
Academic Level (Circle One) 13 14 Other

Mathematics classes completed in high school/college. (List)
_________________________________________________________________
_________________________________________________________________
_________________________________________________________________

Computer classes completed in high school/college. (List)
_________________________________________________________________
_________________________________________________________________
_________________________________________________________________

Do you have a computer for personal use? (Circle)
Yes No

What other computer programming languages can you write? (List)
_________________________________________________________________
_________________________________________________________________
_________________________________________________________________

What computer applications do you use? (List)
_________________________________________________________________
_________________________________________________________________
_________________________________________________________________

How many hours do you spend on the computer per day? (Circle)
Under 2 hours 2-5 hours More than 5 hours

Thank you for your participation.

Margaret G. Mains  Dr. Eunsook Hong
Graduate Student  Faculty Advisor

66
STUDENT PROFILE SURVEY 2

Please circle the correct response or list the information as requested. Your cooperation is greatly appreciated.

Name (Last, First) ____________________________________________

Date___________

Age___________

School (Circle)

Grade Level (Circle or List) 13 14 15 16 Other____

Course (Circle) Programming Graphics

Instructor (Circle) Name Name

Programming Languages (List)


Applications (List)


Graphics (List)


Do you have a computer for personal use? YES NO

Do you use the lab computers? YES NO
Approximately how many hours per week do you use the computer for:

(Circle the number of hours in each category)

<table>
<thead>
<tr>
<th>Category</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>More</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class assignments</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Independent learning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Games</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record Keeping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internet or WWW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Can you apply information that you previously learned in a math class to learning a programming language?  **YES**  **NO**

Can you apply information that you previously learned in a science class to learning a programming language?  **YES**  **NO**

Can you apply information that you previously learned in an English class to learning a programming language?  **YES**  **NO**

Is there a class or instructor that taught you a strategy that assists you in the course/courses that you are taking?

**YES**  **NO**  **If YES, explain**

If you make a mistake in a program/application or a situation does not work, what approach do you use to correct it?

Thank you for your time and assistance.

Margaret G. Mains  
Graduate Student

Dr. Eunsook Hong  
Faculty Advisor

68