379
N81d
NO.2132

ERRORS IN LOOPING AND ASSIGNMENT BY NOVICE
ASSEMBLY LANGUAGE PROGRAMMERS

DISSERTATION

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

For the Degree of

DOCTOR OF PHILOSOPHY

By

Lovie Ann Jeffrey Melkus, M.A.

Denton, Texas

December, 1983

13249m

Melkus, Lovie A., Errors in Looping and Assignment by Novice Language Programmers. Doctor of Philosophy (College Teaching), December, 1983, 98 pages, 15 tables, bibliography, 38 titles.

The problem with which this investigation is concerned is an analysis of errors in looping and assignment made by novice assembly language computer programmers. This analysis is made after subjects write three computer programs. The verbal definition of the problem for each program specifies, without naming, either a for, while, or repeat looping structure. Subjects of this study were students enrolled in an assembly language programming course who had previously completed an introductory computer science course including BASIC programming, and a course covering a high-level language.

The purposes of the study are the following: to determine whether using the appropriate looping structure is related to writing a correct program; to determine the incidence of assignment errors in incorrect programs; to determine the degree of relationship between mathematical aptitude, and the type of assignment errors made; to determine the degree of relationship between use of the appropriate looping structure and mathematical, verbal, and scholastic aptitude; to determine the degree of relationship

between gender and use of the appropriate looping structure, and to determine the degree of relationship between gender and the number of assignment errors.

A phi coefficient of correlation of .29 or .28, significant at the .01 level, was found between using the appropriate loop and writing a correct program for each of the three programs. Sample size was sixty-two. No other significant coefficients of correlation were found. It was determined that a majority of incorrect programs had at least one assignment error related to getting a new value, maintaining a running total, or maintaining a counter.

This report concludes that the significant positive correlation between writing a correct program and using the appropriate looping construct seems to confirm a relationship between expertise and the organization of knowledge, that errors in assignment remain an important reason that the programs of novice assembly language programmers fail, and that the Scholastic Aptitude Test is an inadequate basis upon which to evaluate the aptitude of students for assembly language programming.

TABLE OF CONTENTS

	Page
LIST OF TABLES	v
Chapter	
I. INTRODUCTION	1
Statement of the Problem	
Definitions of Terms	
Purposes of the Study	
Hypotheses	
Significance of the Study	
Assumptions	
Delimitations	
Organization of the Study	
Chapter Bibliography	
II. SYNTHESIS OF RELATED LITERATURE	14
Research Related to Aptitude for	
Computer Science	
Research Related to Student Programming	
Errors	
Theory and Research Related to the	
Organization and Representation	
of Knowledge	
Summary	
Chapter bibliography	
III. STATISTICAL ANALYSES PROCEDURES	39
Instruments	
Population of the Study	
Methods of Collection of the Data	
Hypotheses Testing Procedures	
Chapter Bibliography	
IV. ANALYSIS OF THE DATA	50
Findings	
Discussion	
Discussion of Additional Findings	
Summary	
Chapter Bibliography	

	Page
Chapter	
V. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS . .	79
Summary of Background and Purposes	
Summary of Procedures	
Summary of Findings	
Conclusions	
Recommendations	
Chapter Bibliography	
APPENDIX	93
BIBLIOGRAPHY	95

LIST OF TABLES

Table	Page
I. Frequency Percentages for PASCAL Novices	41
II. Frequency Percentages for PL/I Novices	42
III. Percentage of Students Using Certain Looping Structures and Writing Correct Programs	52
IV. Correlation between Writing a Correct Program and Choosing the Appropriate Loop	53
V. Percentage of Programs without Certain Errors in Assignment	54
VI. Relationship between mathematical Aptitude and Choice of Appropriate Looping Structure	55
VII. Relationship between Mathematical Aptitude and Various Measures of Errors in Assignment	57
VIII. Relationship between Verbal Aptitude and Use of Appropriate Looping Structure	58
IX. Relationship between Verbal Aptitude and Various Measures of Errors in Assignment	59
X. Relationship between Choice of Appropriate Looping Construct and Scholastic Aptitude	60
XI. Relationship between Scholastic Aptitude and Various measures of Errors in Assignment	62
XII. Relationship between Gender and Choice of Appropriate Looping Construct	63
XIII. Relationship between Gender and Various Measures of Errors in Assignment	64
XIV. Percentage of Students Making Assignment Errors in the Implementation of Counters and Running Totals	69

Table	Page
XV. Relationship between Various Measures of Scholastic Aptitude and Writing a Correct Program	74

CHAPTER I

INTRODUCTION

Computer programming is a complex and creative cognitive task, one which Shneiderman (3, p. 2) compares with architecture, creative writing, scientific research, and the composition of serious music. Although the complexity of the subject would seem to imply a correspondingly complex educational task, there has been little empirical research done on the subject of teaching computer programming. In this discipline, as in others, it is important to have research which aids the teacher in the critical process of understanding the student's knowledge, whether that knowledge is accurate or inaccurate (1).

The teaching of loop constructs and assignment statements is of particular concern in computer science because, although they are essential to computer programming, novice programmers often find it difficult to implement them correctly. The relationship of such errors to a theoretical model (4) which represents the knowledge of expert programmers, has been studied in an attempt to understand the knowledge and misconceptions held by novice PASCAL programmers (4, 5) and of both novice and professional PL/I programmers (5). It would also be useful,

therefore, to study errors made by novice programmers in assembly language, a low-level computer language in which programmers, because control structures are not explicitly supplied, must implement the basic loop structures with their own code.

Statement of the Problem

The problem with which this study is concerned is an analysis of errors in looping and assignment made by novice assembly language computer programmers.

Definitions of Terms

For the purposes of this study, the following terms are defined.

1. Assembly language is a machine-specific language that typically generates one machine instruction for each statement coded by the programmer. The assembly language used is appropriate for IBM 360/370 computers.
2. Novice assembly language programmers are students who are enrolled in an introductory assembly language programming course.
3. A high level language is a language in which one statement written by the programmer may generate several machine instructions.
4. Looping is a programming technique that allows a portion of a program to be executed repetitively.

5. The for loop is a construct used when a portion of a code must be executed a specific number of times and that number is known before the first execution of the loop. Execution of a for loop requires use of registers. Implementation of a for loop in assembly language requires that the programmer code a branch-on-count or branch-on-count-register instruction. Both of these instructions automatically decrement the register and branch when the register contains a zero.

6. The repeat loop is a construct used when the number of times a portion of code should be executed is determined by a variable, the value of which should be checked after each execution of the loop. A repeat loop is always executed at least once.

7. The while loop is a construct used when the number of times a loop should be executed is determined by a variable, the value of which should be checked before each execution of the loop. If the control variable has the specified value before the first execution, the while loop will not be executed.

8. Assignment is the process by which a value is associated with a variable.

9. Math aptitude refers to the score derived from the math section of the Scholastic Aptitude Test.

10. Verbal aptitude refers to the score derived from the verbal section of the Scholastic Aptitude Test.

11. Scholastic aptitude refers to the total score on the Scholastic Aptitude Test.

12. A counter is a variable used to count something (e. g., how many times a loop has been executed) within a program.

14. A running total is a variable used to accumulate a total through successive operations in a program.

14. A correct program is one which would achieve the objective in the problem if syntactic errors were removed.

Purposes of the Study

The purposes of this study are the following:

1. To determine whether using an appropriate loop structure is related to writing a correct program, including
 - a. Whether using the for loop when it is appropriate (i. e., when the loop must be executed a specific number of times) is related to writing a correct program;
 - b. Whether using the repeat loop when appropriate (i. e., when the variable controlling how many times the loop is executed should be checked after each execution of the loop) is related to writing a correct program;

c. Whether using the while loop when it is appropriate (i. e., when the variable determining how many times the loop is executed should be checked before each execution of the loop) is related to writing a correct program.

2. To determine the incidence of assignment errors in incorrect programs, including

a. The incidence of getting a new value and assigning it to a variable in incorrect programs;

b. The incidence of assignment errors in the implementation of running totals in incorrect programs;

c. The incidence of assignment errors in the implementation of counter variables in incorrect programs.

3. To determine the degree of relationship between math aptitude, as determined by the Scholastic Aptitude Test (2) math score (SAT-M), and the type of errors made.

4. To determine the degree of relationship between verbal aptitude, as measured by the Scholastic Aptitude Test (2) verbal score (SAT-V), and the type of errors made.

5. To determine the degree of relationship between scholastic aptitude, as measured by the total score on the Scholastic Aptitude Test (SAT-T), and the type of errors made.

6. To determine the degree of relationship between gender and the type of errors made.

Hypotheses

The following hypotheses are used to test the data findings.

1. There will be a significant positive correlation between using the correct loop construct and writing a correct program, including
 - a. Writing a correct program and using the for loop on the test program for which it is appropriate;
 - b. Writing a correct program and using the repeat loop when it is appropriate;
 - c. Writing a correct program and using the while loop when it is appropriate.
2. Assignment statements will be used correctly in the majority of incorrect programs, including
 - a. The majority of incorrect programs will correctly get new values and assign them to variables;
 - b. The majority of incorrect programs will use assignment statements correctly in the implementation of running totals;
 - c. The majority of incorrect programs will use assignment statements correctly in the implementation of counters.

3. There will be a significant positive correlation between math aptitude (SAT-M) and the following variables:

- a. Use of the for construct on the test program for which it is appropriate;
- b. Use of the repeat construct on the test program for which it is appropriate;
- c. Use of the while construct on the test program for which it is appropriate.

4. There will be a significant negative correlation between math aptitude (SAT-M) and

- a. The total number of assignment errors on the three test programs;
- b. The number of assignment errors in getting a new value on the three test programs;
- c. The number of assignment errors in the implementation of running totals on the three test programs;
- d. The number of assignment errors in the implementation of counters in the test programs.

5. There will be a significant positive correlation between verbal aptitude (SAT-V) and the following:

- a. Use of the for construct on the test program for which it is appropriate;
- b. Use of the repeat construct on the test program for which it is appropriate;

- c. Use of the while construct on the test program for which it is appropriate.
- 6. There will be a significant negative correlation between verbal aptitude (SAT-V) and
 - a. The total number of assignment errors on the three test programs;
 - b. The number of assignment errors in getting a new value in the three test programs;
 - c. The number of assignment errors in the implementation of running totals in the three test programs;
 - c. The number of assignment errors in the implementation of counters in the three test programs.
- 7. There will be a significant positive correlation between scholastic aptitude, as measured by the total score on the Scholastic Aptitude Test (SAT-T), and the
 - a. Use of the for construct when it is appropriate;
 - b. Use of the repeat construct when it is appropriate;
 - c. Use of the while construct when it is appropriate.
- 8. There will be a significant negative correlation between scholastic aptitude (SAT-T) and
 - a. The total number of assignment errors on the three test programs;

- b. The number of assignment errors in getting a new value in the three test programs;
 - c. The number of assignment errors in implementation of running totals in the three test programs;
 - d. The number of assignment errors in the implementation of counters in the three test programs.
9. There will be no significant correlations between gender and selected variables that include
- a. Use of the for construct on the test program for which it is appropriate;
 - b. Use of the repeat construct on the test program for which it is appropriate;
 - c. Use of the while construct on the test program for which it is appropriate;
 - d. Total number of assignment errors on the three test programs;
 - e. The number of assignment errors in getting a new value in the three test programs;
 - f. The number of assignment errors in the implementation of running totals in the three test programs;
 - g. The number of assignment errors in the implementation of counters in the three test programs.

Significance of the Study

Despite increased attention to the human factors area of computer science, there has been surprisingly little

research on the teaching of computer science. Although the research which has been done concentrates primarily on determining aptitude for computer science, teachers need information that can help them achieve better success with the students in their classrooms. Because students are frequently required to turn in working programs (i. e., syntactically correct programs which produce correct output), computer science teachers often lose the insight that can accompany correcting errors on students' papers in any discipline. This study examines the errors in looping and assignment that are made on initial attempts to write programs before submission to a computer.

Teachers also need to know which concepts are troublesome for students. Although certain researchers (4, 5) look at a specific area (i. e., looping and assignment), their studies are concerned with high-level languages; it is important to extend this type of study to include assembly language, a computer language that is commonly regarded as difficult by students. The teacher of assembly language needs information from which to begin to compile a picture of students' knowledge. It is also important for teachers who are concerned with individual differences to associate certain variables such as gender and scholastic aptitude not only with overall computer science aptitude

but also with programming performance with regard to the implementation of specific constructs.

This study, therefore, is significant primarily because it will provide a base of knowledge upon which to found improvements in the teaching of computer science. It will extend the study of errors in assignment and looping to assembly language; it will indicate whether gender and scholastic aptitude might be appropriate bases for determining teaching strategies; it will indicate whether novice assembly language programmers can determine when to create the loop constructs supplied by high-level languages.

Assumptions

It is assumed that programs written in a classroom but not submitted to a translator provide an accurate indication of programming errors.

Delimitations

Data obtained from international students, students twenty-six years of age or older, and students with professional programming experience are not included in the analysis so that data results will not be distorted by language problems, maturity, or exceptional background of the subjects.

Organization of the Study

Chapter I of this study presents the problem, purposes, hypotheses, significance of the study, assumptions, and delimitations. Chapter II includes a review of relevant literature, and the statistical methods and procedures used are discussed in Chapter III. Chapters IV and V present the data findings, a summary of the study, and conclusions and recommendations for future research.

CHAPTER BIBLIOGRAPHY

1. Brown, J. S., and R. R. Burton, "Diagnostic Models for Procedural Bugs in Basic Mathematical Skills," Cognitive Science, 4 (April-June, 1978), 379-415.
2. Scholastic Aptitude Test, Princeton, Educational Testing Service, 1970.
3. Shneiderman, Ben, Software Psychology: Human Factors in Computer and Information Systems, Cambridge, Winthrop Publishers, Inc., 1980.
4. Soloway, E., and others, "Cognition and Programming: Why Your Students Write Those Crazy Programs," Proceedings of the National Educational Computing Conference, edited by Diana Harris and Laurie Nelson-Hearn, Iowa City, University of Iowa WEEG Computing Center, 1981, pp. 206-219.
5. Swigger, Kathleen, and Layne Wallace, "Use of the Appropriate Looping Structures: Expert vs. Novice," Proceedings of the Human Factors Society, edited by Richard Edwards, Santa Monica, Human Factors Society, 1982, pp. 999-1003.

CHAPTER II

SYNTHESIS OF RELATED LITERATURE

The presentation of the literature related to this study is divided into three sections. The first of these divisions includes studies that are related to determining aptitude for computer science. The second includes research on analyzing students' programs, detecting students' errors, and correcting students' errors, and the theory that is influential in the teaching of computer science. The third division includes literature concerning the organization and representation of knowledge, which are concepts important to any attempt to understand the knowledge of novice programmers.

Research Related to Aptitude for Computer Science

Much of the research which has been done in the area of computer science education focuses on determining aptitude for computer science. Although results of such research are not always consistent, several studies indicate that mathematical background or ability is often an important element of such an aptitude.

Alsbaugh (2) investigated the relationships between measures in programming two languages, basic assembly

language for the IBM 360 (BAL) and FORTRAN, and scores on a temperament inventory, a programming aptitude test, a critical thinking evaluation, and a scholastic aptitude test. She found that the mathematical background of the student as determined by completed math courses has a correlation of .382 with the FORTRAN score and of .356 with the BAL score, both of which are significant at the .01 level. Alspaugh also found the TTS impulsive and sociable scores to have statistically significant correlations with the measures for both BAL and FORTRAN (-.383, significant at the .01 level, and -.313, significant at the .05 level). She also reports that no measure correlates significantly higher with achievement than with achievement in FORTRAN (3).

Fowler and Glorfeld (10), in devising a classification model to predict success in an introductory computer science course which included programming and other topics, used age, college grade point average (GPA), Scholastic Aptitude Test (22) math score, and the number of math courses completed to determine high- and low-aptitude groups. They found GPA to be the most significant of the four variables. Because they found that age negatively correlates with success, they theorize that older students had avoided taking computer science as long as possible and that high-aptitude students would have taken it earlier.

Some of the most comprehensive work in this area has been done by Wileman, Konvalina, and Stephens (30). They report on a study in which they use a predictor that includes scores for reading comprehension, finding the next term in a sequence, logical reasoning, algorithmic execution, and converting terms by use of a code. They found that the predictor has a correlation of .46, which is significant at the .01 level, with scores on the final examination in an introductory computer science class.

In another study (27), the same researchers obtained for each subject a score on the previously used pretest and data on sex, age, year in college, hours worked each week, previous experience in computer science, estimated high school performance, and estimated college performance. They found that only the last two variables influence scores on the test; scores on the algorithmic and sequential reasoning portions of the test were most important in differentiating students. The authors point out that these scores are closely related to mathematical reasoning ability. Reading comprehension was not found to be an important factor in predicting computer science aptitude, but it should be noted that the reading measure used was the score obtained on a ten-item test which measures comprehension of a computer-related article.

In a later study, Konvalina, Stephens, and Wileman (14) again found that high performance, success in college, and mathematical background are important factors in predicting success in college computer science classes. Further, they found that age, except for the group including subjects thirty-five and older, is significant in predicting success; the older students were more successful. The authors suggest that this contradiction of the findings of Fowler and Glorfeld (10) may result from a difference between the motivation of older students at Texas A&M University, which is primarily a residential university, and that of older students at the University of Nebraska at Omaha, which is primarily a commuter campus.

Mazlack (18) investigated the correlation between success in an introductory FORTRAN course and gender, semester in school, academic discipline, and score on a programming aptitude test. Since he found no significant correlations, Mazlack suggests that lack of significant correlations between either semester in school or academic discipline and success in the FORTRAN course implies that it is not necessary to provide separate courses based on either of those variables.

Peterson and Howe (21) found significant correlations between final grade in an introductory computer science course and high school GPA, college GPA, and high school

grades in math and science. They also found significant negative correlations between course grade and the active, vigorous, impulsive, and sociable measures on a temperament scale. They found significant positive correlations between course grade and the intelligence, verbal, numerical, and clerical scores obtained by students on an aptitude test battery.

Cheney (8) hypothesizes that cognitive style, as well as mathematical ability, might be related to programming aptitude. Describing cognitive style as either heuristic or analytic, Cheney portrays analytic problem solvers as people who use "model building, mathematical analysis, and optimization" (8, p. 286); he describes heuristic thinkers as people who use a less careful approach based on trial and error. Cheney found a Spearman rank-correlation of .82 (significant at the .01 level) between scores on a questionnaire measuring cognitive style and on programming examination. Cheney points out that, although mathematical ability may be involved in analytic cognitive style, using cognitive style as a predictor of success in computer science does not discriminate against those who do not have a strong math background. He also points out that this study may indicate the need for different teaching strategies for students with varying cognitive styles.

Kurtz (16) studied the relationship between abstract reasoning skills and performance in an introductory programming course. Kurtz designed an instrument to determine the ability of an individual to think abstractly, based on levels of intellectual development as defined in The Growth of Logical Thinking from Childhood to Adolescence (13). These are preoperational, concrete, and formal levels of development. An individual moving past the pre-operational level, at which static states are perceived, moves to the concrete level. At this level the individual can classify and generalilze and move from reality to theory. The third level of intellectual development is characterized by abstract thinking, the ability to move from theory to reality, and the ability to examine one's own logical reasoning. According to Piaget, most people move to the concrete level of intellectual development when they are about seven years old; some never reach the formal level. Kurtz's fifteen-item test, which takes about eighty minutes to complete, is designed to measure a student's ability to solve problems related to ten areas (conservation of volume, direct proportion, probabilistic reasoning, inverse proportion, propositional reasoning, correlational reasoning, deductive logic, separation of variables, permutations, combinations) that are characteristic of the concrete, early formal, formal, and late formal levels of intellectual

development. Kurtz administered the test to twenty-three students in his own introductory programming class and found a .7954 coefficient of correlation between test scores and course performance as measured by students' final grades for the semester.

Barker and Unger (4), attempting to develop a predictive instrument for performance in an introductory programming course, modified the Kurtz test so that it can be completed in about forty minutes. They administered their shortened version of the test to 353 students who were studying two computer languages with several instructors. Barker and Unger found a coefficient of correlation of 0.11615 between performance of the modified test and course grade. They believe that their modified version of the Kurtz test may be a useful predictor of success in an introductory programming course when it is combined with other predictive measures.

Research Related to Student Programming Errors

The second area of research related to the present study is concerned with the analysis of students' programs and errors and with the theory of structured programming. In 1981, Soloway and others (24) drew upon the concept of frames developed in the field of artificial intelligence by Minsky (20) and Winograd (31) to study the performance of

novice PASCAL programmers. As a unit of knowledge, a frame includes a description of the object or concept in question as well as links to subgroups, supergroups, and related frames. For example, the frame which describes birthday party might be linked to a supergroup celebration, to subgroups refreshments, presents, and games. Birthday party might be linked to Halloween party as a related frame.

Soloway and others (24) describe frames representing a logical organization of the concepts involved in assignment statements and in for, while, and repeat loops. They use that model to discuss the implications of errors made by novice PASCAL programmers in looping and assignment. The researchers attribute the inability of novice programmers to determine when each of the three types of loops is appropriate to the novices' lack of a knowledge net indicating semantic understanding of the three loop plans. The authors also suggest that the fact that more students correctly use counter variables than running total variables is related to inappropriate frames; the frame representations for running total and counter variables, though similar, should be distinct.

In a 1982 study that uses the same instrument, Soloway and others (25) found that although both novice and intermediate students who chose the correct loop structure were somewhat more likely to write correct programs, the

relationship between those two factors is not strong enough to make choosing the correct looping structure a predictor of program correctness. The researchers point out the peculiar structure needed in order to use a while loop in PASCAL, the language used by students in this study. In order to implement a while loop in PASCAL, the programmer must read the control variable before executing the loop the first time; within the loop, the programmer must first process the variable, then read the next variable. The authors assume that many students are more comfortable with a procedure which would, within the loop, read the variable and then process it. They make that assumption because many students went to extraordinary lengths to convert the normal while procedure to one which read the variable and then processed it within the loop. When the researchers compared those students who used the process variable-read variable structure and who had correct programs with those who used the read variable-process variable strategy, and had correct programs, they found a significant difference. Using the sign test for a group of twenty-four, they found chi equal to seven, significant at the .03 level. Choosing the correct looping strategy proved to be a better predictor of program correctness than choosing the correct loop structure.

In the same study, on the problem requiring a while loop, all novices correctly maintained a counter variable, but 83 per cent correctly maintained a running total even though the two processes are essentially the same. The researchers suggest three possible reasons for novices finding that maintaining a running total is more difficult than maintaining a counter. First, although counting is always a successive activity, most people do not add a column of numbers by maintaining a running total. Second, students memorize how to maintain a counter without realizing that the principles may be used for other purposes. Third, updating a counter involves adding a constant; whereas updating a running total involves adding a variable, and students sometimes have more difficulty working with variables than with constants.

In the same study, 23 per cent of programs written by novices did not correctly get a new value. The researchers suggest two possible explanations for why such a surprising percentage of students failed to perform correctly a basic function of programming. They suggest that some students may confuse a read-variable with a counter-variable, or that students may not understand that the read statement in PASCAL both gets a new value and assigns it to a variable name.

Soloway and others (24) also note that on 27 per cent of programs written by novices, only one variable was used

to refer to both the input number and the running total or counter. They suggest that perhaps novice programmers believe it is within the power of the computer to distinguish between the two uses. Only 8 per cent of the intermediate programmers studied made this error; the difference between the values for intermediate and novice programmers is significant at the .001 level, with chi-square equal to 15.44.

Swigger and Wallace (28) use the test instrument created by Soloway and others (21) to study errors made both by novice PASCAL and PL/I programmers and by novice and professional PL/I programmers. They found no significant differences between use of the appropriate loop or writing a correct program and the experience of the programmer. This finding appears to indicate a need for empirical research into the manner in which expert programmers represent their knowledge.

Bug Finder, a program designed to detect errors in student programs, is described by Bonar and others (4). When students submit syntactically correct programs to Bug Finder, they receive an analysis of their programs. Bug Finder can be used both as an online tutor and as an analyst of a semester's work by an entire class or an individual. The authors believe that an important advantage of Bug Finder is that it allows analysis of student

programs in a realistic setting, i. e., with students actually using computers instead of writing code in a classroom setting. Bug Finder looks for certain pre-defined errors, such as an attempt to divide by zero, failure to initialize a counter, or failure to initialize a total. Most students reported that the Bug Finder was helpful to them. Instructors found that grades on the program for which students used Bug Finder were higher than expected.

In a later article, Soloway and others (26) report on the performance of Bug Finder when tested by having 116 students in an introductory PASCAL programming course use it with a simple looping program. After analyzing the first syntactically correct program submitted by twenty randomly selected students, the researchers found that Bug Finder performed unsatisfactorily, reporting incorrectly almost as many errors (bugs) as it reported correctly. In addition, they found that Bug Finder did not report fifty-nine errors; the program as written should have deducted 15 per cent of these errors.

The researchers believe that much of Bug Finder's ineffectiveness can be explained by the fact that it was tested on a programming problem that was more complex than the problems used in designing Bug Finder. Thus, the potential for error was increased, both by the number of

errors that might occur, and by the number of ways in which each error might occur. In addition, in order to handle the more complex programs, Bug Finder should have been designed to handle dependencies between pieces of code. The researchers conclude by listing two major requirements for an effective program for identification of errors. First, the error detection program needs to include a thorough description of the goals of the program to be tested so that a complete predictive analysis of errors can be made. Second, the error-detection program should be capable of examining a program as a whole instead of as isolated pieces of code.

Gannon (11) studied characteristic errors in ST, which is a small, ALGOL-like computer language. Assigned to write a program which reversed a string, thirty-three graduate and advanced undergraduate students made twenty-seven errors in looping and assignment. Because some errors remained in the programs for more than one execution while students were trying to correct their initial programs, there were 134 error occurrences in the 383 runs which were made before all students had produced correct programs. The looping errors accounted for approximately 10 per cent of errors and occurrences. The looping errors resulted from such factors as failure to initialize a

variable, failure to alter a control variable within a loop, and failure to test for an end-of-file condition.

Waters (29) discusses a method for analyzing loop programs. He suggests looking at loops in terms of four basic plan building methods (PBMs). The first is an augmentation PBM, which consists of a body in which a variable is augmented, and an initialization, which precedes the body. The second is a filtering PBM, which is an augmentation PBM that has an if-then-else clause for a body, with both the then and else clauses null. The third is the basic loop PBM; it is what remains in a loop once augmentation and filtering PBMs have been removed. The fourth is an interleaving PBM, in which two loops are executed together; it is characterized by two separate exit tests. After using the PBM to analyze by hand twenty programs randomly selected from the IBM Scientific Subroutine Package (12) to determine whether the pieces isolated by PBMs were simple and how often loops could not be analyzed by PBMs, they found that it was possible to analyze all of the 187 underlying loops in the twenty programs by means of PBMs. They also found that most pieces resulting from PBM analysis were simple.

Brown and Burton (5, 6) discuss their program to diagnose procedural errors in subtraction; they use a network to depict the various processes involved in the

subtraction procedure. The program is able to detect when a student's procedures (frames or subframes in the student's subtraction network) differ from the correct network. Thus, the program can help to diagnose not only why errors are made but also which errors are made.

As the emphasis in the field of computer science has moved from hardware to software and as software has become increasingly complex, it has become essential to create understandable programs that reflect their purpose in their structure. Dijkstra (9), addressing the problems arising because of the complexity of modern software, advocates that programmers use only three, or possibly four, structures; these structures are simple sequence, the if-then or if-then-else, the do-while, and possibly the case statement. Dijkstra further advocates the use of modular programming--a practice which breaks programs down into components small enough to be understandable. It should be noted that although assembly language does not supply a do-while statement or its variant, the repeat statement, as control structures, it is possible for assembly-language programmers to implement those structures with their own code, thus achieving Dijkstra's underlying goal of clearly written programs which reflect their purpose in their structure.

Theory and Research Related to the
Organization and Representation
of Knowledge

McKeithen and Reitman (9) found that the level of expertise of the programmer differentiated recall performance of programs in logical order but not of those in scrambled order. They also found that, when programmers were asked to recall a list of reserved words in ALGOL W, how those words were chunked is related to the level of expertise of the programmer. They conclude that organization is related to expertise, but their study does not prove that organization can produce greater expertise.

Adelson (1) having studied the recall performance of novice and expert PPL programmers, also found organization of knowledge to be important. She concludes that both novice and expert programmers organize programming knowledge into conceptual categories; however, novices tend to use syntactic categories related to the grammar of a language, while experts use semantic categories related to the meaning of the program.

Winograd (31) discusses the debate concerning whether procedural or declarative representations of knowledge are superior. The proceduralists believe that to know means to know how--that human knowledge is embedded in procedures for using that knowledge. The declarativists, on the other hand, believe that human knowledge consists of a set of

facts and a set of procedures for manipulating those facts. Winograd points out that there is no formal difference between the two views--that it is possible to view everything as data (declarative) (e. g., a program written in a high-level language is data for another program which translates it to machine language) and that it is also possible to view everything as a program (procedural) (e. g., a fact is a simple program accepting as input questions or commands related to its truth).

Winograd (30) lists three major advantages of declarative representation. The first is its flexibility and economy. A declarative representation does not require the advance specification of one of several possible uses for a piece of knowledge, and it can make many uses of a single statement. Another advantage of declarative representation is its understandability and learnability. A third advantage is accessibility and communicability; since natural language uses declarative forms, it is easy to state knowledge in declarative form, whether adding to programs or communicating them to others.

Winograd (31) also lists the major advantages of procedural representation of knowledge. The first is procedural or modeling; much of what we know is intimately bound up with how it is used in a procedure, and it is difficult to represent such knowledge in declarative form.

This ease of representation is traded for the flexibility characteristic of declarative representation. The second advantage of procedural representation arises from the need for heuristic knowledge. Winograd believes that it is not possible for a real system to separate knowledge about strategies from declarative knowledge.

Winograd (31) further supports the value of procedural representation by discussing the nature of programming. He says that declarative representation is appropriate for mathematical applications where each axiom is regarded separately, without regard for how it interacts with others. But he maintains that procedural representation is more appropriate for programming, where interrelationships among subroutines affect the meaning of a program. Procedural representation, he believes, can take into account those interrelationships.

Although Winograd (31) does not want to propose a facile combination of the two theories of representation, he does use the concept of frames to generate a synthesis. Frames, each of which describes the knowledge a person has of a certain subject, are organized into a hierarchical structure that indicates relationships. A given sub-frame may be associated with more than one frame. Associated with a frame are various procedures for its use; for instance, the frame for day (which might have sub-frames

such as holiday, Thursday, and payday) might have associated with it a procedure for using a calendar.

Minsky (21) describes a frame as a "data structure for representing a stereotyped situation" (21, p. 212). The frame has several levels which are tied together by the relationships among the nodes; the top levels are fixed, containing information which is always true about the situation. For instance, a frame for a kitchen might have appliance, table, and cupboards as top-level nodes; stove and refrigerator might be nodes one level lower than from appliance; stove and refrigerator would have several slots indicating various possibilities for each (e. g., gas, electric, wood). These slots may be filled with default assignments which are assumed unless the situation provides information to the contrary (e. g., stoves may always be electric unless otherwise noted). Minsky's frames are linked into systems which may represent such knowledge as cause-effect relations, different views of the same room, or actions. He suggests that the frame systems are tied together by an information retrieval network which can provide a replacement frame when a proposed frame cannot be made to fit reality and when values are assigned to the terminal levels of a frame.

Kuipers (15) discusses several important properties of frames. The first of these properties is description,

which contains a number of features of an object. This descriptive property is filled in by some procedural knowledge that can establish correspondence between the reality experienced and the frame.

Another property discussed by Kuipers (15) is instantiation, a process by which observed values are substituted for default values. Prediction can use an expected description to fill in the actual description and to supply default values when necessary. Justification describes the various levels of confidence attached to different aspects of the description, some of which have been observed, some of which have been observed hastily, and some of which have been supplied by default values. Variation accounts for different sizes and proportions that may be associated with the same general frame. Correction provides the analysis necessary when a proposed frame does not fit reality. Transformation provides new frames when necessary, and perturbation helps to account for small changes in the description with repudiation of the frame.

Other writers suggest the necessity for a similar representation of knowledge without specifically referring to frames. Schwartz (23), discussing the characteristics he considers essential for programmers, includes a knowledge of semantic frameworks. Mayer (18) describes two structural variables that are present when new knowledge

is acquired--external connections and internal connections. External connections link new knowledge to a system of knowledge already established, and internal connections link various aspects of new material.

Summary

This chapter has reviewed the literature relevant to the present study and has divided that literature into three areas. The first of these areas includes studies related to aptitude for computer science. The second area includes research and theory related to the teaching of computer science. The third area includes literature related to the organization and representation of human knowledge.

CHAPTER BIBLIOGRAPHY

1. Adelson, B., "Problem Solving and the Development of Abstract Categories in Programming Languages," Memory and Cognition, 9 (July, 1981), 422-433.
2. Alspaugh, C. A., "Identification of Some Components of Computer Programming Aptitude," Journal of Research in Mathematics Education, 3 (March, 1972), 89-98.
3. _____, "A Study of the Relationships between Student Characteristics and Proficiency in Symbolic Programming," unpublished doctoral dissertation, University of Missouri, Columbia, Missouri, 1970.
4. Barker, R. J., and E. A. Unger, "A Predictor for Success in an Introductory Programming Class Based upon Abstract Reasoning Development," ACM SIGCSE Bulletin, 15 (February, 1983), 154-158.
5. Bonar, J., and others, "Collecting and Analyzing On-Line Protocols from Novice Programmers," Behavior Research Methods and Instrumentation, 14 (April, 1982), 203-209.
6. Brown, J. S., and R. R. Burton, "Diagnostic Models for Procedural Bugs in Basic Mathematical Skills," Cognitive Science, 2 (April-June, 1978), 155-192.
7. Burton, Richard R., "Diagnosing Bugs in a Simple Procedural Skill," Intelligent Tutoring Systems, edited by D. Sleeman and J. S. Brown, New York, Academic Press, 1982, pp. 157-183.
8. Cheney, Paul, "Cognitive Style and Student Programming Ability: An Investigation," AEDS Journal, 13 (Summer, 1980), 285-291.
9. Dijkstra, E., "Notes on Structured Programming," Structured Programming, by O. Dahl, E. Dijkstra, and C. A. R. Hoare, New York, Academic Press, 1972, pp. 1-81.

10. Fowler, G. C., and L. W. Glorfeld, "Predicting Aptitude in Introductory Computing: A Classification Model," AEDS Journal, 14 (Winter, 1981), 96-109.
11. Gannon, J. D., "Characteristic Errors in Programming Languages," Proceedings of the 1978 Annual Conference of the ACM, 1978.
12. IBM, IBM Scientific Subroutine Package Version III Programmer's Manual, IBM, White Plains, 1966.
13. Inhelder, B., and J. Piaget, The Growth of Logical Thinking from Childhood to Adolescence, New York, Basic Books, 1958.
14. Konvalina, John, Larry Stephens, and Stanley Wileman, "Identifying Factors Influencing Computer Science Aptitude and Achievement," AEDS Journal, 16 (Winter, 1983), 106-112.
15. Kuipers, Benjamin J., "A Frame for Frames," Recognition and Understanding: Studies in Cognitive Science, edited by Daniel G. Bobrow and Allan Collins, New York, Academic Press, 1975, pp. 152-184.
16. Kurtz, B. L., "Investigating the Relationship between the Development of Abstract Reasoning and Performance in an Introductory Programming Class," ACM SIGCSE Bulletin, 12 (Winter, 1980), 14-17.
17. Mayer, R., "Instructional Variables in Computer Programming," ERIC Document ED 112 865.
18. Mazlack, Lawrence J., "Identifying Potential to Acquire Programming Skill," Communications of the Association for Computing Machinery, 23 (January, 1980), 14-17.
19. McKeithen, Katherine B., and others, "Knowledge Organization and Skill Differences in Computer Programmers," Cognitive Psychology, 13 (July, 1981), 307-325.
20. Minsky, M., "A Framework for Representing Knowledge," The Psychology of Computer Vision, edited by P. H. Winston, New York, McGraw-Hill, 1975, 211-277.

21. Peterson, C., and T. G. Howe, "Predicting Success in Introduction to Computers," AEDS Journal, 12 (Summer, 1979), 182-191.
22. Scholastic Aptitude Test, Princeton, Educational Testing Service, 1970.
23. Schwartz, J. T., "What Programmers Should Know," Computer Languages, 2 (1976), 21-25.
24. Soloway, E., and others, "Cognition and Programming: Why Your Students Write Those Crazy Programs," Proceedings of the National Educational Computing Conference, edited by Diana Harris and Laurie Nelson-Hearn, Iowa City, University of Iowa WEEG Computing Center, 1981, pp. 206-219.
25. Soloway, E., and others, "What Do Novices Know about Programming?," Directions in Human-Computer Interaction, edited by A. Badre and B. Shneiderman, Norwood, Ablex Publishing Corporation, 1982, pp. 27-55.
26. Soloway, E., and others, "MENO-II: An AI-Based Programming Tutor," Journal of Computer-Based Instruction, 10 (Summer, 1983), 20-34.
27. Stephens, L., J. Konvalina, and S. Wileman, "Group Differences in Computer Science Aptitude," AEDS Journal, 14 (Winter, 1981), 84-95.
28. Swigger, Kathleen, and Layne Wallace, "Use of Appropriate Looping Structures: Expert vs. Novice," Proceedings of the Human Factors Society, edited by Richard E. Edwards, Santa Monica, Human Factors Society, 1982.
29. Waters, R. C., "A Method for Analyzing Loop Programs," IBEE Transactions on Software Engineering, 5 (May, 1979), 237-247.
30. Wileman, S. A., J. Konvalina, and L. J. Stephens, "Factors Influencing Success in Beginning Computer Science Courses," Journal of Educational Research, 74 (March-April, 1981), 223-226.

31. Winograd, Terry, "Frame Representations and the Declarative/Procedural Controversy," Representation and Understanding: Studies in Cognitive Science, edited by Daniel G. Bobrow and Allan collins, New York, Academic Press, 1975, pp. 185-210.

CHAPTER III

METHODS AND PROCEDURES OF THE STUDY

Instruments

The Scholastic Aptitude Test (2) is used in this study to determine math and verbal aptitudes of the population. The Scholastic Aptitude Test, described as "highly perfected" by DuBois (1, p. 646), is designed to measure high school senior students' aptitude for college study. The SAT reports scores with a mean of 500 and a standard deviation of 100. Reliability estimates for the various forms are about .90; validity for the SAT-verbal is .39; for the SAT-math, .33. DuBois reports that reliability, validity and data information have been consistent over the years that this test has been in use.

The instrument used to evaluate students' use of loops and assignment statements is a set of three simple programming problems created for a study made at the University of Massachusetts (3). The verbal definition of each problem specifies, without naming, a particular loop construct. The first problem requires that a loop be executed exactly ten times, thus specifying a for loop. The second problem requires that a loop be executed until a total maintained within the loop is greater than one hundred. That total

is the variable which determines how many times the loop will be executed; because the total should be set to zero at the beginning of the program, the loop will be executed at least once. Because the value of the total variable should be checked after each execution of the loop, the repeat construct is specified. The third problem requires that a loop be executed until the number 99999 is read, but because it is possible that 99999 may be the first number read, the loop may never be executed. The variable to which 99999 is assigned determines how many times the loop is executed; because it should be checked before each execution of the loop, the while loop is specified. Each of the three problems requires the maintenance of a running total variable; the second and third problems require the maintenance of counter variables; the first problem requires a counter variable only if the for loop is not used.

Swigger and Wallace (5) administered this test to both a group of novice PASCAL programmers and a group of novice PL/I programmers. They found that 60.7 per cent of the PASCAL programmers used the for loop when appropriate, 39.3 per cent used the repeat loop when appropriate, and 48.2 per cent used the while loop when appropriate. A chi-square was computed between use of the appropriate loop structure and a correct solution to the problem. For

the PASCAL group, correlation was significant only for problem three. The correlation was significant, however, for all three problems for the PL/I group. With a group of forty-seven subjects, a correlation coefficient of .25, significant at the .05 level, was found for problem one; a correlation coefficient of .28, significant at the .03 level, was found for problem two; and a correlation coefficient of .41, significant at the .01 level, was found for problem three. Frequency percentages for the PASCAL group are reported in Table I, and in Table II for the PL/I group.

TABLE I
FREQUENCY PERCENTAGES FOR PASCAL NOVICES*

Percentage of Students Who:	Problem 1 <u>For</u> Loop	Problem 2 <u>Repeat</u> Loop	Problem 3 <u>While</u> Loop
Used <u>for</u> loop	60.7%	3.6%	3.7%
Used <u>repeat</u> loop	. .	39.3%	29.6%
Used <u>while</u> loop	32.2%	46.4%	48.2%
Used other constructs	7.1%	10.7%	18.5%
Correct program and appropriate loop	16.4%	28.6%	22.2%
Correct program	68.9%	57.1%	29.5%

*After Swigger and Wallace (5, p. 1002).

TABLE II
FREQUENCY PERCENTAGES FOR PL/I NOVICES*

Percentage of Students Who:	Problem 1 <u>For</u> Loop	Problem 2 <u>Repeat</u> Loop	Problem 3 <u>While</u> Loop
Used <u>for</u> loop	74.5%	. .	4.3%
Used <u>repeat</u> loop	2.1%	17.0%	23.4%
Used <u>while</u> loop	21.3%	80.9%	61.7%
Used other constructs	2.1%	2.1%	10.6%
Correct program and appropriate loop	61.7%	17.0%	36.2%
Correct program	76.6%	72.3%	42.6%

*After Swigger and Wallace (5, p. 1002).

Although in both the 1981 study by Soloway and others (3) and the 1982 study by Swigger and Wallace (5), subjects were asked to code the test programs in PASCAL and PL/I, the subjects of this study were asked to code in assembly language. It is possible to implement each construct in any of these languages, although control statements are not built into assembly language as they are built into high-level programming languages.

Population of the Study

The study was conducted at a multi-purpose state university in north-central Texas. Permission was obtained

for data to be collected in introductory assembly language classes during the 1983 spring and summer semesters.

Subjects for the study are students who were enrolled in introductory assembly language classes. Data were gathered in all such classes, the students in which must have completed a prerequisite two semesters' work in computer science or the equivalent that consists of an introductory programming course in BASIC and a programming course in a high-level language such as PASCAL. Although the ability levels of students may vary, the prerequisite assures that all students have been exposed to more than enough training in programming to allow them to complete the simple programs in the test instrument. Furthermore, the variant ability levels, as pointed out by Soloway and others (4), represent real classroom situations.

Data obtained from international students, students twenty-six years of age or older, and students with professional programming experience are not included in the analysis. Data were analyzed from sixty-two students. The data from the total group are used to test hypotheses one, two, and nine. The SAT scores of this population of students, which were necessary to test hypotheses three through eight, were available for thirty-three of the sixty-two students.

Methods of Collection of the Data

The test instrument was administered to the students at a point in each semester after they had been taught to use the branch-on-count (BCT) or branch-on-count-register (BCTR) instructions necessary to implement the for loop in assembly language. Subjects were given a total of fifty minutes to write all three programs; they were not allowed to consult other students, books, or notes. Students who finished early were required to remain in the room. The test instrument was administered by classroom teachers who had been instructed on the conditions of test administration.

The three test programs were presented to the subjects with a stapled-on cover sheet that included directions for taking the test and a section for recording personal data. Subjects were asked to state sex, date of birth, citizenship, social security number, and years of professional programming experience. Each subject's verbal, math, and total scores on the Scholastic Aptitude Test were obtained from the registrar. The test programs and cover sheet are included in Appendix A.

The following items were considered in scoring the tests: whether the appropriate loop was used in each program, whether the program was correct, whether counter variables were used correctly, whether running totals were maintained correctly, and whether data were correctly

assigned to variables. Syntactic errors were not considered unless they were so severe that they obscured the meaning of the program.

Hypotheses Testing Procedures

Nine hypotheses are used to test the data collected for this study. In testing each hypothesis, correlations that achieve the .05 level are considered statistically significant.

The first hypothesis deals with the relationship between using the appropriate loop construct and writing a correct program (see Chapter I definitions). The first hypothesis is tested by computing phi coefficients of correlation between writing a correct program and (a) use of the for loop when appropriate, (b) use of the repeat loop when appropriate, and (c) use of the while loop when appropriate. The level of significance of each correlation is reported.

The second hypothesis deals with the occurrence of assignment errors in incorrect programs. It was tested by determining the frequency of (a) errors in getting new values and assigning them to variables, (b) errors in the implementation of running totals, and (c) errors in the implementation of counters. The percentage of incorrect programs that have each type of listed error is reported.

The third hypothesis deals with the relationship between math aptitude and use of the appropriate loop. It is tested by computing point biserial coefficients of correlation between math scores on the Scholastic Aptitude Test and the (a) use of the for loop when appropriate, (b) use of the repeat loop when appropriate, and (c) use of the while loop when appropriate.

The fourth hypothesis deals with the relationship of math aptitude and assignment errors. It is tested by computing Pearson product moment coefficients of correlation between math scores on the Scholastic Aptitude Test and the (a) total number of assignment errors on the three test programs, (b) number of assignment errors in the implementation of running totals, (c) number or assignment errors in the implementation of counters, and (d) number of errors in assigning a value. The level of significance of each correlation is reported.

The fifth hypothesis deals with the relationship between verbal aptitude and use of the appropriate loop. It is tested by computing point biserial coefficients of correlation between verbal scores on the Scholastic Aptitude Test and the (a) use of the for loop when appropriate, (b) use of the repeat loop when appropriate, and (c) use of the while loop when appropriate. The level of significance of each correlation is reported.

The sixth hypothesis deals with the relationship between verbal aptitude and assignment errors. It is tested by computing Pearson product moment coefficients of correlation between verbal scores on the Scholastic Aptitude Test and the (a) total number of assignment errors in the three test programs, (b) the number of assignment errors in the implementation of counters, (c) the number of assignment errors in the implementation of running totals, and (d) the number of errors in getting a new value. The level of significance of each correlation is reported.

The seventh hypothesis deals with the relationship between scholastic aptitude and use of the appropriate loop. It is tested by computing point biserial coefficients of correlation between total scores on the Scholastic Aptitude Test and the (a) use of the for loop when it is appropriate, (b) use of the repeat loop when it is appropriate, and (c) use of the while loop when it is appropriate. The level of significance of each coefficient of correlation is reported.

The eighth hypothesis deals with the relationship between scholastic aptitude and assignment errors. It is tested by computing Pearson product moment coefficients of correlation between total scores on the Scholastic Aptitude Test and the (a) total number of assignment errors

on the three test programs, (b) the number of assignment errors in getting a new value, (c) the number of errors in the implementation of counters, and (d) the number of errors in the implementation of running totals. The level of significance of each coefficient of correlation is reported.

The first three divisions of the ninth hypothesis deal with the relationship between gender and use of the appropriate loop. They are tested by computing phi coefficients of correlation between gender and the (a) use of the for loop when appropriate, (b) use of the repeat loop when appropriate, and (c) use of the while loop when appropriate. The level of significance of each coefficient is reported.

The last four divisions of the ninth hypothesis deal with the relationship between gender and assignment errors. They are tested by computing point biserial coefficients of correlation between gender and the (a) total number of assignment errors in the three test programs, (b) number of assignment errors in implementation of running totals, (c) the number of assignment errors in getting a new value, and (d) the number of assignment errors in implementation of counters. The level of significance of each coefficient of correlation is reported.

CHAPTER BIBLIOGRAPHY

1. DuBois, Philip H., "College Board Scholastic Aptitude Test," The Seventh Mental Measurements Yearbook, edited by Oscar K. Buros (2 volumes), Highland Park, N. J., The Gryphon Press, 1972, Vol. 1, pp. 646-648.
2. Scholastic Aptitude Test, Princeton, Educational Testing Service, 1970.
3. Soloway, E., and others, "Cognition and Programming: Why Your Students Write Those Crazy Programs," Proceedings of the National Educational Computing Conference, edited by Diana Harris and Laurie Nelson-Hearn, Iowa City, University of Iowa WEEG Computing Center, 1981, pp. 206-219.
4. Soloway, E., and others, "What Do Novices Know about Programming?," Directions in Human-Computer Interaction, edited by A. Badre and B. Shneiderman, Norwood, Ablex Publishing Corporation, 1982, pp. 27-55.
5. Swigger, Kathleen, and Layne Wallace, "Use of Appropriate Looping Structures: Expert vs. Novice," Proceedings of the Human Factors Society, edited by Richard E. Edwards, Santa Monica, Human Factors Society, 1982, pp. 999-1003.

CHAPTER IV

ANALYSIS OF THE DATA

The purpose of this chapter is to present the data obtained in the study. The nine hypotheses were tested by the procedures described in the preceding chapter. The results of these procedures are presented in this chapter following a summary of students' performance on the three problems. Only those correlations which achieve the .05 level are considered statistically significant for the purposes of this study, although other levels of significance are reported.

Findings

Analysis of the programs written for this test reveals that 41.9 per cent of sixty-two students chose the appropriate for loop for the first problem; 35.5 per cent used a repeat loop; 1.6 per cent used a while loop, and 21.0 per cent used other constructs. For this problem, 33.9 per cent of the students wrote correct programs, and 20.9 per cent wrote correct programs with the appropriate looping structure.

For the second problem, 85.5 per cent of the students chose the appropriate repeat loop; 1.6 per cent chose the

for loop; 1.6 per cent chose the while loop, and 11.3 per cent used other constructs. For this problem, 32.3 per cent of the students wrote correct programs; all of the students who wrote correct programs for this problem used the appropriate looping structure.

For the third problem, which requires a while loop, 79.0 per cent of the students chose the appropriate looping structure; no students used a for loop, 6.5 per cent used a repeat loop, and 9.7 per cent used other constructs. For this problem, 33.6 per cent of the students wrote correct programs, and 32.3 per cent wrote correct programs with the appropriate looping structure. Of the sixty-two students, 4.8 per cent either did not attempt or did not complete this program. These data are reported in Table III.

The first hypothesis states that there will be a significant positive correlation between writing a correct program and (a) using the for loop when appropriate, (b) using the repeat loop when appropriate, and (c) using the while loop when appropriate. It was tested by computing phi coefficients of correlation between writing a correct program and using each of the three loops when appropriate.

For the first test program, for which a for loop is appropriate, a correlation of .29 was found between choosing the appropriate loop and writing a correct program; the

TABLE III
PERCENTAGE OF STUDENTS USING CERTAIN LOOPING
STRUCTURES AND WRITING CORRECT PROGRAMS

Percentage of students who:	Problem 1	Problem 2	Problem 3
Used <u>for</u> loop	41.9%	1.6%	none
Used <u>repeat</u> loop	35.5%	85.5%	6.5%
Used <u>while</u> loop	1.6%	1.6%	79.0%
Used other looping structures	21.0%	11.3%	9.7%
Did not complete program	none	none	4.8%
Wrote correct programs	33.9%	32.3%	33.9%
Wrote correct programs with appropriate loop	20.9%	32.3%	32.3%

correlation is significant at the .01 level. For the second test program, for which the repeat loop is appropriate, a correlation of .28, significant at the .01 level, was found between writing the correct program and using the appropriate loop. For the third test program, which requires a while loop, a correlation of .28, significant at the .01 level, was found between writing a correct program and using the appropriate loop. All three divisions of the first hypothesis are accepted because the level of significance achieved in each exceeds the .05 level defined for this study.

Table IV presents the phi coefficients of correlation obtained between writing a correct program and using the appropriate loop for each of the three test programs. The level of significance of each correlation is also indicated.

TABLE IV
CORRELATION BETWEEN WRITING A CORRECT PROGRAM
AND CHOOSING THE APPROPRIATE LOOP

Program	Loop	Phi	p
Program One	For	.29	.01
Program Two	Repeat	.28	.01
Program Three	While	.28	.01

The second hypothesis states that assignment will be used correctly in the majority of incorrect programs. The data indicate that only 42.3 per cent of the incorrect programs had no assignment errors. Thus, the second hypothesis is rejected.

However, each of the three subordinate elements of the second hypothesis is accepted. The first of these subordinate elements states that the majority of incorrect programs will correctly get new values and assign them to variables. Of the incorrect programs, 88.4 per cent performed this function correctly; 92.4 per cent of all programs performed it correctly.

The second subordinate element of the second hypothesis states that assignment will be used correctly in the implementation of running totals in the majority of incorrect programs. Assignment was used correctly in the implementation of running totals in 54.5 per cent of the incorrect programs and in 69.9 per cent of all programs.

The third subordinate element of the second hypothesis states that assignment statements will be used correctly in the implementation of counters in the majority of incorrect programs. The data reveal that 60.3 per cent of incorrect programs either implemented counters correctly or did not require the use of a counter. Of the incorrect programs requiring a counter, 55.1 per cent implemented counters correctly. The data pertaining to the subordinate elements of the second hypothesis are reported in Table V.

TABLE V
PERCENTAGE OF PROGRAMS WITHOUT CERTAIN
ERRORS IN ASSIGNMENT

Error	Incorrect Programs	All Programs
Errors in getting a new value	88.4%	92.9%
Errors in implementation of running totals	54.6%	69.9%
Errors in implementation of counters	60.3% 55.1%	73.8% 69.4%**

*Percentage of incorrect programs requiring a counter.
**Percentage of all programs requiring a counter.

The third hypothesis states that there will be a significant positive correlation between mathematical aptitude and (a) use of the for construct when appropriate, (b) use of the repeat construct when appropriate, and (c) use of the while construct when appropriate. The hypothesis was tested by determining the point biserial coefficient of correlation between mathematical aptitude, as measured by the quantitative score on the Scholastic Aptitude Test (7), and the use of each of the three loop constructs when appropriate.

The data indicate that the coefficient of correlation between math aptitude and use of the for construct on the first test program is .30. The coefficient of correlation between mathematical aptitude and use of the repeat loop on the second program is .33. The coefficient of correlation between mathematical aptitude and the use of the while loop on the third program is .02. None of these relationships is significant at the .05 level. The third hypothesis is rejected. These data are reported in Table VI.

TABLE VI
RELATIONSHIP BETWEEN MATHEMATICAL APTITUDE AND
CHOICE OF APPROPRIATE LOOPING STRUCTURE

Problem	Correlation	Significance
Program One	.30	$p > .05$
Program Two	.33	$p > .05$
Program Three	.02	$p > .05$

The fourth hypothesis deals with the relationship between mathematical aptitude and errors in assignment on the three test programs. It states that there will be a significant negative correlation between mathematical aptitude (SAT-M) and (a) the total number of errors on assignment on the test programs, (b) the number of errors in assignment in getting a new value, (c) the number of errors in assignment in the implementation of running totals, and (d) the number of errors in assignment in the implementation of counters. This hypothesis was tested by determining Pearson product moment coefficients of correlation between mathematical aptitude and each of the four measures of errors in assignment.

The data reveal that the correlation between the total number of errors in assignment and math aptitude is $-.12$, significant at the $.25$ level. The correlation between mathematical aptitude and the number of errors in getting a new value is $-.09$, significant at the $.31$ level. The correlation between mathematical aptitude and the number of errors in the implementation of running totals is $-.16$, significant at the $.19$ level. The correlation between mathematical aptitude and number of errors in the implementation of counters is $-.15$, significant at the $.20$ level. Although all four correlations between math aptitude and measures of assignment errors are negative, none achieves

the .05 level of significance, so the fourth hypothesis is rejected. These data are reported in Table VII.

TABLE VII
RELATIONSHIP BETWEEN MATHEMATICAL APTITUDE AND
VARIOUS MEASURES OF ERRORS IN ASSIGNMENT

Error	r	p
All assignment errors	-.12	.25
Errors in getting a new value	-.09	.31
Errors in implementation of running totals	-.16	.19
Errors in implementation of counters	-.15	.20

The fifth hypothesis states that there will be a significant positive correlation between verbal aptitude (SAT-V) and (a) use of the for construct when it is appropriate, (b) use of the repeat construct when it is appropriate, and (c) use of the while construct when it is appropriate. It was tested by determining point biserial coefficients of correlation between verbal aptitude (SAT-V) and the use of each of the three looping structures when appropriate.

The coefficient of correlation between use of the for construct on the first test program and verbal aptitude is .01. The coefficient of correlation between verbal aptitude and use of the repeat construct on the second test program

is .10, and the coefficient of correlation between verbal aptitude and use of the while construct on the third test program is .22. Because none of these coefficients of correlation is significant at the .05 level, the fifth hypothesis is rejected. These data are reported in Table VIII.

TABLE VIII
RELATIONSHIP BETWEEN VERBAL APTITUDE AND USE OF
APPROPRIATE LOOPING STRUCTURE

Problem	4	Significance
Program One	.01	$p > .05$
Program Two	.10	$p > .05$
Program Three	.22	$p > .05$

The sixth hypothesis deals with the relationship between verbal aptitude (SAT-V) and errors in assignment. It states that there will be a significant negative correlation between verbal aptitude and (a) the total number of assignment errors on the test programs, (b) the number of errors in getting a new value on the test programs, (c) the number of assignment errors in the implementation of running totals in the three test programs, and (d) the number of assignment errors in the implementation of counters in the three test programs. This hypothesis was tested by

determining the Pearson product moment coefficient of correlation between verbal aptitude and each of the four measures of errors in assignment.

The correlation between verbal aptitude (SAT-V) and the total number of assignment errors on the three test programs is .02, significant at the .46 level. The correlation between verbal aptitude and the number of assignment errors in getting a new value is $-.05$, significant at the .40 level. The correlation between verbal aptitude and number of assignment errors in the implementation of running totals is .03, significant at the .43 level; and the correlation between verbal aptitude and the number of assignment errors in the implementation of counters is $-.09$, significant at the .29 level. Because none of these correlations achieves the .05 level of significance, the sixth hypothesis is rejected. These data are reported in Table IX.

TABLE IX
RELATIONSHIP BETWEEN VERBAL APTITUDE AND VARIOUS
MEASURES OF ERRORS IN ASSIGNMENT

Error	r	p
All assignment errors	.02	.46
Errors in getting a new value	$-.05$	.40
Errors in implementation of running totals	.03	.43
Errors in implementation of counters	$-.10$	.30

The seventh hypothesis states that there will be a significant positive correlation between scholastic aptitude, as measured by the total score on the Scholastic Aptitude Test (SAT-T) and the use of each of the three looping constructs in the appropriate test program. This hypothesis was tested by determining the point biserial coefficient of correlation between scholastic aptitude and the use of each of the three looping constructs when appropriate. The correlation between scholastic aptitude and use of the for construct on the first program is .18; the correlation between scholastic aptitude and use of the repeat construct on the second test program is .17, and the correlation between scholastic aptitude and use of the while construct on the third program is .16. Because none of these correlations achieves the .05 level of significance, the seventh hypothesis is rejected. These data are reported in Table X.

TABLE X

RELATIONSHIP BETWEEN CHOICE OF APPROPRIATE
LOOPING CONSTRUCT AND SCHOLASTIC APTITUDE

Problem	r	Significance
Program One	.18	$p > .05$
Program Two	.17	$p > .05$
Program Three	.16	$p > .05$

The eighth hypothesis deals with the relationship between scholastic aptitude (SAT-T) and errors in assignment. It states that there will be a significant negative correlation between scholastic aptitude and (a) the total number of errors in assignment on the three test programs, (b) the number of assignment errors in getting a new value, (c) the number of assignment errors in the implementation of running totals, and (d) the number of assignment errors in the implementation of counters. This hypothesis was tested by determining the Pearson product moment coefficient of correlation between scholastic aptitude and each of the four measures of assignment errors. No significant relationship was found.

The data indicate that the correlation between scholastic aptitude and the total number of assignment errors is .05, significant at the .4 level. The correlation between scholastic aptitude and the number of errors in getting a new value is $-.14$, significant at the .22 level. The correlation between scholastic aptitude and the number of errors in the implementation of running totals is $-.01$, significant at the .49 level; and the correlation between scholastic aptitude and the number of assignment errors in the implementation of counters is $-.13$, significant at the .24 level. None of these correlations achieves the .05 level of significance, so the eighth hypothesis is rejected. These data are reported in Table XI.

TABLE XI
RELATIONSHIP BETWEEN SCHOLASTIC APTITUDE AND VARIOUS
MEASURES OF ERRORS IN ASSIGNMENT

Error	r	p
All assignment errors	.05	.40
Errors in getting a new value	.14	.22
Errors in implementation of running totals	-.01	.49
Errors in implementation of counters	-.13	.24

The ninth hypothesis considers the relationship between gender and selected variables; it is considered in seven divisions. For the first three of these divisions, it is hypothesized that there will be no significant correlation between gender and (a) use of the for construct when appropriate, (b) use of the repeat construct when appropriate, and (c) use of the while construct when appropriate. These divisions of the ninth hypothesis were tested by determining phi coefficients of correlation between gender and use of the appropriate looping construct.

A correlation of .05, significant at the .34 level, was found between gender and use of the for construct on the first test program. A correlation of .11, significant at the .19 level, was found between gender and use of the

repeat construct on the second program; and a correlation of .13, significant at the .15 level, was found between gender and use of the while construct on the third program. Because none of these correlations approached the .05 level of significance defined for this study, these divisions of the ninth hypothesis were accepted. These data are presented in Table XII.

TABLE XII
RELATIONSHIP BETWEEN GENDER AND CHOICE
OF APPROPRIATE LOOPING CONSTRUCT

Problem	Phi	p
Program One	.05	.34
Program Two	.11	.19
Program Three	.13	.15

The last four divisions of the ninth hypothesis consider the relationship between gender and errors in assignment. These divisions were tested by determining point biserial coefficients of correlation between gender and each of the four measures of errors in assignment. No significant relationship was found.

A correlation of .18, significant at the .15 level, was found between gender and the total number of errors in assignment. A correlation of .12, significant at the

.25 level, was found between gender and the total number of errors in getting a new value. A correlation of .23, significant at the .10 level, was found between gender and the total number of errors in the implementation of running totals, and a correlation of .11, significant at the .27 level, was found between gender and the number of errors in the implementation of counters. Because none of these correlations approached the .05 level of significance, these divisions of the ninth hypothesis were accepted. These data are reported in Table XIII.

TABLE XIII
RELATIONSHIP BETWEEN GENDER AND VARIOUS
MEASURES OF ERRORS IN ASSIGNMENT

Error	r	p
All assignment errors	.18	.155
Errors in getting a new value	.12	.254
Errors in implementation of running totals	.23	.104
Errors in implementation of counters	.11	.267

Discussion

The analysis of the results of this study will consider each of the three factors evaluated in the test programs: correctness of the program, use of the appropriate

loop, and errors in assignment. The relationship of these factors to mathematical, verbal, and scholastic aptitude, and to gender will also be considered.

The first hypothesis predicts a significant positive correlation between writing a correct program and using the appropriate loop on each problem. The results of this study indicate that all three correlations are significant at the .01 level. The phi coefficient of correlation between these two factors for the first problem is .29; for the second problem, .28; for the third problem, .28. These findings are similar to those of Swigger and Wallace (10) who administered the same instrument to novice programmers writing in PL/I. Their findings show a Pearson product moment coefficient of correlation between the same two factors of .25 for the first problem, of .28 for the second problem, and of .41 for the third problem. They found a significant correlation for only the third problem when they administered the instrument to novice PASCAL programmers.

Although statistically significant coefficients of correlation were found between writing a correct program and using the appropriate loop, none of the correlations was high enough to be conclusive. However, no data are furnished by the present study to indicate how these two factors are related; it is not within the scope of a correlational study to indicate, for instance, whether the relationship is causal.

The emphasis in modern computer science education on structured programming (3) makes it especially important that the relationship between writing correct programs and using appropriate loops be explored in further studies. The use of appropriate loops is one method of achieving the goal of structured programming that program structure should reflect program meaning. Dijkstra (3) developed his theory of structured programming to help make intelligible the large programs now common in computer science. Although students do not write programs as large as those of professional programmers, novices are taught the principles of structured programming to prepare them for more complex programming tasks. The possibility that implementing a technique of structured programming may also facilitate learning of programming is an issue which should be investigated.

Although the second hypothesis predicts that the majority of incorrect programs will have no errors in assignment, only 38.5 per cent of the incorrect programs had no errors in assignment. The hypothesis is rejected, but the findings may suggest an area of concern to teachers of assembly language. These teachers typically see the final working programs of students in their third semester of work in computer science; the correct programs have no assignment errors, and the teachers may be unaware that

assignment is such a problem for students who, though they are still novice programmers, have had almost three semesters' experience in using assignment.

Examination of the results of the subordinate elements of the second hypothesis reveals which aspects of assignment are difficult for the novice assembly language programmers included in this study. These subordinate elements predict that the majority of incorrect programs will use assignment correctly in getting new values, maintaining running totals, and maintaining counter variables. All three subordinate elements of the second hypothesis were supported by the data.

Using assignment in getting a new value is the least troublesome of these three tasks for the students. Only fourteen of the 183 programs submitted had errors of this type. These fourteen errors were made by six students, five of whom made the same type of error in more than one program. Even though only 9.6 per cent of the programmers made errors of this type, it is noteworthy that a basic function essential to programming in any language remains a problem for some students even after three programming courses.

Maintaining a running total and maintaining a counter, areas of approximately equal difficulty for the students in this sample, produced more errors than did getting a

new value. On problem two, seventeen of the sixty-two students (27.4%) made assignment errors in maintaining running totals; eighteen (29.0%) made errors in maintaining counters. Twelve students made errors in both the implementation of a counter and the implementation of a running total for problem two.

For the third problem, the findings are similar. Twenty students (32.2%) made errors in maintaining running totals; eighteen (29.0%) made errors in maintaining counters. Eleven students made errors in both the implementation of a counter and the implementation of a running total for problem three.

On the first problem, eighteen students (29.0%) made errors in maintaining running totals, while only twelve (19.3%) made errors in maintaining counters. This difference is understandable because any student who used the appropriate for loop for the first problem did not need to use a counter, and consequently did not have the opportunity to make that error. These data are presented in Table XIV.

These findings should be considered by teachers of assembly language. Students often regard assembly language programming as difficult, and teachers recognize the problems caused because assembly language requires a better knowledge of how a computer works and because assembly language often requires more lines of code (thus providing

TABLE XIV
PERCENTAGE OF STUDENTS MAKING ASSIGNMENT ERRORS
IN THE IMPLEMENTATION OF COUNTERS
AND RUNNING TOTALS

Problem	Counters	Running Totals
Program One	19.3%	29.0%
Program Two	29.0%	27.4%
Program Three	33.9%	30.5%

more opportunities for error) to solve a given problem than do high level languages. However, teachers accustomed to grading working programs may well be unaware that students in their third semester of programming still find it difficult to use assignment correctly in the implementation of running totals and counters. Although students should have learned these processes in earlier courses, the presence of assignment errors in the majority of programs submitted for this study indicates that this area is one which should be noted by teachers of assembly language.

The fourth, sixth, and eighth hypotheses deal with the relationship between errors in assignment and mathematical, verbal, and scholastic aptitude. None of these correlations approached significance. For the students in this sample, the number of assignment errors made did not have any consistent relationship to the abilities measured by

the Scholastic Aptitude Test. The lack of such a relationship indicates that further study is needed to determine whether making errors in assignment may be related to some other factor such as logical reasoning processes, cognitive style, or teaching methods.

When Soloway and others (9) assigned the three test programs used in this study to novice PASCAL programmers in an introductory course, the researchers found a significant difference between the number of students using assignment correctly with running totals and the number of students using assignment correctly with counters for the first and third problems. The researchers suggest that this difference in student performance on two nearly identical processes may be explained by their theory that novice programmers have not organized their knowledge of counters and running totals into the same knowledge net, or set of interrelated frames, used by expert programmers.

Although the results of the present study support those of the Soloway study for the second problem, where no significant difference was found, the findings for the third problem are not consistent with those of the previous study. One factor which may help to explain the difference in results is the fact that the students in the present study are in their third, not first computer science course. They may be moving toward the organization of programming

knowledge held by expert programmers (6). However, close examination of the data for this study indicate another important consideration. Twelve students with errors in either maintaining a running total or a counter made errors in both areas. These students, it would appear, did not have difficulty because they had not linked the frames for running total and for counter. Instead, it appears that they did not understand the process basic to both maintaining a running total and maintaining a counter--that of initializing a variable, then adding to it during successive iterations of a loop. Clearly, further study of the organization of knowledge by novice programmers is necessary in order to understand how they organize knowledge and whether they are moving toward the organization of knowledge characteristic of expert programmers.

The present study does not consider the significance of the difference between errors in the implementation of running totals and errors in the implementation of counters for the first problem. This relationship is not considered meaningful by this researcher because use of the appropriate loop makes the use of a counter unnecessary, although solution of the problem still requires the use of a counter. Thus, it is impossible to make a valid comparison.

The third, fifth, and seventh hypotheses consider the relationship between choosing the appropriate loop for each

of the three problems and mathematical, verbal, and scholastic aptitude. Although none of these correlations was statistically significant, the coefficient of correlation between mathematical aptitude and use of the appropriate loop was .30 for problem one, and .34 for problem two. Although these were not significant for a sample of thirty-three, they were similar to the statistically significant correlation of .36 found by Fowler and Glorfeld (4) between the quantitative score on the Scholastic Aptitude Test and success in an introductory programming course. Their sample size was 151. Because none of the abilities measured by the Scholastic Aptitude Test has a statistically significant degree of relationship with any of the variables considered in this study, further studies should be conducted to determine which factors are related to the ability to use looping and assignment in computer programs.

The ninth hypothesis predicts that no significant correlation will be found between gender and use of the appropriate loop for each problem or between gender and four measures of errors in assignment. No significant correlation was found, and none of the correlations found is greater than .12, with most of them less than .15. These results are similar to those found by Fowler and Glorfeld (4) and by Mazlack (6). Fowler and Glorfeld found a correlation of .02 between gender and programming

aptitude; the correlation is not significant. Mazlack, with a sample of 1,350 students found a correlation of .05 between gender and final grade in an introductory programming course. That correlation, because of the large sample size, is significant at the .008 level. Gender appears to be of almost no value in predicting success in computer science, and in this study it has no significant relationship with writing a correct program, using the appropriate loop, or making errors in assignment.

Additional Findings

The relationship between writing a correct program and mathematical, verbal, and scholastic aptitude was measured by determining point biserial coefficients of correlation between writing a correct program and the mathematical, verbal, and total scores on the Scholastic Aptitude Test for each of the three problems. Data pertaining to the relationship between writing a correct program and mathematical, verbal, and scholastic aptitude are presented in Table XV.

Correlations of .21, .29, and .25 were found between mathematical aptitude and writing a correct program for the first, second, and third problems respectively. None of these correlations achieves the .05 level of significance, but the level of correlation is similar to that reported by Fowler and Glorfeld (4) between mathematical aptitude and

TABLE XV
RELATIONSHIP BETWEEN VARIOUS MEASURES OF
SCHOLASTIC APTITUDE AND WRITING
A CORRECT PROGRAM

Aptitude	Problem 1	Problem 2	Problem 3
Mathematical aptitude (SATM)	$r = .21$ $p > .05$	$r = .29$ $p > .05$	$r = .26$ $p > .05$
Verbal aptitude (SATV)	$r = -.09$ $p > .05$	$r = -.12$ $p > .05$	$r = .10$ $p > .05$
Scholastic aptitude (SATT)	$r = .06$ $p > .05$	$r = .02$ $p > .05$	$r = .14$ $p > .05$

success in an introductory programming course. They are also close to the levels of correlation found by Alspaugh (1, 2) between programming proficiency in two languages and measures of mathematical reasoning. The relationship between mathematical aptitude and the ability to write correct programs is one which should be investigated further. Of particular interest might be a study to determine whether a factor such as the ability to reason logically might have a stronger relationship to the ability to write correct programs than does mathematical aptitude.

Even though a correlation significant at .05 level was not found between mathematical aptitude and writing a correct program, that relationship appears to be stronger

than the relationship between writing a correct program and verbal or scholastic aptitude. For the first problem, the point biserial coefficient of correlation between verbal aptitude and writing a correct program is $-.09$; for the second, $-.13$; and for the third, $.10$. The correlation between scholastic ability and writing a correct program is $.06$ for the first problem, $.02$ for the second problem, and $.14$ for the third problem. The fact that a stronger relationship is evident between writing a correct program and mathematical aptitude than between writing a correct program and verbal or scholastic aptitude (of which verbal aptitude is an element) is consistent with previous studies (1, 2, 4, 5, 11) concerned with the prediction of success in introductory computer science courses. Nevertheless, because the reported correlations in these studies between mathematical aptitude and either writing a correct program or success in an introductory course are low even when significant, the relationship of other variables to these two factors should be studied.

Summary

The study supported the first hypothesis, dealing with the relationship between writing a correct program and choosing the appropriate loop and the ninth hypothesis, dealing with the lack of relationship between gender and either choice of the appropriate loop or errors in

assignment. It also supported three subordinate elements of the second hypothesis, dealing with the occurrence of three types of assignment errors in incorrect programs.

CHAPTER BIBLIOGRAPHY

1. Alspaugh, C. A., "Identification of Some Components of Computer Programming Aptitude," Journal of Research in Mathematics Education, 3 (March, 1972), 89-98.
2. _____, "A Study of the Relationship between Student Characteristics and Proficiency in Symbolic and Algebraic Computer Programming," unpublished doctoral dissertation, University of Missouri, Columbia, Missouri, 1970.
3. Dijkstra, E., "Notes on Structured Programming," Structured Programming, by O. Dahl and others, New York, Academic Press, 1972, pp. 1-81.
4. Fowler, George C., and Louis W. Glorfeld, "Predicting Aptitude in Computing: A Classification Model," AEDS Journal, 14 (Winter, 1981), 96-109.
5. Konvalina, John, and others, "Identifying Factors Influencing Computer Science Aptitude and Achievement," AEDS Journal, 16 (Winter, 1983), 106-112.
6. Mazlack, Lawrence J., "Identifying Potential to Acquire Programming Skill," Communications of the ACM, 23 (January, 1980), 14-17.
7. McKeithen, Katherine, and others, "Knowledge Organization and Skill Differences in Computer Programmers," Cognitive Psychology, 13 (July, 1980), 307-325.
8. Scholastic Aptitude Test, Princeton, Educational Testing Service, 1970.
9. Soloway, E. and others, "Cognition and Programming: Why Your Students Write Those Crazy Programs," Proceedings of the National Educational Computing Conference, edited by Diane Harris and Laurie Nelson-Hearn, Iowa City, University of Iowa WEEG Computing Center, 1981, pp. 206-219.

10. Swigger, Kathleen, and Layne Wallace, "Use of the Appropriate Looping Structures: Expert vs. Novice," Proceedings of the Human Factors Society, edited by Richard Edwards, Santa Monica, Human Factors Society, 1982, pp. 999-1003.
11. Wileman, Stanley A., and others, "Factors Influencing Success in Beginning Computer Science Courses," Journal of Educational Research, 74 (March-April, 1981), 223-226.

CHAPTER V

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

Summary of Background and Purposes

Despite the public attention focused on the need for technological skills in modern society and the demand for competent computer scientists, little research has been directed toward the teaching of computer science. Although most studies (1, 2, 4, 5, 10, 15) which have been done in this area focus on aptitude for computer science, some recent studies (12, 13, 14) are concentrated on looping and errors in assignment, two specific areas of interest within computer science education.

Because looping and assignment are essential processes in computer programming which are often areas of difficulty for students, it is important to study them and their relationship to certain variables. Although other researchers have investigated errors in looping and assignment by novice programmers in high level languages, none has investigated such errors by novice programmers in assembly language.

The purposes, then, of this study are to investigate the choice of looping structure and errors in assignment of novice assembly language programmers and to determine

the relationship of those two factors to writing a correct program, to gender, and to mathematical, verbal, and scholastic aptitude. Mathematical aptitude is measured by the Scholastic Aptitude Test quantitative score; verbal aptitude is measured by the Scholastic Aptitude Test verbal score, and scholastic aptitude is measured by the Scholastic Aptitude Test total score.

Summary of Procedures

The study was conducted during the spring and summer semesters of 1983 at a state university. A test instrument written by Soloway and others (12) was administered to students in introductory assembly language classes. The test instrument is comprised of three simple programming problems, for each of which a particular looping construct is appropriate.

The test was given by classroom instructors at a point in the semester after students had learned to use the branch-on-count and branch-on-count register instructions. Students were given fifty minutes to complete the three programs; they were not allowed to refer to books, notes, or other students' work. They were required to remain in the classroom for the entire fifty minutes.

Following analysis of the programs submitted by students, phi coefficients of correlation were used to determine the degree of relationship between writing a correct program

and choosing the appropriate loop, and between choosing the appropriate loop and gender. Pearson product moment coefficients of correlation were used to determine the degree of relationship between mathematical, verbal and scholastic aptitude and each of the following: the total number of errors in assignment, assignment errors in the implementation of running totals, assignment errors in the implementation of counters, and assignment errors in getting new values.

Point biserial coefficients of correlation were used to measure the degree of relationship between choosing the appropriate loop in each problem and each of the following: verbal aptitude, mathematical aptitude, and scholastic aptitude. Point biserial coefficients of correlation were also used to measure the degree of relationship between gender and each of the following: total number of errors in assignment, the number of errors in getting a new value, the number of errors in the implementation of running totals, and the number of errors in the implementation of counters.

In addition, the frequency of each of the three types of assignment errors in each program was calculated. The percentage of all programs having assignment errors was calculated, as well as the percentage of incorrect programs having assignment errors. In addition, the frequency of

each of the three types of assignment errors in each program was determined. The percentage of incorrect programs with each of the four errors was calculated for each problem; the percentage of all programs with such errors was also calculated for each problem.

Summary of Findings

In Chapter IV, data collected pertinent to each hypothesis were analyzed and discussed. A summary of these findings follows.

The first hypothesis predicts a statistically significant correlation between writing a correct program and using the appropriate loop. A phi coefficient of correlation of .29 was found for the first problem; of .28, for the second problem; and of .28, for the third problem. All three of these correlations are statistically significant.

The second hypothesis deals with the occurrence of assignment errors in incorrect programs. Because 53.7 per cent of incorrect programs had some kind of assignment error, the second hypothesis is rejected, but the three subordinate elements of the second hypothesis are accepted. Of the incorrect programs, 88.4 per cent got new values correctly; 54.6 per cent implement running totals correctly, and 60.3 per cent either implement counters correctly or do not require the use of a counter.

The third hypothesis predicts a significant positive correlation between mathematical aptitude, as measured by the Scholastic Aptitude Test (11) quantitative scale (SAT-M), and use of the appropriate looping structure for each problem. For the first problem, the point biserial coefficient of correlation between mathematical aptitude and use of the appropriate looping structure is .30; for the second problem, .33; for the third problem, .02. None of these correlations is statistically significant.

The fourth hypothesis predicts a significant negative relationship between mathematical aptitude (SAT-M) and each of the following: (a) total number of assignment errors on the three programs, (b) number of errors in getting a new value, (c) number of assignment errors in the implementation of running totals, and (d) number of assignment errors in the implementation of counters. The Pearson product moment coefficient of correlation between mathematical aptitude and total number of assignment errors is $-.12$; between mathematical aptitude and number of errors in getting a new value, $-.09$; between mathematical aptitude and number of assignment errors in the implementation of counters, $-.09$; between mathematical aptitude and the number of assignment errors in the implementation of running totals, $-.16$. None of these correlations is statistically significant, so the hypothesis is rejected.

The fifth hypothesis predicts a significant positive correlation between verbal aptitude, as measured by the Scholastic Aptitude Test verbal score (SAT-V), and use of the appropriate loop for each of the three programs. For the first problem, the correlation between verbal aptitude and use of the appropriate loop is .01; for the second problem, .10; for the third problem, .22. None of these correlations is statistically significant. The fifth hypothesis is rejected.

The sixth hypothesis predicts a significant negative correlation between verbal aptitude (SAT-V) and each of the following: total number of assignment errors, number of errors in getting a new value, number of errors in the implementation of running totals, and number of errors in the implementation of counters. The Pearson product moment coefficient of correlation between verbal aptitude and total number of assignment errors is .02; between verbal aptitude and number of errors in getting a new value, $-.04$; between verbal aptitude and number of errors in the implementation of running totals, .03; between verbal aptitude and the number of assignment errors in the implementation of counters, $-.10$. None of these correlations is statistically significant, so the sixth hypothesis is rejected.

The seventh hypothesis predicts a significant positive correlation between scholastic aptitude, as measured by the Scholastic Aptitude Test total score (SAT-T), and use of the

appropriate loop in each of the three programs. For the first problem, the point biserial coefficient of correlation between scholastic aptitude and use of the appropriate loop is .18; for the second problem, .17; for the third problem, .16. None of these correlations is statistically significant; the seventh hypothesis is rejected.

The eighth hypothesis predicts a significant negative correlation between scholastic aptitude (SAT-T) and each of the following: (a) total number of errors in assignment, (b) number of assignment errors in getting a new value, (c) number of assignment errors in the implementation of running totals, and (d) number of assignment errors in the implementation of counters. The Pearson product moment coefficient of correlation between scholastic aptitude and the total number of assignment errors is .05; between scholastic aptitude and the number of errors in getting a new value, $-.14$; between scholastic aptitude and the number of errors in the implementation of running totals, $-.01$; between scholastic aptitude and the number of errors in the implementation of counters, $-.12$. None of these correlations is statistically significant, so the eighth hypothesis is rejected.

The ninth hypothesis predicts that there will be no significant correlation between gender and each of the following: use of the appropriate loop in each program, the total number of assignment errors, the number of assignment errors in getting a new value, the number of

assignment errors in the implementation of running totals, and the number of assignment errors in the implementation of counters. The phi coefficient of correlation between use of the appropriate loop is .05 for the first problem; .11, for the second problem, and .13, for the third problem. The point biserial coefficient of correlation between gender and the total number of assignment errors is .18; between gender and the number of errors in getting a new value, .12; between gender and the number of errors in the implementation of running totals, .22; between gender and the number of errors in the implementation of counters, .11. None of these correlations achieved the .05 level of significance, so the ninth hypothesis is accepted.

Conclusions

The following conclusions are made based upon a consideration of the results of the study.

1. If the abilities measured by the Scholastic Aptitude Test (11) are not significantly related to either errors in assignment, choice of the appropriate looping structure, or writing a correct program, then some other causal factor must be associated with students' success in these aspects of assembly language programming. Although the Scholastic Aptitude Test is designed to measure aptitude for college work, it is apparently inadequate for forecasting success in assembly language programming.

2. The significant positive correlation between writing a correct program and choosing the appropriate looping construct seems to confirm the relationship between expertise and the organization of knowledge indicated by previous studies. Students who understand the purpose of a program and choose a looping structure reflective of that purpose are more likely to write correct programs.

3. The lack of significant correlation between gender and either choice of appropriate looping construct or number of errors in assignment indicates that gender would be a very poor basis for determining teaching strategies. These results confirm those of previous studies (8, 15) indicating that gender is not valuable as a predictor of success in computer science.

4. Assignment errors are an important reason that programs written by novice assembly language programmers fail to achieve the objectives established for the programs. A majority of incorrect programs written for this study had some kind of assignment error which would prevent the program from working correctly.

Recommendations

The following recommendations are made based on a consideration of the results of this study.

1. The relationship between teaching the techniques of structured programming and successful programming efforts

should be investigated. The results of this study indicate a significant positive correlation between choosing the appropriate loop (thereby implementing the principle of structured programming which requires the structure of a program to reflect its meaning) and writing a correct program. Further research should be done in order to determine whether the application of this and other techniques of structured programming may help students to write programs which function correctly the first time they are submitted to a computer.

2. Further research should be carried out to determine how novice programmers organize their knowledge of programming constructs, how that organization differs from that of experts, and whether there is an evolutionary pattern of movement from the organization of knowledge characteristic of novices to that characteristic of expert programmers.

3. Although, as Cheney (4, p. 288) points out, the ability to achieve a good grade on a programming examination is not identical to the ability to write a good program, most of the research concerning aptitude for computer science uses course grades which are heavily weighted by students' performance on tests to measure students' aptitude for computer science. Therefore, there is a need for research which investigates the relationship between the

ability to write correct programs and other significant factors, such as learning style and the ability to use logical reasoning processes.

4. Although several researchers (1, 2, 5, 9, 14) have found either mathematical aptitude or a background in mathematics to be significantly correlated with success in computer science, none of the correlations has been high. Mathematical aptitude or experience clearly is not a definitive factor for computer science aptitude. Higher correlations are reported by both Kurtz (7), who used an instrument measuring levels of intellectual development to predict success in computer science, and Cheney (4) who investigated the relationship between cognitive style and success in computer science. However, both of these researchers had small samples. Further research with larger samples is necessary to study whether either of the factors they studied is highly correlated with the ability to write good programs.

5. A long-range study might allow researchers greater insight concerning the development of semantic knowledge by novice programmers. If novice programmers were tested at regular intervals, researchers would have opportunity to study the developmental process involved in becoming a programmer.

6. The relationship between course grades, the ability to write correct programs and to use appropriate looping constructs should be investigated.

7. Teachers of assembly language programming should be aware that their students may be having difficulty with basic processes such as looping and assignment even after three computer science classes. Teachers should structure their classes so that they are able to see the initial efforts of students to write programs, not only final, correct programs. Some methods of achieving this goal are to have students write brief programs in class, to have students write brief programs for tests, and to utilize a laboratory setting in which a teacher is able to observe students' efforts as they write and submit programs.

8. Teachers of computer science should try to emphasize basic programming constructs and the structure of well-written programs. Although syntactic knowledge often becomes the focus of computer science classes, research seems to indicate that it is the organization of knowledge which distinguishes novice programmers from expert programmers. Teachers should attempt to help students organize their knowledge of programming.

9. Teachers should work with students having trouble writing correct programs as the students design the programs so that students will learn the importance of good structure.

CHAPTER BIBLIOGRAPHY

1. Alspaugh, C. A., "Identification of Some Components of Computer Programming Aptitude," Journal of Research in Mathematics Education, 3 (March, 1972), 89-98.
2. _____, "A Study of the Relationship between Student Characteristics and Proficiency in Symbolic and Algebraic Computer Programming," unpublished doctoral dissertation, University of Missouri, Columbia, Missouri, 1970.
3. Barker, R. J., and E. A. Unger, "A Predictor for Success in an Introductory Programming Class Based upon Abstract Reasoning Development," ACM SIGSCE Bulletin, 15 (February, 1983), 154-158.
4. Cheney, Paul, "Cognitive Style and Student Programming Ability: An Investigation," AEDS Journal, 13 (Summer, 1980), 285-291.
5. Fowler, George C., and Louis W. Glorfeld, "Predicting Aptitude in Computing: A Classification Model," AEDS Journal, 14 (Winter, 1981), 96-109.
6. Konvalina, John, and others, "Identifying Factors Influencing Computer Science Aptitude and Achievement," AEDS Journal, 16 (Winter, 1983), 106-112.
7. Kurtz, B. L., "Investigating the Relationship between the Development of Abstract Reasoning and Performance in an Introductory Programming Class," ACM SIGCSE Bulletin, 12 (February, 1980), 14-17.
8. Mazlack, Lawrence J., "Identifying Potential to Acquire Programming Skill," Communications of the ACM, 23 (January, 1980), 14-17.
9. McKeithen, Katherine, and others, "Knowledge Organization and Skill Differences in Computer Programmers," Computer Psychology, 13 (July, 1980), 307-325.

10. Petersen, C. G., and T. G. Howe, "Predicting Academic Success in Introduction to Computers," AEDS Journal, 12 (Summer, 1979), 182-191.
11. Scholastic Aptitude Test, Princeton, Educational Testing Service, 1970.
12. Soloway, E., and others, "Cognition and Programming: Why Your Students Write Those Crazy Programs," Proceedings of the National Educational Computing Conference, edited by Diana Harris and Laurie Nelson-Hearn, Iowa City, University of Iowa WEEG Computing Center, 1981, pp. 206-219.
13. Soloway, E., and others, "What Do Novices Know about Programming," Directions in Human-Computer Interaction, edited by A. Badre and B. Shneiderman, Norwood, Ablex Publishing Corporation, pp. 27-55.
14. Swigger, Kathleen, and Layne Wallace, "Use of the Appropriate Looping Structures: Expert vs. Novice," Proceedings of the Human Factors Society, edited by Richard Edwards, Santa Monica, Human Factors Society, 1982, pp. 999-1003.
15. Wileman, S. A., and others, "Factors Influencing Success in Beginning Computer Science Courses," Journal of Educational Research, 74 (March-April, 1981), 223-226.

APPENDIX A

COVER SHEET

Social Security Number: _____

Sex: _____ Male _____ Female

Birthdate: _____

United States Citizen: _____ Yes _____ No

Years of professional programming experience: _____

Here are three simple problems which should be coded in assembly language. Please write complete programs, not just program segments. You may use pencil or pen. You will be given 50 minutes to complete all three programs. If you finish before the fifty-minute period is over, sit and look over your work. Do not talk to your neighbor. Do not use books or notes.

Your programs will be used to determine what computer programmers need to learn and how they should be taught.

TEST INSTRUMENT*

Problem One

Write a program which reads 10 integers and then prints out the average. Remember, the average of a series of numbers is the sum of those numbers divided by how many numbers there are in the series.

Problem Two

Write a program which repeatedly reads in integers until their sum is greater than 100. After reaching 100, the program should print out the average of the integers entered.

Problem Three

Write a program which repeatedly reads in integers until it reads the integer 99999. After seeing 99999, it should print out the correct average. That is, it should not count the final 99999.

*E. Soloway, and others, "Cognition and Programming: Why Your Students Write Those Crazy Programs," Proceedings of the National Educational Computing Conference, edited by Diana Harris and Laurie Nelson-Hearn, Iowa City, University of Iowa WEEG Computing Center, 1981, pp. 206-219.

BIBLIOGRAPHY

Books

- Badre, Albert, and Ben Shneiderman, editors, Directions in Human-Computer Interaction, Norwood, Albex Publishing Corporation, 1982.
- Bobrow, Daniel G., and Allan Collins, Representation and Understanding: Studies in Cognitive Science, New York, Academic Press, 1975.
- Buros, Oscar Kirsan, editor, The Seventh Mental Measurements Yearbook (2 volumes), Highland Park, N. J., The Gryphon Press, 1972.
- Dahl, O. J., and others, Structured Programming, New York, Academic Press, 1972.
- Dijkstra, E., A Discipline of Programming, Englewood Cliffs, Prentice-Hall, 1976.
- Newell, Allen, and Herbert Simon, Human Problem Solving, Englewood Cliffs, Prentice-Hall, 1972.
- Shneiderman, Ben, Software Psychology: Human Factors in Computer and Information Systems, Cambridge, Winthrop Publishers, Inc., 1980.
- Sleeman, D., and J. S. Brown, editors, Intelligent Tutoring Systems, New York, Academic Press, 1982.

Articles

- Adelson, Beth, "Problem Solving and the Development of Abstract Categories in Programming Languages," Memory and Cognition, 9 (July, 1981), 422-433.
- Alspaugh, C. A., "Identification of Some Components of Computer Programming Aptitude," Journal of Research in Mathematics Education, 3 (March, 1972), 89-98.
- Barker, R. J., and E. A. Unger, "A Predictor for Success in an Introductory Programming Class Based upon Abstract

- Reasoning Development," ACM SIGCSE Bulletin, 15 (February, 1983), 154-158.
- Bonar, Jeffrey, and others, "Collecting and Analyzing On-Line Protocols from Novice Programmers," Behavior Research Methods and Instrumentation, 14 (April, 1982), 203-209.
- Brown, J. S., and Richard R. Burton, "Diagnostic Models for Procedural Bugs in Basic Mathematical Skills," Cognitive Science, 2 (April-June, 1978), 155-192.
- Brown, J. S., and K. VanLehn, "Repair Theory: A Generative Theory of Bugs in Procedural Skills," Cognitive Science, 4 (October-December, 1978), 379-415.
- Cheney, Paul, "Cognitive Style and Student Programming Ability: An Investigation," AEDS Journal, 13 (Summer, 1980), 285-291.
- Collins, A., "Explicating the Tacit Knowledge in Teaching and Learning," Eric Document ED 163 986, 1978.
- Curtis, Bill, "A Review of Human Factors Research on Programming Languages and Specifications," AEDS Monitor, 21 (March-April, 1983), 24-30.
- Durnin, J. H., and J. M. Scandura, "An Algorithmic Approach to Assessing Behavioral Potential: Comparison with Items Forms and Hierarchical Technologies," Journal of Educational Psychology, 65 (October, 1973), 262-272.
- Fowler, George C., and Louis W. Glorfeld, "Predicting Aptitude in Computing: A Classification Model," AEDS Journal, 14 (Winter, 1981), 96-109.
- Konvalina, John, and others, "Identifying Factors Influencing Computer Science Aptitude and Achievement," AEDS Journal, 16 (Winter, 1983), 106-112.
- Kurtz, B. L., "Investigating the Relationship between the Development of Abstract Reasoning and Performance in an Introductory Programming Class," ACM SIGCSE Bulletin, 12 (February, 1980), 14-17.
- Mayer, Richard, "Instructional Variables in Computer Programming," Eric Document ED 112865, 1978.

- Mazlack, Lawrence J., "Identifying Potential to Acquire Programming Skill," Communications of the ACM, 23 (January, 1980), 14-17.
- McKeithen, Katherine, and others, "Knowledge Organization and Skill Differences in Computer Programmers," Cognitive Psychology, 13 (July, 1980), 307-325.
- Peterson, C. G., and T. G. Howe, "Predicting Success in Introduction to Computers," AEDS Journal, 12 (Summer, 1979), 182-191.
- Reitman, J. S., "Skilled Perception in Go: Deducing Memory Structure from Inter-Response Times," Cognitive Psychology, 8 (July, 1976), 336-356.
- Reitman, J. S., and H. Rueter, "Organization Revealed by Recall Orders and Confirmed by Pauses," Cognitive Psychology, 12 (October, 1980), 554-581.
- Schwartz, J. T., "What Programmers Should Know," Computer Languages, 2 (1976), 21-25.
- Soloway, E., and others, "Meno-II: An AI-Based Programming Tutor," Journal of Computer-Based Instruction, 10 (Summer, 1983), 20-34.
- Stephens, Larry J., and others, "Group Differences in Computer Science Aptitude," AEDS Journal, 14 (Winter, 1981), 84-95.
- Waters, Richard C., "A Method for Analyzing Loop Programs," IEEE Transactions on Software Engineering, SE5 (May, 1979), 237-247.
- Wileman, Stanley A., and others, "Factors Influencing Success in Beginning Computer Science Courses," Journal of Educational Research, 74 (March-April, 1981), 223-226.

Publications of Learned Organizations

- Edwards, Richard, editor, Proceedings of the Human Factors Society, Santa Monica, Human Factors Society, 1982.
- Engel, Gerald L., editor, Proceedings of the 1978 Annual Conference of the Association for Computing Machinery, New York, Association for Computing Machinery, 1978.

Harris, Diane, and Laurie Nelson-Hearn, editors, Proceedings of the National Educational Computing Conference, Iowa City, University of Iowa WEEG Computing Center, 1981.

Technical Publications

IBM, IBM Scientific Subroutine Package Version III Programmer's Manual, IBM, White Plains, 1966.

Tests

Scholastic Aptitude Test, Princeton, Educational Testing Service, 1970.

Unpublished Materials

Alspaugh, C. A., "A Study of the Relationship between Student Characteristics and Proficiency in Symbolic and Algebraic Computer Programming," unpublished doctoral dissertation, University of Missouri, Columbia, Missouri, 1970.