DOCUMENT RESUME

ED 305 903 IR 013 758

AUTHOR Linn, Marcia C.
TITLE Autonomous Classroom Computer Environments for Learning. Progress Report and Annotated Bibliography.
INSTITUTION California Univ., Berkeley. School of Education.
SPONS AGENCY National Science Foundation, Washington, D.C.
PUB DATE May 88
GRANT MDR-84-70364
NOTE 13p.
PUB TYPE Reference Materials - Bibliographies (131) -- Reports - Research/Technical (143)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Academic Achievement; Annotated Bibliographies; Case Studies; *Classroom Environment; Comparative Analysis; *Computers; Computer Science Education; Feedback; High Schools; *High School Students; *Intermode Differences; *Programing

ABSTRACT

This document provides both a brief progress report for the Autonomous Classroom Computer Environments for Learning (ACCEL) project and an annotated bibliography of publications from this project, the Computers and Problem Solving Project, and other recent publications from the ACCCEL (Accessing the Cognitive Consequences of Computer Environments for Learning) project and the ARP (Adolescent Reasoning Project). Two major project activities are described. One of the activities consisted of designing expert solutions to computer programming problems in order to communicate the techniques used by experts to solve programming problems. Designed to evaluate the effectiveness of these expert solutions, the other activity consisted of contrasting three educationally defensible alternatives for using these approaches and assessing the relative effectiveness of these approaches in 14 pre-college classrooms. The findings of this assessment are reported: students either completed all three activities or one of the three; there were significant differences between the three different conditions for using the expert solution; and students learned more about programming when they participated in all activities from writing the program to reading the expert solution. Relationships between performance on the case study and the instructional provisions in the classroom were also assessed, and three factors were found to contribute to student learning: (1) access to computers; (2) feedback on student work; and (3) individual and small group assistance by the teacher. The project report contains four references, and the annotated bibliography contains 35 references. (EW)

Progress Report for the Autonomous Classroom Computer Environments for Learning

National Science Foundation Grant MDR 84-70364

Marcia C. Linn, Principal Investigator

May, 1988

The Autonomous Classroom computer Environments for Learning project set out to substantially increase the learning outcomes from programming instruction by a) analyzing expert techniques for solving programming problems, b) identifying ways to communicate key techniques used by experts to pre-college students in programming courses, and c) determining whether these expert approaches to problem solving varied in effectiveness depending on the instructional provisions found in pre-college programming courses and the learning activities of students.

Primary Activities During Quarter

While programming represents a relatively mature use of technology in instruction, clear guidelines are not available to teachers on the most effective methods of teaching programming. Historically, programming classes have built on the experiences of expert programmers who taught themselves. Students were provided with assignments and access to computers and were expected to learn through trial and error and unguided discovery. Feedback consisted primary of what happened when students ran their programs.

The initial focus of the ACCEL project was to examine the instructional provisions in existing programming classes (e.g., Linn, 1985; Linn & Dalbey, 1985; Linn, Sloane, & Clancy, 1987). We found that some teachers continued to emphasize discovery learning, some emphasized problem solving procedures, and some provided extensive off-line feedback on student problem solving behavior.

The project also analyzed the behavior of expert programmers. This analysis suggested that students need opportunities to practice their programming skills, but they also need fairly explicit instruction in algorithms and procedures for solving new problems. Effective programmers build a library of algorithms or what the project has called "templates" that represent complex programming procedures. Experts tend to conceptualize these procedures in a generalized format which could be referred to as pseudo-code.

To communicate the techniques used by experts to solve programming problems, the ACCEL project designed expert solutions to computer programming problems. These expert solutions illustrate skills used by expert programmers, but rarely taught in pre-college, and even college, programming courses. First, the expert solutions provided generalizable templates represented in pseudocode. Second, the expert solutions illustrated the design decisions that programmers typically engage in to select between alternative appealing methods for solving a programming problem. In order to illustrate these design decisions, the case studies or expert solutions were of fairly complicated computer programs, typically 300 - 1000 lines long. Third, the expert solutions illustrated how experts intelligently search for program bugs, rather than linearly simulating the operation of a program. Fourth, the expert solutions emphasized implementing computer programs piece by piece, and testing each piece as it is implemented. Such a procedure greatly reduces the debugging task, because when a problem arises one knows that the bug is in the new code that is being tested. In addition, the expert solutions emphasized how programmers test their programs. They discussed the use of typical cases, as well as the use of extreme cases. As a result, the expert solutions pointed out a strategy for testing programs not commonly used by pre-college students. Pre-college students often fail to test their programs at all, or use haphazardly selected test cases, rather than systematically investigating the accuracy of their programs.

The expert solutions also implemented a recommendation commonly made by expert programmers. Many experts report that they learned more about programming by reading the programs written by others than by writing programs on their own. In a book entitled by *Programmers at Work*, for example, Susan Lammers interviewed Charles Simonyi, who reported "I had the complete listings of the Algol compiler which I had studied inside and out ... The SNOWBAL compiler I wrote at Berkeley, for example, was just a variation on the same theme. I think that the Algol program is still in my mind and influences my programming today. I always ask myself, if this were part of the Algol Compiler, how would they do it." The expert solutions help students learn how to use the programs of others to guide their own programming activities.

Classroom Investigation

To evaluate the effectiveness of the expert solutions, the ACCEL project has contrasted three educationally defensible alternatives for using these approaches and assessed the relative effectiveness of these approaches in fourteen pre-college classrooms.

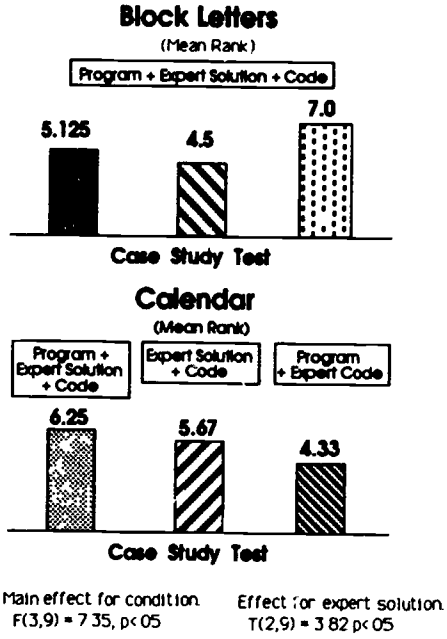
To use expert solutions, students could: (a) write the program for the problem, (b) read the experts' description of the program solution, (c) examine the code generated by the expert, (d) answer study questions about the expert solution, in order to analyze the solution more carefully; and (e) take a case study test which assesses their understanding of the expert solution. To investigate the effect of the expert solution, we contrasted the three conditions shown in Table 1. The first condition included all possible activities. The second condition included all of the activities, except having the students write the program themselves. Students started with the expert's program and went from there. The third condition omitted the description of the expert solution, but did include the expert code. Thus the students wrote the program and then looked at the code generated by the expert.

Table 1: Description of Conditions

	Program & Expert Solution & Code	Expert Solution & Code	Program & Expert Code
Write the program for the problem	X		X
Read the expert solution	X	X	
Run the expert code	X	X	X
Answer the study questions	X	X	X
Take the Case Study Test	X	X	X

To compare these three conditions, we first administered a case study, called Block Letters, to all participating classes and established a baseline of performance. Then we randomly assigned classes to one of the three conditions and administered a second case study called Calendar. As shown in Figure 1, ten classes completed both cases. There were significant differences between the three conditions for using the expert solution. Students learned more about programming when they participated in all of the activities, from writing the program to reading the expert solution. Students were least successful on the case study test when they were not given access to the expert solution, but were only given the opportunity to run the expert code. Thus, having expert commentary on the code appears to be extremely helpful for students when learning about computer programming.

Figure 1: Effect of Case Study Condition on Student Performance
N = 10 classes



Classroom Characteristics

We assessed the relationship between performance on the case study and the instructional provisions in the classroom. The instructional provisions we assessed are shown in Table 2. We determined the characteristics of classrooms by asking students and teachers to report on the classrooms (Sloane & Linn, in press). In general, students and teachers were in good agreement concerning the nature of classroom activities.

Analysis of the relationship between classroom characteristics and performance on the case study revealed that three factors were important to student learning. First, as would be expected, students benefit from access to computers. In pre-college settings, access to computers is primarily a function of the teacher's willingness to be present in the

Table 2: Classroom Characteristics Questionnaire: Variation Across Classes

Class Variable	Range of Scores	(F-value)	Interpretation
Computer Access			
# days on-line	2.30 to 5.00	(8.56*)	1-2 days to 4-5 days
Hours computer avaliable	1.74 to 2.71	(13.21*)	0-5 hrs/wk to 16+ hrs
Prop. on-line assignments	3.14 to 5.00	(10.35*)	About 1/2 to almost all
Explicitness			
# days teacher lectures	1.67 to 4.00	(20.17*)	0-1 days to 3-4 days
# days teacher works indiv.	1.50 to 3.86	(6.45*)	0-1 days to 3-4 days
Require. for computer access	0.00 to 3.00	(10.65*)	None to show plan & code
Quality of explicit instruction (how often teacher uses strategies)	2.84 to 3.87	(9.76*)	Rarely to often
Feedback			
Presence of feedback	3.33 to 4.70	(12.58*)	Sometimes to always
Type of feedback	2.37 to 4.00	(6.96*)	Rarely to often
Tests on language features	2.35 to 3.91	(6.96*)	Rarely to often
Tests on Pascal procedures	1.80 to 2.69	(3.69*)	Rarely to sometimes
Grading criteria	2.00 to 4.00	(4.63*)	Correctness only to correct & design, style

* F (18,340), p < .001

classroom before or after school. Therefore, access to the computer is partly a school variable and partly a teacher variable. Students with greater access to computers were more likely to succeed on both the computer program and the case study test.

Second, students benefited from feedback on their work. Classes where assignments were returned with comments and returned regularly were more successful on the programming assessments than were classes where there was less feedback available.

Third, students were more successful when teachers were able to provide individual assistance and worked with small groups of students on particular problems.

In summary, classrooms offering programming differ substantially from one to another and these differences do affect student learning.

Conclusions

We are still identifying characteristics of effective programming instruction and analyzing the relationship between student learning activities, classroom characteristics, and programming proficiency. The ACCEL studies suggest that curriculum materials that emphasize techniques used by experts and classroom provisions that provide access to computers, feedback on problem solutions, and explicit instruction on areas of difficulty will contribute to effective understanding of programming. The case studies of expert solutions appear to impart the design, testing, and debugging skills students need for successful programming. Furthermore, the effectiveness of these materials varies with the characteristics of classrooms.

Future Directions

Increases in computer access are difficult to implement given the time demands on pre-college teachers and the preparation level characteristic of those teaching programming. As a result, it seems useful to increase student access to expert solutions and to find ways to dynamically present the information available in expert solutions such that they can be used by students in independent or self-paced learning situations. This question is the focus of our subsequent work.

References

Linn, M. C. (1985). The Cognitive Consequences of Programming Instruction in Classrooms. *Educational Researcher, 14*(5), 14-29.

Linn, M. C. & Dalbey, J. (1985). Cognitive Consequences of Programming Instruction: Instruction, Access, and Ability. *Educational Psychologist, 20*(4), 191-206.

Linn, M. C., Sloane, K. D., & Clancy, M. J. (1987). Ideal and Actual Outcomes from Precollege Pascal Instruction. *Joural of Research in Science Teaching, 24*(5), 467-490.

Sloane, K. D. & Linn, M. C. (in press). Instructional Conditions in Pascal Programming Classes. In R. Mayer (Ed.), *Teaching and Learning Computer Programming: Multiple Perspectives.*

Annotated bibliography available upon request. Please write:

Marcia C. Linn
Graduate School of Education
Tolman Hall
University of California at Berkeley
Berkeley, CA 94720

This material is based upon research supported bu the National Science Foundation under grant nos. DPE-84-70364 and MDR-84-70514. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of National Science Foundation.

ANNOTATED BIBLIOGRAPHY
Selected Publications

ACCEL Project

1. Husic, F., Linn, M. C., & Sloane, K. (1988). *Adapting instruction to the cognitive demands of programming* Unpublished manuscript, University of California, ACCEL Project, Berkeley.

   This study examined class conditions and programming instruction for 8 introductory and 8 advanced placement Pascal classes. While introductory students required more explicit or direct instruction, advanced placement students performed better in situations that provided less guidance and more autonomy over their learning environment. The results suggest that curriculum for programming instruction needs to be adaptable in order to match the cognitive demands of different learning groups.

2. Sloane, K. & Linn, M. C. (in press). Instructional conditions in Pascal programming classes. In R. Mayer (Ed.), *Teaching and learning computer programming: Multiple research perspectives.* Hillsdale, NJ: Lawrence Erlbaum Associates.

   This study investigates the relationship between two measures of students' programming proficiency and instructional practice in 14 Pascal programming classes at the high school level. Proficiency measures assessed proficiency in writing programs and in understanding, analyzing, and modifying existing programs. We found that instructional conditions, including explicit emphasis on problem solving, extensive computer access, and precise feedback, are highly predictive of students' performance on each of the measures. The results from this study help clarify the nature of exemplary instructional environments and encourage researchers to examine explicit instruction more closely.

3. Rohwer, W. D. & Thomas, J. W. (1988). *The role of autonomous problem-solving activities in learning to program.* Unpublished manuscript, University of California, ACCEL Project, Berkeley.

   Variation in high school students' achievement in computer programming courses in Pascal was examined as a function of individual differences in self-reported engagement in autonomous problem-solving activities and of differences in features of the students' courses. Also examined were relationships between course features and extent of engagement in different problem-solving activities. A total of 107 students in eight Introductory

courses and 79 students in seven Advanced Placement courses completed a self-report questionnaire about their problem-solving activities, a second questionnaire about features of their courses, and a criterion test requiring the reformulation of code in an existing program. The results of regression analyses indicated that student differences in autonomous problem-solving activities account for substantial variance in criterion performance, especially in Advanced Placement courses. Whereas these relationships generally held across courses, certain features of these courses were related to differences in engagement in selected problem-solving activities.

4. Kersteen, Z., Linn, M. C., & Clancy, M. (1988). Previous experience in the learning of computer programming: The computer helps those who help themselves. *Journal of Educational Computing Research, 4*(3), 321-333.

   This study investigates the role of previous experience with computers as a predictor of performance in college computer science courses. We discuss the interaction of gender, previous experience with computers, and computer science course performance. Results revealed disparate amounts of prior computing experience among males and females. Nevertheless, males and females earned similar grades in introductory courses.

5. Gelman, R. & Linn, M. C. (1987). On the use of hands-on materials in science class. *Continuum* (A publication of PATHS and PRISM), *May/June,* 3.

   This article discusses implications of the *Establishing a Research Base* report (C&PS report #8) for classroom practice.

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

   This paper investigates the relationships between instructional practice and learning outcomes in 14 Pascal programming classes at the high school level. In particular, we examine the role of extensive on-line access, explicit instruction, and extensive feedback in instructional practice.

7. Mandinach, E. & Linn, M. C. (1987). The cognitive effects of computer learning environments. *Journal of Educational Computing Research, 2*(4), 411-427.

   In this study we examined the student characteristics that lead to success in programming courses. Results indicated that the most successful students did not progress far along the chain of cognitive accomplishments of

programming, gained their skills primarily from classroom instruction, and were not necessarily high in general ability or owners of home computers. Implications for classroom instruction are discussed.

8. Clancy, M. (1985). Large programs in Advanced Placement Computer Science. *The Computing Teacher, 12*(9), 60-61.

   This paper discusses the advantages of using large programs to teach computer science.

COMPUTERS AND PROBLEM SOLVING PROJECT (C&PS)

1. Eylon, B. & Linn, M. C. (in press). Learning and instruction: An examination of four research perspectives. *Review of Educational Research.*

   Recent research in science education examines learning from four distinct perspective which we characterize as a focus on concept learning, a developmental focus, a differential focus, and a focus on problem solving. This paper illustrates how these perspectives, considered together, offer new insights into the knowledge and reasoning processes of science students. An integrated examination of the four research perspectives strongly suggests that in-depth coverage of several topics will teach students far more than will fleeting coverage of numerous science topics.

2. Friedler, Y., Nachmias, R., & Linn, M. C. (in press). Learning scientific reasoning skills with microcomputer-based laboratories. *Journal of Research in Science Teaching.*

   This study contrasted the effects of two aspects of scientific investigation: observation and prediction. Students' scientific reasoning skills were developed in the content domain of temperature and heat energy using a microcomputer-based laboratory (MBL) environment. Four eighth grade classes were divided into two emphasizing observation and two emphasizing prediction. Findings included equal gains for observation and prediction groups in subject matter knowledge and the ability to use scientific reasoning skills as part of the problem-solving process. Differences were found between students' development of observation and prediction skills and the incorporation of these skills into their problem-solving process.

3. Friedler, Y., Nachmias, R., & Songer, N. (in press). Teaching scientific reasoning skills: A case study of a microcomputer-based curriculum. *School Science and Mathematics.*

To prepare students to live in and contribute to our rapidly changing society, science educators must look for ways to encourage and develop critical thinking skills. In our study, we investigated the educational potential of microcomputer-based laboratories to foster such inquiry skills in an eighth-grade physical science curriculum. Of the project's three main objectives, (1) to teach the subject matter, (2) to teach graph interpretation skills, and (3) to foster students' scientific reasoning skills, this paper focuses on the third objective -- namely the development and implementation of a scientific reasoning Skill Development Mode.

4. Linn, M. C. (in press). Designing science curricula for the information age. Colorado Springs, CO: Biological Sciences Curriculum Study.

   Recent research on provides guidance to those designing curricula for the information age. This paper summarizes procedures for designing science curricula based on current instructional theories. Questions addressed include how these curricula will elicit and sustain lifelong interest in learning, how they can help students construct robust, general conceptions of science phenomena to eventually replace less powerful conceptions, and how they can prepare students for future learning.

5. Linn, M. C. (in press). Establishing a science and engineering of science education. In A. di Sessa, M. Gardner, J. Greeno, F. Reif, & A. Schoenfeld (Eds.), *Toward a scientific practice of science education.* Hillsdale, NJ: Lawrence Erlbaum Associates.

   This paper summarizes a conference, entitled "Toward a Scientific Practice of Science Education," held at the Lawrence Hall of Science, Berkeley, California, and jointly sponsored by the Lawrence Hall of Science and the Graduate School of Education at the University of California, Berkeley.

6. Linn, M. C. (in press). *Science education and the challenge of technology.* To appear in *Informational technologies and science education* (1988 yearbook of the Association for the Education of Teachers in Science). Washington, D. C.: ERIC Clearinghouse.

   This paper analyzes the relationship between science education and technology over the last 15 years, identifying promising trends, and recommending policies for the future. Efforts to incorporate technology into science education fall into three stages. At first, technological tools such as computer-presented text and question-and-answer software were used to mimic established instructional procedures. In the second stage, progress involved

making expert tools such as simulations, microworlds, and real-time data collection available to students. Users found that these tools were not sufficient to teach the thinking skills of experts but could be effective if used with curriculum materials that drew on research on learning and instruction. In the third stage educators are redefining the roles of teachers, technology, textbooks, and experiments, as well as rethinking the goals of science education. The result is an improved model of instruction.

7. Linn, M. C. & Songer, N. B. (1988, April). *Cognitive research and instruction: Incorporating technology into the science curriculum.* Paper presented at the annual meeting of the American Educational Research Association, New Orleans, LA.

   In these investigations, we combine advances from research on learning and instruction with advances in educational technology. Our goal is to improve students' understanding of aspects of thermodynamics. The technological advance we study, real-time data collection, frees students from the tedium of recording, analyzing, and displaying data. The challenge to curriculum designers involves taking full advantage of this capability in teaching students about thermodynamics. The advances in research on learning and instruction we incorporate characterize the learner as a) actively constructing a view of the natural world, b) coming to science class with isolated conceptions rather than integrated ideas, c) benefitting from robust models of scientific phenomena, and d) capable of learning self-monitoring skills.

8. Songer, N. (1988, June). *Using the microcomputer in an elementary preservice teacher training workshop.* Presentation to the National Educational Computing Conference (NECC), Dallas, Texas.

9. Linn, M. C. (1987). Learning more with computers as lab partners (An apple a day). *Science and Children, Nov/Dec,* 15-18.

   This paper presents the learning outcomes and advantages of using computers as silent laboratory partners in a one-semester physical science class. In particular, outcomes from using microcomputer-based laboratory (MBL) software using temperature probes, light probes, and heat pulsars to collect data are discussed.

10. Stein, J. (1986-87). The computer as lab partner: Classroom experience gleaned from one year of microcomputer-based laboratory use. *The Journal of Educational Technology Systems, 15*(3), 225-236.

This paper presents a formative evaluation of one microcomputer-based laboratory (MBL) system and its accompanying semester-length junior high physical science curriculum. This paper draws on data from classroom observations and student and teacher interviews to outline some pinnacles and pitfalls encountered in a year of MBL use, and incorporates these into a model for the integration of MBL into science curricula.

11. Linn, M. C., Layman, J., & Nachmias, R. (1987). Cognitive consequences of microcomputer-based laboratories: Graphing skills development. *Contemporary Educational Psychology, 12*(3), 244-253.

    Students' difficulties with graph interpretation may stem from their inappropriate representations of graphs. Real-time data collection and graphic display of results offer a dynamic representation of graphing. Evaluation of student response to this instruction reveled that students gained robust and coherent understanding of graphing that generalized to new knowledge domains.

12. Nachmias, R. & Linn, M. C. (1987). Evaluations of science laboratory data: The role of computer-presented information. *Journal of Research in Science Teaching, 24*(5), 491-506.

    This paper examines how students critically evaluate information acquired in the science laboratory, par ularly computer-presented information. We examine the role of factors that influence such assessment, like knowledge of science principles, and contrast the activities in a science laboratory with current thinking about epistemology in the philosophy of science.

13. Linn, M. C. (1986). Computer as Lab Partner Project. *Teaching Thinking and Problem Solving, May/June.* (Hillsdale, NJ: Lawrence Erlbaum Associates.)

    This article provides an overview of the Computer as Lab Partner Project, its focus, objectives, and preliminary activities.

14. Linn, M. C. (1987). Establishing a research base for science education: Challenges, trends, and recommendations. *Journal of Research in Science Teaching, 24*(3), 191-216.

    On January 16-19, 1986, mathematicians, scientists, educators, and curriculum and technology experts convened at Berkeley for a planning conference on research and science education. This report describes the themes that emerged from the discussions at the conference, and makes four recommendations

intended to encourage the development of an integrated research base in science education and to infuse science teaching with ideas and techniques informed by research and dedicated to meeting the challenge of change in a technological world.

15. Linn, M. C. (Ed.). (1987). Cognitive consequences of technology in science education [Special issue]. *Journal of Research in Science Teaching, 24*(4/5).

    These issues feature 12 papers on the use of technology in science education.

16. Kirkpatrick, D. (1986). Technology in education: A middle school teacher's experience. *Technology and Learning* (Lawrence Erlbaum Associates newsletter.)

    In a column entitled "*Teacher's Perspective,*" Doug Kirkpatrick, a mentor teacher at Foothill Middle School, shares his experiences in using microcomputer-based laboratories and in training science teachers to incorporate technology into their programs.

17. Songer, N. (1987). *Encouraging conceptual constructions: Exemplary software tools and their role in exemplary learning environments.* Unpublished manuscript, University of California, Education in Math, Science, and Technology, Berkeley.

18. Linn, M. C. (1987). *Education and the challenge of technology* [Proceedings of a conference on Technology and Teacher Education]. Cupertino, CA: Apple Computer, Inc.

    To examine the potential of technology for improving education in general and teacher education in particular, the Graduate School of Education at the University of California, Berkeley, with support from Apple Computer, invited leading educators, administrators, researchers, industry experts, and state officials to a conference on Technology and Teacher Education in Monterey, California from August 5 - 8, 1986. This report summarizes the proceedings from this conference and offers four interrelated recommendations to foster the infusion of technology into education.

19. Nachmias, R., Stein, J. S., & Kirkpatrick, D. (1987, April). *Computer as lab partner: Students' subject-matter achievements.* Paper presented at the annual meeting of the National Association for Research in Science Teaching, Washington, D. C.

In order to carry out the studies of the Computer as Lab Partner Project, a semester-length microcomputer-based laboratory (MBL) physical science curriculum was developed and has been refined and evaluated over three years. This paper presents an account of the preliminary development and design of that curriculum, evaluates the curriculum in terms of the students' subject matter achievements, and suggests ways in which MBL may be effectively integrated with science instruction to produce conceptual gain.

20. Stein, J. S., Nachmias, R., & Friedler, Y. (1987). *An experimental comparison of two science laboratory environments: Traditional and microcomputer-based.* Unpublished manuscript, University of California, Computers and Problem Solving Project, Berkeley.

    This paper examine the cognitive consequences for science students of two modes of laboratory data collection: traditional manual data recording and a system of microcomputer-based laboratories (MBL). Classroom processes, learning outcomes, and students' perspectives for a boiling-point experiment in each environment are compared. The advantages and disadvantages of MBL are discussed, along with provisions in each environment that foster the goals of science laboratory learning.

21. Nachmias, R., Friedler, Y., & Linn, M. C. (1987). *The role of programming environments in Pascal instruction.* Unpublished manuscript, University of California, Computers and Problem Solving Project, Berkeley.

    Programming environments that support the problem-solving skills of experts are being developed for novices. Two experiments investigated the advantages of the interactive programming features and optional tools in the Macintosh Pascal and Instant Pascal programming environments. The first experiment assessed when precollege programming students use the unique capabilities of the environment. The second experiment employed activities to enhance debugging skill and compared performance of students using Instant Pascal to that of students using a traditional programming environment. Results suggest techniques for using the tools and demonstrate possible advantages.

Other recent publications from the ARP* and ACCCEL** Projects.

1. Hyde, J. S. & Linn, M. C. (1988). Gender differences in verbal ability: A meta-analysis. *Psychological Bulletin, 104*(1), 53-69.

2. Linn, M. C. & Dalbey, J. (1986). Cognitive consequences of programming instruction: Instruction, access, and ability. *Educational Psychologist, 20*(4), 191-206.

3. Hyde, J. S. & Linn, M. C. (Eds.). (1986). *The psychology of gender: Advances through meta-analysis.* Baltimore: Johns Hopkins University Press.

4. Linn, M. C. & Petersen, A. C. (1986). A meta-analysis of gender differences in spatial ability: Implications for mathematics and science achievement. In J. S. Hyde & M. C. Linn (Eds.), *The psychology of gender: Advances through meta-analysis* (pp. 67-101). Baltimore: Johns Hopkins University Press.

5. Linn, M. C. (1985). Fostering equitable consequences from computer learning environments. *Sex Roles, 13*, 227-240.

6. Linn, M. C. (1985). The cognitive consequences of programming instruction in classrooms. *Educational Researcher, 14*(5), 16-29.

* ARP: Adolescent Reasoning Project

** ACCCEL: Assessing the Cognitive Consequences of Computer Environments for Learning