SEQUENCING AS A FACTOR ASSOCIATED WITH STUDENTS' ABILITY TO LEARN PROGRAMMING

DISSERTATION

Presented to the Graduate Council of the
University of North Texas in Partial
Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

By

Hassanali Honarvar, B.S., M.S.

Denton, Texas

August, 1991

Honarvar, Hassanali, Sequencing as a Factor Associated with Students' Ability to Learn Programming. Doctor of Philosophy (College Teaching), August, 1991, 72 pp., 6 tables, bibliography, 62 titles.

This experimental study concerned the performance of students in computer programming courses following exposure to a sequencing computer-aided instruction (CAI) drill and practice. The order of the procedures for the experiment was pretest, treatment, and posttest.

The students from two sections of a BASIC programming course were the subjects of the study. A pretest of BASIC programming language was administered to all students at the beginning of the course. According to their pretest results, students were divided into matched experimental and control groups. The sequencing CAI treatment was administered to the experimental group, and an alternate treatment was administered to the control group. Then all of the students followed to complete the BASIC programming language course. The posttest of BASIC programming language was administered at the completion of the course.

The performance levels of the following groups were compared: experimental and control groups, below average experimental and above average experimental groups, below average experimental, and below average control groups. Subjects' placement in the above average and below average

groups was based on their performance in the pretest. The following is a summary of the major findings of this study.

1. The experimental group did not have a significantly higher gain on the posttest than the control group.

2. The below average experimental group did not have a significantly higher gain on the posttest than the above average experimental group.

3. The below average experimental group did not have a significantly higher gain on the posttest than the below average control group.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER I

Introduction

A substantial percentage of students in beginning computer science courses find programming to be a more challenging task than they originally expected. In order to find ways to better advise students before they take computer science courses, researchers have developed several computer science aptitude predictor tests (Alspaugh 1972; Bauer, Mehrens, and Vinsonhaler 1968; Peterson and Howe 1979b; Fowler and Glorfeld 1981; Kurtz 1980; Kovalina, Wileman, and Stephens 1983; Sidbury 1986a). These tests are an attempt to estimate the chance of student success in introductory computer courses before students actually invest their time, money, and efforts in the courses.

The Konvalina Aptitude Test, designed by Konvalina, Wileman, and Stephens (1983), measures knowledge of sequencing, logic, algebraic word problems, and calculation areas of mathematics. These mathematical areas have been found to be effective predictors of students' success rate in introductory computer science courses. According to results of the Konvalina, Wileman, and Stephens study, students who performed better in these mathematical areas also performed better in computer science programming courses.

In theory, increases in students' knowledge of sequencing, logic, algebraic word problems, and calculation should elevate their achievement in programming. This study is an attempt to demonstrate the association of students' success rate in programming by improving mathematical skills in the area of sequencing.

Statement of the Problem

The problem of this study concerns the performance of students in computer programming following exposure to a computer-aided instruction (CAI) drill in sequencing and practice exercises.

Purpose of the Study

The purpose of this study was to determine whether or not student success rates improve in computer programming after completing CAI sequencing drills and practice exercises.

Hypotheses

The following hypotheses were developed for this study:

1. College students who complete a CAI drill in sequencing and then complete an introductory computer programming course will perform

significantly better in programming than students who complete only an introductory computer programming course.

2. Treatment students who perform below average in the BASIC programming language pretest will have a higher gain on the posttest than treatment students who perform above average on the pretest.

3. Treatment students who perform below average on the BASIC programming language pretest will perform higher on the posttest than control students who perform below average on the pretest.

Definition of Terms

Sequence is defined as "to put a set of symbols into an arbitrarily defined order; an arbitrarily defined order of a set of symbols; (i.e., an orderly progression of items of information or of operation in accordance with some rule)" (Sippl 1986, 252). The set of symbols used in this study includes numbers, letters, and a combination of both.

Background and Significance of the Study

Rapid development in computer-related fields has created a tremendous demand for computer science graduates. This demand has left computer science departments with two problems. The first is how to control excessive growth in student enrollment in introductory programming courses. The second is how to provide adequate faculty and computer resources for computer science students.

In light of the limited resources available to computer science educators and the increasing demand for computer education, researchers have concentrated on developing more effective tools for advising students who are interested in obtaining an education in computer science. Many studies have been conducted for the purpose of developing reliable computer science aptitude tests. Most of the resulting tests measure cognitive skill, personality traits, past academic achievement, or mathematical background. The findings of some of these studies are reviewed in the following paragraphs.

In the area of personality traits, Alspaugh (1972) used a personality predicting aptitude test. She found that students who were successful at programming had a low level of impulsiveness and a high level of reflectiveness. Grade-point average (GPA) was used in some studies as a representative of past academic achievement, and it has been correlated with student programming performance. Bauer, Mehrens, Vinsonhaler (1968); Peterson and Howe (1979b); and Fowler and Glorfeld (1981) all found college GPA to be the best predictor of success in introductory computer programming courses. Kurtz (1980) and Barker and Unger (1983) developed aptitude tests to measure the area of abstract thinking. Barker and Unger concluded that the test of abstract thinking in conjunction with other advising information provides a useful predictor of students' course performance.

Other researchers have studied the effects of mathematical knowledge on student performance in introductory computer science courses. Many areas of mathematics have been studied in relation to computer programming performance. The studies conducted in this area indicate a strong correlation between performance in introductory computer science courses and mathematical background (Sukhen and Mand 1986; Hancock 1988). Most researchers concerned with prerequisites have studied the effect of different areas of mathematics on computer programming.

Research shows that the area of discrete mathematics is related to performance in introductory computer science courses (Sukhen and Mand 1986). In his study of the effect of mathematics on performance level in introductory computer science courses, Sidbury (1986a, 45) concluded: "There is a strong indication that students who take discrete mathematics make higher grades in computer science than do the students who take the algebra-calculus sequence of courses."

Konvalina, Wileman, and Stephens (1983) found sequencing ability to be an effective predictor of performance level in introductory computer science courses. Given that students' ability in sequencing is a valid predictor of success in introductory computer science courses, it seems reasonable that strengthening this area would improve students' success rates. As an attempt to verify such improvement this study is significant in that it provides suggestions for ways to improve student performance, thereby reducing the number of students failing the courses. In turn, the findings of this study provide recommendations for using computer science faculty and computer facilities more efficiently.

Experimental Procedures

Introductory computer science students at Winona State University in Minnesota made up the population for this study. The sample included 80 students from two sections of an Introductory BASIC Programming Language course at Winona State University's Computer Science Department. The two sections of the Introductory BASIC Computer Programming class were taught by the same instructor. Students of both sections were divided into two equivalent groups: control and experimental.

The tool used for measuring students' levels of mastery of programming in BASIC computer language was a domain-referenced test in BASIC programming. The content of the test was designed and validated by professors who teach the BASIC programming language. Test reliability is reported in terms of coefficient of stability.

A CAI drill, written to present exercises in sequencing, was given to the experimental group during the first week of the semester. The exercises for the sequencing CAI program were taken from a doctoral study by Cannara (1976) at Stanford University which was designed for teaching children computer programming and a pattern sequencing of a Test of Foreign English Language Practice manual. The drill was written for the IBM PC/XT (TM), AT or any model of the IBM PS/2 (TM) microcomputer under IBM DOS 3.3 and above. A faculty member of Winona State University administered the tests and monitored the CAI sequencing drill in the Computer Science Department computer lab. Each student was allotted a two- to three-hour session to complete the CAI sequencing drill and practice exercises.

After completing the CAI drill, students in both of the groups proceeded through the introductory computer programming course. At the end of the course, a posttest (same as pretest) in the BASIC programming language was given to all of the subjects in both groups.

Statistical Analysis

A one-factor experimental research design where the independent variable is the treatment and the dependent variable is the performance level of the students in the BASIC programming language test was used for this study. Simple analysis of variance was used to compare the two groups involved in each hypothesis in order to determine if a significant difference existed among the groups for each hypothesis. The following comparisons were performed for the three hypotheses:

1. Compare the posttest mean score of the first group (sequencing treatment) with the second group (control group).

2. Compare the posttest mean score of treatment students who scored below average on the pretest with the mean score of treatment students who scored above average on the pretest.

3. Compare the mean score of treatment students who scored below average on the pretest with control students who scored less than average on the pretest.

Limitations of the Study

One limitation of this study was the sampling method used to establish the experimental and control groups. The subjects for this experiment were chosen from one university, therefore, the findings of the experiment are limited to similar universities.

A second limitation was the amount of time the experimental groups had to learn sequencing. Because of the students' limited time to contribute to the study, the treatment was limited to two to three hours.

Summary

An overview of the problem, the purpose of the study, background and significance of the study, the experimental procedures, statistical analysis, and limitations of the study are presented in Chapter I. A review of related literature is included in Chapter II. Chapter III contains the methods and procedures used for data collection and data analysis. The findings of the research are discussed in Chapter IV. Chapter V includes the summary, conclusion, and recommendations for future research.

CHAPTER II

REVIEW OF THE RELATED LITERATURE

Introduction

As Knuth (1974, 517) points out, "computer programming is an art, because it applies accumulated knowledge to the world, because it requires skill and ingenuity, and especially because it produces objects of beauty." Whether defined as an art (Knuth 1974) or as a science (Gathers 1986; Gries 1981), programming has been identified as a difficult activity to learn by many researchers (Pea and Kurland 1984b; Soloway et al. 1982). Chris Hancock (1988), a Project Associate at Harvard's Educational Technology Center, affirms the view that teaching and learning computer programming at the introductory level are both difficult.

On the other hand, the availability of opportunity in computer-related fields has created an increasing enrollment of college students in the introductory computer programming courses as observed by Denning (1980, 618), Gray (1974), and Rothman and Mosmann (1972). Increasing numbers of interested students in introductory programming courses and the difficulty level of computer programming have resulted in an obvious need for effective methods for advising students who are enrolled in introductory programming courses.

In an effort to better advise students and to potentially direct them into successful educational experiences in computer programming, many researchers have conducted studies related to students' success rate in programming courses. Most of the previous research has been concentrated on the identification of important factors which influence students' success rate in introductory computer programming courses. These researchers have evaluated factors such as student knowledge of mathematics, overall performance in high school and college, personality traits, and psychological elements in an effort to predict students' success rate in introductory programming courses. Other researchers have attempted to identify skills that, once improved, could aid students in learning computer programming. Still other researchers have examined methods of improving the curriculum for computer science programs and computer programming courses, and have suggested methods for enhancing teaching styles in those courses.

The results of such research provide educational curriculum developers with the information necessary to determine prerequisites for students who plan to take introductory computer programming courses. Both students and colleges can benefit from such information by effectively using their resources to provide and obtain an education in the area of computer programming.

Factors Associated with Students' Success Rate in Introductory Computer Programming Courses

Identifying factors that affect student performance in programming is by far the most frequently researched topic related to student performance in introductory computer programming courses. The available literature reports research on mathematics knowledge, past and present performance, personality traits, and psychological elements as they relate to the student performance in introductory computer programming courses. This review of related literature is divided into the following sections: the sequencing of cognitive skills, mathematics as a factor affecting students' programming ability, and personality traits and psychological elements in programming.

Sequencing as a Factor Associated with Students' Ability to Learn Programming

Many cognitive skills are associated with computer programming, some are identified as more essential for programming than others. Konvalina, Stephens, and Wileman (1981) determined that sequencing cognitive skill is among the most important skills for predicting students' computer programming success rate.

The cognitive skill of sequencing is important for problem solving, particularly in computer programming. However, there appears to be little formal research involving sequencing skills of introductory computer science students before their exposure to introductory computer programming courses. In most cases, the limited training students receive in sequencing is offered as a part of other problem-solving classes, such as mathematics courses. The importance of the cognitive skill of sequencing in computer programming and the area of problem solving has been studied by several researchers.

In search of strategies for teaching computer programming, McEntyre (1977), from the University of California at Berkeley, identified the following skills as necessary in an introductory programming course: sequencing control, precision, and manipulation of variables. McEntyre clearly identifies sequencing as an essential skill in programming. Jansson, Williams, and Collens (1987) define the task of programming as one requiring the formulation of an algorithm by stating and analyzing a problem, defining a sequence of operations to solve the problem, and obeying the rules of syntax and semantics of a computer language in entering the program into a computer to test the solution. Hostetler argues that, for identification of factors for predicting student performance rate, diagramming is correlated with the students' final numerical score in a computer programming course. Diagramming is defined "a test of ability to analyze a problem and order the steps for solution in a logical sequence" (Hostetler 1983, 41).

Researchers at the University of Nebraska discovered sequencing, among other mathematical skills, to be an effective predictor of students' success rate in introductory programming courses (Konvalina, Stephens, and

Wileman 1981; Konvalina, Wileman, and Stephens 1983; Stephens, Wileman, and Konvalina 1981). In their 1983 study, Konvalina, Wileman, and Stephens administered a computer science aptitude predictor to students enrolled in an introductory computer course in order to determine potential for success. The test consisted of problems in five areas of mathematics. Sequencing with the $\underline{P} < .01$, among other mathematical factors, proved to be an effective factor for predicting achievement in the class.

Although many facts support the importance of sequencing as an essential skill in computer programming, this study seems to be a first attempt to investigate the association of a computer-aided instruction (CAI) sequencing drill and practice on achievement in an introductory programming course.

Mathematics as a Factor Associated with Students' Ability to Learn Programming

Performance rate in mathematics has been correlated with achievement rate in computer programming. The effect of such findings is reflected in the mathematics prerequisite of many computer science programs. Students' performance in secondary-school and college-level mathematics has been evaluated as a prerequisite to computer science programs.

Peterson and Howe (1979a, 183) concluded that "an able high school student successful in mathematics and science will probably be a successful student in Computer Science." Alspaugh (1972) also found that mathematical background is an important factor in the success rate of students in introductory programming courses. Campbell and McCabe (1984, 109) discovered that the Scholastic Aptitude Test (SAT) mathematics and verbal, high school rank, and high school background in mathematics and science are the most predictive variables for forecasting success in computer science, engineering, and other science fields. Sukhen and Mand (1986) found significant correlations between the average grades of students in college-level mathematics courses and performance in computer science courses. Students with higher levels of high school and college mathematics knowledge obtained higher grades in computer science courses. Werth (1986) examined the relationship between many predicting factors and students' grade in introductory computer science courses at the University of Texas at Austin in 1986. She found significant correlation between the letter grade in the computer science course, the number of hours worked, and the number of high school mathematics classes taken. According to Gathers (1986), the two significant factors in the successful placement of college freshman computer science majors in their first computer course are the American College Test (ACT) English and University of Tennessee at Martin (UTM) mathematics placement test scores. Konvalina, Stephens, and Wileman (1983) studied eight factors regarding performance in an introductory computer course and concluded that a very critical relationship exists between students' performance in high school and their success at the college level. Their research also showed that a high school mathematics background is important to success in computer science. As a result, they encouraged advisors to recommend participation in high school mathematics to students at the earliest possible point. In a more isolated field of mathematics, Barker and Unger (1983) report that abstract reasoning development is an effective measure of success rate in a programming course.

Personality Traits and Psychological Elements Associated with Students' Success Rate in Introductory Computer Programming Courses

"Because of the complex nature of the programming task, the programmer's personality--his individuality and identity--are far more important factors in his success than usually recognized" (Weinberg 1971, 95). In his book, <u>The Psychology of Computer Programming</u> Weinberg also states that it is pure speculation to equate the programmer's personality traits to engineers, mathematicians, or any other field. There is a genuine need for analysis of personality traits and their association on students' programming ability.

In an attempt to identify some of the personality traits associated with students' success rates in programming tasks, Alspaugh (1972) found that the more successful programming student might be expected to have a personality associated with a low level of impulsiveness and sociability and a relatively high level of reflectiveness. In a similar study, Van Merrienboer of the University of Twente, The Netherlands, studied the relationship between the personality traits (reflection-impulsivity) and achievement in an introductory programming course in high school. He defines reflection-impulsivity as: "related to the quality of problem solving, when several alternatives are available and the current solution is not immediately obvious" (Van Merrienboer, 1988, 182). Persons with impulsive traits are characterized as gathering their information less systematically and carefully than individuals with reflective traits, as spending less time considering possible solutions or planning in advance, and as prone to make more errors. In a programming activity, Van Merrienboer states, persons with impulsive traits are likely to code a program as soon as they associate a certain detail in the problem description with a new language feature. Their understanding of a program is at a single-line level rather than the overall design level. However, he found that individuals with reflective traits consider different alternative solutions to the problem before they attempt to code a program. This planning technique of those with reflective traits could be helpful in constructing schemes or templates of blocks of codes. The use of schemes has proven to be an effective technique and is practiced by expert programmers (Clements 1979; Messer 1981). Van Merrienboer concluded that the program comprehension of persons with reflective traits is superior to the comprehension of those with impulsive traits. In knowledge of programming syntax, no significant difference was found between individuals with impulsive traits and those with reflective traits. In a related study, Clements and Gullo (1980) found that impulsivity is modifiable to a certain degree. They noted a positive effect of computer programming on reflection-impulsivity traits. Clements and Gullo also found that it is possible to force persons who have a tendency to be impulsive into a reflective strategy through exposure to appropriate instructional materials. According to Deimel and Moffat (1982) and Pea (1986), instructional strategies that are based on comprehension, modification, and amplification of written programs may force persons with having impulsive traits into a reflective strategy, resulting in better success in an introductory programming course.

Not all studies dealing with the identification of the personality traits and the psychological elements of students have revealed factors for effectively predicting student success rate. Hostetler (1983) studied sixteen different personality categories and did not find any significant correlation with students' success in computer science courses. Corman (1986), of the University of North Texas, also found that psychological and personality variables make little contribution to success rate of students in an introductory programming class.

Weinberg (1971) explains that failure of personality tests in identifying programming success rates among college students may be partly the result of inadequacies of the test. Another reason, Weinberg suggests, may be an inadequate understanding of which personality traits play an important role, and in what section of the programming process. For this reason, Weinberg recommends that additional research be conducted in order to identify personality traits that are related to computer programming success rates.

Improving Students' Ability to Learn Programming

Much effort has been spent in an effort to improve students' learning effectiveness in computer programming education. The following sections describe current research in the secondary education computer science curriculum, higher education computer science curriculum, and computer programming teaching methodologies as they relate to students' performance in computer programming performance and students' ability to learn computer programming.

Secondary Education Computer Science Curriculum and Students' Performance in Programming

Computer science in the secondary schools is a vital part of computer science education because it is where students are exposed to computer programming for the first time. Lack of a carefully designed computer training program at the secondary level can sometimes be harmful, and can have irreversible consequences. In reporting on his 1980 study of the secondary school computer science curriculum, Sedlmeyer, of Purdue University, notes:

> we have encountered an increasingly large proportion of students in our Introductory Programming classes who have had prior programming experience. Unfortunately, this "experience" has proven more harmful than helpful.

> . . . It was evident that these students had been taught programming language syntax--not programming skills . . . too often such a student is unwilling to change even when confronted with the gross inadequacies of his programming efforts. (Sedlmeyer 1980, 168)

It is justifiable to say that design and implementation of an appropriate computer science curriculum for high school has become an urgent task for educators. The goal of such a curriculum must be the preparation of secondary school students for higher education in computer science. Computer science programs at secondary schools are faced with a number of inadequacies. The Poirot (1979) study identified the three major problems associated with computer education at the secondary level as: (a) teacher proficiency, (b) hardware access, and (c) curriculum development.

Many solutions have been suggested to alleviate the problems of qualified secondary-level computer science teachers. Training for teachers to receive certificates in data processing has been implemented as a means to develop better qualified teachers. To improve teacher training, in its Recommendations for School Mathematics of the 1980s, the National Council of Teachers of Mathematics recommends that teacher education programs for all levels of mathematics should include computer literacy, experience with computer programming, and the study of ways to make the most effective use of computers and calculators in instruction.

As a result of tremendous advancement in integrated circuit technology, which has driven microcomputer prices down to an affordable range, hardware access is not as much of a problem as it was a decade ago. Time-sharing network systems are another feasible alternative for a more centralized operation.

On the other hand, the development of a secondary school curriculum which meets the demands of the ever-advancing computer industry continues to be a challenging problem for computer science programs. Sedlmeyer describes the goal for a programming course for high school students who are considering the pursuit of a data processing or computer science degree:

> 1. The student should understand the basic logical, physical and functional components of computer system and how they interact (computer architecture and organization).
> 2. The student should acquire the problem solving skills necessary to develop a computer program from a problem definition.
> 3. The student should be introduced to structured programming techniques.
> 4. The student should write programs for representative applications.
> 5. The student should develop an appreciation for the role of program documentation in program design, debugging and maintenance. (Sedlmeyer 1980, 169)

The Task Force on Computer Science of the Association of Computer Machinery Elementary and Secondary School Subcommittee (1983) recommends that a curriculum for a computer science course in high school should be oriented primarily toward teaching problem-solving skills rather than being vocationaly oriented. Enough information on data processing or computer professional careers should be included to create awareness of this possibility in the student. Therefore, such a course should consist of applications-oriented, general interest topics, with emphasis on practical use of the computer as a tool for problem solving for the individual and for society as a whole. The sequence of this course should vary considerably, depending on several factors (i.e., programming language or use of structured programming techniques), but an important consideration is that hands-on experience with a computer should start immediately at the beginning of the course. According to the task force, students should be running library programs (usually emulations or games) as a preliminary to writing their own programs. This provides working models for student programming and maintains a high interest level. The beginning of the course is also the time to eliminate "computerphobia" in those who are afraid of computers. This problem is generally not as common among secondary school students as in the general population.

Higher Education Computer Science Curriculum and Students' Performance in Programming

Although computer literacy at the secondary school level is recognized as an important part of student education in computer science, the majority of this training takes place at the college level. It is not surprising, therefore, to find that a large portion of previous research in this areas has been concentrated on computer science education at the college level. Research literature on computer programming at the college level is found in the areas of program curriculum, teaching methods, and curriculum.

The undergraduate curriculum in computer science has been greatly influenced by the Association of Computer Machinery's Curriculum Committee on Computer Science reports of 1968 and 1979. Berztiss (1987) argues that new curricula are needed to emphasize both the theory and implementation aspects of computer science. He identifies the following five essential fields of computer expertise: research, development, implementation, maintenance, and application user. Each field has a different level of knowledge requirements. For research and development, Berztiss recommends a Bachelor of Science degree with forty-five credit hours in computer science. For implementation expertise, he recommends a Bachelor of Arts degree with thirty credit hours in computer science. Finally, for maintenance and users, Berztiss recommends a minor in computer science. Berztiss supplements the recommended computer science curriculum with additional emphasis on mathematics courses.

In a curriculum-related study, Sidbury (1986b, 140) compared the effect of traditional mathematics and discrete mathematics and concluded that "students who take discrete mathematics do better in their beginning computer science course than do students who take traditional mathematics." Course curriculum and teaching methodology have been focal points in computer-programming-course-related studies,

Noonan (1979) suggests a two-semester course in BASIC or PASCAL that emphasizes algorithm development. The first-semester course concentrates on the programming language and the second-semester course expands on advanced programming techniques such as structure programming, modularity, and block programming for the development of large programs.

The curriculum for introductory programming courses has been addressed in many contexts. Samurcay (1985) suggests that many concepts and procedures, at various levels of complexity, are involved in programming. Lower-level concepts must be taught before the concepts of higher complexity can be introduced.

Dupras, Lemay, and Mili (1984) suggest that programming correctness is the focal point of the beginning course. They explain that, "the view that the art of programming can't be separated from the science of correct programming . . . program correctness must be omnipresent from the very first contact of the student with programming" (Dupras, Lemay, and Mili 1984, 151). They recommend emphasis in programming methodology as well as programming languages.

Soloway (1986) points out the techniques for programming methodology versus teaching only programming language syntax and semantics. He states that students should be given explicit instruction in vocabulary terms such as mechanism, explanation, goal, plan, roles of programming discourse, and plan composition methods. These techniques, he suggests, can also be advantageous in other problem-solving situations. In another course curriculum study, Bulgren and Wetzel (1982) recommend PASCAL as the programming language choice, instead of FORTRAN, for an introductory course in programming. The PASCAL programming language offers support for structured programming.

Teaching Methodology and Students' Performance in Programming

The outstanding technique in teaching methodology is the problem-solving approach (Bulgren 1987; Campbell 1984; Cook 1980; Gabrin 1982; Meinke 1981; Whipkey 1984). In her study, Campbell (1984) points out that preliminary programming courses which emphasize problem-solving techniques have a significantly positive effect on subsequent performance in traditional PASCAL courses. Most of the problem-solving methods emphasize algorithm development as a major step in programming development (Campbell 1984; Culik and Rizki 1983; Whipkey 1984). Other important techniques are structure programming, top-down design, and stepwise refinements (Campbell 1984; Cook 1980).

Other methodologies include the use of no machine-specific languages, suggested by Bulgren (1987), for the purpose of problem solving. For the workshop environment, the evolutionary method of try, test, and repeat is recommended (Smith 1981). This method relies on the techniques of teaching-by-example, intuition, and reasonableness.

CAI is also an effective method of teaching computer programming concepts. Using this method, students interactively use computer software to drill and practice computer programming concepts. In some cases, CAI can speed up the students' learning process (Aikin 1981).

Summary

From the existing research it is clear that programming ability of students is predictable. The success rate of students can be predicted by using a combination of past and present performance at secondary school and college level, performance in mathematics, and personality traits such as impulsive-reflectivity. It seems apparent that little research has been done in the area of sequencing, logic, and other problem-solving skills on the performance of students in an introductory programming course.

Sequencing, although an essential skill in programming, has so far only been correlated to the success rate of students in programming. The association of sequencing has never been studied as an isolated factor in assisting students to learn programming. Research concerning the psychological and personality traits of programming students is limited, and the exact role of personality traits on programming success rates of students is not clear. Future research into the relationship of cognitive skills and personality traits and learning programming is essential for the development of effective instructional material for programming courses.

CHAPTER III

RESEARCH METHODS

Introduction

The population, experimental group, experimental procedure, and experimental treatment are discussed in this chapter. The research type and steps involved in conducting the research are reviewed in the experimental procedure section. The computer-aided instruction (CAI) treatment tool, BASIC programming language test, and the method of applying the treatment are fully discussed in the experimental treatment section. Discussion of the treatment of data and a chapter summary are also included.

Population

The population for the experimental and control groups included students from two sections of an Introductory BASIC Computer Programming Language course. This course was conducted at Winona State University in Minnesota during a fall quarter.

Experimental Groups

Seventy students from two sections of an Introductory Basic Programming Language course at Winona State University were participants in the study. Students were divided into equivalent experimental and control groups. To achieve equivalent experimental and control groups, students' scores on a pretest of the BASIC programming language were sorted from the top score to the bottom score. Then, from the top of the list, each pair of students was randomly assigned to an experimental or control group. A toss of a coin was used for randomly dividing each pair of students. The sample group of students was divided into two groups of thirty-five each. The size of the groups was large enough to allow for drop-outs without affecting the statistical analysis. The first group was designated as the experimental group and the second was designated as the control group.

Experimental Procedures

One-factor experimental research using pretest-posttest control group design with a matching techniques used for this study. Experimental research is typically used to explore the effect of a treatment (i.e. new curriculum, new method of teaching, etc.) on a population. This study is most closely modeled after the experimental research design because it investigates the association of CAI drill in sequencing with the ability of students in introductory computer science courses to learn programming. The pretest-posttest control group design with matching technique was selected from among the variety of designs in experimental research as the model. Pretest-posttest control group design is commonly used in educational research.

Walter Borg (1983) notes that experimental design effectively controls for the eight threats to internal validity: history, maturation, testing, instrumentation, regression, selection, mortality, and interaction effects. In order to obtain additional precision in the statistical analysis of the data, the matching technique was added to the experimental design.

The purpose of the matching technique is to divide subjects into two equivalent groups--control and experimental. Although many procedures can be used to achieve matching groups, the ranking method was chosen for this study. The ranking method sorts students' scores from the pretest. It measures the dependent variable or variables correlated with the dependent variable in ascending order and then it randomly divides each pair of two top scores from the sorted list into two groups. The result is two matched groups that are fairly equivalent in background. One of the groups is used as the control group and the other is used as the experimental group. The steps in pretest-posttest control group with matching technique used for this study are as follows:

> 1. Administer measures of the dependent variable or of a variable closely correlated with the dependent variable to the research subjects.
> 2. Assign subjects to matched pairs on the basis of their scores on the measures described on step 1.
> 3. Randomly assign one member of each pair to the experimental group and the other member to the control group.
> 4. Expose the experimental group to the experimental treatment and, if appropriate, administer a placebo or alternative treatment to the control group.

5. Administer measures of the dependent variables to the experimental and control groups.
6. Compare the performance of the experimental and control group on the posttest(s) using tests of statistical significance. (Borg 1983, 44)

A BASIC programming language test, described in the following sections, was administered to all of the students in both sections of the class as a pretest. Then, according to the procedure explained previously, students were paired and randomly divided into experimental and control groups based on their pretest scores. After students were divided into two groups, the experimental group was exposed to CAI drill in sequencing treatment while the control group was exposed to an alternative treatment. The experimental treatment is discussed in detail in the experimental treatment section. The students of both classes then completed the normal activities of their Introductory BASIC Programming class. At the end of the term, a test on the BASIC programming language was administered as the posttest to both sections of the class. The scores of the experimental students were compared with those of the control students' using analysis of variance as the tests for statistical significance.

Experimental Treatment

This section includes a discussion of the sequencing CAI treatment instrument, the sequencing CAI treatment procedure, and the BASIC programming language test. The instrument used for the experimental treatment was a CAI drill which was designed and developed for practicing sequencing cognitive skills.

The CAI treatment on sequencing was based on a drill-and-practice strategy which was designed to improve students' sequencing abilities. The sequencing CAI drill program is presented in Appendix A. Many researchers have indicated that CAI can be at least as effective as traditional instruction and that using CAI typically reduces the required time for students to master the material (Aikin 1981). Another reason why the CAI method was used for this study was that no specific lecture time for sequencing was required. This approach provided the students freedom to practice the sequencing lessons at their convenience. The CAI drill on sequencing was carefully designed to allow all students to complete the sequencing problems at their own pace.

The CAI drill on sequencing is made up of forty multiple-choice questions in sequencing patterns. These questions were taken from a pattern sequencing section of a Test of Foreign English Language practice manual and a sequencing instruction tool by Alexander B. Cannara (1976) at Stanford University. The drill was designed for the IBM PC or compatible machine using IBM DOS 3.3 and higher. The CAI drill lesson on sequencing includes all of the necessary instructions for students. At the beginning of the drill, the student's name is requested and verified for correctness. Names are requested to facilitate personal interaction with the students and for recording student performance. Two pages of instructions and examples are presented at the beginning of the CAI lesson. An explanation of how to answer the multiple-choice questions of the sequencing drill and practice exercises is also included. The practice exercises and forty sequencing drill questions are presented, one at a time.

Students are provided with two opportunities to choose the correct choice before the computer presents the correct answer. The correct answer presented by the computer is accompanied with an explanation. For every question that is answered correctly on the first try, two points are gained; on the second try, only one point is gained. Once the computer presents the correct answer, students do not gain any points for that question. However, the questions that are answered incorrectly in the first round are recorded and are presented later for review. At the end of the lesson, the computer presents the questions that were answered incorrectly by the student. This allows the student a second chance to answer the questions. This technique helps to motivate students to pay attention to every question. Each student's name and final score were recorded along with the student's response to every question.

A frequent difficulty encountered with experimental research is a lack of subjects for study. It is often difficult to obtain enough student volunteers to establish acceptable control and experimental groups, particularly if the study requires a considerable amount of time. In this study, two sections of an introductory BASIC computer programming course at Winona State University were selected as participants. The research procedure was conducted by a computer science professor in two sections of an Introductory Computer Science course. The Introductory Computer Science course primarily teaches BASIC programming language. Research activities were presented to the students as part of course activities; and therefore, all of the students were involved in the research. No differences existed in the teaching styles for the sessions of the BASIC programming language course (both were taught by the same instructor).

A week after students completed the pretest in BASIC programming language, they were asked to participate in a laboratory exercise using educational software. Computer diskettes containing the sequencing CAI drills were given to the experimental student group. Members of the control group were provided with randomly selected software such as games and puzzles. The lab provided each student with a PC XT or a PC AT personal computer. Under a supervised laboratory environment, subjects in the experimental group were asked to complete sequencing exercises while subjects in the control group were asked to complete its software games. All of the diskettes were collected at the completion of the student session. Sequencing CAI drills were separated and examined for completion scores by the lab supervisor. All of the subjects in the experimental group completed the sequencing CAI exercises. The CAI drill and practice exercises on sequencing took students two to three hours to complete. The control students spent about the same amount of time using the randomly-selected software.

The BASIC computer programming language test which was used as the pretest (at the beginning of the quarter) and as the posttest (at the end of the quarter) was designed by the computer science department at Winona State University for evaluating students in BASIC programming language courses. A sample of the test is presented in Appendix B. Students were required to take the tests as part of the course. The BASIC programming language test was administered by the instructor in the class session during the first week of the term as the pretest and during the last week of the term as the posttest. All of the tests were scored by the same instructor, with 155 as the highest possible score.

Content validity of the test was previously verified by the Computer Science Faculty at Winona State University. The test had been in use by the BASIC Computer Programming Language course faculty for the evaluation of students for several years. The test was originally modified by the designer to obtain the optimum reliability as well. Consistency in test results over a period of several years was proof of its reliability. A coefficient of stability measure was calculated in a test-retest situation for the purpose of estimating the reliability of this test by Winona State University. The test was administered twice in a one-month period. A correlation coefficient of .76 was reported for the test; thus, the reliability of the test was confirmed.

Treatment of Data

Scores on the pretest in the BASIC programming language were used to construct two matching groups--experimental and control. However, posttest scores were statistically analyzed for the validation of the three stated hypotheses.

Hypothesis 1 predicted that students in the experimental group would score higher on the posttest than students in the control group. Therefore, the posttest scores of students in the experimental group were expected to be significantly higher than the posttest scores of students in the control group. Analysis of variance was used to verify this hypothesis.

Hypothesis 2 predicted that students in the treatment group who performed below average on the pretest would have a higher gain on the posttest than students in the treatment group who performed above average on the pretest. Because the students' gain on the posttest was being questioned, the appropriate statistical procedure to verify this hypothesis was analysis of variance.

Hypothesis 3 predicted that students in the treatment group who scored lower than average on the pretest would score higher on the posttest than students in the control group who scored lower than average on the pretest. Analysis of the variance was also used to verify this hypothesis.

Summary

In summary, the population of the study was made up of students in two sections of an Introductory BASIC Programming course at Winona State University. Students were divided into experimental and control groups based on their pretest performance on a BASIC programming language test. Students in the experimental group were exposed to a CAI drill on sequencing, while students in the control group was exposed to alternative software. Both groups of students completed the requirements of their Introductory BASIC Programming classes. At the end of the term, a posttest of the BASIC programming language was administered to all of the students. Analysis of variance was used to compare the difference of the experimental group and the control group posttest scores for all three hypotheses.

CHAPTER IV

PRESENTATION OF FINDINGS

Introduction

The findings of the study are presented in this chapter. A review of the experiment is also presented in this section. Pretest findings are presented and discussed; the three hypotheses are presented and discussed individually; and tables of statistical data for the hypotheses are presented and discussed. A summary of the findings is presented at the end of the chapter.

Experiment Review

The data obtained through procedures described in Chapters I and III are from a single-factor experimental research design where the independent variable was the teaching method and the dependent variable was the performance of the students on a BASIC programming language test. Three hypotheses were investigated in this study. The statistical procedure used to evaluate the three hypotheses was analysis of variance. A test of the BASIC programming language was administered in both the pretest and posttest. Test scores ranged from 0 to 155 points. The pretest was given to students in two sections of an Introductory Computer Science course. The total number of students in the two sections of the course was 70, 10 students dropped by the end of the quarter.

Because only two groups were used for comparison, both t-test and ANOVA worked well for the statistical analysis. As stated by Slakter (1972), ANOVA is appropriate for $j = 2$, as well as for $j > 2$. A person may well wonder whether a two-sample case should be handled by using the t distribution or methods of using the F distribution. Actually, for the test of H0: $u1 = u2$, the two methods are equivalent. However, in this study, ANOVA was chosen for the statistical procedure because it compares the groups' overall distribution and not just their means.

The statistical software package that was used for this study was ySTAT (1987). An important point to remember in comparing groups for the test of significant difference is the validation of the underlying assumptions for such tests. These basic assumptions are normality of population distribution, equal variance, and independence. Slakter (1972, 652) states the fragility of the assumption: "While the assumptions of a model can be satisfied in only one way, they can be violated in infinite number of ways." The three assumptions are examined briefly in this section.

The first assumption is the normality of the population distribution. This assumption is considered safe with the populations encountered in educational research (Slakter 1972, 652). Equal variance is achieved by dividing the sample into the same size as closely as possible. Independence is accomplished by careful supervision. This ensures that one subject's responses do not affect the responses of other subjects. In the following sections, the result of the pretest is examined. A statistical analysis and discussion of each of the three hypotheses is then presented.

Performance of Students Prior to Treatment

A pretest was administered to students in both classes. The mean, standard deviation, F value and the variance between the pretest scores of the experimental and control groups are presented in Table 1.

Table 1.--Difference in Pretest Scores for Experimental and Control Groups

Group	N	Mean	SD	F	P
Experimental	35	7.1	12.99	0.001	0.977
Control	35	7.2	12.42		

There was a total of 70 students in the two sections of the course. The scores of students in the pretest were used to divide students into two matched groups--experimental and a control group. There were 35 students in each group. From the possible 155 points, the students' average was around 7. As shown in Table 1, students' scores were on the low side of the scale. The results show that although some students were knowledgeable in the BASIC programming language, the majority of the students were not familiar with the BASIC programming language before taking the Introductory Computer Science course. The mean score on the pretest for the students in the experimental group was 7.1, and the students' mean score for the control group was 7.2. These scores indicate the similarity of the two groups.

Simple analysis of variance with the two groups' scores was conducted. ANOVA with the P value of .97 indeed supported the null hypothesis that both groups were very similar.

Performance of Students Following Computer-Assisted Instruction

Hypothesis 1 was designed to determine whether college students who completed a computer-aided instruction (CAI) drill in sequencing and then completed an introductory BASIC programming course would perform significantly better in programming than the students who completed only the introductory BASIC programming course. As described in Chapter III, the experimental group participated in the CAI drill in sequencing and practice exercises before attending the BASIC programming language course. A posttest in the BASIC programming language was administered to both groups at the end of the quarter. The mean, standard deviation, F value, and variance between the posttest scores of the experimental and control groups are presented in Table 2.

Table 2.--Difference in Posttest Scores for Experimental and Control Groups

Group	N	Mean	SD	F	P
Experimental	30	125.30	14.53	0.896	0.344
Group	30	121.87	13.55		

The experimental group's mean on the posttest was 125.30, which was 3.43 points higher than the control group's mean of 121.87. The control group's mean was slightly lower than the experimental group's mean. To compare the experimental group's performance in the posttest with the control group's performance, an analysis of variance was conducted with two samples of the experimental group's and the control group's posttest scores.

The F value is significant at P = .34, and is beyond the expected alpha value of .050. Therefore, the null hypothesis is accepted and the difference between the experimental and the control groups' performance in the posttest is not significant.

The experimental group averaged slightly higher than the control group on the posttest of the BASIC programming language. This difference is not significant for the alpha level of .05. Obviously, the effect of participating in a CAI sequencing drill on the ability of students to learn programming is not a significant one. There is a 34 percent chance that the difference between both groups' mean scores is due to random errors. The difference indicates that participating in the CAI drill on sequencing had a positive effect on the learning behavior of students, but not a statistically significant effect. Some of the reasons for the lack of a significant difference between the experimental and the control groups' performance on the posttest could be isolation of the sequencing factor, the limitation of the sample size, and the short treatment period dictated by the study.

Hypothesis 2 predicted that treatment students who performed below average on the BASIC programming language pretest would have higher gains on the posttest than treatment students who performed above average on the pretest.

The pretest of the BASIC programming language scores of 35 students in the experimental group were 7.1. As the results of the pretest revealed, most of the students scored zero points; thus, the grading distribution was heavily skewed to the low side of the grade range. Twelve of the experimental students scored above average and, thus, made up the above average group. Eighteen experimental students scored below average and, thus, made up the below average group. Five students in the experimental group dropped out of the study during the quarter. The results of the analysis of variance for the pretest scores of above average and below average experimental groups are presented in Table 3.

Table 3.--Difference in pretest scores for Above Average and Below Average Experimental Groups

Group	N	Mean	SD	F	P
Above average	12	19.75	16.46	27.207	0.00
Below average	23	1.70	2.75		

Analysis of variance revealed a P value of zero for the pretest scores of the two groups. This testifies to the significant difference between the above average group and the below average group performance on the pretest. As mentioned earlier, 5 students in the below average group dropped out of the study, which left the below average group with 18 students against the 12 students in the above average group. The mean, standard deviation, F value and variance between the posttest scores of the above average and below average experimental groups are presented in Table 4. The above average experimental group's mean on the posttest was 6.03 higher than the below average experimental group's mean.

Hypothesis 2 predicted that experimental students who performed below average on the pretest would have a higher gain on the posttest than experimental students who performed below average. A simple analysis of the variance was performed between the experimental below average group's

Table 4.--Difference in Posttest Scores for above Average and Below Average Experimental Groups

Group	N	Mean	SD	F	P
Above average	12	128.92	12.80	1.251	.263
Below average	18	122.89	15.45		

posttest scores and the experimental above average group's posttest scores. The results of the study indicate that the difference between the two groups' performance on the posttest of the BASIC programming language is not significant. Therefore, this hypothesis is rejected.

Analysis of the variance failed to provide a significant F ratio between the posttest results of the experimental below average group and the experimental above average group. The fact that both the experimental below average group and the experimental above average group performed noticeably well, was due to the effectiveness of the Basic Programming Course. The reason that the experimental above average group scores were slightly higher than the experimental below average group may be attributable to higher prior knowledge of the above average group versus the below average group. Participation in the CAI drill on sequencing did not cause a significantly higher gain in scores of the experimental below average group than that in scores of the experimental above average group.

Hypothesis 3 predicted that experimental students who performed below average on the BASIC programming language pretest would perform better on the posttest than control students who performed below average on the pretest.

The experimental and the control group pretest scores were each divided into above average and below average groups. The scores of both groups in the pretest were compared using analysis of variance. Data for this comparison are presented in Table 5. The mean of the experimental below average group and the control below average group pretest was 1.70, which

Table 5.--Difference in Pretest Scores for Experimental Below Average and Control Below Average Groups

Group	N	Mean	SD	F	P
Experimental below average	23	1.70	2.75	0.00	1.00
Control below average	23	1.70	2.75		

indicate the similarities of the two groups. The P value of 1.00 also indicates that the two groups are very similar.

After the BASIC programming language course was completed, 18 students in each of the experimental and control below average groups participated in the posttest. The mean, standard deviation, F value, and variance between the posttest scores of the below average experimental and below average control groups are presented in Table 6.

Table 6.--Difference in Posttest Scores for Experimental Below Average and Control Below Average Groups

Group	N	Mean	SD	F	P
Experimental below average	18	122.89	15.45	0.50	0.479
Control below average	18	119.56	12.70		

The experimental below average group mean is higher than the control below average group mean. To substantiate the significance of this difference, a simple analysis of variance was calculated with the two samples of experimental and control below average groups' scores. The result of analysis of variance indicates a P value of .479 which (Table 1) is beyond the alpha toleration level of 0.05. Therefore, the null hypothesis is accepted. Thus, there is no significant difference between the experimental and control below average groups' posttest performance.

Students in the below average group who participated in the experiment by completing a CAI drill on sequencing did not performed better than the below average students who did not participate in the CAI drill on sequencing. The short treatment period may have contributed to the lack of a significant improvement in the experimental group.

Summary of Major Findings

The major findings of this study were the following:

1. The experimental group did not have a significantly higher gain in the posttest than the control group.

2. The below average experimental group did not have a significantly higher gain on the posttest than the above average experimental group.

3. The below average experimental group did not have a significantly higher gain on the posttest than the below average control group.

CHAPTER V

SUMMARY, DISCUSSION, CONCLUSIONS, AND RECOMMENDATIONS

Computer Science programs continue to be a demanded field of study as computers are applied in new industries. The pressure of high demand for computer programming courses and inadequate information about computer programming prerequisite skills contribute to the current lack of effective computer science advisement programs. The fact that a number of students do not benefit from their computer science courses because they are not adequately prepared has resulted in the low utilization of already limited resources.

An increased understanding of the skills needed to learn computer programming would be a valuable aid to computer science departments in advising students and could improve the success rates in computer science courses. Based on Konvalina, Wileman, and Stephens's (1983) discovery that sequencing, among other factors, is related to students' success rates in introductory programming courses, the purpose of this research was to determine whether or not student success rate would improve in computer programming after completing a computer-aided instruction (CAI) sequencing drill and practice exercises.

For this study, a CAI program was developed to teach pattern recognition sequencing. Two groups of students were used, one as an experimental group and one as the control group. A BASIC programming language test was used to measure the students' knowledge of BASIC programming language.

After students' scores were obtained in a pretest, they were ranked from highest possible score (155) to lowest possible score (0). The ascending list of students' pretest scores was divided into two approximately equivalent groups. These matched groups were assigned as experimental and control groups. Sequencing CAI software were given to the experimental group and games and puzzle software were given to the control group as a computer laboratory assignment. After students had completed the laboratory assignment, both groups attended the BASIC programming course. At the completion of the course, a BASIC programming test, which was used as the pretest, was administered to both groups as the posttest. The data collected from the posttest were statistically studied for each of the three hypotheses. Analysis of variance was used to compare the posttest scores of the experimental group with scores of the control group for the three hypotheses. Analysis of variance with two samples was calculated for the experimental and control groups for each hypothesis.

Summary of Major Findings

Although the findings for this research are limited, they are important as they relate to the performance of students in introductory computer programming courses.

1. No significant difference was found between experimental students' posttest scores and control students' posttest scores for the alpha level of 0.05 in an analysis of variance study.

2. Experimental students who scored below average on the pretest did not show a significant gain on the posttest when compared to the experimental students who scored above average on the pretest.

3. Students in the experimental below average group did not have a significantly higher mean than students in the control below average group on the posttest of BASIC programming language.

Discussion of Findings

1. The most important finding of this research may be that sequencing CAI as a complementary educational tool for the introductory computer programming students did not have a statistically significant impact on students' performance in the course. This finding resulted in the rejection of the hypothesis that sequencing exercises would have a positive significant effect on the ability of students to learn programming. This may be due to the fact that there are many cognitive skills involved in programming. Sequencing,

which is one of them, did not provide enough impact by itself to significantly improve experimental students' scores which compared to control students' scores. However, the comparison of the means for both groups reveals that participation in a CAI drill on sequencing did have some positive effect on experimental students' scores. It is possible that participation in a CAI drill on sequencing combined with other cognitive programming skill treatments such as logic, could result in significant improvement.

2. It was hypothesized that students in the experimental group who scored below average on the pretest would have a higher gain on the posttest than experimental students who scored above average on the pretest. It was found that neither group scored significantly higher on the posttest of BASIC programming language. Therefore, participation in the CAI drill on sequencing did not cause a statistically significant gain in the posttest scores of below average experimental students when compared with the scores of above average experimental students who used the same CAI drill. This seems to indicate that the BASIC programming course was effective in raising the experimental below average students' knowledge of BASIC programming to a level comparable to the experimental above average students. Therefore, participation in the CAI drill on sequencing did not noticeably improve the experimental below average students' performance in BASIC programming versus the experimental above average students' performance.

3. The experimental below average group had a nonsignificant but higher mean on the posttest than did the control below average group. The hypothesis predicted that the experimental below average group would have a significantly higher performance on the posttest than the control below average group. This hypothesis was rejected. Participation in a CAI drill on sequencing did not significantly improve the ability of the experimental below average group to learn programming when compared with the control below average group. The higher performance by the experimental below average group (3.33 points) is an indication that participation in a CAI drill on sequencing had a nonsignificant positive effect on students' performance. The combination of the CAI drill with other computer programming cognitive skills, such as logic, might have a significant effect on the ability of students to learn programming.

Conclusions

1. Participation in CAI drills on sequencing alone does not appear to have enough impact to cause a significant improvement in students' ability to learn programming.

2. Participation in CAI drills on sequencing alone did not have a significant impact on the performance of the experimental below average group on the test of BASIC programming language versus the control below average group.

3. Participation in CAI drills on sequencing alone did not improve the performance of experimental below average students significantly over the experimental above average students on the test of the BASIC programming language.

4. Participation in CAI drills on sequencing did not negatively effect students' ability to learn programming.

Recommendations

Based on the findings of the study, the following recommendations are made:

1. A complementary study is recommended which uses the same tools but applies the CAI treatment more than one time during the term of the BASIC programming language course. The use of a larger group of students is also advised.

2. A study on the combination and interactions of multiple computer programming cognitive skills such as sequencing, logic, and algebraic manipulation is recommended.

3. A comprehensive study of other prerequisite areas for computer programming courses is recommended.

4. A similar study with an earlier posttest is also suggested.

APPENDIX A

SEQUENCING COMPUTER-AIDED INSTRUCTION DRILL PROGRAM

```
100       NQ = 40
200       DIM W(40)
300       D$ =  CHR$ (4)
400       FOR I = 1 TO 40:W(I) = 0: NEXT I
500       PRINT D$;"OPEN SEQUENCE.DAT"
600       PRINT D$;"CLOSE SEQUENCE.DAT"
700       PRINT D$;"PR#3"
800       HOME
900       INPUT "PLEASE ENTER YOUR LAST NAME AND FIRST NAME:
          ";N$
1000      HOME : PRINT "HI ";N$
1100      PRINT "THE FOLLOWING IS A DRILL AND PRACTICE
          PROGRAM IN SEQUENCING. THE PROGRAM WILL"
1200      PRINT "PRESENT YOU WITH TOTAL OF 40 MULTIPLE CHOICE
          QUESTIONS. FOR EACH QUESTION FOUR"
1300      PRINT "ANSWERS ARE GIVEN FROM WHICH YOU HAVE TO
          CHOOSE THE CORRECT ONE."
1400      PRINT "EXAMINE THE SEQUENCE OF THE LETTERS OR
          NUMBERS AND FIND THE PATTERN THEY FOLLOW. BASED ON
          THE PATTERN CHOOSE THE ANSWER TO FILL UP THE BLANK
          SPACES."
1500      PRINT "IF YOU"
1600      PRINT "CHOOSE THE WRONG ANSWER ON THE SECOND
          CHANCE, COMPUTER WILL DISPLAY THE"
1700      PRINT "CORRECT ANSWER AND SOME EXPLANATION."
1800      PRINT "USE OF PAPER, PENCIL AND CALCULATOR IS
          RECOMMENDED."
1900      PRINT
2000      PRINT "GOOD LUCK"
2100      VTAB 24: INVERSE : PRINT "PRESS ANY KEY TO
          CONTINUE";: NORMAL : GET C$: PRINT
2200      NC = 0
2300      FOR II = 1 TO 2
2400      RESTORE
2500      FOR I = 1 TO NQ
2600      FOR J = 1 TO 6
2700      READ A$(J): NEXT J
2800      READ CA$
2900      IF W(I) = 1 THEN  GOTO 5000
3000      NT = 1
3100      HOME : FOR J = 1 TO 5: PRINT A$(J): PRINT : NEXT J
3200      PRINT : PRINT "ENTER THE CORRECT CHOICE: ";: GET
          UA$
3300      IF UA$ <  > CA$ THEN  GOTO 4100
3400      IF W(I) = 1 THEN  GOTO 3800
3500      W(I) = 1
3600      NC = NC + 1
```

```
3700      IF NT = 1 THEN NC = NC + 1
3800      PRINT
3900      VTAB 23: INVERSE : PRINT "CORRECT ANSWER,
          CONGRATULATIONS!": NORMAL
4000      INVERSE : PRINT "PRESS ANY KEY TO CONTINUE";:
          NORMAL : GET C$: GOTO 5000
4100      IF NT = 2 THEN  GOTO 4700
4200      NT = NT + 1
4300      PRINT
4400      VTAB 23: INVERSE : PRINT "WRONG ANSWER, TRY
          AGAIN!": NORMAL  4500  INVERSE : PRINT "PRESS ANY
          KEY TO CONTINUE";: NORMAL : GET C$
4600      GO TO 3100
4700      PRINT : PRINT "CORRECT ANSWER IS: ";CA$: PRINT
          "EXPLANATION: "; A$(6)
4800      PRINT
4900      VTAB 24: INVERSE : PRINT "PRESS ANY KEY TO
          CONTINUE";: NORMAL : GET C$: GOTO 5000
5000      NEXT I
5100      NEXT II
5200      PRINT "NC = ";NC
5300      FOR I = 1 TO NQ: PRINT W(I): NEXT I
5400      PRINT D$;"APPEND SEQUENCE.DAT"
5500      PRINT D$;"WRITE SEQUENCE.DAT"
5600      PRINT N$;",",NC
5700      PRINT D$;"CLOSE SEQUENCE.DAT"
5800      HOME : SPEED= 20: VTAB 12: HTAB 20: PRINT "THANK
          YOU FOR YOUR TIME"
5900      VTAB 14: HTAB 25: PRINT "GOOD LUCK"
6000      SPEED= 255: PR# 0
6100      DATA  1. 2 7 10 15 18 23 26 31 34 39 _ _ _,A. 44 46
          51,B. 42 47 50,C. 34 23 14,D. 23 34 45,"ALTERNATE
          THE ADDITION OF 3 AND 5 TO A NUMBER TO GET THE NEXT
          ONE",B
6200      DATA  2.  ABABBABABBAB_ _ _,A. A B B,B. B B A,C. B
          A B,D. B B B,"ABABB IS BEING REPEATED, SO AFTER THE
          LAST AB THERE  COMES A B B",A
6300      DATA  3.  9 A 8 C 7 E 6 _,A. 5,B. A,C. 3,D. G,"AS
          NUMBERS ARE DROPPING BY ONE THE LETTERS ARE
          ADVANCING BY TWO. TWO LETTER AFTER E IS LETTER G",D
6400      DATA  4.  9 12 11 14 13 16 15 18 _ _ _,A. 18 20
          19,B. 17 20 19,C. 12 13 20,D 20 18 17,"THE NUMBERS
          ARE GOING UP BY 3 AND DROPPING BY 1, SINCE 15 WAS
          INCREASED BY 3 TO 18 THEN THE NEXT NUMBER IS
          18-1=17, 17+3=20 AND 20-1=19",B
6500      DATA  5. B A D C C F E H G _ _ _,A. H I L,B. J I
          L,C. J L I,D. NONE OF THE ABOVE IN ALPHABETICAL
          ORDER, SO 3 LETTERS AFTER G IS J AND THE LETTER
          BEFORE J IS I AND NEXT IS L",B
```

```
6600      DATA  6.  QQLQQQQLLLQQLQQQQLLLQ_ _ _,A. QQQ,B.
          QLQ,C. LQQ,D. LQQ,"QQLQQQQLLL IS BEING REPEATED, SO
          THE NEXT THREE  LETTERS ARE QLQ",B
6700      DATA  7. 27 24 22 19 17 14 12 9 _ _ _,A. 7 4 1,B. 4
          2 3,C. 7 4 2,D. NONE OF THE ABOVE,"SUBTRACT 2 AND 3
          ALTERNATIVELY,  9-2=7, 7-3=4 AND 4-2=2",C
6800      DATA  8.  A Z B Y C X D _ _ _,A. W A D,B. W E V,C.
          B S E,D. NONE OF THE ABOVE,"THE LETTERS ARE
          ALTERNATING FROM THE BEGINNING AND THE END OF THE
          ALPHABET",B
6900      DATA  9.  32 27 29 24 26 21 23 _ _ _,A.  18 20
          15,B. 20 23 21,C. 17 20 22,D. 18 20 16,"THE PATTERN
          IS SUBTRACTING 5 AND ADDING 2, SO THE NEXT NUMBERS
          ARE 18 20 AND 15",A
7000      DATA  10.  1 12 121 1212 12121 _ _,A. 121211
          21112,B. 121211 1212121,C. 121212 1212121,D. NONE
          OF THE ABOVE,"PAD THE PREVIOUS NUMBER ALTERNATELY
          WITH 1 AND 2",C
7100      DATA  11.  8 10 13 17 22 28 35 _ _ _,A. 43 50 60,B.
          40 42 45,C. 43 52 62,D. 43 52 63,"THE NUMBERS ARE
          INCREMENTING BY THE PREVIOUS INCREMENT PLUS 1 WITH
          2 AS THE FIRST INCREMENT. THE CORRECT ANSWER IS 43
          52 AND 62",C
7200      DATA  12.  147 144 137 141 138 131 135 132 125 _ _
          _,A. 129 126 119,B. 130 131 132,C. 129 17 119,D.
          129 126 120,"THE NUMBERS ARE DECREASING BY 3 AND BY
          7 AND INCREASING BY 4",A
7300      DATA  13.  A Z C X E V G _ _ _,A. H I T,B. T I R,C.
          I T R,D. T I S,"THE LETTERS ARE ALTERNATING FROM
          THE BEGINNING AND END OF THE ALPHABET ",B
7400      DATA  14.  J 1 P 3 M 5 J 8 P 1 M 3 J 5 P 8 M 1 J 3
          _ _ _,A. M 3 J,B. P 3 J,C. P 3 M,D. P 5 M,"LETTERS
          JP AND M ARE BEING REPEATED WHILE THE NUMBERS 1 3 5
          AND 8 ARE BEING REPEATED",D
7500      DATA  15.  1 2 6 24 120 _ 5040 _,A. 640 320320,B.
          700 42320,C. 720 40320,D. NONE OF THE ABOVE,"120 IS
          MULTIPLIED BY 6 TO GET 720, 720 IS MULTIPLIED BY 7
          TO GET 5040 WHICH IS MULTIPLIED BY 8 TO GET
          40320",C
7600      DATA  16.  1 3 4  7 11 18  2 30 32  5 8 13  5 12 _
          3 18 _,A. 17 19,B. 17 21,C. 19 20,D. 21 17,"THE
          ADDITION OF THE FIRST TWO NUMBERS IN EACH GROUP
          RESULTS IN TO THE THIRD NUMBER FROM THE SAME
          GROUP",B
7700      DATA  17.  15 4 19  5 80 85  11 2 13  6 3 _  4 _ 14
          _ 8 29,A. 9 10 21,B. 10 12 2,C. 9 12 20,D. 9 10
          20,"THE ADDITION OF THE FIRST TWO NUMBER IN EACH
          GROUP RESULTS IN TO THE THIRD NUMBER FROM THE SAME
          GROUP",A
```

```
7800      DATA  18.  2 3 5 8 13 21 _ _ _,A. 34 54 80,B. 23 34
          55,C. 18 23 34,D. 34 55 89,"THE ADDITION OF TWO
          CONSECUTIVE NUMBERS GIVES THE NEXT NUMBER",D
7900      DATA  19.  1 2 3 6 11 20 37 _ _ _,A. 57 94 165,B.
          68 125 230,C. 63 123 228,D. NONE OF THE ABOVE,"THE
          ADDITION OF THREE CONSECUTIVE NUMBERS RESULTS IN
          THE FOURTH",B
8000      DATA  20.  5000 6000 5900 6900 6800 7800 7700 ___
          ___ ___,A. 8000 7000 6000,B. 8700 7600 9600,C. 8700
          8600 9600,D. 9600 8700 7000,"ADDING A 1000 AND
          SUBTRACTING 100 IS THE ALTERNATE PATTERN",C
8100      DATA  21.  13 18 20 19 24 26 25 30 32 _ _ _,A. 31
          35 37,B. 31 36 38,C. 32 35 37,D. 40 43 45,"DEDUCT
          1, ADD 5 AND ADD 2 ALTERNATIVELY IS THE GENERAL
          PATTERN",B
8200      DATA  22.  4 14 21 26 36 43 48 58 65 _ _ _,A.  70
          80 87,B. 70 72 75,C. 75 80 82,D. 70 80 83,"THE
          PATTERN IS TO ADD 10, 7 AND 5 ALTERNATELY",A
8300      DATA  23.  7 3 4 8 4 5 10 6 7 14 10 11 22 _ _ _,A.
          20 21 20,B. 23 21 22,C. 18 19 30,D. 18 19 38,"THE
          PATTERN IS TO DEDUCT 4 ADD 1 AND MULTIPLY BY 2
          ALTERNATELY",D
8400      DATA  24. 64 32 16 8 4 _ _ _,A. 2 1 1,B. 2 1/4
          1/2,C. 2 1 1/2,D. 1 2 4,"DIVIDE EACH NUMBER BY TWO
          TO GET THE NEXT ONE",C
8500      DATA  25.  1 11 20 _ 1 A _ TKA 1 _ 20 11 _ _ KTK
          _,A. 1 K 1 1 K K,B. 11 K 11 1 A A,C. A K 11 1 K
          A,D. NONE OF THE ABOVE,"FIRST TEN CHARACTERS ARE
          BEING REPEATED TWO TIMES",B
8600      DATA  26.  ACEG GACE EGAC CEGA ____ ____,A. ACGA
          GACE,B. CEAG ACEG,C. ACEG GACE,D. AGEC GAEC,"THE
          LAST LETTER OF EACH SET IS ROTATED TO THE BEGINNING
          OF THE NEXT SET OF LETTERS",C
8700      DATA  27.  XOXOOOO OXOXOOO OOXOXOO _______
          _______,A. XOXOXOO XXOOXXO,B. XOOXXX OOOXXXO,C.
          OOOXOXO OOOOXOX,D. NONE OF THE ABOVE,"THE LAST
          LETTER OF A GROUP IS MOVED TO THE BEGINNING OF THAT
          GROUP TO MAKE THE NEXT GROUP",C
8800      DATA  28.  2 4 3 6 5 10 9 18 17 _ _ _,A. 34 33
          66,B. 34 32 35,C. 32 30 35,D.34 33 65,"MULTIPLY BY
          TWO AND SUBTRACT ONE ALTERNATELY TO GET THE NEXT
          NUMBER",A
8900      DATA  29. 49 48 46 43 39 34 _ _ _,A. 28 21 11,B. 21
          13 28,C. 23 21 13,D. 28 21 13,"SUBTRACT THE NUMBER
          FROM THE PREVIEW PREVIOUS SUBTRACTOR PLUS ONE,
          SINCE 5 WAS SUBTRACTED FROM 39 TO GET 34 THEN
          34-6=28,28-7=21 AND 21-8=13",D
9000      DATA  30.  8 9 7 10 6 11 5 12 _ _ _,A. 4 11 3,B. 12
          13 11,C. 4 13 2,D. 4 13 3, "ALTERNATE THE ADDITION
```

```
        AND THE  SUBTRACTION OF THE PREVIOUS INCREMENTER OR
        SUBTRACTOR PLUS ONE, 12-8=4, 4+9=13 AND 13-10=3",D
9100    DATA  31.  1 3 6 8 16 18 36 _ _ _,A. 38 76 78,B. 30
        40 50,C. 36 18 1,D. 38 76 71,"ALTERNATE THE
        ADDITION OF TWO AND MULTIPLICATION OF TWO",A
9200    DATA  32.  5AA 10BB 12CD 17DG 19EK _ _,A. 21GH
        25LM,B. 24FP 26GV,C. 20CD 22 EF,D. 23EH 25JK
9300    DATA  "THE NUMBERS ARE INCREMENTED BY 5 AND 2, THE
        FIRST LETTER IS MOVED UP ONE AND THE SECOND IS
        INCREMENTED TO THE ALPHABETICAL POSITION OF THE
        FIRST AND SECOND LETTERS",B
9400    DATA  33. D G F H K J L O N P S R _ _ _,A. T W U,B.
        W V T,C. T W V,D. W V T,"THE SEQUENCE IS TO COUNT
        UP 3 LETTERS, BACK UP ONE LETTER AND COUNT UP TWO
        MORE LETTERS",C
9500    DATA  34.  A1 B2 D4 Z26 J10 C3 A1 G7 N14 M13 E_ A_
        B_ _3 _16,A. 3 4 2 D F,B. 5 1 2 C P,C. 3 4 1 C P
        C,D. 5 2 1 E D F," EACH LETTER POSITION IN THE
        ALPHABET FOLLOWS THE LETTER",B
9600    DATA  35.  5 10 20 40 80 160 _ _ _,A. 320 640
        1100,B. 200 300 1200,C. 320 600 1200,D. 320 640
        1280,"EACH NUMBER IS MULTIPLIED BY TWO TO GET THE
        NEXT NUMBER",D
9700    DATA  36.  120 115 109 102 94 __ __ __,A. 85 75
        64,B. 81 76 98,C. 99 106 111,D. 85 75 63,"SUBTRACT
        ONE PLUS THE PREVIOUS SUBTRACTOR FROM A NUMBER TO
        GET THE NEXT ONE",A
9800    DATA  37.  A B C _ A 1 2 3 2 1 A B C B A 1 _ 3 2 1
        _ _ C B A,A. C A 1 2,B. B 2 A B,C. C 2 A B,D. C 1 A
        B,"THERE IS A SYMETRIC POINT AT THE SECOND C
        LETTER",B
9900    DATA  38.  B M N _ B 2 13 14 13 2 B _ N M B 2 _ 14
        13 2 _ _ N M B,A. N M 14 M B,B. N M 12 B 2,C. 2 M
        13 B M,D. M M 13 B M,"THERE IS A SYMERTIC POINT AT
        SECOND N LETTER",D
10000   DATA  39. 2 6 18 54 162 _ _,A. 324 648,B. 486
        1458,C. 100 50,D. 81 40,"MULTIPLY EACH NUMBER BY
        TWO TO GET THE NEXT ONE",B
10100   DATA  40. A C E C E G E G I G _ _ _,A. I K I,B. J K
        J,C.K I K,D. I J I,"UP TWO, UP TWO, DOWN TWO IS
        REPEATED AS THE PATTERN",A
```

APPENDIX B

COMPUTER BASIC PROGRAMMING LANGUAGE TEST

COMPUTER SCIENCE BASIC PROGRAMMING TEST

NAME ____

TRUE or FALSE Questions: (two point each)

____ 1. If X$ = "ABCDEFGH", then LEN(LEFT$(X$, 4)) is equal to 8.

____ 2. The statement MID$(A$, 1, 4) and LEFT$(A$, 4) are the same in string values.

____ 3. If X$ has the value MADAM, then MID$(X$, 3, 3) has the value D.

____ 4. If A is any number, then LEN(A) gives the number of digits in A.

____ 5. When string fields or constants are compared, they must always contain the same number of characters.

____ 6. In the ASCII code, the letter T has a higher value than the letter Z in the collating sequence.

____ 7. The following example is a valid statement that contains the VAL function: 400 ON VAL(R) GOSUB 2000,3000,4000

____ 8. The statement 100 PRINT CHAR$(66) prints a B.

____ 9. A BASIC programming line may contain more than one BASIC command.

____10. String constants appearing in relational expressions must be quoted.

____11. The statement 100 PRINT ASC (CHR$ (67)) will print C.

____12. It is possible to assign both string variables and numeric variables using the same READ statement.

13. Show the output for the program below. Be sure to use exact columns.

```
10 LET A$ = "MARY HAD A LITTLE LAMB"
20 FOR I = LEN(A$) TO 1 STEP -3
30 LET B$ = MID$(A$,I,1)
40 PRINT B$;
50 NEXT I
60 PRINT
70 PRINT "MILLER"; MID$(A$,9,9)
80 END
```


----

14. Rewrite the following FOR NEXT loop code using a WHILE loop.

```
100 FOR I = 1 to 50 STEP 5
120 PRINT I, I * I
130 LET M = I + 4
140 NEXT I
150 END
```

15. Write a BASIC program that will count the numbers of "E"s in a string that is input at run time.

16. Show the output of the following program given below.

```
100 REM PRINT USING STATEMENTS
110 REM
120 LET A$ = "MARY LOU RETTON"
130 LET A = 1234.567 : LET B = 105 : LET C = 195.3
140 LET D = 422.39
150 LET B$ = "###   ###.##"
160 REM THE PRINT USING STATEMENTS
170 PRINT USING "!"; A$
180 PRINT USING "\  \"; A$ : REM 2 spaces between
      slashes
190 PRINT USING "&": A$
200 PRINT USING "#"; A
210 PRINT USING "####.## "; A, B
220 PRINT USING "$$####.##"; C
230 PRINT USING B$; C, B
240 PRINT USING "**#########"; A, B
250 PRINT USING "B + C IS ####.##"; B + C
260 PRINT USING "####,.##" A
300 END
```

```
-----------------------------------------------
-----------------------------------------------
-----------------------------------------------
-----------------------------------------------
-----------------------------------------------
-----------------------------------------------
-----------------------------------------------
-----------------------------------------------
-----------------------------------------------
```

17. The ____________ command erases the screen and moves the cursor to the upper left hand corner.
18. The ____________ statement allows you to position the cursor on the screen at a certain row and column.
19. What do each of the following commands do?
    (a) KILL
    (b) SHELL

20. Complete the program which will create a file MONTHS with the given data (month (M$)) and (days (D) for each records.

```
100 DATA JAN,31,FEB,28,MAR,31,APR,30,MAY,31
105 DATA JUN,30,JUL,31,AUG,31,SEP,30,OCT,31
110 DATA NOV,30,DEC,31
115 REM STATEMENT TO OPEN THE FILE FOR OUTPUT
120 OPEN ______________________________________
130 FOR i = 1 TO 12
140 READ m$, d
150 WRITE _____________________________________
160 NEXT I
170 CLOSE ______________
200 END
```

21. Write a program which will read the MONTHS file and print out the month and days of each month in the file. (Use file created in question 20.) Use a WHILE loop.

22. Consider the following program:

```
100 PRINT "TYPE IN THREE NUMBERS"
105 INPUT A, B, C
110 IF B = 0 THEN 500
120 IF C > 25 THEN 180
130 IF A <= 10 THEN 160
140 GOSUB 300
150 GOSUB 100
160 GOSUB 200
170 GOTO 100
180 GOSUB 100
190 GOTO 100
200 LET B = A + C
210 LET A = 2 * A
220 LET C = 0
230 GOSUB 400
240 RETURN
300 LET B = A - C
310 PRINT B
320 RETURN
400 PRINT A; B; C
410 RETURN
500 PRINT A; C; "STOP"
999 END
```

What output will be printed for each of the following sets of inputted data.

(a) Type in three numbers
? 5, 20, 10

(b) Type in three numbers
? 20, 5, 12

(c) Type in three numbers
? 10, 20, 30

(d) Type in three numbers
? 0, 0, 0

23. Show the output for the program below. Be sure to be exact on the rows and columns.

```
10 PRINT "7777777"
20 LET N = 1
30 LET S = 6
40 PRINT TAB( S ); N + S
50 LET N = N + 1
60 LET S = S - 1
70 IF S >= 1 THEN GOTO 40
80 END
```

```
---------------------------------------------
---------------------------------------------
---------------------------------------------
---------------------------------------------
---------------------------------------------
---------------------------------------------
---------------------------------------------
---------------------------------------------
```

24. Complete the BASIC statements needed to sort the month names in array MONTH in ascending alphabetic order.

```
100 DIM MONTH(12)
110 DATA JANUARY, FEBRUARY, MARCH, APRIL, MAY,
    JUNE,
115 DATA JULY, AUGUST
120 DATA SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER
130 FOR I = 1 TO 12 : READ M$(I) : NEXT I
140 REM SORT ROUTINE SELECTION SORT
```

25. Given:

```
A$ = "MARY HAD A LITTLE LAMB"  ASCII FOR "A" IS 65
B$ = "ABCDEFGHI"
C$ = "1988"
D$ = "1970"
 E = 20
 F = 70
```

(a) C$ + D$ = ____________________

(b) LEFT$(A$,8) = ____________________

(c) MID$(B$,2,1) + MID$(B$,9,1) + MID$(B$,7,1) =______

(d) RIGHT$(A$,11) = ____________________
(e) VAL(D$) + E = ____________________
(f) STR$(F) = ____________________
(g) CHAR$(F) = ____________________
(h) LEN(A$) = ____________________

26. The statement below will print a random integer in the range of ________ through _________

```
100 PRINT INT( 21 * RND(4) + 5)
```

27. Consider the following program.

```
100 DIM B$(12)
110 INPUT ANS$
120 FOR J = 1 TO 12
130 READ B$(J)
140 NEXT J
150 IF ANS$ = "YES" THEN 200
160 PRINT B$(2), B$(7) + B$(9), B$(8)
170 GOTO 300
200 PRINT
   B$(10),B$(12),B$(2),B$(1),B$(11),B$(5),B$(3)
250 DATA
   "JACK","SEVEN","EAT","AGO","COULD","SCORE"
260 DATA "NO", "YERAS", "FAT", "FOUR", "SPRAT",
   "AND"
300 END
```

(a) What is assigned to B$(10) when the program is executed?

(b) What string is assigned to B$(3) when the program is executed?

(c) What message would be printed if the string "YES" were typed after the RUN for input?

(d) What message would be printed as a result of typing the word "OK" after RUN for input?
? "OK"

BIBLIOGRAPHY

Aikin, John O. 1981. A self-paced first course in computer science. SIGCSE Bulletin 1 (February): 78-85.

Alspaugh, Carol Ann. 1972. Identification of some components of computer programming aptitude. Journal for Research in Mathematics Education 3 (March): 89-98.

Association of Computer Machinery's Curriculum Committee on Computer Science. 1968. Curriculum 68: Recommendations for academic progress in computer science. Communication of ACM 11, 3 (March): 151-97.

________. 1979. Curriculum 78: Recommendations for the undergraduate program in computer science. Communication of ACM 22, 3 (March): 147-66.

Barker, Ricky J., and E. A. Unger. 1983. A predictor for success in an introductory programming class based upon abstract reasoning development. ACM SIGCSE Bulletin 15, 1 (February): 85-90.

Bauer, Roger, William A. Mehrens, and John F. Vinsonhaler. 1968. Predicting performance in a computer programming course. Educational and Psychological Measurement 28 (March): 1159-64.

Berztiss, Alfs. 1987. A mathematically focused curriculum for computer science. Communication of the ACM 30, 5 (May): 356-65.

Borg, Walter R. 1983. Educational research an introduction. 4th ed. New York: Longman.

Bulgren, William G. 1987. An introductory algorithm teacher. ACM SIGCSE Bulletin 19, 1 (February): 292-99.

Bulgren, William G., and Gregory F. Wetzel. 1982. Introductory computer science courses. ACM SIGCSE Bulletin 14, 1 (February): 133-39.

Campbell, Patricia F. 1984. The effect of the preliminary programming and problem solving course on performance in a traditional programming course for computer science majors. ACM SIGCSE Bulletin 16, 1 (February): 56-62.

Campbell, Patricia F., and G. P. McCabe. 1984. Predicting the success of freshman in computer science major. Communication of the ACM 27, 11 (November): 108-13.

Cannara, Alexander Bellows. 1976. Experiments in teaching children computer programming. Ph.D. diss., Stanford University.

Clements, D. H. 1979. Effects of logo and CAI environments on cognition and creativity. Journal of Educational Psychology 78, 4 (June): 309-318.

Clements, D. H., and D. F. Gullo. 1980. Effects of computer programming on young children's cognition. Journal of Educational Psychology 76, 6 (July): 1051-58.

Cook, Robert N. 1980. Structured programming using BASIC. ACM SIGCSE Bulletin 12, 1 (February): 40-49.

Corman, Larry S. 1986. Cognitive style, personality type, and learning ability as factors in predicting the success of the beginning programming student. SIGCSE Bulletin 18, 4 (December): 80-83.

Culik K., and M. M. Rizki. 1983. Logic versus mathematics in computer science education. ACM SIGCSE Bulletin 15, 1 (February): 14-20.

Deimel, L. E., and D. V. Moffat. 1982. A more analytical approach to teaching the introductory programming course. In Proceedings of the NECC, ed. Howard Johnson, 114-18. Columbia: The University of Missouri.

Denning, P. J. 1980. U.S. productivity in crisis. Communication of the Association for Computer Machinery 23 (March): 617-19.

Dupras, Marcel, Fernand Lemay, and Ali Mili. 1984. Some thoughts on teaching first-year programming. SIGCSE Bulletin 16, 1 (February): 148-53.

Fowler, George C., and Louis W. Glorfeld. 1981. Predicting aptitude in introductory computing: A classification model. AEDS Journal 2 (Winter): 96-109.

Gabrin, Philippe. 1982. Integration of design and programming methodology into beginning computer science courses. ACM SIGCSE Bulletin 14, 1 (February): 85-87.

Gathers, Emery. 1986. Screening freshman computer science majors. SIGCSE Bulletin 18, 3 (September): 44-48.

Gray, J. D. 1974. Predictability of success and achievement level of data processing technology students at the two-year post-secondary level. Ph.D. diss., Georgia State University.

Gries, D. 1981. The science of programming. New York: Springer.

Hancock, Chris. 1988. Context and creation in the learning of computer programming. For the Learning of Mathematics 8, 1 (February): 18-30.

Hostetler, Terry. R. 1983. Predicting students' success in an introductory programming course. ACM SIGCSE Bulletin 15, 3 (September): 41-5.

Jansson, Lars C., Harvey D. Williams, and Robert J. Collens. 1987. Computer programming and logical reasoning. School Science and Mathematics 87, 5 (May): 371-79.

Knuth, Donald E. 1974. Computer programming as an art. Communications of the ACM 17 (December): 517-30.

Konvalina, John, Larry Stephens, and Stanley Wileman. 1981. Identifying factors influencing computer science aptitude and achievement. Programmed Learning and Educational Technology 20, 2 (Winter): 84-95.

Konvalina, John, Stanley A. Wileman, and Larry J. Stephens. 1983. Math proficiency: A key to success for computer science students. Communication of the ACM, 26, 5 (May): 377-82.

Kurtz, B. L. 1980. Investigating the relationship between the development of abstract reasoning and performance in an introductory programming class. ACM SIGCSE Bulletin 12, 2 (February): 110-17.

McEntyre, Diane S. 1977. Structures and strategies for teaching computer programming in an introductory course. Ph.D. diss., University of California, Berkeley.

Meinke, John G. 1981. Alternatives to the traditional first course in computing. ACM SIGCSE Bulletin 13, 1 (February): 57-60.

Messer, S. B. 1981. Reflection-impulsivity: A review. Psychological Bulletin 83, 6 (January): 1026-1052.

National Council of Teachers of Mathematics. 1980. Recommendation for school mathematics of the 1980s. Communication of ACM 12, 1 (February): 105-20.

Noonan, Robert E. 1979. The second course in computer programming: Some principles and consequences. SIGCSE Bulletin 11, 1 (February): 187-91.

Pea, R. D. 1986. Language-independent conceptual "bugs" in novice programming. Journal of Educational Computing Research, 2, 1 (February): 25-36.

Pea, R. D., and D. M. Kurland. 1984a. On the cognitive effects of learning computer programming: A critical look. Technical Report No. 9. New York: Center for Children and Technology, Bank Street College.

________. 1984b. On the cognitive prerequisites of learning computer programming. Technical Report No. 18. New York: Center for Children and Technology, Bank Street College.

Peterson, Charles G., and Treavor G. Howe. 1979a. Predicting academic success in introduction to computers. AEDS Journal 12, 4 (Summer): 182-91.

________. 1979b. Predicting academic success in introduction to computers. AEDS Journal 2 (Fall): 34-42.

Poirot, J. L. 1979. Computer education in the secondary school: Problems and solutions. In Proceedings of the 10th SIGCSE Technical Symposium, by SIGCSE. St. Louis: SIGCSE, 166-70.

Rothman, S., and C. Mosmann. 1972. Computer and society. Chicago: Science Research Associated, Inc.

Samurcay, Renan. 1985. Learning programming: An analysis of looping strategies used by beginning students. For the Learning of Mathematics an International Journal of Mathematics Education 5, 1 (February): 37-43.

Sedlmeyer, Robert L. 1980. A college preparatory course in computer programming. SIGCSE Bulletin 12, 1 (February): 166-70.

Sidbury, James R. 1986a. A statistical analysis of the effect of discrete mathematics on performance of computer science majors in beginning computing classes. ACM SIGCSE Bulletin 1 (February): 42-48.

________. 1986b. A statistical analysis of the effect of discrete mathematics on the performance of computer science majors in beginning computer classes. Communication of the ACM 12, 6 (April): 134-40.

Sippl, Charles J. 1986 Computer dictionary, New York: Howard W. Sams and Company, Inc.

Slakter, Malcolm J. 1972. Statistical inference for education research. New York: Addison-Wesley Publishing Company.

Smith, Jeffrey W. 1981. A method for teaching programming. ACM SIGCSE Bulletin 13, 1 (February): 252-55.

Soloway, Elliot. 1986. Learning to program = learning to construct mechanisms and explanations. Communication of the ACM 29, 9 (September): 850-58.

Soloway, Elliot, K. Ehrlich, J. Bonar, and J. Greenspan. 1982. What do novices know about programming. In Directions in human computer interactions, ed. B. Shneiderman and A. Badre, 318-22. New Jersey: Ablex Publishing Company.

Stephens, Larry J., Stanley Wileman, and John Konvalina. 1981. Group differences in computer science aptitude. AEDS Journal 14, 2 (Winter): 84-95.

Sukhen, Dey, and Lawrence R. Mand. 1986. Effects of mathematics preparation and prior language exposure on perceived performance in introductory computer science courses. ACM SIGCSE Bulletin 18, 1 (February): 144-48.

Task force on computer science of the association of computer machinery elementary and secondary school subcommittee report. 1983. Communication of ACM 12, 3 (March): 58-65.

Van Merrienboer, Jeroen J. G. 1988. Relationship between cognitive learning style and achievement in an introductory computer programming course. Journal of Research on Computing Education 3 (Winter): 181-86.

Weinberg, Gerald M. 1971. The psychology of computer programming. New York: Van Nostrand Reinhold Company.

Werth, Laurie H. 1986. Predicting student performance in a beginning computer science class. SIGCSE Bulletin 18, 1 (February): 95-105.

Whipkey, Kenneth L. 1984. Identifying predictors of programming skill. ACM SIGCSE Bulletin 16, 4 (December): 36-42.

ySTAT, Spreadsheet Statistical Package for IBM PC. 1987. Rochester, N.Y.: MING TELECOMP, Inc.