ED 290 459 IR 013 115

AUTHOR Nachmias, Rafi; And Others
TITLE Variables--An Obstaclo to Children Learning Computer Programming. Technical Report No. 8.
INSTITUTION Tel-Aviv Univ. (Israel). Computers in Education Research Lab.
PUB DATE Jul 86
NOTE 23p.; Paper presented at the Annual Meeting of the American Educational Research Association (San Francisco, CA, April 16-20, 1986). For related reports, see IR 013 108-114.
PUB TYPE Reports - Research/Technical (143) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Abstract Reasoning; Age Differences; Comparative Analysis; Computer Literacy; *Concept Formation; Difficulty Level; Foreign Countries; Intermediate Grades; *Microcomputers; *Preadolescents; Problems; *Programing; Psychological Studies
IDENTIFIERS *Israel; *Variables

ABSTRACT

The difficulties that younger students experience in understanding concepts related to the use of variables in computer programming are examined through descriptions of two studies: (1) detailed case studies of six highly intelligent children--three fourth graders and three sixth graders--who learned to program in BASIC during 60 hours of instruction under the careful observation of research staff; and (2) a study in a regular classroom environment with 73 children in the fourth and sixth grades who had 12 two-hour hands-on lessons in programming. The programming textbook used in both studies taught the use of variables in the second half of the course, thus permitting comparison of the acquisition of programming concepts related to variables and the acquisition of concepts that do not involve variables. The results of the first study indicated that the fourth graders, in contrast to the sixth graders, were unable to learn the concepts associated with variables, even though they had understood the concepts without variables. The results of the second study indicated that the inability to understand concepts associated with variables was related more to the level of academic achievement than to grade level. It is suggested that four factors may explain the difficulties younger students experience in learning to use variables: (1) the level of abstraction in using variables; (2) the dynamic nature of the values of variables; (3) the degree of complexity in using variables; and (4) the level of reasoning required. Measures that may help to overcome the influence of these factors are suggested, and a 25-item bibliography is provided. (EW)

THE COMPUTERS IN EDUCATION RESEARCH LAB.

המעבדה לחקר יישומי מחשבים בחינוך

VARIABLES - AN OBSTACLE TO CHILDREN LEARNING COMPUTER PROGRAMMING

Rafi Nachmias, David Mioduser, David Chen

Tecnical Report No. 8

July 1986

TEL AVIV UNIVERSITY SCHOOL OF EDUCATION

בית הספר לחינוך אוניברסיטת תל אביב

UNIT FOR COMMUNICATION & COMPUTER RESEARCH IN EDUCATION

היחידה לחקר תקשוב בחינוך

THE COMPUTERS IN EDUCATION RESEARCH LAB.   המעבדה לחקר יישומי מחשבים בחינוך

VARIABLES - AN OBSTACLE TO CHILDREN LEARNING COMPUTER PROGRAMMING

Rafi Nachmias, David Mioduser, David Chen

Tecnical Report No. 8

July 1986

School of Education Tel Aviv Univercity Tel Aviv 69978 Israel

'Variables' - An Obstacle To Children Learning Programming

Rafi Nachmias[1], David Mioduser, David Chen

School of Education, Tel-Aviv University, Israel.

(1) also the Lawrence Hall of Science, University of California, Berkeley.

This paper was presented at the Annual Meeting of the American Educational Research Association April 19, 1986.

This paper was written while the first author was a Fulbright scholar at the University of California at Berkeley. Support from the United State-Israel Education Foundation is appreciated. The authors wish to thank John Nides and Elana Rienin for their help.
Requests for reprints should be sent to Rafi Nachmias, LH-22, Lawrence Hall of Science, University of California, Berkeley, CA 94720.

INTRODUCTION

As microcomputers become increasingly popular in both homes and schools in Israel, a growing number of students are engaged in introductory programming courses, in formal and informal education systems. Many of these new programmers are young children in elementary school who are learning to manipulate the programming environment as well as write, modify, and run simple programs. When viewing an exemplary performance of some of these young students, many people rush to the conclusion that young children can learn programming language very easily. Such a conclusion is probably inaccurate; there is cumulative evidence from recent research that learning programming language is neither as easy nor natural as it was thought to be (Kurland, Mawby, Cahir, 1984; Dalbey, Linn1985). Many students encounter difficulties in understanding and applying basic programming concepts (Bonar 1984: Soloway et al. 1982; Perkins 1985; Sleeman 1984).

Computer programmings is a rather complex activity which requires a wide range of skills and knowledge (Shiel, 1980). Difficulties encountered by novices may stem from the need to simultaneously use a large number of basic skills. Some of these skills already exist in the repertoire of the learner, while others must still be learned.

Some descriptive models have been proposed as tools for classifying and organizing these skills into a conceptual framework. Shneiderman (1980) differentiates between syntactic knowledge and semantic knowledge. He suggest that these two kinds of knowledge be organized in a spiral form (Shneiderman, 1977). Mayer (1979) proposes a subdivision of the BASIC language into 8 levels which begins with a transaction level and concludes to a computer program level.

Nachmias, Mioduser, and Chen (1984) suggest that the skills and basic concepts required of the beginning programmer can be divided into five domains of knowledge:

1) **Handling the machine.** This includes technical skills connected with the use of a computer system and the understanding of the information flow that takes place between its units when a computer program is executed;

2) **Mastering the programming language.** This includes learning the syntax and the semantics of the programming language.
3) **Perceiving the computer program structure.** This includes the order of execution of the statements (flow of control), different types of program structures and the capability of algorithmic and procedural thinking;
4) **Comprehending the logical dynamics of the computer program.** In particular, this includes the ability to follow how the values of the variables change when a program is being executed, and the ability to manipulate and apply logical rules to variables;
5) **Knowledge specific to the task.** In order to write a program, specific knowledge is needed about the problem which the program is being intended to solve.

It has been suggested that two types of learning are involved in the acquisition of programming language. The first type concerns learning within each of the above knowledge domains, and the second concerns integration of these various domains.

This paper focuses on the first type of learning of concepts within the fourth domain which are related to variables. The ability to use variables is a key step in the development of mathematical and scientific thinking in children; consequently learning to use variables is one of the main concerns of mathematics and science education. Understanding the concept of variables is also central to computer programming as a problem solving tool. Though it is possible to write simple programs without variables, in order to realize the full potential, power, and beauty of programming, the learner needs to use variables extensively in computer programs.

The findings of this paper are based upon two studies in which elementary school students learned to use variables in a computer programming course. These studies part of a larger study on the acquisition of several basic programming concepts by novice learners (Nachmias, 1985), were carried out in the Computers in Education Research Lab at Tel-Aviv University's School of Education. The participants in the studies were fourth grade and sixth grade students (ages 10-12). The first study was carried out under laboratory conditions; the second was done in a regular classroom environment.

The objectives of this paper are:

- To suggest that the understanding of concepts in the fourth domain of knowledge that relates to variables, is one of the sources of difficulties in novices learning programming.

- To point out through a comparison of the achievements of fourth and sixth graders, that children do not understand these concepts pertaining to variables before the sixth grade.

- To describe some difficulties young students encounter when they learn to use variables in programming, and to suggest factors that can cause or increase these difficulties.

METHOD

DESCRIPTION OF THE STUDIES

Conclusions of this paper are based upon the findings of two studies:
The first study consisted of detailed case studies of six highly intelligent children (three fourth graders and three sixth graders) who learned to program in BASIC during a period of four months (about 60 hours of instruction). The students studied programming individually using the learning materials developed, and interaction between them was encouraged. they were guided by research staff who observed their work, provided help when necessary. and directed the students to activities according to their level of understanding and ability. Detailed observation of the students' learning processes and difficulties were recorded throughout the course. A comprehensive final test was administered and final projects of students were documented and saved. Because of the high aptitude of the students, the one-to-one student to teacher ratio, and the unlimited access to computers, the first study is considered to have been conducted under ideal conditions.

The second study was conducted in a regular classroom environment in two fourth grade classes and two sixth grade classes in an upper middle class elementary school in Tel-Aviv. A preliminary total of 11 students participated. A questionnaire was used initially to exclude any students who had previous knowledge of programming. The final number of subjects participating in the second study was 73. Forty-five students were in the fourth grade and 28 in the sixth grade; 31 were boys and 42 were girls. The instruction included 12 two hour hands-on lessons, and there was an average of one computer per two students. The instructors were the authors of this paper; the textbook used was the same as the one used in the first study. During the course of the study, five tests were administered. At the conclusion of the course, a final test in the form of an interview, using a computer was administered to each student.

Description of the data collection instruments used in the two studies, as well as the principles in developing the learning material can be found elsewhere (Nachmias, 1985; Nachmias, Mioduser, Chen, 1984; Nachmias, Mioduser, Chen, 1985).

THE INTRODUCTION OF VARIABLES IN THE COURSE

In the programming textbook used in the two studies (Nachmias, Mioduser, Chen, 1985a), students are not required to master the concept of variables too early in the course. Instead, in the first eight chapters of the textbook, concepts and activities that do not require a direct use of variables are introduced, such as graphics command, random variables, infinite loops, numerical calculation, and printing statements.

The concept of a random number as a number "chosen" by the computer was introduced quite early in the course to allow the teaching of loop constructs as well as to enable students to cope with more complex programs such as simple animation and probability games.

Teaching numerical calculation and print statements at the conclusion of the first part of the course enabled the students to integrate the concepts and skills from the first three domains of knowledge they had previously learned. It was intended that a solid foundation for the introduction of variables to young students be provided in the first part of the course.

In the second part of the course, students learned to use variables. The children were taught that a variable is a designated location in the computer memory which contains a value that can be manipulated while the program runs.

The following uses of variables in computer programs were gradually introduced to the students:

1) **Variables as parameters in action commands** (e.g. PLOT X,Y to draw a point at a certain position on the screen). Students initially used variables just to store constant numbers. Later students learned to manipulate values assigned to variables.

2) **Using variables to store the values of mathematical expression** (e.g. DISTANCE=VELOCITY*TIME). Students in the fourt grade in Israel are not familiar with the English language; consequently, very short variable names (e.g. X.Y. A$) were used. The mathematical expressions used in the course were kept simple, because of the young age of

the students. For example, students were introduced to a program that calculates the number of days, hours, minutes, and seconds in a given number of years.

3) **Using a variable as a counter** (e.g. N=N+1). A counter was added to a simple program, such as ones that simulate a coin tossing and that display drills on the screen.

4) **Using a variable as an operator in a logical expression.** in conditional statements (e.g. IF X>0, THEN...) Conditionals were only used in computer programs to execute a command only if a certain condition was met, or to stop infinite loops, or to branch programs.

Examples of these uses of variables were presented to students in short programs which they were encouraged to understand, run, test, and modify. The students were then asked to write programs carrying out similar tasks.

The way variable values are manipulated is of great importance in programming. Four methods for changing the value of a variable were presented to the students during the course:

a) Assigning a constant value to a variable using the assignment statement (e.g. X=30, A$="Shalom").
b) Assigning a random number to a variable (e.g. X=RND(1)*100).
c) Changing the value of a variable while the program runs by using the input statements (e.g. INPUT S, GET A$).
d) Incrementing the value of a variable. Changes could be either simple (e.g. X=X+1), complex (e.g. X=3*X+4), or consist of more than one variable (e.g. A=2*B+C).

The data types used in the course were integers, real numbers, and alphanumeric strings. More complex data structures such as array were beyond the scope of this course.

RESULTS AND INTERPRETATION

COMPREHENSION OF VARIABLE RELATED CONCEPTS

By dividing the teaching of programming intc two parts, with and without the use of variables, it was possible to compare the acquisition of programming concepts related to variables to concepts which did not depend upon the use of variables.

Study 1

In study 1, the children in the fourth grade achieved a good understanding of programming without variables. They wrote and ran simple programs that demonstrated their creativity and their ability to apply the concepts they had learned. When the students tried to learn concepts associated with variables however, their level of understanding almost ceased to exist. It was as if the students had reached an insurmountable barrier. In contrast, the sixth grade students usually succeeded in overcoming this barrier.

In the first part of the course, sixth graders worked on more difficult problems than the fourth graders did. They also acquired new material at a faster rate than the fourth graders. In the second part of the course, the sixth graders continued to modify programs and to solve problems by writing their own programs, albeit with more difficulty. In contrast the fourth graders, rather than writing their own programs, r[illegible] examples that had been prepared for them, without achieving a thorough understanding of the concepts involved. In a sense they turned from being programmers to program users.

The previous pattern was also reflected in student scores on the final test, which consisted of five parts. Each part required the students to concentrate on a different programming concept. Students were asked to identify syntax errors, describe the function of statements, predict the output of a program, modify slightly a given program, and to write a short program using the concept underlying that part of the test.

Table 1 presents the mean scores of both age groups for each of the five parts of the test.

TABLE 1: FINAL TEST SCORES OF STUDY 1

PROGRAMMING CONCEPTS AND SKILLS	MEAN SCORES*	
	FOURTH GRADERS	SIX GRADERS
basic graphics	69	90
random numbers & infinite loops	71	84
random numbers & printing	34	86
variables	21	68
input & conditionals	0	65

*** The maximal score for each part is 100.**

The following trends are observable from these results:

- It is apparent that the fourth graders were fairly successful in mastering the first two concepts, which involved programming without using variables. They had more difficulty in printing random numbers, and they failed completely to use variables as required in the final two parts of the test.

- The scores of the sixth graders in the first three parts of these were excellent. In the final two parts there was a moderate decrease in their understanding.

- Sixth graders scored better than fourth graders for each part of the test, but the most significant difference in results was in their comprehension of variables.

Each student was required to do a final project during approximately ten hours of on-line programming. Each student worked on a Pacman-like program that required users to move a square thro'' : maze with the use of the keyboard without hitting the maze walls. "Eating" other col .quares resulted in the user being given credit points. Sixth graders used variables much more extensively in their programs that the fourth graders. Al of the sixth grade students used variables as counters to keep track of credit points; none of the fourth graders did so.

One of the fourth graders found an original way to keep track of credits. His program printed a star every time the user earned a credit. At the end of the game, the user could count the number of stars. Apparently this student was not able to write a program that used a variable as a counter.

All of the fourth graders used .ne maximum amount of the time allotted for work on their projects. The sixth graders, however, were able to finish their projects in about half of the allotted time. They then worked on a second project which consisted of graphics animation similar to that used in video games such as Space Invaders. In these second projects variables were employed more extensively and in more complex ways.

It was clear from the final projects, that the older students had achieved competency in using and applying variables, but younger students lacked the intellectual maturity to comprehend the concept of the variable.

Study 2

An evaluation of the scores of the final test given to students in the second study shows a marked decr ase in comprehension of concepts pertaining to variables. This test was considered highly reliable because it consisted of individual interviews with students along with actual use of the computer.

Table 2 presents a summary of the mean scores for each part of the final test (the maximal score for each part was 100). Scores of students were grouped into three categories according to their level of academic achievement in school.

TABLE 2: FINAL TEST SCORES OF STUDY 2 (N= 64)
(The mean scores are presented in percents)

concepts:	graphics statements (4 items)				random number (2 items)				loops (3 items)				variables (4 items)				overall score (15 items)			
academic level:	1	2	3	all	1	2	3	all	1	2	3	all	1	2	3	all	1	2	3	all
4th grade	69	77	83	77	21	44	65	47	50	56	72	61	20	33	59	41	43	54	72	59
6th grade	86	84	98	89	31	50	75	53	64	68	79	70	20	28	69	39	54	59	82	65
all students				81				49				64				40				61

level of academic achievements : 1-low 2-intermediate 3-high

Table 2 shows that most students understood the concept of infinite loops, and almost all students succeeded in running graphics programs. The level of students' understanding of the concepts of random numbers and variables, however, was low. Students with low and intermediate academic achievement scored extremely low in the part that tested the understanding of variables.

The difference in levels of students' academic achievement, appears to explain in this study the wide variation between students better than the difference in ages.

STUDENTS' DIFFICULTIES IN USING VARIABLES

An analysis of difficulties encountered by students while using variables suggests four factors which can cause or increase the difficulty students experience in learning to use variables:

1. Level of abstraction

The more abstract a concept, the more difficulty young students have in grasping it. Using variables as representatives of a general phenomena increases the level of abstraction. Young students can understand the use of constant numbers as parameters in a computer command for example, far easier than they can comprehend the more general representation of the same command with variables as parameters. Thus, a child who can understand a concrete location on the screen, identified by two numbers (e.g. 2,3), may not be able to understand the meaning of a point at X,Y, which represents a set of points rather than a single point.

Even the sixth grade students were not able to cross the intellectual bridge between the concrete and abstract representation and to internalize the abstract meaning of variables:
Example: Children in the sixth grade who could easily draw lines were asked to draw a 50 units long vertical line which started in the location (X,Y). None of the students succeeded in this task. Even after having the solution explained to them, the students could not draw such a line when the required length was abstracted to 'Z' units.

2. The dynamic nature of variables

As mentioned previously, there are four methods for assigning a value to a variable:
-Assigning a constant number to a variable;
-Assigning a random number to a variable;
-Changing the value of a variable while the program is being executed, using input devices;
-Incrementing the value of a variable.

There is a crucial difference in terms of the ability of students to understand how value is assigned to a variable using the first three methods, and using the fourth method. The first three assign a value that is independent of the former value, whereas the fourth method required that the new value be calculated from the former value (e.g. X=X+2). In order to follow this change, students must hold in their short term memory a series of numbers. They have to figure out the first number in the series as well as device the mathematical rule for determining the rest of the following numbers in the series. This ability is not required to master the first three methods for assigning values to variables.

In table 3, four very short programs are shown that illustrate the four methods for changing the value of variable and the resulting load of short term memory (STM) incurred in each case.

TABLE 3: FOUR METHODS FOR CHANGING THE VALUE OF A VARIABLE

METHOD	**EXAMPLE**	**REQUIREMENT OF STUDENT STM LOAD**
use of a constant number	10 X=20 20 PRINT X 30 GOTO 10	20
use of a random variable	10 X· RND(1)*40 20 PRINT X 30 GOTO 10	any number the computer chooses
enter a new value while the program runs	10 INPUT X 20 PRINT X 30 GOTO 10	any number the user chooses
incrementing the value of a variable	10 X=X+2 20 PRINT X 30 GOTO 10	{2,4,6,8,10 ... } the first number the mathematical rule

The example shown above which uses the fourth method illustrates one of the simplest formulas for incrementing the value of a variable. When the formula for calculating the next number in the series is slightly more complex such as X=2*X+1, the resulting series of numbers is {1,3,7,15,31 ... } which is difficult for young students to remember and understand.

The following example illustrates the difficulty of young learner in using the fourth method. *Example:* A fourth grader in study 1 was instructed to run a program that printed the following series of sequential numbers: { 0,1,2,3,4,5 .... }. He was then asked to modify the program so it would only print the even numbers { 0,2,4,6,8 ...}. The students proceeded to change the program so it read as follows:

```
ORIGINAL PROGRAM        MODIFIED PROGRAM

10 PRINT X              10 PRINT X,X=2
20  X=X+1                20 X=X+1
30 GOTO 10              30 GOTO 10
```

Apparently, the student totally failed to comprehend the principles underlying the method for incrementing of the value of a variable.

3. Degree of complexity

Variables can be implemented in a computer program in varying degrees of complexity, defined in part by the following factors (all the examples were taken from Applesoft Basic):

- The number of parameters in the statement, either one (e.g. COLOR=3), or more (e.g. HPLOT 0,0 TO 100,100).
- The number of variables actually used as parameters in a statement. (e.g. HLIN 0,40 AT X vs. HLIN X,X AT X).
- The way variable appear in the command, either directly (e.g. HPLOT X,X), or as a result of a calculation (e.g. HPLOT 180-X,X).

It was apparent from the two studies that each additional parameter and calculation added to the degree of complexity and greatly increased the demand on students' short term memory load as they coped with new material, making learning more difficult. This is illustrated by the following:

Example: In the second study, the mean percentage of correct answers to questions pertaining to plot commands with <u>two</u> parameters was 75. In the same test, the mean percentage of correct answers to questions pertaining to line commands with <u>three</u> parameters was only 46.

Example: In programs containing many parameters, student performance was very low. In the first study, a simple program that drew rectangles was presented to the students. The program required the user to input four numbers: two numbers defined the location of the upper left corner of the rectangle, one which defined the length, and one which defined the width. After the user entered the numbers, the program drew a rectangle with these specifications. The students ran the program with data specified in the textbook. Next, they were asked to use the program to draw "a very long rectangle" anywhere on the screen. Surprisingly, none of the students, including the sixth graders, succeeded in performing this task.

4. Level of logical reasoning required

Variables contained in logical expressions are used in programming to do selective acts. Undoubtedly, the ability to reason logically is a prerequisite for being able to use conditionals in programming. Some students who participated in the two studies lacked this ability.

Example: Some students who understood the meaning of the express X<1 could not understand the meaning of the express X<Y.

Example: A sixth grader was required to write a program that asked the user to enter two numbers. If the numbers are equal, the program should print a certain message; if they are unequal the program was supposed to print a different message. The student's program read as follows:

```
10 INPUT A
20 IF A=A THEN...
30 IF A≠A THEN...
```

Evidently this sixth grader lacked understanding in applying logical rules.

Example: Fourth graders ran a six line program which generated a random number without displaying it. The program then asked the user to guess the number. After a student inputted his guess the program responded by displaying "too big", "too small" or "you found it". Fourth graders commonly replied to the display "too big" by inputting a still larger number. They failed to comprehend the logical rule underlying the game.

DISCUSSION

It has been argued that the lack of the ability to comprehend the concepts relating to variables acts as a barrier to young students learning programming. The students in the first study were highly intelligent and motivated, and learning conditions were close to ideal. Nevertheless, participating fourth grade students totally failed to understand the concepts pertaining to variables and sixth grade students experienced many difficulties in learning them. Thus, it may be concluded that other students of this age will also find understanding tne concept of variables difficult to understand.

These conclusions support Carlson's claim that, "Even very bright children under the age of 12 may be slow in mastering the more abstract parts of programming" (Carlson, 1983). These conclusions are also in agreement with the results of a study done at the Bank Street College of Education, that showed students seldom use variables in writing LOGO programs (Mawby, 1984).

Four possible factors affecting the level of difficulty experienced by students learning to use variables were suggested above: the level of abstraction in using variables, the dynamic nature of the values of variables, the degree of complexity in using variables, and the logical reasoning required. Recognizing the influence of these factors on learning, it might be possible to overcome some of the difficulties encountered by students, through enacting the following measures:

- Postpone the introduction of variables to a later stage in the learning process as suggested by Luehrmann (1983) and by Nachmias, Mioduser and Chen (1984). It has been suggested that young students first master the skills from three of the domains of knowledge presented in the Nachmias, et al. model --machine handling skills, the language features and flow of control of the computer program -- before introducing the concept of variables. It is essential to present programming concepts to young children gradually and build upon the knowledge they have mastered previously.

- Displaying the changes in the value of variables on the screen during the execution of a program can help students to better understand the dynamic nature of variables. In current window-oriented programming environments like MacPascal (Apple Computer Inc.), Instant Pascal (Think Technologies, Inc.), and Boxer (diSessa, 1984), this display is a built-in feature. Attaching a window to a program variable is analogous to attaching a voltmeter to an electronic circuit (Bobrow, Stefik, 1986). It gives on-line feedback to the programmer and it may reduce the demand of the short term memory load of young learners while they are programming. More research is needed to determine the extent of students' benefit from the on-line feedback features of the programming environment.

- Integrating concrete models into the learning environment, as suggested by Mayer (1981); duBoulay, O'Shea and Monk (1981); and Mioduser, Nachmias and Chen (1984), may contribute to the construction or completion of students' mental models of computer programs, and thereby decrease the level of abstraction which is raised by the introduction of variables. Further research and development on interactive computer software which attempt to construct such mental models may prove to be of great value.

- Developing the students' repertoire of templates employed to perform commonly encountered tasks in programs, as argued by Linn (1985), may reduce the complexity of dealing with more than one variable in a program.

- Teaching logical reasoning prior to introducing conditionals may reduce the number of students' misconceptions and errors in programming instruction.

This paper has examined programming language acquisition from a developmental point of view. The studies presented dealt with only two points on the developmental curve: fourth grade and sixth grade. Fourth grade was considered representative of a concrete operational stage, and the sixth grade as representative of the possible beginning of the formal operation stage. It may be concluded from this research that the understanding of concepts related to variables does not occur before the sixth grade (age of 12). There is a need for further developmental research on the use of variables in higher ages to determine whether difficulties posed by learning the concepts relating to variables act as barriers to older novice students as well.

Many research studies have dealt with far transfer from programming toward fostering high level cognitive skills (Salomon, Perkins, 1984; Pea, Kurland, 1984;. Pea, 1984). It is believed that there is a place for more research to investigate closer transfer; such as how learning and using variables influence students' ability in similar concepts acquisition in mathematics and science.

REFERENCES

Bonar, J., "Understanding The Bugs of Novice Programmers", *Learning Research and Development Center*, University of Pittsburgh, October 1984.

Bobrow D. G., & Stefik M. J., "Perspective on Artificial Intelligence Programming", *Science*, Vol. 231, February 1986, 951-957.

Carlson E.-H., "Teach Your Kids Programming", *Creative Computing, April 1983*, . 168-176.

Dalbey, J., & Linn, M. C., "The demands and requirements of computer programming: A review of the literature", *Journal of educational Computing Research*, Vol. 1, 1985.

diSessa, A., "Boxer: An Integrated Personal Computing Environment", *Educational Computing Group*, Laboratory of Computer Science, M. I. T. , Cambridge, 1984.

duBoulay, J. B. H.; O'Shea, T.; Monk, J., "The Black Box Inside the Glass Box: Presenting Computing Concepts to Novices", *Int. J. Man-Machine Studies 14*, 1981, 237-249.

Kurland, M. D.; Mawby, R.; Cahir, N., "The Development of Programming Expertise in Adults and Children", in: Kurland, M. D.(ed.), *Developmental Studies of Computer Programming Skills*, Bank Street College of Education, Technical Reoport NO. 29, AERA Annual Meeting, 1984.

Linn, Marcia C., "The Cognitive Consequences of Programming Instruction in Classrooms", *Educational Researcher*, May 1985, 14-29.

Luhermann, A., "Slicing Through the Spaghetti Code", *The Computer Teacher*, April 1983.

Mawby R., " determining Students' Understanding of Programming Concepts", Paper presented in AERA meeting, April 1984.

Mayer, R.E., "A Psychology of Learning BASIC", *Communication of the ACM 22*, November 1979, 589-593.

Mayer, R. E., "A Psychology of How Novices Learn Computer Programming", *Computing Surveys*, Vol. 13, No. 4, 1981, 121-141.

Mioduser, D.; Nachmias, R.; Chen, D., " Teaching Programming Literacy To Non-Programmers: The Use of Computerized Simulation.", *The Computer in education Research Lab.*, research report No. 15, Tel-Aviv university, Israel, 1985.

Nachmias, R., "Teaching Programming Language to Children", unpublished Ph.D. Thesis, Tel-Aviv University, 1985.

Nachmias, R.; Mioduser, D.; Chen, D., "Acquisition of Basic Computer Programming Concepts By Children", *The Computers in Education Research Lab., Research Report No. 14*, Tel-Aviv University, Israel, 1985.

Nachmias, R.; Mioduser, D.; Chen D., *Programming is a Child's Game* (Hebrew), Masada, Israel, 1985a.

Nachmias, R.; Mioduser, D.; Chen, D., "A Cognitive Curricular model for Teaching Computer Programming to Children", *The Computer in Education Research Lab.*, research report No.3, Tel-Aviv University, Israel, 1984.

Pea, R. D.; Kurland M. D., "On The Cognitive Effects of Learning Computer Programming", *New Ideas Psychol.*, Vol. 2, No. 2, 1984, 137-168.

Perkins, D. N.; Martin F., " Fragile Knowledge and Neglected Strategies in Novice Programmers", Harvard Graduate School of Education, 1985.

Salomon, G.,; Perkins, D. N., "Rocky Roads to Transfer: Rethinking Mechanisms of a Neglected Phenomenon", Paper Presented at the *Conference on Thinking*, Harvard Graduate School of Education, 1984.

Sheil, B. A., "Coping with complexity", Proceedings of Houston Symposium III, *Information and Society*, 1980.

Shneiderman, B., "Teaching Programming: A Spiral Approach to Syntax and Semantics", *Computers and Education 1*, 1977, 193-197.

Shneiderman, B., *Software Psychology: Human Factors in Computer and Information Systems*, Winthrop, Cambridge, MA., 1980.

Sleeman D.; Putnam, R. T.; Baxter, J. A.; Kuspa, L. K., "Pascal and High School Students: A Study of Misconceptions", *Occasional Report, No. 9, Technology Panel Study of Stanford and the Schools,* Stanford, 1984.

Soloway, E.; Ehrlich, K.; Bonar, J.; Greenspan, J., "What Do Novices Know About Programming?" in: Shneiderman, B.; Badre, A.(eds.), *Directions in Human-Computer Interactions*, Hillsdale, NJ., Ablex, 1982.