DOCUMENT RESUME

ED 350 981 IR 015 791

AUTHOR Jang, Younghee
TITLE Cognitive Transfer of Computer Programming Skills and Analogous Problem Solving.
PUB DATE Apr 92
NOTE 45p.; Paper presented at the Annual Conference of the American Educational Research Association (San Francisco, April 20-24, 1992).
PUB TYPE Reports - Research/Technical (143) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS Calculus; *Cognitive Processes; Higher Education; *Intermode Differences; Learning Laboratories; Learning Strategies; *Lecture Method; Microcomputers; *Problem Solving; *Programing; Programing Languages; Research Design; Statistical Analysis; *Transfer of Training
IDENTIFIERS Nested Loops; PASCAL Programing Language

ABSTRACT

This study investigated the cognitive benefits of learning how to program by determining the degree of cognitive transfer of programming skills at a construct level to solving analogous problems in other domains. Subjects, who were students enrolled in four sections of the beginning Pascal programming course and two sections of a calculus course, were assigned to the experimental group or one of two control groups. For both the experimental group (n=42) and the first control group (n=51), the programming course consisted of 150 minutes of lecture and 150 minutes of laboratory per week. Both groups were given instruction in programming and in the language Pascal through lectures and laboratory activities lasting 6 weeks and 4 weeks respectively. Only the experimental group was taught loop and nested loop constructs and the corresponding Pascal codes whose outputs embodied the combination and intersection schema of Piaget. The second control group (n=38) was given instruction in calculus consisting of 200 minutes of lectures per week for 7 weeks. Results showed significant effects of learning the nested loop construct on solving analogous problems in other domains. By using a tracer, the degree of learning of the nested loop construct was positively related to the degree of transfer. Students who scored higher on the achievement test, which was used in determining understanding and mastery of the nested loop construct, scored higher on the transfer posttest than those who scored lower on the achievement test. Students who learned the nested loop construct and who did not perform well on the transfer posttest showed significant improvement after a hint was given. (Contains 7 tables and 32 references.) (Author/ALF)

Paper presented at the 1992 National Meeting of the American Educational Research Association, San Francisco

Cognitive Transfer of Computer Programming Skills and Analogous Problem Solving

by

Younghee Jang
Southwest Regional Laboratory
Los Alamitos, California

The experimental results described in this paper were part of the doctoral research conducted by the author at the University of California, Los Angeles. Partial support was provided by two University Fellowships and Graduate School of Education Dean's Circle Fellowship.

Cognitive Transfer of Programming Skills and Analogous Problem Solving

This study investigated the cognitive benefits of learning to program by determining the degree of cognitive transfer of programming skills at a construct level to solving analogous problems in other domains. Subjects were students in a four year college enrolled in the beginning Pascal programming course and a calculus course. Four sections of the programming course and two sections of the mathematics course were selected to serve either as experimental group, control group one, or control group two. For both the experimental group (n=42) and the first control group (n=51), the programming course consisted of 150 minutes of lecture and 150 minutes of laboratory per week. Both groups were given instruction in programming and in the language Pascal through lectures and laboratory activities lasting 6 weeks and 4 weeks respectively. Only the experimental group was taught the loop and nested loop constructs and the corresponding Pascal codes whose outputs embodied the combination and intersection schema of Piaget. The second control group (n=38) was given instruction in calculus consisting of 200 minutes of lectures per week for seven weeks. Results showed significant effects of learning the nested loop construct on solving analogous problems in other domains. By using a tracer, the degree of learning of the nested loop construct was positively related to the degree of transfer. Students who scored higher on the achievement test, which was used in determining understanding and mastery of the nested loop construct, scored higher on the transfer posttest than those who scored lower on the achievement test. Students who learned the nested loop construct and who did not perform well on the transfer posttest showed significant improvement after a hint was given.

Technologists and educators believe that future classrooms will be populated with computers. Either as a medium to store and transmit knowledge or as a vehicle to enhance mental abilities, the computer is considered vital to the education of our next generation. Cognitive benefits of learning to program have been claimed by proponents of computer education (Papert, 1980), and are often used as the justification for teaching programming in our schools. Proponents claim that learning to program will help our students learn to think more creatively and become better problem solvers. Papert (1980) believes that, when children are allowed to write programs and explore in a LOGO environment, "powerful intellectual skills are developed in the process." Bork (1981) expounds the view that analytical thinking skills applicable to a great number of areas can be learned in the context of computer programming. There is some theoretical and logical support for this view. Problems in many different domains require substantial planning and exploration before a solution becomes apparent. In implementing a solution, the problem solver is often required to transform the problem into a suitable representation, which is often symbolic and mathematical. The solution is then carried out in an algorithmic fashion. Programming often involves planning, revision of plans, debugging, choice of representation, and algorithmic thinking. It is these characteristics of problem-solving found in programming activities that provide the basis for many of the claims.

Empirical studies into cognitive benefits of programming are of more recent origin. Since there is not a large body of codified work, it is difficult to view the studies with great specificity. Generally, the studies can be classified into three categories. The studies in the first category were empirical studies of classroom instruction of LOGO or BASIC over a fixed period of instruction. Generalized cognitive benefits were measured through correlational methods. Most of the studies in this group came earlier chronologically (Howes, O'Shea, & Place, 1980; Soloway, Lochhead, & Clement, 1982; Hines, 1983; Clements & Gullo, 1984; Milojkovic, 1984; Miller, Kelley & Kelley, 1988). The results were mixed, and there were methodological questions raising doubts about the results.

However, the overall indication is that positive results are possible under certain restricted conditions, such as under well-taught and structured instruction. Pea and Kurland (1984) provided a detailed analysis of a general framework for understanding studies on the cognitive effects of learning computer programming. Part of their conclusion was that the cognitive consequences of different levels of programming skill would provide far more relevant information for guiding the process of education than standard correlation studies.

Methodological concerns such as the failure to determine the degree of learning and to relate what was learned to what was transferred resulted in another group of studies. The primary focus of these studies was not necessarily in demonstrating the cognitive benefits of programming but rather tried to achieve a better understanding of programming skills or use a more specific measure of cognitive benefit.

These studies (Clement, Kurland, Mawby, & Pea, 1986; Kurland, Pea, Clement, & Mawby, 1986; Natasi, Clements, & Battista, 1990) were more focused on particular aspects of programming. Clement et al 's study (1986) predicted and obtained a significant correlation from pretests of analogical reasoning to posttests of programming ability to write subprocedures usable for several different programs. In Kurland et al's study (1986), programming experience (as opposed to expertise) did not appear to transfer to other domains which shared analogous formal properties. The failure to transfer could be due to the rudimentary understanding of programming. This study indicates the importance of the determination of the level of programming skill as part of any transfer study.

Recent studies (Carver, 1988; Klahr & Carver, 1988; Dyck & Mayer, 1989; McGrath, 1989; Clements, 1990; Clements, 1991) were based on detailed models with tighter control over the instructional methods. They were either influenced by or had taken into account some of the studies on cognitive transfer in the area of cognitive psychology (Ellis, 1985; Gick & Holyoak, 1983; Holyoak, 1985; Sternberg, 1985). Some of these studies asked more specific questions. The results were sharper but with less general

applicability. Klahr and Carver (1988) addressed an issue generally not dealt with in the experimental studies on programming, namely that effective transfer of knowledge depends on its degree of learning. They used an explicit model of the debugging process and showed that students who acquired better debugging strategies performed better on transfer tests of debugging in nonprogramming contexts. This experiment can be viewed as teaching a metacognitive debugging strategy which has wide general applicability across different domains.

However, none of these studies were focused at the level of specific programming constructs. If specificity in addressing what is learned to what is transferred is desired, then a study cannot be based on an entire segment of instruction. It must be refined to a level where the programming concepts or constructs taught can be identified. A fundamental result in programming methodology is that all structured programs can be expressed in terms of three basic programming constructs: sequencing, if-then-else, and the while-loop (Boehm & Jacopini, 1966). Sequencing is simply doing one instruction after another in temporal order. Because of its perceived simplicity, there has been no study conducted on its cognitive effect. Studies on the other two constructs are therefore of great significance. However there has only one such study.

This one study was on the if-then-else construct. It was conducted by Seidman (1981), and analyzed in Seidman (1989). This pioneering study was on the cognitive effects of the *if-then* and *if-then-else* statements on logical reasoning. Seidman postulated that if transfer of learning occurred and subjects interpreted the ordinary-language conditional statement in the material conditional manner, then a negative effect would be expected. The result showed a negative effect on the experimental group on reasoning with the ordinary material conditional. As pointed out by Seidman, the ordinary material conditional does not have the same semantics of the conditional *if-then* of programming languages. This study points out the importance of using analogous problems in other

domains in a transfer study as unintended cognitive effects may result in transfer to similar problems with different underlying structures.

The available evidence seems to show that learning to program results in *weak*, if any, transfer of cognitive skills to other domains. However, the lack of positive results on cognitive transfer in some of the studies on programming could be due to the design of the experiments being too general to detect specific transfer or due to a lack of mastery of the initial learning. In addition, it could also be due to a lack of cues as the work of Holyoak and Gick (1980, 1983) showed that both adults and children have difficulty in solving analogous problems, but if the subjects are cued or the context is made more explicit, transfer to analogous problems does occur.

Objective of Study

The objective of the present study was to investigate the cognitive benefits of learning to program by determining the degree of cognitive transfer of programming skills to solving analogous problems in other domains. The study addressed the following questions: (a) Do the procedural thinking skills in programming constructs and activities transfer to analogous problem situations in other domains ? (b). Does a greater degree of learning of the skills in programming activities lead to a greater degree of transfer to analogous problems in other domains ? and (c) Does a greater degree of transfer result upon receiving a hint ?

The present study differed from the methodology generally employed in current research into the cognitive benefits of computer programming in its approach and in its specificity. First, it was focused on a specific construct of computer programming, namely that of the nested loop. The transfer effect of a single programming construct to isomorphic problems in other domains was studied, rather than the cognitive benefit of an entire course of instruction as measured by some general tests, whose relation to the materials learned may not be entirely clear. Second, the transfer was to analogous

problems in other domains. By using the procedural manifestation of the nested loop as a tracer, not only the end result of transfer but also degrees of transfer could be determined. In almost all of the previous studies, only generalized cognitive benefits were considered. Since the experimental results on generalized cognitive benefits were mixed, and since there was no precise relation between what was learned and what was transferred, the present study tried to address some of these difficulties by focusing on an important programming construct, and tried to provide a more conclusive answer to the question of cognitive transfer in the area of computer programming.

Intersection Schema and Combination Schema

The problems used in this study were inspired by a set of combination and permutation problems used by Inhelder and Piaget (1969) in their studies with children's formal operational ability. The tasks studied by Piaget and Inhelder involved the generation of pairs from sets of items, for example one version was on the mixing of chemicals. Piaget's "intersection" procedure is described below. In forming all possible pairs from two sets A and B, the abstract mathematical notion Cartesian product, $A \times B$, is the set consisting of all ordered pairs (x,y), where x is from A and y is from B. There are many different ways of forming all these pairs. Piaget envisioned the matrix schema, with the elements of A along one dimension and the elements of B along the other in a matrix, and the ordered pairs listed in a tabular form. A person with the matrix conception can systematically list all the possible pairs, for example by going along each row until the pairs are exhausted, and then continuing with the next row. This planful behavior of listing the pairs row by row is referred to as the *combination procedure schema.* More complex isomorphs exist between multiple nested loop and the Cartesian product of a collection of sets, which were also used in the present study.

In the case of the two sets being the same, as in many of the tasks (e.g., Piaget's chemical mixing task) where the order of the entries in a pair is immaterial, the *intersection procedure schema* (for a set A = {1,2,3,4,5}) is as follows:

	1	2	3	4	5
1		(1,2) --->	(1,3) --->	(1,4) --->	(1,5)
2			---> (2,3) --->	(2,4) --->	(2,5)
3				---> (3,4) --->	(3,5)
4					----> (4,5)
5					

According to Piaget, children with this matrix conception could exhaust all the combinations by choosing pairs planfully from beginning to end. Children without the matrix conception typically would make pairs without an orderly pattern, and hence could not be certain of exhausting all possible pairs.

The Loop Construct

The combination and intersection schema are embedded procedurally in one of the central programming constructs, namely that of the nested iterative loop, which is used in numerous sorting and searching algorithms. Prototypic nested loops, in Pascal, are the following:

```
for I := 1 to N do                for I := 1 to N do
    for J := I to N do                for J := I+1 to N do
        Action(I,J);                      Action(I,J);
```

whose outputs are isomorphic to the combination procedure schema and intersection procedure schema respectively. The outputs of these programs (with N = 4, and *Action(I,J)* being write the values of I and J =) are

11	12	13	14	12	13	14
21	22	23	24	23	24	
31	32	33	34	34		
41	42	43	44			

respectively. In the first case, the static output is the result of a procedural manifestation of the Cartesian product of $A \times A$, where A is the set 1, 2, 3 and 4. Procedurally, the first problem is isomorphic to the combination procedure of Piaget. With its well-defined procedural steps (as indicated by the arrow -->), it served as a tracer in the next phase of the experiment.

11 --> 12 --> 13 --> 14 -->
21 --> 22 --> 23 --> 24 -->
31 --> 32 --> 33 --> 34 -->
41 --> 42 --> 43 --> 44

Procedurally, the second problem is isomorphic to the intersection procedure of Piaget. With its well-defined procedural steps (as indicated by the arrow -->), it served as a tracer in the determination of the degree of transfer in this experiment.

12 --> 13 --> 14 -->
23 --> 24 -->
34

Method

Subjects

Students in a four year college enrolled in the beginning Pascal computer programming course and mathematics course were part of the study. Four sections of the programming course and two sections of the mathematics course were selected to serve either as experimental group, control group one or control group two. The number of

subjects in each section ranged from 23 to 33. Forty-two (35 males and 7 females) out of 58 students in the experimental group, 51 (38 males and 13 females) out of 64 in the control group one and 38 (33 males and 5 females) out of 62 in the control group two had scores on all the measures administered. Fifty-three out of 184 subjects either enrolled in the section after the study had begun, withdrew from the course during the study, or failed to take one of the tests. Subjects majored in 21 different fields of studies.

Design.

The design consists of three groups. Each group comprised of two sections. Four experienced computer science and mathematics instructors participated in this study. One instructor taught both sections of the experimental group, two instructors taught each section of the control group one, and the remaining instructor taught both sections of control group two. Two of the four Pascal sections were randomly assigned as the experimental group and the other two as control group one. The two calculus sections were assigned as control group two. The first control group was used to rule out the possible contribution to cognitive transfer from the initial phase of instruction in programming up to but not including the loop instruction. The second control group was used to control for self-selection factors into a programming class. Two sets of the transfer test, forms A and B, were counterbalanced within the experimental group, control group one, and control group two.

Measures

The measures administered in this study were the Transfer Tests (used as a pretest and posttest for assessing transfer) and the Achievement Test (used to assess the learning of the nested loop construct). The transfer test was used to determine the degree of transfer from the programming domain to different domains. Two sets of questions, Form A and Form B, were constructed for use in the pretest and posttest to render them less readily

recognized as identical tests. Table 1 displays the problems in Form A and Form B. Each form contained three problems analogous to the nested loop constructs in nonprogramming domains. Both forms contained isomorphic problems with different embedding story situations. Two of the three problems dealt with the use of the intersection schema to solve the problem, and the remaining one dealt with the combination schema. Students were given thirty minutes to complete the transfer test.

Insert Table 1 about here

The achievement test was used to determine the subject's understanding and mastery of the nested loop concept. It consists of four parts. The first part contained three program comprehension problems to assess the understanding of the codes involving the nested loop construct. For each problem, subjects were asked to write down the output of the given segment of code. The execution of the nested loops in these problems is the intersection schema and the combination schema by which the transfer tasks can be solved.

Parts two, three, and four were designed to assess the ability of the subjects to write correct Pascal code involving the loop construct to generate given outputs. The second part consisted of two problems. The first problem was a simple test to see whether the subject knew the basic loop construct. The second problem tested whether the subject could apply his or her knowledge in problem 1 to produce a given output. The third part consisted of two questions. The first was similar to the first problem of part two except it was slightly more complex. The second problem was designed to test mastery and skillful application of the nested loop to generate a given pattern. The problem was judged to be quite difficult for most subjects. A subject who solved this problem certainly had mastered the nested loop construct. The fourth part consisted of two problems. The first and second problem asked for the code to generate given outputs which embodied the intersection and combination schema respectively.

The achievement test assessed the subjects' learning in four areas/concepts: single loop construct (part 2 problem 1 and part 3 problem 1), intersection schema (part 1 problem 2, part 2 problem 2, and part 4 problem 1), combination schema (part 1 problem 1, part 1 problem 3 and part 4 problem 2), and nested loop construct (part 3 problem 2). Students were given thirty minutes to complete the test. Table 2 presents representative excerpts from the achievement test.

Insert Table 2 about here

Treatment

For both the experimental group and the first control group, the beginning Pascal programming course consisted of 150 minutes of classroom lecture and a 150 minutes of laboratory session each week. Each student had his or her individual terminal for laboratory work. The instruction was coordinated and uniform across all sections. All the sections followed the same instructional schedule, used the same textbook, and covered the same content. The text was *Oh! Pascal* by Doug Cooper and Michael Clancy, published by W. W. Norton & Company. The same set of laboratory activities was also used in all laboratory sessions. All the students in the experimental group and the first control group were given instruction in programming and in the language Pascal through lectures and laboratory activities.

Control Group One Introductory concepts of programming such as variables, input, output, and assignment statements were taught prior to the loop construct. A simple treatment of functions and procedures was also given prior to the loop construct. These topics lasted until the 4th week of the instruction for the first control group.

In the first week, a brief history of computers was followed by an introduction to the computing facilities and the basic commands of the operating system, including how to edit and compile a program. Instruction on programming and Pascal began in the second

week by concentrating on programs with simple input and output commands. Towards the end of the second week, the concepts of variables and debugging were covered. In the third week, expressions and assignment statements were the central topics. Progressively more complex programs were presented to illustrate these new topics. Toward the end of the third and in the fourth week, programming with procedures and function subprograms were covered. In addition, top down design and stepwise refinement were introduced. The use of procedures and functions to support the programming methodology was discussed. More advanced concepts such as the scope of a variable, value parameters and variable parameters were covered in the fourth week.

The laboratory sessions paralleled the classroom lectures. In the first week, students were taught how to log onto their computer accounts, and execute simple system commands, such as how to send and receive electronic mail, how to save and retrieve files. In the second week, students were asked to debug simple Pascal programs. They were asked to find and correct the syntax errors in the given program. In the third week, the laboratory assignment was to write a program to solve a numerical problem involving arithmetic expressions and assignment statements. In the fourth week, a more difficult program involving addition, subtraction, multiplication and division of rational numbers was assigned. In order to write this program, a function subprogram which computed the greatest common factor of two numbers was given to the student. In this lab, students became more familiar with the use of functions and procedures. In general, the laboratory assignments lagged behind the lecture to allow time for the students to absorb the concepts.

Experimental Group The instructional program was the same as the first control group during the first four weeks. The experimental group received instructions on the loop construct in the 5th and 6th weeks of instruction. The single loop construct was introduced using the *for* statement. This was followed by instruction and examples on the nested loop construct. The students were taught codes involving the nested loop construct whose outputs embodied the intersection and combination schema. Students wrote codes

for problems involving the loop construct in their programming activities in the laboratory sessions.

In the fifth week, the loop construct was introduced as a mechanism for repetition. This was followed by an explanation of the syntax diagram of the **for** statement. Additional examples of the single **for** loop were given. The primary example was to introduce the iteration scheme via the example of finding the sum of the first 100 positive integers. The nested **for** loop was introduced following the single loop instruction. The fact that the inner loop is to be executed entirely before the outer loop counter variable could be changed to a new value was emphasized. The concept of a one-dimensional array was briefly discussed, as it was used in some of the lecture examples. The instructional examples were all chosen from the textbook. The application of the nested loop construct to sorting was mentioned without going into the detailed working of the program. Use of the nested loops was illustrated by many examples, including an elementary sorting method. The different forms of the nested loop used in these examples embodied both the combination and the intersection schema.

The lecture treatment concluded with instructional materials on the structure of loops and the values of the counter variables. A table of M rows and N columns can be indexed by two variables, I and J. The instructor explained that the indexing would be useful in the study of two dimensional arrays. A systematic way of going through all the entries of a table can be represented by a nested loop construct. Assignments embodying the combination and intersection schema were given to the students to be completed in the laboratory session. Students were asked to write complete Pascal programs to generate the same output as those displayed on paper. This constituted the treatment in the fifth and sixth weeks of instruction.

Control Group Two The calculus course consisted of 200 minutes of lecture per week. The instruction was uniform across both sections. All the sections in the second

semester calculus course used the same text, and covered the same topics. The text was *Calculus, Third Edition* by Howard Anton, published by John Wiley & Sons.

The instructional program for the seven-week duration followed closely the text used in the course. Both sections were taught by the same instructor. Students in the second control group had finished one semester of calculus, which covered the basic concepts of analytical geometry, and differential and integral calculus. Applications of the differential calculus were also a part of the first course. Hence instruction started with a review of the basic properties of the definite and indefinite integral. In the second week, integrals involving the trigonometric functions were covered. In the third week, the definition of the natural logarithmic function as the integral of the function 1/t from 1 to x, and its properties were covered. In the fourth week, the exponential function was introduced as the inverse of the natural logarithmic function. In the fifth week, applications of integrals of these functions to physics and economic problems were covered. In the sixth and seventh weeks, techniques of integration such as substitution, integration by parts, and methods of partial fractions were introduced. Hence the treatment of the second control group was primarily on specific functions, such as the logarithmic and exponential functions, and techniques of evaluating integrals of these two functions together with the more common polynomial and trigonometric functions.

Procedure

The study was conducted in the classes during regularly scheduled hours. Students in the programming and the mathematics sections were informed by the instructors that they were participating in a study but not informed of the nature of the study. Students were asked to attend all the classes. The Transfer Test (pre- and post-test) and the Achievement Test were administered by the instructors. For the programming sections, these tests were administered at the beginning of a laboratory period; for the mathematics sections at the beginning of a class period. The transfer tests were not announced to the students prior to

administration; the achievement test was. The pretest and posttest forms of the transfer test were counterbalanced in the experiment.

Administration of premeasure In the second week of instruction prior to the instruction on Pascal, subjects in four sections of programming course and two sections of mathematics course were given the written transfer test consisting of either Form A or Form B. The subjects completed the transfer test in one single laboratory or class period. The transfer test was 30 minutes. After the completion of the test, the instructor of each section continued with the normal activities of that day.

Administration of achievement test In the seventh week of instruction, the achievement test was administered only to the subjects in the experimental group. This test was given after the instructor finished with the nested loop in the sixth week. The subjects were informed of the achievement test one week prior to its administration. They were tested for their understanding and mastery of the nested loop construct. The test was 40 minutes in duration.

Administration of postmeasure For the postmeasure, the subjects in all three groups were given 30 minutes to complete the transfer test. The test was completed by the subjects in one single laboratory or class period. After the completion of the test, the instructor of each section continued with the normal activities of that day.

Experimental group In the eighth week of instruction, one week after the achievement test, the subjects were given the transfer test Form B and Form A respectively. During the ninth week, subjects who did not do well on the last item of the transfer posttest were given the same item again with a hint. The hint was in written form below the statement of the problem. It read as follows: *The nested loop construct you have studied may be useful in solving this problem.*

Control Group One In the fourth week of instruction, just prior to the students' entry into the instructional phase where the loop construct was covered, the subjects in the first control group were given the transfer test Form B and A respectively.

Control Group Two During the eighth week of instruction, subjects in the second control group were given the transfer test Form B and Form A respectively.

Scoring

Achievement Test To score the achievement test, part 1 was given a total of 6 points with 2 points for each problem and a total of 6 points for part 2 with 2 and 4 points for each problem respectively. Also, part 3 was given a total of 9 points with 2 and 7 points for each problem respectively and a total of 6 points for part 4 with 3 points for each problem. Thus, the total maximum score for the achievement test was 27 points. Partial credit was given depending on the degree of correctness of subject's answer. The degree of learning of the nested loop construct was given by the score of the subject on the achievement test. Table 3 contains excerpts of the answers and scoring protocol.

Insert Table 3 about here

Transfer Test The analysis of the solutions and the assignment of scores to the problems in Form A and Form B required a method for detecting the presence of the intersection procedure and the combination procedure as explained in the section on the loop construct. Each problem was given four points for a possible total of 12 points. The amount of partial credit given depended on the degree of manifestation of the intersection procedure or the combination procedure in student's solutions. There were numerous systematic or nonsystematic ways to list all the pairs or all the combinations, which yielded a complete listing of the possibilities. It should be emphasized that in this experiment one was looking for a solution which displayed the procedural manifestation of the nested loop construct, namely the intersection and combination schema, not just for any correct solution.

Intersection schema In scoring subject's answers on the transfer test, each of the two problems which embodied the intersection procedure was given a maximum of 4 points with a total score of 8 points. Both test forms A and B for the pretest and posttest were graded in the same manner. The subject who demonstrated clear understanding of the intersection procedure whose list of pairs coincided with the order of pairs as given in the procedural manifestation of the nested loop construct was given 4 points. The amount of partial credit given depended on the degree of manifestation of the intersection procedure. A correct sequence for problem 1 of Form A with names of John, Ann, Nancy, Peter, Paul, Tina, Robert and Mary is shown in Table 4.

Insert Table 4 about here

This is structurally isomorphic to the output from a nested loop of the type

```
for I := 1 to 8 do
    for J := I+1 to 8 do
        writeln (I, J);
```

The order of the people in the social introduction could vary from subject to subject. Subjects could work from different arrangements of the people, for example, Nancy, Peter, Paul, Tina, Robert, Mary, Ann, and John. As long as their sequence of pairs coincided with the procedural manifestation of the nested loop construct, the solution was scored as correct.

Combination schema The solutions involving the combination schema were graded in a similar fashion as the problems involving the intersection schema. Both forms A and B in the pretest and posttest were scored in the same manner. In scoring a subject's answers, the problem which embodied the combination procedure was given a total score of 4 points. A solution list of combinations coinciding with the order in the procedural manifestation of the nested loop construct was given the maximum points. In this case, the

subject clearly demonstrated understanding of the combination procedure. The amount of partial credit given depended on the degree of manifestation of the combination procedure. A correct sequence for problem 2 of Form A is

	(Salad Chicken Cake)	-> (Salad Chicken Pie)	-> (Salad Beef Cake)
->	(Salad Beef Pie)	-> (Salad Ham Cake)	-> (Salad Ham Pie)
->	(Soup Chicken Cake)	-> (Soup Chicken Pie)	-> (Soup Beef Cake)
->	(Soup Beef Pie)	-> (Soup Ham Cake)	-> (Soup Ham Pie)

which is structurally isomorphic to the output from a nested loop of the type:

```
for I:= 1 to 2 do
    for J:=1 to 3 do
        for K:= 1 to 2 do
            writeln (I,J,K);
```

with I, J and K indexing the elements of the sets {Salad, Soup}, {Chicken, Beef, Ham} and {Cake, Pie} respectively. The above sequence is a procedural manifestation of the output of the nested loop. All solutions which were permutations of the three sets {Salad, Soup}, {Chicken, Beef, Ham}, and {Cake, Pie} or permutations of the elements within each set were considered correct as long as procedurally a subject was using the combination schema from the nested loop construct. There are 6 possible permutations of the three sets, and in each permutation, there are 24 possible orderings of the elements. Hence there are altogether 144 different possible correct sequences a subject could give.

Results and Discussion

This section examines the effects of the three treatment conditions - "programming and instruction on the nested loop construct": experimental group versus "programming but

no instruction on the nested loop construct": control group one versus "no programming": control group two - on performance on the Transfer Test and the Achievement Test which measured understanding of and mastery of the nested loop construct.

To examine the initial section differences across groups, a one-way analysis of variance was performed on the scores of the transfer pretest. Results of the analysis showed no significant section differences ($F(5,125) = 1.91, p < .10$). As a result, the two sections in each group were combined.

A t-test was performed on the mean scores of the transfer pretest to examine possible transfer test form differences. The transfer pretest means with standard deviations for form A was 7.39 (4.53) and for form B was 7.77 (3.89). Results of a t-test revealed no significant differences between the mean scores of test form A and test form B ($t = -.51$, $p < .61$). Therefore the two test forms were treated as equivalent.

Transfer of Knowledge of Programming Construct to Solve Analogous Problems

The question of whether procedural thinking skills in programming constructs and activities transfer to analogous problem situations in other domains was examined. A one-way analysis of variance (ANOVA) was used to analyze the data . The dependent variable was the gain score (mean difference score between posttest and pretest) on the transfer test. The transfer pretest, posttest, and gain means and standard deviations for each group are shown in Table 5. Results of a one-way ANOVA showed a significant difference for the group ($F(2,128) = 5.49, p = < .005$). The subjects who learned the nested loop construct positively transferred the skill to analogous tasks in other domains.

Insert Table 5 about here

Results of a Tukey's HSD post hoc analysis indicated that subjects who were given instruction on the nested loop construct (experimental group) performed significantly better

on the transfer posttest relative to their performance on the transfer pretest than those who were taking programming but not given instruction on the nested loop construct (control group one) and those who were not taking programming (control group two). Results of a Tukey's HSD post hoc analysis also revealed that subjects in control group one did not perform significantly better than those in control group two. As can be seen in Table 5, the experimental group showed a substantial gain compared to control group one and control group two.

Since subjects were tested for two different schema, the intersection schema and the combination schema, in the transfer test and the result of an ANOVA showed a significant gain for the experimental group, a comparison between the mean scores on each schema for the experimental group was performed to examine where those significant gains were located in terms of these schema. For example, the significant gain could be due to significant gain from one schema only rather than from both. To resolve this, a t-test was performed. For the experimental group, the transfer pretest means and posttest means with standard deviations on the intersection schema are 5.31 (2.90) and 7.0 (1.70) respectively and on the combination schema are 2.31 (1.92) and 3.14 (1.57) respectively. The comparison between the mean scores on the transfer pretest and posttest for the intersection schema and the combination schema showed significant mean differences ($t = 3.76, p < .001$) and ($t = 3.24, p < .002$) respectively.

The above reported the effects of the knowledge of nested loop construct on the transfer test. When comparing the performance of the subjects who had received instruction on the nested loop construct to those who had not, the experimental group outperformed both control groups and showed a significant gain, thus showing a positive transfer of the procedural thinking skills in a programming construct to analogous problem situations in other domains. On the other hand, those in control groups who did not have any instruction on the nested loop construct showed no significant improvement. From the analysis of performance on the items of intersection schema and combination schema,

subjects showed a significant improvement on both schema after learning the nested loop construct.

Effects of Degree of Learning on Degree of Transfer

The question of whether a greater degree of learning of the nested loop construct would lead to a greater degree of transfer was examined. The subjects' scores on the achievement test were summarized in several ways for use in subsequent analyses: subscores on subject's ability to predict the outcome from the given segment of code; subject's ability to write correct Pascal code involving the loop construct to generate given outputs; subscores on the problems which embodied the intersection schema; subscores on the problems which embodied the combination schema; subscores on the problems which dealt with the single loop; and subscores on the problems which dealt with the mastery loop.

The achievement test means, standard deviations, and maximum points for each entry for the experimental group are shown in Table 6. First, to examine the relationship

---

Insert Table 6 about here

---

between performance on the transfer pretest and on the achievement test, a multiple regression analysis was performed. The transfer pretest score accounted for a significant portion of variance in achievement test scores, $R^2 = .27$, $F(1,40) = 14.54$, $p < .001$. The correlation between the transfer pretest and the achievement test was .52.

To examine the effect of degrees of learning on transfer, a multiple regression analysis was performed on the transfer posttest by entering the transfer pretest scores first and then the scores on the achievement test to control the performance on the transfer pretest. Results of the analysis revealed that the transfer pretest accounted for a significant portion of variance, $R^2 = .30$, $F(1,40) = 16.82$, $p < .0002$. The achievement test also

accounted for a significant portion of additional variance, change in $R^2 = .17$, $F(2,39) = 17.33$, $p < .0001$, Beta = .49.

Since the results of the multiple regression analysis revealed a significant effect of achievement of the nested loop construct on the transfer posttest, several multiple regression analyses were performed to determine the predictability of the achievement test subscores on the transfer posttest. A multiple regression analysis was performed to examine the predictability of subscores on the ability to generate the correct output and to write the correct code involving the loop construct with given outputs on the transfer posttest. The scores on the transfer pretest were entered first and then the subscores on the ability to generate the output and to write the correct code. Results of the analysis revealed that the transfer pretest accounted for a significant portion of variance, $R^2 = .30$, $F(1,40) = 16.82$, $p < 0.0002$. Only the subject's ability to write correct code involving the loop construct accounted for a significant portion of additional variance, change in $R^2 = .17$, $F(3,38) = 11.21$, $p < 0.0001$, Beta = .41. The subject's ability to predict the output did not show a significant effect ($t = 1.21$, $p < .23$).

A multiple regression analysis was performed on the transfer posttest using the subscores of the problems in the achievement test which embodied the intersection schema and the combination schema. The scores on the transfer pretest were entered first and then the subscores on the intersection schema and the combination schema. Results of the analysis revealed that the transfer pretest accounted for a significant portion of variance, $R^2 = .30$, $F(1,40) = 16.82$, $p < 0.0002$. Only the subject's score on the intersection schema accounted for a significant portion of additional variance, change in $R^2 = .23$, $F(3,38) = 13.40$, $p < 0.0001$, Beta = .43. The subject's score on the combination schema did not show a significant effect ($t = 1.57$, $p < .12$).

The subscores of problems in the achievement test which dealt with the single loop and mastery loop were used in an analysis to determine the predictability of the achievement of the nested loop construct on the transfer posttest. The scores on the transfer pretest were

entered first and then the subscores on the single loop and the mastery loop. Results of the analysis revealed that the transfer pretest accounted for a significant portion of variance, $R^2 = .30$, $F(1,40) = 16.82$, $p < 0.0002$. The basic loop did not accounted for a significant portion of additional variance ($t = 1.11$, $p < .28$). The mastery loop also did not show a significant effect ($t = 1.10$, $p < .28$). Correlations and partial correlations between the transfer posttest and the predictors are shown in Table 7

Insert Table 7 about here

The above analyses examined the effects of performance on the achievement test, which measured the understanding and the mastery of the nested loop construct, on performance on the transfer posttest. The results showed that subjects who scored higher on the achievement test scored significantly higher on the transfer posttest than those who scored lower on the achievement test. This finding indicates that the degree of learning of the nested loop construct was positively related to the degree of transfer to analogous problems in other domains. The finding also revealed that the subjects who scored higher on the problems in which they were required to write correct codes involving the nested loop construct performed significantly better on the transfer posttest than those who scored lower on these problems. However, the problems asking the subjects to predict the output of a given code segment did not indicate a similar effect on the transfer posttest. This is consistent with the claim of Fay and Mayer (1988) that the ability to write a program is a more complex and demanding skill and requires knowledge of both the syntax and the semantics of a programming construct. Subjects with these skills were more likely to transfer their skills to solving problems in other domains. To be able to produce the output of a given code segment may not require one to have as complete an understanding of the code, thus not necessarily resulting in a positive transfer of the skill.

Results also revealed that the subjects who scored higher on the problems involving the intersection schema in the achievement test scored significantly higher on the transfer posttest than those who scored lower on these problems. This finding indicated that the degree of learning of the nested loop construct which embodied the intersection schema was positively related to the performance of solving analogous problems in the transfer posttest. However, the problems which embodied the combination schema did not show a similar effect. This could be due to the fact that most of the subjects learned the nested loop construct embodying the combination schema to a high degree (mean 6.24 out of a maximum score of 7 in the achievement test).

The achievement on single loop problems did not show a positive effect on the transfer posttest. Subjects who scored higher on the single loop problems did not score significantly higher on the transfer posttest than those who scored lower on these problems. The single loop problems were easy for many of the subjects. Twelve subjects achieved maximum scores, and another 11 had almost perfect scores (3.5 out of 4) on both of the single loop problems. In addition, none of the single loop problems embodied the intersection schema or the combination schema. This provides a plausible explanation as to why performance on the single loop problems was not positively related to the performance on the transfer posttest. Also, the effect of the achievement on the mastery loop problem did not show a positive effect on the transfer posttest. Subjects who scored higher on the mastery loop problem did not score significantly higher on the transfer posttest than those who scored lower on this problem. The mastery loop problem was too difficult for almost all the subjects. Only two subjects managed to solve this problem. Most subjects scored very low on this problem. The mastery loop problem required not only understanding of the nested loop construct, but also mathematical characterization of the loop indices, which turned out to be too difficult for almost all the subjects. As a result, the performance of the subjects on this problem did not show a positive effect.

Effects of Hint on Transfer

This section examines whether subjects who learned the nested loop construct but did not perform well on the transfer posttest would show improvement after receiving a hint. Those subjects (N = 20) who did not perform well on the last item of the transfer posttest which embodied the intersection schema were given the same item again with the hint that the learned nested loop construct might be useful in solving the problem. The transfer posttest means with standard deviations before the hint and after the hint are 1.15 (1.39) and 3.90 (0.31) respectively. The comparison between the mean scores before and after the hint showed significant mean differences ($t = 8.50$, $p < .0001$). After receiving the hint, subjects showed a significant improvement on that same problem compared to their performance without the hint.

Conclusion

This study examined the cognitive effects of learning to program in a procedural language by restricting it to the micro-level of a specific programming construct. The approach was different from the methodology generally employed in current research into the cognitive benefits of learning to program. The transfer effect of an important programming construct to isomorphic problems in other domains was studied, rather than the cognitive benefits of an entire course of instruction as measured by some general tests, whose relation to the materials learned might not be entirely clear, or difficult to ascertain.

The transfer of the programming skills from the nested loop to analogous problems in other domains was studied. It investigated whether a greater degree of learning of the skills in programming would lead to a greater degree of transfer to analogous problems in other domains. Finally, it investigated whether a greater degree of transfer would result upon the receipt of a hint.

The mastery of the programming construct was measured by a carefully constructed achievement test. The cognitive benefit was measured by the subject's ability to solve analogous problems in other domains. The procedural manifestation of the nested loop construct, which embodied the intersection or the combination schema of Piaget and Inhelder, was used as a tracer in the solutions of the subjects in the transfer task. In other words, among all the correct solutions to the transfer tasks, only those which displayed the procedural manifestation of the schema embodied in the nested loop construct received credit. This design ensured that what was transferred was indeed what was learned.

The results of this experiment were of significance and interest. Significant transfer was found for those who received instruction on programming and the nested loop construct (experimental group). Indeed, several of the correct solutions to the transfer tasks consisted of Pascal programs which clearly showed that the subjects knew that the nested loop construct could be used to enumerate the pairs to provide a solution to the transfer tasks. No transfer was found for those who did not receive instruction on the nested loop construct.

These results, with the procedural manifestation of the nested loop construct as a tracer in the solutions of the subjects to the transfer problems, clearly established the direct relationship between what was learned and what was transferred. Past studies had failed to establish a direct relationship, with specificity, between what was learned and what was transferred as the learning came from an entire course of instruction. In the very few studies where attempts were made, there were still questions about either the transfer task or the materials learned. In the only study (Seidman,1989) where programming constructs (*if-then* and *if-then-else*) were studied, the study did not relate the construct to analogous problems in other domains. It investigated the cognitive effects of learning the construct on the knowledge of the ordinary conditional, which was not semantically equivalent to the programming construct. The present study was the first in the area of cognitive benefits of learning to program where analogous problems were identified at the programming

construct level in the learning situation and in the testing situation for cognitive transfer. This direct relationship between the learning tasks and transfer tasks provided a more definitive answer to questions about positive cognitive benefits of learning to program.

Furthermore, the degree of learning was found to be positively related to the degree of transfer. This agrees with the findings of Khlar and Carver studies(1988) on the transfer of debugging skills. In other studies where learning was assessed, the test was over a broad spectrum of programming skills, and hence the exact degree of learning in the programming construct or constructs which bore a direct relation to the transfer tasks was not determined as it was not possible to factor out their contributions.

The achievement test in this study, which measured understanding of the loop construct, was carefully constructed to consist of problems to measure three types of learning. The first type of learning consisted of asking for the output given a code segment. Correct solution of these problems gave an indication that the subjects understood the semantics of the nested loops construct. The second type of learning consisted of writing Pascal code given a desired output. This required not only an understanding of the semantics of the loop construct, but also a competent skill level of writing code in the given programming language. The third type of learning consisted of mastery of the nested loop construct.

The results indicated that the type 2 problems predicted the performance on the transfer posttest whereas type 1 problems did not. This finding implied that subjects who understood the semantics of the nested loop construct and were capable of writing code were able to transfer whereas understanding the semantics alone was not necessarily sufficient for transfer. The type 3 problem was too difficult for almost all the subjects. Only two out of 42 subjects solved this problem correctly. These two subjects obtained the highest scores in the achievement test, and also obtained perfect scores on the transfer tasks. However, twenty-two other subjects also obtained perfect scores. This suggests, that in a transfer situation, being an "expert" in one domain beyond a certain competence

level may not provide additional advantage in solving transfer problems in other domains. This could be accounted for by observing that in order to solve the transfer tasks, the critical ingredient was the abstraction of the intersection and combination schema by the subjects while learning the nested loop construct, and the ability to map these schemata onto analogous problems when confronted with the transfer tasks. Exceptional skills in programming and a deep understanding of the nested loop construct as exemplified by the two subjects who solved the type 3 problem were not necessary. However, one should be cautious about the preceding observation because only two subjects were able to solve the type three problems. A future study is needed to investigate this.

Finally, the positive effect of a hint was established With the hint, the mean score of the subjects was 3.90 out of 4, compared with 1.15 out of 4 before the hint. Only two subjects scored less than 4. These two subjects scored 3 which indicated that a very minor error was made. Twenty-two out of the 42 subjects scored 4 out of 4 on the transfer posttest without a hint. Hence it could be concluded with certainty that all subjects had learned the intersection schema from their study on the nested loop construct. This further strengthened the fact that the inability of a subject to transfer could not be ascribed to the possibility that the subject had not learned the underlying intersection schema. The failure to transfer could be due to the subjects not connecting the transfer situation to the learning situation.

Implication of the Approach in this Study

Several points could be noted about the present study. It is focused on a central programming construct, the nested loop construct, which is used in numerous important algorithms. This construct is covered in all courses on programming and used extensively in many programming solutions to computational problems in other domains. Hence this construct's cognitive benefits contribute to the overall cognitive benefits in learning to program.

Another point is its specificity in that it is focused on a specific segment of the instructional programming curriculum. It is not unusual that in other subject areas, such as mathematics or physics, only a particular segment of its curriculum is investigated. For example, the interdomain transfer of learning arithmetic progression in a high-school algebra class to isomorphic problems in physics (Bassok & Holyoak, 1989) was studied rather than the entire algebra course. However, in the area of learning to program, almost all the research studies had been on an entire course of instruction. Therefore, the relation of what was learned to what was transferred had never been delineated clearly. Most of the studies therefore found either no cognitive benefits or only benefits of a generalized nature. Even when positive cognitive benefits were found, it was not clear as to their origin; that is, the benefits could be due to the instructional method, the computer environment, the programming language, the underlying programming constructs or even sociological or psychological factors. By dealing with one construct, it was possible to design the experiment to carefully rule out extraneous factors which might have contributed to the transfer.

This study is replicable since the use of a well-defined tracer in determining the degree of transfer allows other investigator to replicate and verify the results. The materials and the method of instruction can also be easily replicated.

It should also be noted that this study is extendible. By modifying the learning environment, the instructional materials, or the teaching style, future studies can be constructed on top of this precise framework to answer questions of cognitive transfer of learning programming under different sets of conditions and of learning different programming constructs.

There is a limitation to the present study. In a study of cognitive benefits, the degree of difficulty of the transfer problems has an impact on the result. For example, in a study by Pea and Kurland (1984) where the transfer tasks involved a number of complex skills, the subjects failed to show any significant transfer even after two years of study on

programming. The solution to the transfer problem on the combination schema in this study was isomorphic to the triple nested loop. These transfer problems turned out to be relatively easy for most subjects. It suggests that analogous problems on the combination schema embedded in more complex contexts should be used in future studies.

This study holds a few implication about teaching programming. One of the most important goals of education is to produce students who are critical thinkers and good problem-solvers, and students who can transfer what they learn in the classroom to solve problems in society. This study confirmed that learning to program resulted in cognitive benefits to the students. The students were able to transfer their programming skills to solve analogous problems in other domains.

One of the results was the success of all the subjects in transferring what they learned to analogous tasks upon receipt of a hint. The failure to transfer was due to the inability of some of the subjects to see the analogous nature of the transfer tasks to the nested loop construct, and the failure to abstract the intersection schema from the instructional examples during the learning phase. Since those who failed to transfer could transfer with a hint, it suggested that it is important to teach metacognitive strategies and applicability of programming knowledge to other domains while teaching programming. When students encounter new situations, they can monitor their own problem-solving strategies, and ask whether knowledge from programming could apply.

Furthermore, subjects who learned not only the semantics but also learned to write code were able to transfer. One of the perennial debates of the educational computing community is about the importance of writing code in a programming course. It seems that the results of this experiment strongly suggest that in order to derive cognitive benefits from learning to program, students should become also proficient in writing codes.

In conclusion, this study set out to examine the cognitive benefits of learning to program by focusing on an important programming construct. In adopting a fine grain approach at the programming construct level and the use of a tracer, the cognitive effects of

learning to program were determined with specificity unattainable before. The results showed that there was cognitive transfer, and these results could be accepted with greater confidence than in any previous studies. Furthermore, factors which impacted on the outcome were controlled to a degree hitherto unobtainable. This study demonstrated the advisability of this approach, and the methodology can be used in future studies on the cognitive benefits of learning to program and also in studies on procedural knowledge in other domains.

REFERENCES

Boehm, C. & Jacopini, G. (1966). Flow diagrams, Turing machines, and languages with only two formation rules. *Communication of the Association of Computing Machinery, 9*, 5, 366-371.

Bork, A. (1981). *Learning with Computers*. Bedford, MA: Digital Press.

Bassok, M. & Holyoak, K. J. (1989). Interdomain transfer between isomorphic topics in algebra and physics. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 15* (1), 153-166.

Carver, S.M. (1988). Learning and transfer of debugging skills: Applying task analysis to curriculum design and assessment. In R. E. Mayer (Ed.). *Teaching and learning computer programming: Multiple research perspectives* (pp. 259-298). Hillsdale: Lawrence Erlbaum Associates, Publishers.

Clements, A. C., Kurland, D. M., Mawby, R., & Pea R. D. (1986). Analogical reasoning and computer programming. *Journal of.Educational Computing Research, 2* (4), 473-486.

Clements, D. H. (1985). Research on LOGO in education: Is the turtle slow but steady, or nor even in the race? *Computers in the Schools, 2* (2/3), 55-71.

Clements, D. H. (1986). Effects of LOGO and CAI environments on cognition and creativity. *Journal of Educational Psychology, 78*, 309-318.

Clements, D. H. (1990). Metacomponential development in a LOGO programming environment. *Journal of Educational Psychology, 82* (1), 141-149.

Clements, D. H. (1991). Enhancement of creativity in computer environments. *American Educational Research Journal, 28* (1), 173-187.

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

Dyck, J. L., & Mayer, R. E. (1989). Teaching for transfer of computer program comprehension. *Journal of Educational Psychology, 81*(1), 16-24.

Ellis, H. C. (1965). *The transfer of learning*. New York: Macmillan.

Fay, A. L. & Mayer, R. E. (1988). Learning LOGO: A cognitive analysis. In R. E. Mayer (Ed.). *Teaching and learning computer programming: Multiple research perspectives* (pp. 54-74). Hillsdale: Lawrence Erlbaum Associates, Publishers.

Gick, L. M., & Holyoak, K. J. (1980). Analogical solving. *Cognitive Psychology, 12,* 306-355.

Gick, L. M., & Holyoak, K. J. (1983). Schema induction and analogical transfer. *Cognitive Psychology, 15*, 1-38.

Hines, S. N. (1983, July-August). Computer programming abilities of five-year-old children. *Educational Computer*, 10-12.

Holyoak, K. J. (1985). The pragmatics of analogical transfer. In G. H. Bower (Ed.), *The psychology of learning and motivation* (Vol. 19, pp. 59-87). New York: Academic Press.

Howe, J. A. M., O'Shea, T. and Plane, F. (1980). Teaching mathematics through Logo programming: An evaluation study. In R. Lewis and E. D. Tagg (Eds.) *Computer assisted learning: Scope, progress and limits* (pp. 85-102). New York: North-Holland Publishing Company.

Inhelder, B., & Piaget, J.(1969). In E. A. Lunzer & D. Papert (Trans.), *The early growth of logic in the child: Classification and seriation*. New York:Norton.

Klahr, D., & Carver, M. (1988) Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology, 20,* 362-404.

Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school. *Journal of Educational Computing Research, 2* (4), 429-458.

McGrath, D. (1988). Programming and problem solving: Will two languages do it? *Journal of Educational Computing Research, 4* (4), 467-484.

Miller, R. B., Kelly, G. N., & Kelly, J. T. (1988). Effects of LOGO computer programming experience on problem solving and spatial relations ability. *Contemporary Educational Psychology, 13*, 349-357

Milojkovic, J. D. (1984). *Children learning computer programming: Cognitive and motivational consequences.* Unpublished doctoral dissertation, Stanford University.

Nastasi, B. K., Clements, D. H., & Battista, M. T. (1990). Social-cognitive interactions, motivation, and cognitive growth in LOGO programming and CAI problem-solving environments. *Journal of Educational Psychology, 82* (1), 150-158.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* New York: Basic books.

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

Seidman, R. H. (1981, April). *The effects of learning a computer programming language on the logical reasoning of school children.* Paper presented at the Annual Meeting of the American Education Research Association, Los Angeles, CA.

Seidman, R. H. (1989). Computer programming and logical reasoning: Unintended cognitive effects. *Journal of Educational Technology Systems, 18* (2) , 123-141.

Soloway, E., Lochhead, J., & Clement, J. (1982). Does computer programming enhance problem solving ability? Some positive evidence on algebra word problems. In R. Seidel, R. Anderson, & B. Hunter (Eds.), *Computer Literacy.* New York: Academic Press.

Sternberg, R. J. (1985). *Beyond IQ: A triarchic theory of human intelligence.* Cambridge, MA: Cambridge University Press.

Table 1
Problems on the Transfer Test

Form A	Form B
1. You are the host of an evening event where you have brought together people who will work with you on a charity project. They have never met each other before. You would like all of them to get to know each other by introducing them on an individual basis, one person to another. Please write down the list of pairs to be introduced in the order that they occur to you. These people go by the names of John, Ann Nancy, Peter, Paul, Tina, Robert, and Mary.	1. You are the coordinator of a charity tennis event where you have brought together some of the best tennis players. To maximize the charity contribution, you want to have all of them play against each other. Your task is to produce a list such that each player would play against the rest of them in single matches. Please write down the list of pairings in the order that they occur to you. These players go by the names of Boris, Stefan, Ivan, Andre, Jimmy, Matts, Steven, and John.
2. You find yourself in a restaurant which caters to the wishes of the customers by offering them some freedom in designing their own dinner. You are to select one item from each category to form your own meal. Please write down the list of the possible combinations of items from each category in the order that they occur to you. The items in each category are listed below: Items: First Category: Salad or Soup Second Category: Chicken, Beef or Ham Third Category: Cake or Pie	2. You are the stage lightning manager for a horror play. Three color guns are available to you to provide the lightning of the different scenes. All three guns would be focused on the "monster" when it is on stage. You have to show the director how the colors combine. Please write down the list of the possible combinations of colors you have to try by selecting a color from each gun. The colors from each gun are listed below: Colors First Gun: Red or Blue Second Gun: Green, Purple or Yellow Third Gun: Brown or Orange
3. You are the chief chemist of a project to analyze how two different chemicals interact with each other to produce colors. The chemicals to be studied are the following: Sodium Hydroxide (NaOH), Hydrochloric Acid (HCl), Ferric Chloride ($FeCl_3$), Sodium Carbonate (Na_2CO_3), Alizarin ($C_{14}H_8O_4$), Phenolphthalein ($C_{20}H_{14}O_4$), Thymol Blue ($C_{27}H_{30}O_5S$), and 5-Nitrosalicylic ($C_7H_5NO_5$). You need to obtain the resulting colors by mixing the given chemicals, two at a time. Please write down the list of pairs of chemicals in the order that they occur to you. Your assistant chemists will conduct the actual testing according to the list you produce.	3. You are the chief chemist of a project designed to discover which two of various chemicals would produce an explosive reaction when combined together. You need to obtain the possible interactions to make sure that the ones which lead to explosive reactions will be detected. The chemicals to be studied are the following: Potassium Metal (K), Chloric Acid ($HClO_3$), Lithium Aluminum Hydride ($LiAlH_4$), Potassium Permanganate ($KMnO_4$), Water (H_2O), Sucrose ($C_{12}H_{22}O_{12}$), Glycerol ($C_3H_8O_3$), and Sulfuric Acid (H_2SO_4). Please write down the sequence of pairs of chemicals in the order that they occur to you so that your assistant chemists will conduct the tests.

Table 2

Excerpts of Representative Problems from Achievement Test on Loop Construct

1. What is the output of each of the following segments of code?

(a)
```
for K := 1 to 5 do
    begin
        for J := K+1 to 5 do
            write (K,J);
        writeln
    end;
```

(b)
```
for I := 1 to 2 do
    for J := 3 to 4 do
        for K := 1 to 3 do
            begin
                write (I,J,K);
            writeln
            end;
```

2. Write a Pascal nested loop code segment to generate on the screen Output on the screen is :

```
|     *
|   * * *
| * * * * *
|* * * * * * *
```

where | denotes the left edge of the screen

3. Write code to generate the following output on the screen:

```
1 6
1 7
1 8
1 9
2 7
2 8
2 9
3 8
3 9
4 9
```

Table 3

Excerpts of the Answers and Scoring Protocol of the Achievement Test

Write a Pascal nested loop code segment to generate the output on the screen as specified.
Output on the screen is:

```
|*
|**
|***
|****
|*****
```

where | denotes the left edge of the screen

Correct Answer:	*Points:*
for *K* := 1 **to** 5 **do**	**1**
begin	
for *J* := 1 **to** K **do**	**2**
write ('*');	**1/2**
writeln;	**1/2**
end;	

This problem is given a total of 4 points. The amount of partial credit depends on the degree of correctness of the subject's answer. If subject makes one or more of the mistakes described below, the amount of points underlined is deducted accordingly.

1/2 point:
a. The subject uses an incorrect value for the initial or the final value of the counter variable in the outer loop.
b. The subject uses an incorrect value for the initial value of counter variable in the inner loop.
c. The subject omits either the *write* or the *writeln* statement in his/her program.
d. The subject doesn't complete the *for* statement with correct syntax.

1 point: The subject uses an incorrect value for the initial and the final value of the counter variable in the outer loop.

1 1/2 points: The subject uses an incorrect expression for the final value of counter variable in the inner loop.

Table 4

An Example of a Solution Based on the Intersection Schema with Names of John, Ann, Nancy, Peter, Paul, Tina, Robert, and Mary

John/Ann -->	John/Nan -->	John/Peter-->	John/Paul -->	John/Tina -->	John/Rob -->	John/Mary
-->	Ann/Nan -->	Ann/Peter -->	Ann/Paul -->	Ann/Tina -->	Ann/Rob -->	Ann/Mary
-->	Nan/Peter -->	Nan/Paul -->	Nan/Tina -->	Nan/Rob -->	Nan/Mary	
-->	Peter/Paul-->	Peter/Tina-->	Peter/Rob -->	Peter/Mary		
-->	Paul/Tina -->	Paul/Rob -->	Paul/Mary			
-->	Tina/Rob -->	Tina/Mary				
-->	Rob/Mary					

Table 5

Pretest and Posttest Means and Standard Deviations for the Transfer Test

	Experimental		Control One		Control Two	
Group	M (SD)	n	M (SD)	n	M (SD)	n
Pretest	7.60 (3.86)	42	7.96 (4.28)	51	7.03 (4.55)	38
Posttest	10.17 (2.53)	42	8.94 (3.48)	51	7.47 (4.68)	38
Gain	2.57 (3.27)	42	.98 (2.63)	51	.45 (3.24)	38

Note: Maximum point is 12.

Table 6

Means and Standard Deviations for the Achievement Test

	Experimental Group		
	M (SD)	Maximum Points	n
Achievement Test	16.20 (4.38)	27	42
Predict Output	4.82 (1.27)	6	42
Write Code	11.66 (3.82)	21	42
Intersection Schema	6.31 (1.96)	9	42
Combination Schema	6.24 (1.15)	7	42
Single Loop	2.58 (1.29)	4	42
Mastery Loop	1.34 (1.66)	7	42

Table 7

Intercorrelations and Partial Correlations Between the Transfer Posttest and the Predictors

	Predictors	Correlations with Transfer Posttest	Partial Correlations Controlling for Other Variables in the Set
Set 1	Transfer Pretest	.54**	.25
	Achievement Test	.64**	.42
Set 2	Transfer Pretest	.54**	.34
	Predict Output	.35*	.19
	Write Code	.61**	.43
Set 3	Transfer Pretest	.54**	.35
	Intersection Schema	.62**	.48
	Combination Schema	.42**	.25
Set 4	Transfer Pretest	.54**	.44
	Basic Loop	.39*	.18
	Mastery Loop	.35*	.18

* $p < .05$. ** $p < .01$.