

**Restructuring the CS 1 Classroom: Examining the Effect of
Open Laboratory-Based Classes vs. Closed Laboratory-Based
Classes on Computer Science 1 Students' Achievement and
Attitudes Toward Computers and Computer Courses**

by

Jean Foster Henderson

**Bachelor of Arts
Mathematics
University of North Alabama
1974**

**Master of Arts
in Mathematics
University of Alabama in Huntsville
1976**

**Master of Science
in Computer Science
University of Alabama in Huntsville
1994**

**A dissertation submitted to the Graduate School of
Florida Institute of Technology
in partial fulfillment of the requirements
for the degree of**

**Doctor of Philosophy
in
Science Education**

**Melbourne, Florida
November, 2007**

UMI Number: 3285777

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform 3285777

Copyright 2007 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

© Copyright 2007 Jean Foster Henderson

All Rights Reserved

The author grants permission to make single copies Jean Foster Henderson

We the undersigned committee
hereby approve the attached dissertation

Restructuring the CS 1 Classroom: Examining the Effect of Open Laboratory-
Based Classes vs. Closed Laboratory-Based Classes on Computer Science 1
Students' Achievement and Attitudes Toward Computers and Computer Courses.

by

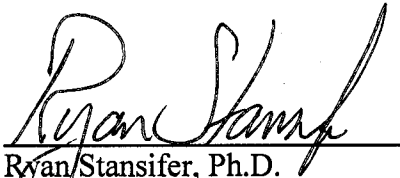
Jean Foster Henderson



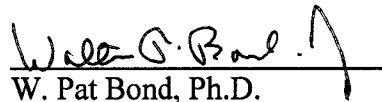
Michael A. Gallo, Ph.D.
Associate Professor
Science/Mathematics Education
Committee Chair



David E. Cook, Ph.D.
Professor/Department Head
Science/Mathematics Education



Ryan Stansifer, Ph.D.
Associate Professor
Computer Science



W. Pat Bond, Ph.D.
Associate Professor
Computer Science

ABSTRACT

TITLE: Restructuring the CS 1 Classroom: Examining the Effect of Open Laboratory-Based Classes vs. Closed Laboratory-Based Classes on Computer Science 1 Students' Achievement and Attitudes Toward Computers and Computer Courses.

AUTHOR: Jean Foster Henderson

MAJOR ADVISOR: Michael A. Gallo, Ph.D.

The purpose of this study was to assess the effect of classroom restructuring involving computer laboratories on student achievement and student attitudes toward computers and computer courses. The effects of the targeted student attributes of gender, previous programming experience, math background, and learning style were also examined. The open lab-based class structure consisted of a traditional lecture class with a separate, unscheduled lab component in which lab assignments were completed outside of class; the closed lab-based class structure integrated a lab component within the lecture class so that half the class was reserved for lecture and half the class was reserved for students to complete lab assignments by working cooperatively with each other and under the supervision and guidance of the instructor. The sample consisted of 71 students enrolled in four intact classes of Computer Science I during the fall and spring semesters of the 2006–2007 school year at two southern universities: two classes were held in the fall (one at each university) and two classes were held in the spring (one at each university). A counterbalanced repeated measures design was used in which all

students experienced both class structures for half of each semester. The order of control and treatment was rotated among the four classes. All students received the same amount of class and instructor time.

A multivariate analysis of variance (MANOVA) via a multiple regression strategy was used to test the study's hypotheses. Although the overall MANOVA model was statistically significant, independent follow-up univariate analyses relative to each dependent measure found that the only significant research factor was math background: Students whose mathematics background was at the level of Calculus I or higher had significantly higher student achievement than students whose mathematics background was less than Calculus I. The results suggest that classroom structures that incorporate an open laboratory setting are just as effective on student achievement and attitudes as classroom structures that incorporate a closed laboratory setting. The results also suggest that math background is a strong predictor of student achievement in CS 1.

TABLE OF CONTENTS

ABSTRACT.....	iii
LIST OF TABLES.....	xii
LIST OF FIGURES.....	xiv
ACKNOWLEDGEMENTS.....	xv
1. INTRODUCTION.....	1
Background and Purpose.....	1
Background.....	1
Purpose.....	9
Definition of Terms.....	10
Research Questions and Hypotheses.....	13
Research Questions.....	13
Research Hypotheses.....	14
Study Design.....	15
Significance of the Study.....	16
Study Limitations and Delimitations.....	18
Limitations.....	18
Delimitations.....	20
2. REVIEW OF RELATED LITERATURE.....	24
Introduction.....	24

Overview of Underlying Theory	24
Constructivism	24
Individual Constructivism.....	25
Social Constructivism	29
Learning Style Theory.....	32
Metacognitive Theory	36
Closing Comments.....	39
Review of Past Research Studies	40
Studies on Student Attributes.....	40
Studies on Learning Styles.....	53
Studies on Restructured Classrooms Involving Computer Labs.....	59
Studies on Self-Regulated Learning.....	82
Summary and Study Implications	85
3. METHODOLOGY.....	89
Population and Sample.....	89
Population	89
Sample.....	91
Power Analysis	93
Instrumentation	95
Attitude towards Computers and Computer Courses.....	95
Kolb's Learning Style Inventory.....	97

Teacher-Based Assessments	98
Laboratory Assignments	99
Programming Assignments	102
Unit Exams.....	103
Course Description.....	104
Procedures	106
Research Methodology/Design	106
Study Implementation	108
Closed Laboratory-Based Class (Treatment)	109
Open Laboratory-Based Class (Control).....	110
Human Subjects Research Issues	111
Threats to Internal Validity	112
Subject Characteristics	112
Mortality.....	113
Location.....	113
Instrumentation	114
Testing.....	116
History.....	116
Maturation	116
Subject Attitude.....	117
Regression.....	117

	Implementation	117
	Treatment Verification and Fidelity	118
	Data Analysis	119
4.	RESULTS	122
	Introduction.....	122
	Descriptive Statistics.....	122
	Overview	122
	Kolb's Learning Style Inventory.....	123
	Achievement	124
	Attitude.....	127
	Inferential Statistics.....	131
	Overview	131
	Preliminary Analysis.....	133
	Missing Data	133
	Outlier Analysis	135
	Regression Assumptions	136
	MANOVA Analyses	141
	Overview.....	141
	Testing the MANOVA Model for Significance.....	142
	Y_1 (Achievement) Follow-Up	145
	Y_2 (Attitude) Follow-Up	149

	Attribute-Treatment Interaction (ATI) Analyses	152
	Supplemental Analysis	152
	Results of Hypothesis Testing.....	155
	Hypothesis 1	155
	Hypothesis 2.....	156
	Hypothesis 3	157
	Hypothesis 4.....	158
	Hypothesis 5.....	159
	Hypothesis 6.....	160
5.	CONCLUSIONS, IMPLICATIONS, AND RECOMMENDATIONS	161
	Summary of Study	161
	Summary of Findings	166
	Conclusions and Inferences	168
	Research Question 1	168
	Research Question 2	173
	Research Question 3	175
	Mathematics Background.....	176
	Student Gender.....	178
	Previous Computer Programming Education	180
	Learning Styles	180
	Research Question 4	181

Research Question 5	183
Research Question 6	184
Implications Relative to Prior Theory and Research	185
Theoretical Implications.....	186
Constructivism	186
Learning Styles.....	187
Implications Relative to Prior Research.....	188
Prior Studies on Class Structure and Laboratories.....	188
Prior Studies on Student Characteristics	195
Implications for Educational Practice	201
Generalizability, Limitations, and Delimitations	203
Generalizability	203
Study Limitations and Delimitations	204
Limitations	205
Delimitations.....	206
Recommendations for Research	210
Recommendations for Future Research Relative to Study Limitations and Delimitations	211
Recommendations for Future Research Relative to Implications	217
Recommendations for Practice Relative to Implications	221
REFERENCES	224
APPENDIX A: INSTRUMENTS	238

Sample of KLSI-3.1	239
Attitude towards Computers and Computer Courses Scale	240
Test 1	243
Test 2	248
Test 3	254
Test 4	261
APPENDIX B: LAB AND PROGRAMMING ASSIGNMENTS	268
Lab Assignments	269
Scoring Rubric for Lab Assignments	352
Programming Assignments	353
Scoring Rubric for Programming Assignments	360
APPENDIX C: CLASSROOM AND COURSE INFORMATION	361
APPENDIX D: INSTITUTIONAL REVIEW BOARD	367
APPENDIX E: RAW DATA	378

LIST OF TABLES

CHAPTER 3

3.1	Summary of Student Characteristics.....	92
3.2	Summary of Teacher Characteristics	93
3.3	Power Analysis and Calculated Powers for $\alpha = .05$	94
3.4	Session Assessments	99
3.5	Session Topics.....	104
3.6	Treatment Implementation Schedule	108
3.7	Characteristics of the Variables	121

CHAPTER 4

4.1	Summary of Students' Learning Styles from Kolb's (2005) Learning Styles Inventory-Version 3.1	124
4.2A	Achievement Results by School Location	126
4.2B	Achievement Results by Semester	126
4.3	<i>t</i> Test Comparisons of UNA vs. ASU and Fall vs. Spring Relative to Achievement and Attitude	126
4.4A	Attitude Results by School Location.....	128
4.4B	Attitude Results by Semester	128
4.5	Summary of Results of Student Attitude towards Computers and Computer Courses	129
4.6	Students' Mean Subscale Scores for Attitude towards Computers and Computer Courses	132
4.7	Missing Data Results.....	135
4.8	Overall MANOVA Results.....	143

4.9	MANOVA Results for Independent Variables	144
4.10	Summary of Follow-Up Univariate Hierarchical Regression Analysis for Achievement.....	146
4.11	Summary of Follow-up Tests Relative to $Y_1 = \text{Achievement}$	147
4.12	Summary of Follow-Up of the Independent Variable “Math Background”	148
4.13	Summary of Follow-Up Univariate Hierarchical Regression Analysis for Attitude	150
4.14	Summary of Follow-up Tests Relative to $Y_2 = \text{Attitude}$	151
4.15	Summary of Results Comparing the Full Dataset (UNA and ASU) to UNA Data Only	154

CHAPTER 5

5.1	Summary of Hypotheses Tests Results	169
-----	---	-----

LIST OF FIGURES

CHAPTER 2

- 2.1 Kolb's (1984) learning style preferences 34

CHAPTER 3

- 3.1 The relationship between the activities of students'
laboratory assignments and the six levels of Bloom's (1956)
taxonomy..... 101

ACKNOWLEDGEMENTS

I appreciate the time, effort, and especially the patience of Dr. Bond, Dr. Cook, and Dr. Stansifer, my dissertation committee. Most particularly, I am grateful to Dr. Gallo, the chair of my dissertation committee, for his support and encouragement throughout my time as a graduate student. The success of this dissertation is directly due to his many hours spent in consultation with me and in editing my less than optimal writings. It was a privilege to work with him. Finally, I am thankful for the assistance and tolerance of my colleagues and the administration at the University of North Alabama during the years I have worked toward this degree.

CHAPTER 1

INTRODUCTION

Background and Purpose

Background

The first undergraduate computer science course, as defined in the 1978 ACM curricula guide (ACM, 1979) and later updated in the 2001 curricula guide (ACM, 2001), is a gateway course for computer science majors. The fundamentals taught in this course provide the foundation for later studies. Traditional introductory undergraduate computer science courses generally include a component in which students learn to program using the syntax of a specified language. The language used has changed as the computer science field has matured and often varies from school to school. In the past languages such as Fortran, Pascal, and BASIC were used. Today schools often use C, C++, or Java.

Enrollment in introductory computer science courses and the number of computer science majors has declined during the last few years (Corritore, Hickman, Grandgenett, & Hitchcock, 1999). Speculation for this decline has focused on “outside” influences, including economic problems, interest in the Information Systems major and its associated courses, and the diminishing job market in which many computer science jobs are being outsourced to other countries. Although these external influences have merit, recent conjecture is that the enrollment ebb is grounded in the increased difficulty of the major’s

requirements, including learning a programming language and the related assignments of the introductory courses. Many teachers of these introductory courses surmise that students are having difficulty becoming fluent in a computer language. Since its inception, various studies have been conducted that investigated possible causal factors of student achievement in these courses. For example, early studies such as Konvalina, Stephens, and Wileman (1983) found that computer science students' math proficiency was important to achievement and Taylor and Mounfield (1991) reported that students who had a previous structured programming course were more successful than those who did not have such a course. Other studies also identified various student characteristics of successful students in the introductory course (Butcher & Muth, 1985; Campbell & McCabe, 1984; Wilson, 2001). Although the models postulated by these studies were only able to explain a small amount of the variance among students who succeeded, several studies identified student characteristics that correlated with success (Evans & Simkin, 1989). Some of the characteristics identified were mathematics ability, previous experience with computers, and gender.

Students' learning styles was another student characteristic of interest in some previous studies. These studies were conducted in an effort to understand the differences between how people perform in different academic disciplines and situations. In an early study of students enrolled in a computer applications course, Davidson, Savenye, and Orr (1992) found students' learning styles, as classified by

the Gregorc Style Delineator, to be an important consideration. For example, students classified as Abstract Sequential learners had higher achievement scores in the class than those classified as Abstract Random learners. In a study of computer science majors, Crosby and Stelovsky (1995) compared the learning styles of beginning students to those of successful senior-level students. Using the Meyers-Briggs Type Indicator (Meyers, 1952, 1975), they found the majority of beginning students had the sensing cognitive style while almost all of the senior-level students displayed the intuitive cognitive style. Avitabile (1998) found that students in an introduction to computer science class whose lecture included multimedia instruction did better than those in a class based on the traditional lecture method but found no difference between the test scores of those students classified as intuitive and those classified as sensing. Avitabile also found no interaction between student learning styles and mode of presentation.

In a further effort to make the course content more palatable and to lower the high attrition rate, many introductory undergraduate computer science programs have begun restructuring their course to incorporate a laboratory component. This new approach is different from what traditionally has been done. For example, in the past most (if not all) introductory computer science courses were structured using a classroom-lecture-with-outside-programming-assignments paradigm. That is, courses were lecture-based and programming assignments were prepared outside

of class at the student's convenience. This approach is commonly referred to in the literature as *open laboratory instruction* (Thweatt, 1994).

In contrast to this, the restructured approach is based on a "closed-and-structured" paradigm that involves teacher guidance and is either a direct component or an extension of a class (Thweatt, 1994). Known as *closed laboratory instruction*, students prepare laboratory programming assignments in a structured setting under the auspices of an instructor. This concept is analogous to labs that are traditionally associated with science courses such as physics or biology.

Usually the closed lab is implemented in two ways: as a separate laboratory session scheduled in addition to class time or a laboratory component integrated within the structure of the class. Classes and their separately scheduled lab components are usually conducted in separate classrooms/lab rooms and may have different class and lab instructors. A closed lab that integrates the lab component within the structure of the class utilizes a classroom-laboratory setting that contains a whiteboard, an instructor's computer and projection system, and individual student computers. In both types of closed laboratories, programming assignments are discussed and prepared individually or in student-pairs with the instructor as a resource person. Regardless of the approach, the laboratory component is an extension of classroom instruction.

Positive anecdotal evidence concerning the effectiveness of closed-lab based courses relative to increasing student achievement and lowering attrition has

been articulated at computer science faculty meetings and conferences. In contrast, though, little empirical evidence exists and of the few empirical studies conducted, findings have been mixed. In one of the first studies that examined the effect of closed versus open labs in an introductory computer science course, Thweatt (1994) required students in the closed lab classes to finish their lab assignments in the school's computer laboratory during a scheduled lab period; students in the open lab classes were required to schedule time to finish their assignments either in the school's laboratory or on their home computers. Thweatt reported that students who were in the closed lab setting outperformed their counterparts in the open lab setting on the comprehensive final exam. In contrast to these findings, Hartel and Hertzberger (1995) in an early meta-analysis of current literature found little conclusive evidence that closed laboratory-based computer science courses were superior to traditional lecture-based (i.e., open laboratory) courses. In a later study, Corritore et al. (1999) also found no difference between two closed-lab approaches: hands-on and interactive demonstration.

Independent of these open vs. closed lab studies, several early studies were conducted that focused solely on the effect of restructuring undergraduate introductory computer science courses to include some type of laboratory. From these studies, various recommendations or suggestions were reported. For example, Chavey (1991) found that the most effective laboratory activities were those in which the material was previously introduced in a lecture-based class. Geitz (1994)

found that a supervised hands-on laboratory increased student retention and improved programming performance. Thomas and Upah (1998) reported that students needed guidance and monitoring in order to shape their learning process in a laboratory setting. These recommendations served as guidance for this study.

Other previous studies also examined the relationship between students' learning styles and achievement in both the laboratory and lecture portions of undergraduate computer science courses. Houston (1993) reported there was no correlation between student achievement and learning styles in the classroom or the lab component of a college-level computer science course, but she did find that students whose learning styles were matched with the instructor's teaching style performed better in the lab portion of the course. Learning styles in this study were categorized using the Gregorc Learning Style Delineator. Crosby and Stelovsky (1995) compared hypermedia laboratory instruction to lecture-based laboratory instruction. They found a significant difference in the achievements of groups classified by their learning style. Based on the Meyers-Briggs Type Indicator (Meyers, 1956, 1962), "intuitive" students did better than "sensing" students. They also found a strong interaction effect between the type of laboratory instruction and students' learning styles.

Despite the paucity of research findings, many computer science programs now include a laboratory component in their beginning courses and there are an increasing number of computer science textbooks with associated laboratory

manuals (Corritore et al., 1999). A survey performed by Dale (2005) revealed that 50% of schools included a separate laboratory with their introductory course, 28% integrated laboratory exercises within their lectures, and 22% reported no lab with the course.

When examined from a learning theory perspective, the use of a laboratory component makes sense because the concept of any type of laboratory experience should promote student achievement. A class structured to include a closed lab, for example, is the manifestation of social constructivist epistemology, which posits that: (1) knowledge is formed from experience, (2) students must take an active role in their learning; (3) learning should be a collaborative process that occurs in realistic settings, and (4) learners need to choose their own path and activities (Ormrod, 2004). Commensurate with this perspective, students in a closed lab are given one or more problems to solve and resource material to investigate in order to create an inquiry type environment. Students are in control of and responsible for their own learning. They base their learning on their own experiences as they work individually or in pairs to solve problems assigned by the lab instructor. They generate their own rules about programming and the syntax of the language through their experiences using available resource material. The laboratory instructor serves as a mentor who provides guidance as the students write sample code for the concepts. In this setting, students are in what Vygotsky (1987) called the zone of proximal development (ZPD), which refers to a range of tasks that

students cannot yet perform independently but can perform with the help and guidance of others. In short, through this experience, the lab instructors provide the necessary scaffolding for students to successfully learn to develop algorithms and code computer programs.

Alternately, a class structured to include an open laboratory is a product of individual constructivism. This perspective differs from social constructivism in that the process of constructing knowledge occurs separately within each learner, not collectively as a group. Students in the open laboratory class are expected to schedule time outside the classroom to complete their laboratory assignments and to obtain help from the lab instructor if necessary. They can work on their personal computer or in a school computer laboratory. To be successful, however, they must be cognizant of their own metacognition and be self-regulated learners. Both involve self-motivation, goal setting, planning, attention control, application of learning strategies, self-monitoring, and self-evaluation. Self-regulated learners have been shown to set high academic goals, to learn more effectively, and to achieve at high levels in the classroom (Ormrod, 2004).

In summary, the current body of literature is sparse and mixed when it comes to understanding the effect classroom restructuring with laboratories has on student achievement. On the one hand, many anecdotal studies report laboratory-based courses increase student achievement and lower attrition. Other, more systematic studies, however, challenge these results. Adding to this confusion is the

presence of contemporary learning theory, which supports the notion that all labs, regardless of structure or presentation mode, are appropriate for increasing student achievement.

Given these mixed results and the apparent disconnect between theory and practice, this study was offered to help uncover additional information of the relationship between laboratory activities and computer science education. By examining these issues from a systematic field-based trial perspective, this study endeavored to provide empirical evidence that laboratory activities benefit all students regardless of their student attributes and regardless of how the classes were structured.

Purpose

The purpose of this study was to assess the effect of classroom restructuring involving computer laboratories on student achievement and on their attitudes toward computers and computer courses. The effects of targeted student attributes (gender, mathematics background, previous computer programming education, and learning styles) were also evaluated. The classroom structures were a closed lab-based class (i.e., control), which involved traditional lecture and a separate unscheduled lab component, and an open lab-based class (i.e., treatment), which integrated the lab component within the lecture class. Students in an introduction to computer science course completed specific laboratory assignments that complemented the coursework in both settings.

Definition of Terms

Key terms used throughout this study were operationally defined as follows:

1. *Achievement* referred to the mastery of specific laboratory and programming assignments from an introductory computer science course evidenced by the student's grade on an instructor-designed rubric for scoring laboratory and programming assignments. Achievement was further measured using students' session test grades.
2. *Attitude toward computers and computer courses* was defined as personal feelings about computers and computer courses. Student attitudes were assessed by Newby and Fisher's (1997) Attitude towards Computers and Computer Courses (ACCC) instrument.
3. *Classroom restructuring* was defined as a traditional class in which instruction time was altered to accommodate a laboratory component. For example, in a traditional 50-minute class that meets 3 days per week, instead of devoting the entire 150 minutes to instruction, a part of this time (e.g., 50% or 75 minutes per week) was devoted to a laboratory component designed to complement the instruction.
4. *Closed laboratory-based class* was defined as a traditional class that was restructured to include a lecture component and a laboratory component. Class periods were equally divided between lecture and laboratory. Computers were available during the laboratory period for

students to work individually or in pairs on laboratory assignments at their own pace with a lab instructor available for assistance. Students were expected to complete the assignments during the laboratory period, but were not penalized for not doing so. Proof of completion of the exercises was due at the next class meeting. A diagram of this type of classroom's physical structure is given in Appendix C.

5. *Introduction to computer science course (CS 1)* was defined as the 1st-year introduction to programming course specified by the 2001 ACM curriculum recommendations for computer science education (ACM, 2001).
6. *Laboratory assignments* consisted of short programming problems, program code to change, and subprograms to insert in longer prewritten programs. For example, a short programming problem might require students to write a program that finds the average, variance, and standard deviation of a fixed group of numbers and output the results in a simple labeled format. As another example, students might be required to write a procedure to get a list of numerical data from a file then insert that procedure in a prewritten program that finds the average of the numbers and outputs the results.

7. *Learning style* was defined as students' preferred behaviors or approaches for acquiring, organizing, and recalling knowledge. This was measured by Kolb's (2005) Learning Style Inventory-Version 3.1.
8. *Mathematics background* was defined as the highest level of college mathematics courses completed with a grade of C or better.
9. *Open laboratory-based class* was defined as a traditional class in which student laboratory assignments are completed outside of class at the convenience of the students with no instructor supervision. Students could make use of a university provided computer laboratory or their own personal home computers. Hard and soft copies of revised or completed programs were to be turned in at the next class period.
10. *Previous computer programming education* as defined as the completion of a computer programming course with a final grade of C or better. This was self-reported by students responding to the question "Have you passed with a C or better a computer programming course in high school or college?"
11. *Programming assignments* consisted of longer programming problems that were to be completed by students at their own convenience outside of class. These problems paralleled the classroom instruction and the laboratory assignments. Writing these programs involved using techniques learned in the laboratories.

Research Questions and Hypotheses

Research Questions

The overall research question that guided this study was, “What are the relationships among the targeted student attributes and the different classroom structures on undergraduate computer science students’ achievement in and attitudes toward computers and computer courses.” The specific research questions investigated were:

1. What is the effect of classroom restructuring (open lab-based vs. closed lab-based) of an undergraduate introductory computer science class on student achievement?
2. What is the effect of classroom restructuring (open lab-based vs. closed lab-based) of an undergraduate introductory computer science class on students’ attitudes toward computers and computer courses?
3. What is the effect of the targeted student attributes (gender, previous computer programming education, math background, and learning styles) on student achievement?
4. What is the effect of the targeted student attributes (gender, previous computer programming education, math background, and learning styles) on students’ attitudes toward computers and computer courses?
5. What is the interaction effect between the targeted student attributes (gender, previous computer programming education, math background,

and learning styles) and treatment (open lab-based vs. closed lab-based) relative to student achievement?

6. What is the interaction effect between the targeted student attributes (gender, previous computer programming education, math background, and learning styles) and treatment (open lab-based vs. closed lab-based) relative to students' attitudes toward computers and computer courses?

Research Hypotheses

The research hypotheses that corresponded to the research questions follow:

Hypothesis 1

Students in a restructured undergraduate introductory computer science class that incorporates a closed laboratory will have higher student achievement than students in a restructured undergraduate introductory computer science class that incorporates an open lab.

Hypothesis 2

Students in a restructured undergraduate introductory computer science class that incorporates a closed laboratory will have more positive attitudes toward computer science than students in a restructured undergraduate introductory computer science class that incorporates an open lab.

Hypothesis 3

The targeted student attributes (gender, previous computer programming education, math background, and learning styles) will have a positive relationship with student achievement.

Hypothesis 4

The targeted student attributes (gender, previous computer programming education, math background, and learning styles) will have a positive relationship with students' attitudes toward computers and computer science courses.

Hypothesis 5

There will be a non-zero interaction effect between the targeted student attributes (gender, computer programming experience, math background, and learning styles) and treatment (open lab-based vs. closed lab-based) relative to student achievement.

Hypothesis 6

There will be a non-zero interaction effect between the targeted student attributes (gender, previous computer programming education, math background, and learning styles) and treatment (open lab-based vs. closed lab-based) relative to students' attitudes toward computers and computer courses.

Study Design

A counterbalanced repeated measures design involving intact classes was used in this study. Repeated measures is an experimental research design in which

a single group of participants in the study experience all treatments. In this design participants serve as their own control (Creswell, 2005). In a counterbalanced repeated measures design, participants are placed in groups and each group experiences the treatments in a different order (Stevens, 2002). In Summer 2006, a pilot study was initiated. For the full study, intact classes at two different schools over 2 semesters, Fall 2006 and Spring 2007, were used for a total of four classes. Each 14-week semester was divided into two 7-week sessions. In this way, the students in each of the four classes experienced both class structure treatments (one per session). The open laboratory-based class served as the control structure with the closed laboratory-based class structure serving as the experimental treatment. Pre-existing student attributes of math background, previous computer programming education, and gender were measured and used as independent variables and to establish group equivalency between classes and schools. Students' transcripts were examined for these student attributes at the beginning of each semester. All groups were administered an attitude survey at the end of each treatment session. Students were administered researcher-constructed unit exams at the middle of and at the end of each treatment session to assess their achievement in the course for each treatment.

Significance of the Study

Many past studies have been conducted to determine characteristics of the successful computer science student. As personal computers have become more

prevalent and high school computer courses a reality, the profile of an entering computer science student has changed. In addition to the changing student, the structure of the typical first computer science course has also changed. As computer laboratories have been integrated into these early courses, questions have been raised as to how these laboratories affect not only students' achievement but also their attitudes toward computers and computer courses. In light of the fact that so many schools now include scheduled laboratories as part of their beginning computer science courses it is startling to see few studies showing the efficacy of laboratories incorporated into the classroom structure of computer science courses. Because computer laboratories constitute a significant on-going expense, it is important that well-designed, empirically-based formal studies be conducted that address the question of whether laboratories are an efficient use of resources and how those laboratories should be structured as part of the class.

This study provided further information about the types of classroom structure that benefit students in beginning undergraduate computer science courses and thus lead to lower levels of attrition in those courses. It also provided information about the correlation between lab setting and student attributes such as previous computer programming education, math background, gender, and learning styles. Because each laboratory setting was associated with different areas of current learning theory, this study tested those theories relative to computer science

courses and added to the current body of educational learning theory knowledge in the area of classroom structures and types of laboratory activities.

Study Limitations and Delimitations

Results of this study were restricted by the following limitations and delimitations, which effectively marked the study boundaries. Thus, any interpretation, explanation, or generalizations of reported results should take into account these limitations and delimitations. As a note to the reader, limitations are events or conditions outside the control of the researcher that limit the generalizations of study results; delimitations are additional restrictions imposed by the researcher that further limit the study.

Limitations

1. **Sample selection and participant assignment.** A sample of convenience was used for this study and consisted of intact classes of students who self-enrolled in the targeted course, Computer Science 1. Neither random selection of participants nor random assignment of individual participants to treatment order was possible. This limitation was mitigated somewhat by the initial random assignment of intact classes to treatment order coupled by the rotation between schools and semesters. Similar studies should take into account the self-enrollment nature of this sample as well as the use of intact classes.
2. **Course Curriculum.** The content of the targeted course, Computer Science 1, was designed to match the recommendations of the ACM curricular guide

(ACM, 2001). This is the accepted standard for CS 1 type courses throughout the United States. Additionally, all programming was done in C++ on IBM compatible microcomputers using Microsoft Windows XP as the operating system. This was the default language and hardware platform at the targeted universities. The reader should note that many schools use different languages and operating systems. Thus, any generalization of these results should be considered relative to the language and operating system used.

3. Student Demographics. The demographics of the sample used in this study, which affected its results, were beyond my control. Students who comprised this study were a mixture of freshmen, sophomores, and community college transfers with junior status. The majority of the participants were computer science, computer information system, and math majors. The targeted student characteristics of gender, previous computer programming education, and learning styles had no significant effect on achievement or attitude, but math background was found to have a significant effect on achievement. Similar studies with different sample demographic conditions might result in different findings.

Delimitations

1. Study location. The sample was comprised of students who were attending one of two public regional liberal arts universities located in northern Alabama. These schools were selected due to their convenience for the researcher. Although a comparison analysis of available schools' and U.S. colleges' undergraduate computer science student demographic data indicated that the sample from these two schools was reasonably representative of all U.S. colleges, it is possible that studies using other locales could have different results.
2. Time of study. The sample consisted of students who self-enrolled for credit in an introductory computer science course during one of two 14-week semesters: Fall 2006 and Spring 2007. Computer science is a rapidly changing field. Thus, previous or later studies, or studies conducted during the summer term, could result in different findings.
3. Duration of the study. This study was implemented in 2 consecutive semesters. This was necessary because a sufficiently large sample size based on power analysis was not feasible in a single term due to historically low per-term enrollment in the introductory computer science course at the targeted universities ($M = 20$ per school). Semester and school variables were included in the statistical analysis to control for potential semester and school effects. If

the study were limited to 1 semester or extended over a greater time, it might have provided different results. Only studies with similar durations should be compared to this study.

4. Study design. A counter-balanced repeated measures design was used for this study. Each semester was divided into two equal sessions and participants from four intact classes (two classes per semester) experienced treatment and control (i.e., class structures), one each session. This design was chosen due to circumstances at the participating institutions. Other study designs could lead to different results.
5. Achievement instruments. This study measured student achievement using researcher-prepared laboratory activities, programming assignments, and multi-chapter unit tests based on traditional introductory computer science textbooks, lab manuals, and textbook-provided chapter tests. These instruments were reviewed for content validity by the researcher. They also had a calculated reliability coefficient between .74 and .83, which exceeded the generally accepted minimum measure for deriving inferences within educational research (Cohen et al., 2003). It is possible that different instruments could yield different results.
6. Instrument scoring. Researcher-designed rubrics were used to score programming and laboratory assignments. Two of the participating instructors scored the instruments. The inter-rater reliability of .93 indicated acceptable

consistency between scorers. Because student achievement results were directly related to these rubrics and by the scorers' interpretations of how to use them, different results might be achieved with different rubrics or scorers.

7. Participating instructors. The researcher, a female assistant professor with 20 years teaching experience (10 years teaching computer science courses), was the class and lab instructor for the classes at UNA. A male adjunct professor with 10 years experience teaching computer science courses was the class and lab instructor for the fall class at ASU. A female assistant professor with 8 years experience teaching computer science was the class and lab instructor for the spring class at ASU. Although instructor differences were examined indirectly via the school/semester variables of Set B, it is still conceivable that they may have contributed in part to this study's results. The reader is cautioned not to ignore this potential instructor effect in comparison studies.
8. Class design. All groups received 150 minutes of classroom time per week. The control group had 150 minutes of lecture plus class discussion, and the experimental group had their instruction time divided between lecture and laboratory. This design of integrating the closed lab into the regularly scheduled class time was chosen due to restrictions at the targeted schools. Readers should take this into account when comparing results from other studies that use a longer laboratory scheduled separately from class time.

9. Attitude instrument. Attitudes were assessed using Newby and Fisher's (1997) Attitude towards Computers and Computer Courses instrument, which was designed to measure college level students' attitudes taking a computer science course with a laboratory. Although the reliability coefficient of this instrument based on sample data was .95, different results might be found using a different attitude scale.
10. Learning styles. Students' learning styles were assessed using Kolb's (2005) Learning Style Inventory–Version 3.1 (KLSI-3.1). As noted earlier in this chapter, there are several different methods to categorize students' learning styles. Thus, it is conceivable that studies using a different learning styles instrument (e.g., the Myers-Briggs Type indicator or the Gregorc Style Delineator) most likely will get different results. .
11. Achievement measure. Achievement was measured using the scores obtained from unit exams, laboratory assignments, and programming assignments. The weighting of these different assessments relative to students' final achievement scores was researcher-determined and based on her teaching experience. Other studies that use different assessment methods or weights to measure achievement might obtain different results.

CHAPTER 2

REVIEW OF RELATED LITERATURE

Introduction

This chapter contains three sections. The first section provides an overview of the theoretical foundation on which this study was based: individual and social constructivism, self-regulated learning, and learning style theory. The second section is a review of past research studies related to this study. The third section is a summary of the major findings of these past studies and their implications to this study.

Overview of Underlying Theory

Constructivism

Constructivism is a perspective or philosophy of cognitive learning theory. Constructivists believe that learners must create new knowledge for themselves and hence places learners in an active role. “Instead of just listening, reading, and working through routine exercises, they discuss, debate, hypothesize, investigate, and take viewpoints” (Perkins, 1999, p. 7). There are two general categories of constructivism: individual and social. The former involves individuals constructing their own knowledge and understanding independent of others; the latter involves individuals constructing knowledge and understanding through dialogue with others. A discussion of each follows.

Individual Constructivism

Individual constructivism suggests that students construct knowledge from personal (i.e., individual) experiences. This perspective stems from Piaget's (1964) developmental theory, which makes several assumptions about the cognitive development of children: (1) Children construct knowledge from their experiences; (2) children are active and motivated learners; (3) knowledge is not a true reflection of reality—it is invented, not discovered; (4) knowledge is personal and idiosyncratic; (5) children learn through complementary processes of assimilation and accommodation; (6) the process of equilibration or reformation promotes progression toward increasingly more complex levels of thought; and (7) effective learning requires open-ended challenging problems for the learner to solve. Thus, constructivism is grounded in the belief that knowledge is invented, is personal, and idiosyncratic.

Constructivists believe that students construct mental representations of objects and events, which are then used to make sense of new situations (Driver & Scanlon (1988). According to Piaget (as cited in Nussbaum & Novick, 1982), students' knowledge and mental representations are organized as schemas, which are groups of similar thoughts or actions. "A schema is generally used to denote a person's existing conceptual framework" (Nussbaum & Novick, p. 184). It is a person's private understanding, alternative framework, or preconception.

Piaget's (1964) developmental theory posits that when we experience new information that is incongruent with our prior knowledge, we construct new knowledge. Thus, learning only takes place when we make an effort to reconcile new knowledge with what we already know. Piaget referred to this learning process as assimilation and accommodation. When new information can comfortably be integrated into existing schema, assimilation occurs. During the process of assimilation, students are said to be in a state of equilibrium. Otherwise, when the new information cannot be comfortably integrated, students are said to be in a state of disequilibrium or cognitive dissonance. They must recognize this inconsistency and change their existing schema or create new schema. This is called accommodation. These two processes, assimilation and accommodation, are complementary and go hand-in-hand during learning (Ormrod, 2004).

Because we construct knowledge based on what we already know, learning occurs when we create new or build on our existing knowledge. Each idea we learn facilitates our ongoing intellectual development. This view is summarized by Phillips (2000, p. 7):

[the] constructivist view is that learners actively construct their own ("internal" some would say) sets of meanings or understandings; knowledge is not a mere copy of the external world, nor is knowledge acquired by passive absorption or by simple transference from one

person (a teacher) to another (a learner or knower). In sum
knowledge is made not acquired.

If a person's view of the world, that is, his or her knowledge, is constructed or invented, then it is personal and individual. There is a difference between a person's private understanding and public knowledge. Thus, students can often state some part of public knowledge but because they have compartmentalized that knowledge from their own private understanding, the knowledge has no meaning to them (Pines & West, 1986). Meaningful learning is defined as "relating new information to knowledge already stored in long-term memory" (Ormrod, 2004, p. 222). Meaningful learning is the result of active engagement by the learner and knowledge is constructed based on previous experience (Yager, 1991). Teachers must be cognizant that the same lesson can result in very different learning in different students.

The individual nature of constructivist epistemology makes myths of absolute truth and objective reality (Segal, 2001). We individually construct what we know, thus reality is dependent on the mind of the individual. There is no reality that is separate from an individual's construction of it. Who we are, our experiences, our interests, our values and our needs, are all part of what we construct as knowledge (Heshusius, 1995).

Constructivists reject the idea that the teacher is the source of most knowledge in the classroom. Thus, a teacher's role is different under

constructivism. The constructivist teacher's task is to guide students in their search for knowledge (Brooks & Brooks, 1993). Consequently, teachers in constructivist classrooms encourage students to be thinkers, involve students in the entire problem-solving task, play a less central role in the classroom by facilitating students' construction of their own unique knowledge, encourage students to engage in dialogues so that they can develop intellectually, and put in place scaffolds to help students develop and learn. Poplin (1988b, p. 395) asserts that "the more control educators have over the content, the less likely students will be to maintain and generalize skills and or strategies." Thus, in the constructivist classroom, conversations start with students and end with the teacher, or start with students and end with other students (Knight, 2003).

In a constructivist classroom, mistakes are indicators that learning is taking place; they are an essential component of the construction of knowledge. Errors are seen as part of the process of risk taking and meaningful learning. "Constructivists ... seek to create environments where 'penalty-free' errors can emerge and be realized" (Poplin, 1988a, p. 409). Constructivists believe that it is often naïve to assume that there is only one "right answer." Thus, errors are to be condoned and teachers should not correct students as they are learning. Heshusius (1995, p. 182) explains, "the wrong answer can be perfectly right where it is the result of a personal and often complex process the child goes through."

When applied to a 1st-year college-level computer science course, the strategies of constructivism provide the foundation for students who must work in the world of computer programming. Although programming languages have inflexible rules of syntax, students (and employees) are still expected to become innovative problem solvers. Relative to the current study, the classes structured to include an open laboratory were a product of individual constructivism. Students in this class structure were required to schedule time to work on laboratory assignments. They could work on their personal computer or in a school computer laboratory. Help from the instructor was available only if they requested it. As they worked on an assignment, mistakes were made that often resulted in error messages or incorrect answers. Students made use of notes from classroom discussions, reference books, and the error messages generated by the compiler to construct their understanding of the day's topic such as FOR loops and the process of using it in solving problems.

Social Constructivism

Although all constructivists believe that learners construct their own knowledge from experiences, social constructivists also have a contextual view of learning; that is, learning is influenced by the physical and social contexts of the situation, and hence students can acquire behavior and beliefs simply by watching and imitating others. Individuals within a group of learners share ideas and information. They communicate and construct meaning and understanding

together. Social constructivists include teachers, parents, and students in their community of learners. All members of the learning community have a part in the construction of knowledge in social constructivism (Ormrod, 2004).

Both Piaget and Vygotsky (as cited in Ormrod, 2004) proposed that social interaction is critical for cognitive development. For example, one assumption by Piaget (1964, p. 178) relative to social constructivism is that “interaction with one’s physical and social environments is essential for cognitive development.” The concept of social constructivism, then, is that people learn (i.e., construct meaning and understanding) when working together.

Vygotsky (1987) is considered to be one of the first social constructivists. He viewed learning as a social process and considered communication among students to be very important. He stressed that working cooperatively on tasks and problems, especially when children work with more advanced individuals (e.g., teachers, parents, or more advanced students), enable children to develop more sophisticated strategies and thought processes.

Vygotsky (1987) proposed that students have two different ability levels: actual and potential. Students’ actual ability level is the extent to which they can perform tasks independently without anyone’s help. Students’ potential ability level is the extent to which they can perform tasks with the assistance of people more advanced and competent than themselves. Vygotsky believed that new knowledge is acquired through students’ potential ability. Thus, “real” cognitive growth occurs

when we attempt new tasks that we can only accomplish under someone else's guidance. Vygotsky referred to this as the zone of proximal development (ZPD), which is a range of tasks that students can perform with the assistance of someone else but cannot yet accomplish on their own. Thus, students advance cognitively only when working on tasks within their zone of proximal development.

Vygotsky's (1987) concept of ZPD is incorporated into the idea of scaffolding, which occurs when students work on activities and tasks in their ZPD with the guidance of other more competent individuals. While working on these tasks, students become more adept and can later perform such tasks without help. Thus, the zone of proximal development moves and shifts as some tasks are mastered and more complex ones take their place (Ormrod, 2004).

Relative to the current study, classes that incorporated a closed laboratory were a product of social constructivism. Students in this class structure were expected to research a topic such as counted (FOR) loops before the classroom lecture/discussion and then provide the direction for that discussion by asking questions and making comments. In this type of class, the students as a group, under the direction of the instructor, were in charge of the discussion. Additionally, during the laboratory portion of the class, students were expected to experiment individually or collaboratively in pairs to discover methods to use FOR loops to solve the assigned problems. Students also were encouraged to investigate and share with their classmates different solutions to the same problem.

Vygotsky's ZPD also was manifested via the closed laboratory-based class. For example in the lecture portion of these class structures, the instructor guided the students in a discussion while completing an example program of counted FOR loops. The instructor then provided guidance as students worked to complete the laboratory assignments such as those from the topic of FOR loops. In contrast to this scenario, students in the open laboratory-based class worked without the presence of the instructor to complete the laboratory assignment on FOR loops.

In summary, regardless of the orientation (individual or social), constructivism presumes that "knowledge is created, or constructed, by individuals or groups and not simply acquired" (Byrnes, 1996). In individual constructivism, knowledge is individually formed; it is based on the central thesis that people learn not by acquiring fixed knowledge that already exists but by making their own individual sense of the world around them. As such, it is the teacher's role to guide students in making that sense and in the construction of their own knowledge. In social constructivism, the presence of a sociocultural milieu is acknowledged in addition to individual cognition. Thus, a social constructivist learning environment is characterized by students working in groups, working with a mentor (akin to Vygotsky's ZPD), or working individually.

Learning Style Theory

Research indicates there are differences in the way students gather and process information and in the way they prefer to learn. These unique differences

refer to students' learning styles, which can affect a wide range of learning behaviors (Davidson, 1990). Different classroom structures might have affected students with different learning styles in different ways. Students with one type of learning style might have preferred a certain classroom structure while students with another type of learning style might have had no classroom structure preference.

Several learning styles models and corresponding instruments have been developed to help identify a person's learning style relative to a particular model. One such model was Kolb's (1976) learning styles inventory (KLSI). Kolb's model was influenced by several earlier theories, including: Dewey (1897), who stressed the need for learning grounded in experience; Lewin (1939) who emphasized the importance of active participation by learners; and Piaget's (1964) theory that intelligence is the result of the interaction of the person and the environment (Kolb, 1984).

In Kolb's (1984) model there are two orthogonal continuums along which learners can be situated (see Figure 2.1). In one continuum the learning style varies from concrete experience (CE) to abstract conceptualization (AC). Students who perceive information toward the concrete experience end of this continuum generally prefer to learn by doing, by experience, or by direct contact with real objects, real situations, and the environment. Students who perceive information toward the abstract conceptualization end of this continuum generally prefer to

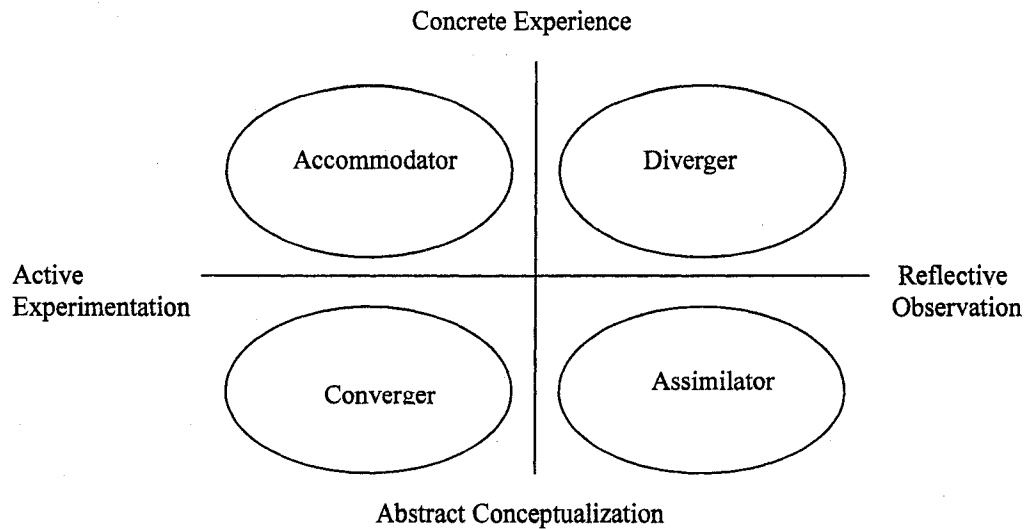


Figure 2.1. Kolb's (1984) learning style preferences.

learn by thinking. This learner grasps philosophical concepts and is able to understand and think about theories. In the second continuum the learning style varies from active experimentation (AE) to reflective observation (RO). Students whose learning style is toward the active experimentation end of this continuum generally like to experiment and do not mind taking risks. Students whose learning style is toward the reflective observation end of this continuum generally like to observe, imitate, and reflect upon how things work.

Kolb's (1984) theoretical model has four learning styles, one for each quadrant created by the two orthogonal continuums (see Figure 2.1). The four learning styles are converger, diverger, assimilator, and accommodator.

Convergers' strengths lie in the practical application of ideas and prefer to solve problems with one correct answer through deductive reasoning. Divergers'

strengths lie in their imaginative ability. They perform better when solving problems through “brainstorming.” Assimilators like to create theoretical models. They excel in the use of inductive reasoning. Accomodators prefer to carry out plans and experiment. They are risk-takers and often solve problems in an intuitive trial-and-error manner. Kolb’s (1976) Learning Style Inventory was developed to classify students’ learning styles relative to one of these four categories. It is a self-reported instrument in which students rank four possible words in each of nine sets. One of the four learning styles is represented by each of the four. This instrument is discussed in more detail in Chapter 3.

Students’ learning styles affect the manner in which they prefer to learn as well as their ability to learn in particular settings. Given the current study’s use of two different classroom structures, it was likely that students with different learning styles might prefer and be more successful in different class settings. For example, when studying the counted (FOR) loop, assimilators might have preferred the structure of the open laboratory-based class because they excel in using induction to find solutions. In contrast, accommodators might have preferred the structure of the closed laboratory-based class where they were encouraged to try different FOR loops in their march toward solving the given problem. One aspect of this study was to examine possible interaction effects between learning style and classroom structure.

Metacognitive Theory

A key aspect of contemporary learning theory is the concept of metacognition, which is “the uniquely human ability to monitor one’s own reflective activity” (Edwards, 1994, p. 14). Ormrod (2004) believes that students’ metacognition is strongly related to their awareness and prior knowledge when engaged in a learning activity. These include (p. 324):

1. Being cognizant of what one’s own learning and memory capabilities are and of what learning tasks can realistically be accomplished.
2. Knowing which learning strategies are effective and which are not.
3. Planning an approach to a learning task that is likely to be successful.
4. Using effective learning strategies.
5. Monitoring one’s present knowledge state.
6. Knowing effective strategies for retrieval of previously stored information.

Metacognition includes students’ knowledge of how they think and learn, and the techniques they use to control and enhance their learning processes. Metacognition guides the student during learning and monitors the success of various learning strategies. Thus, in addition to focusing on teachers’ instructional approach and students’ learning styles as tenets to effective learning, contemporary learning theorists believe that learning is also a function of students’ metacognitive ability.

Inherent in a student's metacognition is self-regulation, which evolved from social learning theory. In particular, self-regulated learners exhibit the following characteristics:

- **Self-motivation:** Individuals have the desire, which comes from within, to complete a learning task. As a part of this self-motivation, self-regulated learners have high self-efficacy in regards to their ability to perform a learning task.
- **Goal setting:** Self-regulated learners set realistic goals for their learning activities.
- **Planning:** Self-regulated learners make definite plans for learning sessions and use their time wisely.
- **Attention control:** Self-regulated learners are focused on the task during learning sessions and do not allow distracting thoughts or actions.
- **Application of learning strategies:** Self-regulated learners know and practice different, appropriate learning strategies.
- **Self-monitoring:** During a learning session, self-regulated learners periodically check their progress and, if necessary, modify their learning strategies and goals.
- **Self-evaluation:** At the end of a learning session, self-regulated learners assess the outcomes of their efforts to determine if they have met their own goals for the session. (Ormrod, 2004, p. 327)

Self-regulated learners are very good at information processing (Borkowski & Thorpe, 1996). Moreover, they not only possess cognition and metacognition, they are also motivated to use effective learning strategies in their learning sessions (Pintrich & De Groot, 1990). Cognitive and metacognitive strategies are the basis for constructing knowledge. However, motivation is key to student engagement. Pintrich, Marx, and Boyle (1993, p. 167) emphasize the problems in viewing learning in a

...cold, or overly rational, model of conceptual change that focuses only on student cognition without considering the ways in which students' motivational beliefs about themselves as learners and the roles of individuals in a classroom learning community can facilitate or hinder conceptual change.

Unmotivated students do not organize their knowledge or use what they know. The best learning strategies will be useless if students do not use them. Thus, when reviewing the attributes of the self-regulated learner, motivation is just as an important consideration as cognition and metacognition (Pintrich & De Groot, 1990).

Self-regulated learners not only possess the knowledge of what they know, but also what they need to know, strategies to learn what they need to know, and are highly motivated to learn. They revise their learning strategies as needed when difficulties arise. Self-regulated learners also must make decisions as to when and

where they will study. They have been shown to set high academic goals, to learn more effectively, and to achieve at high levels in the classroom. Self-regulated learning is consistent with effective learning as portrayed by social learning theorists and cognitivists (Ormrod, 2004).

All successful computer science students should use the techniques of self-regulated learning. This is especially true in situations where students must schedule time outside of class to complete assignments and in situations where they must work unsupervised. As an example, consider the running example of counted (FOR) loops. Students in a class structured with open laboratories would be expected to schedule time to complete the laboratory assignment involving FOR loops in addition to a homework programming assignment that includes the use of a FOR loop. Students had the choice to work on both assignments either in the computer laboratory or on their own personal computer. Students were not supervised or assisted by an instructor while working on assignments. Thus, to be successful in this endeavor, students must possess a high degree of metacognition and be capable of self-regulating their learning.

Closing Comments

In the context of this study, the learning theories presented in this section provided the basis for determining the different classroom structures. For example, classes structured to include both types of lab were manifestations of constructivist epistemology. Students constructed their own rules about programming and the

syntax of the language through their experiences using available resource material. Additionally, classes that included an open laboratory required students to become self-regulated learners. The unstructured nature of the open laboratory required successful students to depend on their own motivation to schedule time and effort for laboratory assignments. Thus, contemporary learning theories support the notion that all labs, regardless of structure or presentation mode, are appropriate for improving student attitude and achievement. In this study, hypotheses deduced from these theories were tested to determine which class structure was more effective relative to student achievement and students' attitudes toward computers and computer courses.

Review of Past Research Studies

Prior studies relevant to the proposed study include those that (1) examined the relationship between student attributes and success in computer science courses and (2) investigated the effect of different computer science classroom structures involving laboratories on student learning and retention. A summary of these studies follows.

Studies on Student Attributes

Butcher and Muth (1985) performed a 2-year study on 269 first-semester freshmen enrolled in two fall CS 1 courses at West Virginia University. The purpose of the study was to predict students' performance in the introductory computer science course and first-semester grade point average. The results of

their study were also used to set admission requirements for CS majors at West Virginia. Initially 13 independent variables were used: four ACT sub-scores and composite score, high school rank, high school class size, high school GPA, highest high school math course taken, completed high school computer course, number of high school physics and chemistry courses completed, number of science, math, and computer classes completed, and high school rank as a percentile of rank to class size. All student attributes were obtained from high school transcripts and student ACT reports. Exam average, lab average, final course grade, and semester grade point average (CGPA) were used as dependent variables. All IVs except high school class size and completed high school computer course had a significant correlation with the DVs. The three IVs with the strongest correlation, ACT math, ACT composite, and HS GPA, were selected for further analysis.

Using a multiple regression strategy, a two-variable regression equation involving the ACT math subscore and HS GPA was significant at $\alpha = .05$ for the four DVs ($R_{Exam}^2 = .395$, $R_{Lab}^2 = .242$, $R_{Grade}^2 = .366$, and $R_{CGPA}^2 = .417$). No other variable contributed a significant amount to the prediction of the dependent variables. Butcher and Muth (1985) concluded that it is possible to predict student success of 1st-semester freshman in an introductory computer course by using information from the ACT report and high school transcripts. They qualified this conclusion by noting that more than 50% of the variance in course grade remains unexplained and thus any prediction is simply an educated guess.

Although Butcher and Muth's (1985) results were informative and helped confirm the relationship between ACT math subscores and high school GPA in predicting student success in an introductory computer course, there were several flaws of their study that limited generalizability. First, participants were limited to 1st-semester freshmen, which implies that the results cannot be generalized to the general population of CS 1 students. Second, there was no report on the validity and reliability of their instruments. Third, although their reported sample size was large ($N = 269$), the initial sample comprised 372 students: 66 withdrew and 37 were deleted because of inadequate high school transcripts. Nearly one-third of the initial sample was dropped. Thus, it is possible that the resulting sample was biased. Fourth, data from the two fall semesters were combined into a single sample. Although it is reasonable to assume the two samples could be combined, statistical procedures that described this were not reported and hence, the use of two separate samples could bias the results. Finally, one-way analyses of variance for all independent and dependent variables and three simultaneous multiple regression analyses were conducted. These separate analyses lead to inflated alpha levels, which increase the chances that the results occurred randomly. Given the presence of four DVs, a better strategy would have been MANOVA/MANCOVA. In this study, care was taken to not repeat these errors.

Taylor and Mounfield (1991) also examined the relationship between student attributes and success in computer science. The sample was comprised of

240 Louisiana State University students enrolled in an introduction to computer science course during one semester of 1989. Independent variables included gender, home computer ownership, type of prior computer course (e.g., application, general programming, structured programming, etc.), typing ability, and work status. These data were self-reported by students using a survey instrument. The dependent variable was students' end-of semester grades, and successful completion of the course was defined as a final course grade of C or better.

Using a cross tabulation with the chi-square statistic and Goodman and Kruskal's lambda follow-up tests, only type of prior computer course and work status were significant. More specifically, percent successful scores were reported for students with a previous programming course (52%), students with a previous structured programming course (74%), and students who worked 40 or more hours per week (92%). Given these results, Taylor and Mounfield (1991) concluded that students who had a high school structured programming course were the most successful in an introductory computer science course and those who had any type of high school programming course also had an advantage.

The poor manner in which this article was written made it difficult to assess the threats to internal validity and generalizability of the study. For example, no information was given on the type of instruments the instructors used to assess student grades, and there is an inherent error potential when students self-report information on a survey. These issues notwithstanding, the targeted attributes were

relevant to the current study and hence were examined for their influence on student achievement.

A longitudinal study by Hagen and Markham (2000) was conducted at Monash University's School of Computer Science and Software Engineering. As part of that study, researchers correlated computer programming experience with student success in the first of two 1st-year computer programming courses. In this initial phase of the study, 121 of 360 students agreed to participate in the study, but only 75 students completed all the instruments of the study. Three survey instruments were used to determine student biographical data, student previous programming experience and formal study, student experiences with the programming environment, and student evaluation of the semester. Student performance was determined by a semester-long programming assignment (stage 1 and stage 2), two tests, a final exam, and the final mark in the class.

Six separate t tests were performed and the results indicated that students with prior programming experience performed better on programming assignments (stages), tests, final exams, and final marks than those without programming experience ($t_{\text{stage1}} = -2.29$, $t_{\text{test1}} = -3.00$, $t_{\text{test2}} = -2.83$, $t_{\text{stage2}} = -3.47$, $t_{\text{exam}} = -1.83$, $t_{\text{final mark}} = -2.90$). All differences were significant except for the grades on the final exam. An analysis of variance was conducted between students who had previous experience with between 0 and 3 languages ($n_0 = 12$, $n_1 = 26$, $n_2 = 21$, $n_3 = 16$). The

results showed a significant difference in all assessments based on the number of languages ($F_{\text{stage1}} = 3.14$, $F_{\text{test1}} = 5.67$, $F_{\text{test2}} = 4.95$, $F_{\text{stage2}} = 5.36$, $F_{\text{exam}} = 3.73$, $F_{\text{final mark}} = 5.46$). Finally, a second ANOVA was conducted between students who had previous formal studies with between 0 and 3 languages ($n_0 = 23$, $n_1 = 20$, $n_2 = 17$, $n_3 = 15$). These results also showed a significant difference in all assessments, except the final exam, based on the number of languages ($F_{\text{stage1}} = 3.56$, $F_{\text{test1}} = 3.96$, $F_{\text{test2}} = 4.31$, $F_{\text{stage2}} = 4.44$, $F_{\text{exam}} = 1.23$, $F_{\text{final mark}} = 3.37$). The authors concluded that students in an introductory programming course benefited from having prior programming language experience. They also indicated that the extent of this benefit was related to the number of languages the students had studied.

Hagen and Markham's (2000) study suffered from many of the same threats of the first two studies. As with Taylor and Mounfield's (1991) earlier study, the use of a survey in which students self-reported the values of the independent variables can lead to errors. Additionally, the use of a series of independent t tests involving two dependent variables is an inappropriate statistical strategy. This strategy leads to inflated alpha levels, which increase the probability that any result occurred by chance

Another study involving student attributes was Wilson's (2002), which involved 105 students (19 of whom were female) in CS 1 at a mid-western university during the first half of the 2000 spring semester. The study was conducted to determine factors that promote success in a CS 1 type course and

what differences appear in these factors between genders. Twelve predictor variables were used: gender, prior programming experience, prior non-programming computer experience, encouragement to pursue computer science, self-efficacy in the class, comfort level in the class, work style preference, math background, game playing, and attribution of success/failure to luck, ability, effort, and difficulty of task. Two instruments, a researcher-developed questionnaire and the Computer Programming Self-Efficacy Scale (Ramalingam & Wiedenbeck, 1998) were used to collect data. The questionnaire was validated for face validity by Psychology professors and content validity by Computer Science professors. It had a test-retest reliability that ranged between .72 and 1. The self-efficacy scale had an overall alpha reliability of .98.

Wilson's (2002) study was guided by the result of a one-way ANOVA involving previous term students ($n = 48$) and previous research studies. Using midterm grades as her dependent measure, Wilson found that the overall proportion of variance explained by the 12 IVs was statistically significant ($F_{11, 92} = 6.13, p < .0001$) but only comfort level ($p = .0002$), math background ($p = .005$), and attribution of success/failure to luck ($p = .0233$) were individually significant in the full model. Neither gender nor previous programming experiences was significant. However, when the different types of experiences were compared as predictors of student achievement, previous formal computing course and game playing were both significant.

As was the case with the previously reviewed studies presented in this section, the results of Wilson's (2002) study were informative to this study but also were of some concern. For example, although the overall sample size of Wilson's study was reasonably large ($N = 105$), only 19 students were female, which could explain why she did not find a significant gender effect. Another concern of Wilson's study is the lack of attention to validity and reliability of the midterm examination, which was used to measure the dependent variable. Finally, the duration of the study, which was conducted during the first half of a semester, was not long enough to make any widespread generalizations.

Two recent studies by Pillay and Jugoo (2005) investigated the effects of certain student characteristics on programming performance. The characteristics studied were student problem solving ability, gender, learning style, first language, and previous computer experience. The sample consisted of students enrolled in a first-course in Java programming at two African schools: University of KwaZulu-Natal and Mangosuthu Technikon. An intact class of 67 students was used in the first study and an intact class of 30 students was used in the second study. Problem solving ability was measured by students' performance in a previous mathematics class in the first study, and by their performance in a software development class in the second study. Learning style was measured using Kolb's Learning Style Inventory (1996). Previous computer experience referred to computer literacy, and programming performance and exam scores were used as dependent variables. All

students in the first study had taken an introductory computer literacy course, and all students in the second study spoke English as a second language.

Hypothesis testing for both studies was done using multiple t tests at $\alpha = .05$. Results of the first study found: (1) no significant gender difference ($n_M = 45$, $n_F = 22$) on either programming scores ($p = .25$) or exam scores ($p = .41$); and (2) a significant first-language effect on programming scores ($p = .004$) in favor of those whose mother tongue was English ($n = 38$) vs. those who spoke English as a second language ($n = 31$); there was no effect on exam scores, however ($p = .11$). Results of the second study found: (1) no significant gender difference ($n_M = 17$, $n_F = 13$) on either programming scores ($p = .38$) or exam scores ($p = .39$); and (2) no significant previous computer experience effect on either dependent measure between students who had previous computer experience ($n = 21$, $p = .34$) and those who did not ($n = 9$, $p = .33$). When examined from a correlation perspective, there was a significant positive correlation between problem solving and both dependent measures (programming and exam scores) in both the first study ($r = .41$, $p = .002$; and $r = .29$, $p = .03$) and the second study ($r = .62$, $p = .001$; and $r = .70$, $p = .0001$). Finally, Assimilators performed better than Divergers in the first study, but there was no significant correlation between learning style and performance in the second study. Based on these results, Pillay and Jugoo (2005) concluded that performance in a Java programming course is independent of gender and previous programming experience. They also concluded that problem solving

ability affects performance and students who speak English as a second language are at a disadvantage. They did not draw any conclusion about the effects of learning style on performance.

Although Pillay and Jugoo's (2005) study involved a different set of students and culture than this study (African students vs. northern Alabama students), the results of their study were still informative. For example, my research setting was similar, my anticipated sample size was approximately the same, and I examined a set of similar student attributes. The primary difference between this study and Pillay and Jugoo's was the statistical strategy. Pillay and Jugoo used a series of independent *t* tests involving two dependent variables. As was noted earlier, this strategy leads to inflated alpha levels, which increases the probability that any significance occurred by chance. In my study, I used a multivariate approach, which protected against such probability pyramiding.

A final study involving student success and student characteristics was conducted by Bergin and Reilly (2005) at the National University of Ireland Maynooth during the 2003–2004 academic year on 96 students enrolled in a 2-semester introduction to programming module. The module consisted of a 1.5-hour problem based learning workshop, a 1.5-hour laboratory, and 3 hours of lecture per week. The purpose of the study was to determine the influence on programming performance of 15 student attributes: prior academic experience, prior computer experience, self-perception of their understanding of the module, comfort level on

the module, and targeted cognitive skills. Performance on the module was based on continuous assessment (30%) and a final exam (70%).

Two instruments were used to collect the independent data: a researcher-developed survey and a locally-developed cognitive test. Thirty students completed the survey and 80 completed the exam during the 2nd semester of the module. An a priori analysis indicated there was no significant difference between the mean overall module scores of the class and the sample, $t(124) = 0.80, p = 0.43$.

Pearson correlations were calculated between students' class performance and the study's independent variables to better understand which variables to include in a regression analysis. First, correlations were calculated for students' rank on Irish Leaving Certificates in Mathematics ($n = 30, r = .46$), Biology ($n = 10, r = .75$), Chemistry ($n = 11, r = .4$), Physics ($n = 18, r = .59$), and the highest science score ($n = 28, r = .48$). The correlations for Mathematics and highest science score were significant at the .01 level; the correlations for Biology and Physics were significant at the .05 level. A significant correlation, $r = .31, p = .01$ was found between performance on the cognitive test and performance on the module. Comfort level on the module was also found to be significant, $r = .516, p < .01$. A strong significant relationship between students' self-perception of their understanding in the course and their module performance was found, $r = .76, p < .01$.

A series of *t* tests for independent samples on the difference in means were also performed for the dichotomous variables. Students were separated into groups: those with no non-programming experience with computers ($n = 4$), those with some non-programming experience with computers ($n = 26$), those with no programming experience ($n = 25$), and those with programming experience ($n = 5$). The results of *t* tests based on differences between group means ($M_{\text{no non-prog}} = 49\%$, $M_{\text{some non-prog}} = 45\%$, $M_{\text{no prog}} = 46\%$, $M_{\text{some prog}} = 44\%$) showed no significant difference in the overall module. This was also the case for the individual research factors: Gender ($n_M = 60$, $M_M = 49\%$; $n_F = 36$, $M_F = 51\%$); age (under 23: $n = 92$, $M = 50\%$; 23 or older: $n = 4$, $M = 56\%$); work style preference (individual: $n = 12$, $M = 50\%$; group: $n = 18$, $M = 43\%$); part-time job ($n_{\text{no}} = 18$, $M_{\text{no}} = 47\%$; $n_{\text{yes}} = 12$, $M_{\text{yes}} = 45\%$); and encouragement by others ($n_{\text{no}} = 21$, $M_{\text{no}} = 44\%$; $n_{\text{yes}} = 9$, $M_{\text{yes}} = 50\%$).

Several regression analyses were performed to determine the earliest indicators of programming performance. A step-wise regression analysis was performed using gender, previous academic experience, cognitive test score, previous programming and non-programming experience, encouragement from others, work style preference, and hours working at part-time job. The model was significant ($F_{2, 27} = 7.11$, $p < .01$, adjusted $R^2 = 30\%$) with significant coefficients for LC Math ($\beta = 0.39$, $p < .0001$) and gender ($\beta = -0.38$, $p = .01$). Another step-wise regression model was performed using all the student attributes of the

previous model and scores from the first test, students' comfort level, and students' self-perception of their level of performance in the course. This model was also significant ($F_{4, 23} = 26.03, p < .001, \text{adjusted } R^2 = 79\%$). Significant coefficient values were found for understanding ($\beta = 0.51, p < .0001$), gender ($\beta = -0.50, p < .0001$), comfort level ($\beta = 0.30, p = .022$), and LC Math ($\beta = 0.20, p = .047$). The authors concluded that the strongest relationship existed between students' perception of their understanding of the module (as measured half-way through the module) and their final performance.

As with Pillay and Jugoo's (2005) study, participants in Bergin and Reilly's (2005) study were from a different culture than those of the current study. Nevertheless, their results were informative because the study involved many of the same student characteristics as the current study. It is important to note, though, that some of the methods Bergin and Reilly used threatened the validity and reliability of their conclusions. For example, a survey was used in which students self-reported many of the values used for the independent variables. As mentioned earlier, this can lead to errors. Additionally, no validity or reliability measures were given for the cognitive test, the final exam, or other instruments used in calculating the final grade. Without such measures, it is difficult to determine the validity or reliability of results and the conclusions based on those results. As with many of the other studies on student characteristics, Bergin and Reilly chose to

calculate several Pearson correlations and perform multiple independent t tests, which may have led to inflated alpha levels.

Studies on Learning Style

Davidson and Savenye (1992) conducted a study to determine if there was a relationship between learning styles and computer performance measures and course grade in a computer concepts and application course. Sixty-eight students at a Texas university participated. The independent variable was students' learning style as measured by the Gregorc (1984) Style Delineator, which has an internal consistency score of $\alpha \geq .85$ for the four scales, Abstract Sequential, Abstract Random, Concrete Sequential, and Concrete Random. Dependent measures were scores on two course projects (Project 1 and Project 2), midterm and final exam scores, total points, and the final course grade. Two intact classes with different instructors were used.

Pearson correlation coefficients were calculated among the four learning style scores and the study's dependent measures. Significant positive correlations were reported between Abstract Sequential (AS) scores and midterm ($r = .29, p < .02$), Project 2 ($r = .33, p < .02$), final exam ($r = .33, p < .005$), total points ($r = .38, p < .001$), and final course grade ($r = .28, p < .02$); and significant negative correlations were reported between Abstract Random (AR) scores and midterm ($r = -.27, p < .01$), final exam ($r = -.30, p < .01$), total points ($r = -.32, p < .007$), and

final course grade ($r = -.28, p < .02$). No other significant correlations were reported.

Given these significant correlations, subjects were then identified as high or low AS and high or low AR relative to the respective AS and AR means. An ANOVA indicated that there was a significant difference between the total point scores of low vs. high AS ($F_{1, 67} = 5.22, p < .05$) and of low vs. high AR ($F_{1, 67} = 4.72, p < .05$). Davidson and Savenye (1992) concluded that learning styles had a significant effect on performance. The differences in effect were between sequential (those who like structured problems) and random learners (those who prefer loosely structured tasks). This effect was seen in overall total points earned, Project 2, and the final exam, which involved programming in BASIC.

Although a significant learning styles effect was reported, the results of Davidson and Savenye's (1992) study are suspect. From a statistical perspective, and as noted earlier, studies that involve multiple dependent measures run the risk of inflated alpha levels if not approached from a multivariate perspective. Such is the case here. Second, no information was provided about the data collection instruments, including attention to instrumentation validity and reliability. The most troubling aspect, though, is that no consideration was given to student differences (aside from learning styles) or instructor differences. This concern was addressed in the current study by using a repeated measures approach, which

eliminated the subject characteristics threat to internal validity because students served as their own control.

Crosby and Stelovsky (1995) conducted a 15-week study in a CS 2 type course at the University of Hawaii to (1) examine the effectiveness of multimedia presentations and (2) determine if there is an interaction effect between presentation mode and learning styles. Fifty-six students were randomly assigned to one of two presentation groups (multimedia and traditional). The same instructor taught both groups. The traditional group met in a classroom where the instructor used a white board and view graphs. The multimedia group met in computer lab where each student worked at an individual computer running the demonstration program. Students rotated between both groups so that they all experienced both instructional approaches. Four different 50-minute laboratory sessions were used. A pretest was given before each laboratory and a posttest was administered 2 days later. Students' learning styles were categorized into sensing ($n = 35$) and intuitive ($n = 21$) based on the Meyers-Briggs Type indicator (Meyers, 1962, 1975). Fifteen subjects (8 sensing and 7 intuitive), did not complete all the posttests. Independent measures were presentation mode (multimedia or traditional), learning style (sensing or intuitive), and test question type (text or graphical). The dependent measure was posttest score.

Results from a three-factor ANOVA showed a significant difference in posttest scores by instructional method ($F_{2, 38} = 23.485$, $p < .0001$) and a significant

interaction effect between subjects' cognitive style and instructional method ($F_{2, 38} = 15.448, p \leq .0002$). Although multimedia instruction had a positive effect on both learning styles, sensing students made more dramatic gains than intuitive students. Crosby and Stelovsky (1995) concluded that the use of multimedia courseware is beneficial for all students and that its use not only is helpful to students in general, but can also make computer science a viable major for different kinds of students.

Avitable (1998) tried to replicate the results of Crosby and Stelovsky's (1995). Avitable's study involved 32 Introduction to Computer Science students at the College of Saint Rose, a small liberal arts college in Albany, New York. Study design, independent and dependent variables, and study methods were similar to Crosby and Stelovsky's study. In an effort to validate the instrument used for testing student learning (which was not reported by Crosby and Stelovsky), two independent experts in computer science education reviewed the test and judged it a valid test of knowledge in the subject area. Only one laboratory session was used in Avitable's study and students were randomly assigned to traditional (control) or multimedia (treatment) labs. Results from a two-way ANOVA were in contrast to Crosby and Stelovsky's results: There was no significant interaction effect between learning style and presentation method relative to posttest scores ($F_{1, 28} = 0.48, p > .05$), and there was no significant difference in posttest scores by learning style ($F_{1, 28} = 1.58, p > .05$). There was, however, a significant difference in posttest scores by presentation mode ($F_{1, 28} = 10.83, p < .05$), which was similar to

Crosby and Stelovsky's findings. Hence, Avitable agreed with Crosby and Stelovsky that multimedia instruction benefited all kinds of students and was superior to traditional instruction.

Avitable (1998) reported that because of the small sample size, power was only .54 and recommend repeating the study with a larger sample size. Avitable also noted that the small sample size could have biased the results of his study. Neither Crosby and Stelovsky (1995) nor Avitable, however, acknowledged the potential testing effect the pretest could have had on posttest scores. At the very minimum, the pretest should have been held constant statistically by being treated as a covariate.

Byrne and Lyons (2001) conducted a study at the University of Ireland's Department of Informational Technology on honors humanities majors enrolled in a 1st-year BASIC programming and logic course. The purpose of the study was to examine the relationship between student results in a 1st-year programming course and student attributes of gender, prior computing experience, learning style, and previous academic performance. The course was comprised of 2 hours per week of lecture and 2 hours per week of laboratory. Student performance was determined by 20 weekly assignments (30%) and a written final exam (70%), and the 110 students (43 male, 67 female) who completed the course were participants in the study. A *t* test was performed to determine the significance of student gender.

There was no significant difference between male and female students on the dependent variable, final grade ($t = .19$, $F = .90$).

One hundred of the participants self-reported limited previous experience with computers, but only 10 self-reported prior programming experiences. The mean final grade of the group of students with prior programming experience was higher than that of students with only limited previous experience with computers ($M_{\text{limited}} = 41.7$, $M_{\text{programmed}} = 47.6$).

Ninety-one of the participants were categorized by Kolb's (1999) LSI into four learning style groups: convergers ($n = 34$), assimilators ($n = 31$), divergers ($n = 15$), and accommodators ($n = 11$). Convergers performed best overall ($M = 45.0$), but this difference was not significant (no measures reported).

Pearson correlation factors were computed between students' performance and students' scores on Irish Leaving Certificates of Mathematics ($n = 110$, $r = .353$), Science ($n = 35$, $r = .572$), English ($n = 110$, $r = .088$), and Foreign Language ($n = 99$, $r = .119$). The correlations for English and Foreign Language were not significant; Mathematics and Science were significant at the .01 level.

Byrne and Lyons (2001) concluded that gender and learning style did not appear to affect programming performance, although the largest group, convergers, scored higher than the other groups. Additionally, students with prior programming experience outperformed those with limited experience with

computers. Finally, there was a clear link between programming ability and existing aptitude in math and science.

As with Bergin and Reilly's (2005) study, the participants of Byrne and Lyons's (2001) study were from a different culture than those of the current study. Additionally, Byrne and Lyons used students who were majoring in humanities. In spite of this, this study is included because it targeted some of the same attributes as the current study and it involved students in a programming class. The major concern of this study is again the methods used in the statistical analysis of the raw data. Means were compared and conclusions reached without testing the significance of the differences. Multiple correlations were calculated leading, again, to inflated alpha levels.

Studies on Restructured Classrooms Involving Computer Labs

A large body of research exists concerning restructured classrooms involving computer labs and how such restructuring affects students' achievement and attitudes. In the use of computers and in computer science courses, this research has often focused on including scheduled structured laboratories as part of course requirements. The research studies reviewed in this section investigated several types of restructured classrooms involving various types of laboratories.

A 3-week study by Beckenstein and Staunton (1998) involved 30 first graders in two elementary schools in Virginia. Their hypothesis was that students in a classroom structured with free access to a small number of computers would

spend more time on the computer and enjoy that time more than students in a classroom structured to include a weekly laboratory in a separate classroom with a large number of computers. Two intact classes at each school were used. In one school all students attended a weekly 45-minute computer laboratory and were required to complete scheduled activities. In the other school, two computers were available for student use and three students per day were required to complete scheduled computer activities.

The only independent variable was group membership (i.e., lab or in-class computer use). Two dependent measures were assessed: computer use time and computer use enjoyment. Beckenstein and Staunton (1998) used attendance records in the structured laboratories to gather their data on time spent using computers by the laboratory group. They performed several systematic observations of students performing the laboratory activities to determine average time spent on-task by the laboratory students. Students in the other group were trained to self-record their time-in and time-out on log sheets kept near the computers. A researcher-developed "Enjoyment of Computer Usage" survey was used to measure computer-use enjoyment. The instrument was pilot tested to ensure it was age-appropriate.

Raw data were given for each question on the survey instrument but no statistical tests of significance were reported. Based on descriptive statistics results, Beckenstein and Staunton (1998) concluded that the two in-class computer groups

used their computers more ($M_1 = 128$ min, $M_2 = 92$ min) than the two regularly scheduled laboratory groups ($M_3 = 26.75$ min, $M_4 = 20.5$ min). They also concluded that both groups enjoyed their time on the computer but that the in-class group believed more strongly computers were helping them learn and the laboratory group wanted more time to use computers. Beckenstein and Staunton reported several limitations that affected the validity of their results and the generalizability of their conclusions. They included:

- use of the log by the young children to self-report time on computer could have resulted in incorrect time data;
- individual absences at one school led to loss of data;
- inclement weather during the study caused school closure during the study and led to loss of data as well as disruption of normal class routine;
- different levels of home computer use between the two groups probably due to different socio-economic levels for the students in the two schools; and
- different teacher and school philosophies as to computer use.

In addition to these limitations, there were several other flaws that limited its validity and generalizability. Although Beckenstein and Staunton tested the survey instrument to ensure the questions were age-appropriate in a pilot study, no tests were done (or at least none was reported) to ensure the validity and reliability of the instrument. The short duration of the study also could have led to the presence of the Hawthorne effect and biased the results. Perhaps the most serious threat in their

study, though, was the lack of appropriate inferential statistical techniques to justify the conclusions.

Smith (1996) reported anecdotally on the effect of shifting from a lecture-centered classroom structure to a lab-centered classroom structure using a Process Education approach in four computer science courses: System Analysis and Design, C and Assembly Language, Database, and Simulation. "Process education incorporates cooperative learning, discovery-based learning, journal writing, and extensive assessment" (Smith, 1996, p. 2). Students were organized into groups of four. Each student had a specific role in the class. The teacher's role was as a facilitator in the learning process rather than its leader. Each class included a short review of the previous class, a team quiz on the reading assignment, presentation of the answers on the quiz, a discovery learning activity, and a short time period for end of class processing.

Smith (1996) concluded that students were learning more and gaining greater confidence in their ability to learn as a result of this approach. He noted that he at first experienced difficulty teaching this type of class and that teachers require training to learn to use the Process Education approach. This article described one professor's experiences using the Process Education approach. Because no statistical information were given (neither descriptive nor inferential), the author's conclusions cannot be accepted as statistically valid and hence has limited generalization.

In their study on collaborative learning in computer science courses, Sabin and Sabin (1994) investigated the effect of this instructional approach on student attitudes and achievement. Thirty-one students participated in the semester-long study at Loyola College in Baltimore, Maryland. Intact classes were used with 18 in the traditional classroom and 13 in the treatment group. The traditional (control) group was taught using the lecture method; the treatment group was taught using collaborative learning strategies. Students were divided into groups and during class time worked together to solve assigned problems. Programming projects were assigned as out-of-class work for each major topic covered.

The independent variable was group membership, and the dependent measures were attitudes toward computers and achievement. Attitudes were assessed using a computer attitude survey, which was given prior to and after treatment. Achievement was defined as scores on a computer programming posttest and final class grade, and a computer programming test was administered as a pre-assessment. Student demographic data (gender, major, computer ownership, and parent computer use) were also collected using a survey. Pre- and posttest scores showed there was improvement in programming knowledge in both groups. A *t* test indicated that only the treatment group's achievement level improved significantly at $\alpha = .01$. No significant differences in attitudes existed between groups. Although Sabin and Sabin (1994) reported that their results were biased by the small sample size, they nevertheless concluded that

collaborative learning had a positive effect on the achievement. Anecdotal observations by the instructor of the collaborative learning course were used to justify conclusions that collaborative learning resulted in a friendly, comfortable classroom. In addition to the sample size issue, other concerns include instrumentation, mortality, subject characteristics, and testing threats to internal validity. As a result, readers should be cautious when generalizing Sabin and Sabin's results to other samples and populations.

Priebe (1997) conducted a 9-week quasi-experimental study employing a pretest, posttest, control group design. The purpose of their study was to compare the comprehension, logical reasoning ability, and attendance of a traditionally structured lecture class to a class structured using the tenets of cooperative education. Participants in the study were students enrolled in a CS 2 type course at a large southwestern university during the summer term of 1996. Initially 67 students were part of the study but only 49 completed all the instruments. Two intact classes were used for treatment and control group. All students had successfully completed a CS 1 type course. The class structure for the control group was traditional and consisted of three 90-minute lecture sessions and one 90-minute discussion session per week. The class structure of the treatment group consisted of three 90-minute sessions using a cooperative learning environment and one 90-minute discussion session. Both groups were given outside programming assignments. Students were given a pretest and posttest to measure content

comprehension levels and logical reasoning ability. Two previously validated instruments, the Burton Comprehension Instrument (Burton, 1992) and the Propositional Logic Test (Piburn, 1989), were used. Daily attendance was measured by taking a headcount in each class.

Priebe (1997) found no significant difference between the two groups in concept comprehension when pretest scores were used as a covariant (ANCOVA, $F_{1,46} = 0.053, p = .471$). A multivariate analysis of variance for repeated measures was performed on the scores on the pretest and posttest for logical reasoning. No significant difference was found for the two groups (MANOVA, $F_{1,47} = 2.00, p = .164$). A significant difference at $\alpha = .05$ was found between the attendance in the two classes (ANOVA, $F_{1,44} = 5.05, p = .03$). Priebe concluded that classes structured using the tenets of cooperative learning learned just as well as classes structured in a traditional manner and that the students in the cooperative learning classes attended more regularly than those in the traditional class. He also noted that homework from the cooperative learning class was neater and more complete and that students engaged in conversations more often. Thus, the two classes had a different character.

Concerns about Priebe's (1997) study included time of implementation, lack of random assignment, and mortality. Participants in his study were from a summer session, which is not indicative of general class populations. No random assignment was used either in individual student assignment to treatment or control

or in intact class assignment to treatment or control. This may have biased the results. The study also had a greater than 25% mortality rate. Statistical measures should have been used to account for this large loss of data. Without such measures, the results of the study were most likely biased.

An empirical study by Duplass (1995) investigated the effectiveness of a supervised laboratory in a computer applications course. The study was conducted during the Spring 1994 term using two intact classes ($n_{\text{Control}} = 26$, $n_{\text{Treatment}} = 27$) of teacher education students enrolled in a computer applications course at the University of South Florida. Students from both groups attended separate but similar lectures taught by the same teacher. Treatment group students also attended four 2-hour supervised laboratories. All students were required to complete nine out-of-class assigned projects. During laboratory time the instructor was available to assist and answer student questions as they worked on the assigned projects.

To show group equivalency, a pretest was given on the BASIC language and concepts of microcomputers and application software. Students also reported their self-assessment of their personal competence with computer software on a scale from 1 to 5 with 1 representing novice and 5 representing expert. Duplass found no significant group difference on pretest scores ($t = 0.74$, $p = .56$), and the mean score on the self-reported computer competence questionnaire for both groups was 1.49. Thus, the two groups were equivalent on the dependent measure and in their self-assessed computer competence level. All students successfully

completed the out-of-class projects and reported the amount of time needed to complete each project. At the end of the semester, the pretest was re-administered as the posttest.

A paired *t* test on pre- vs. posttest differences for each group showed a significant difference in the change scores for both groups ($t = -5.55, -4.54, p < .0001$ for both). A *t* test on posttest scores, however, showed no significant difference between groups ($t = -0.03, p = .974$). Duplass (1995) also found very little difference in the average self-reported time to complete the projects but performed no statistical tests on this measure. Duplass concluded that a laboratory supplement does not improve students' command of the basic concepts of contemporary computer software. Given the limited time students have to commit to their education and limited institutional resources, Duplass did not recommend supervised laboratories as a lecture supplement.

Concerns about Duplass's (1995) study are similar to those expressed earlier. First is the lack of randomization. Although the two groups were statistically equivalent on computer knowledge and competence, other subject characteristics could have biased the results. Second, by not treating the pretest as a covariate, a testing threat to internal validity could have been present. Third, no information about the data collection instrument was reported, including any attention given to its reliability and validity. Finally, although Duplass's sample was comprised of teacher education majors, his overall conclusion that

supplementary labs are not warranted is a classic example of expressing “conclusions as truth” (Raths, 1973).

Thweatt (1994) conducted a study on students in CS 1 (a beginning programming class) to determine if student performance in a class with closed lab experiences, the treatment group, was higher than student performance in a class with open laboratory experiences, the control group. There were two motivations behind the decision to perform the study: recommendations from the joint ACM/IEEE task force on the Core of Computer Science and the continuing poor performance of students in CS 1. The study was implemented during the Fall and Spring semesters of the 1992–93 school year at Middle Tennessee State University in Murfreesboro, Tennessee.

During the fall semester, CS 1 students were divided into two balanced classes by college GPA and/or SAT-ACT scores. These two classes were randomly assigned to closed or open lab experiences. During the spring semester, students self-chose to register for the classes with closed or open lab experiences. One hundred fifteen students initially began the study; 45 completed it in the classes with open laboratory experiences, and 35 completed it in the classes with closed laboratory experiences.

The lecture portion of both classes was the same with the same tests and assignments. The same lab assignments were given to both classes. The lab assignments were discussed briefly during the lecture portion of the open lab

classes but students were expected to do the assignments on their own time and submit them the next class period. The closed lab classes had a 2-hour laboratory scheduled each week during which they were expected to complete the laboratory assignments. A graduate student and the course instructor were available during the lab to answer individual questions. During the fall semester, two different instructors taught the two groups. During the spring semester, the same instructor taught both groups. Student performance was assessed using a comprehensive final exam developed by a departmental committee.

A regression analysis was performed to determine if pre-existing predictor variables for the comprehensive final exam needed to be included in the final ANCOVA analysis. Only GPA was found to be significant. The four class groups were then tested for initial class GPA differences. Data analysis indicated that the classes with open laboratory experiences had a slightly higher GPA than the classes with closed laboratory experiences but the difference was not significant. A questionnaire was administered at the end of the spring semester to gather data on perceived previous computer-related experience. A chi-square test of independence on this data showed there was no significant difference in perceived previous computer-related experience between the two fall groups, the two spring groups, and the two combined (fall and spring) groups.

An ANCOVA with GPA as the covariate and an ANOVA showed: (1) no significant difference in mean final exam scores between the Fall open and closed

lab classes ($M_{\text{Fall-open}} = 76.9$, $n = 21$, $M_{\text{Fall-closed}} = 82.3$, $n = 18$; ANCOVA, $p = .421$; ANOVA, $p = .138$); (2) a significant difference in mean final exam scores between the Spring open and closed lab classes in favor of the closed lab ($M_{\text{Spring-open}} = 74.4$, $n = 17$, $M_{\text{Spring-closed}} = 81.0$, $n = 24$; ANCOVA, $p = .009$; ANOVA, $p = .053$); and (3) a significant difference in combined mean Fall and Spring final exam scores between open and closed lab classes in favor of the closed lab ($M_{\text{F\&SP-open}} = 75.6$, $n = 35$, $M_{\text{F\&SP-closed}} = 81.7$, $n = 45$; ANCOVA, $p = .014$; ANOVA, $p = .013$). Thweatt (1994) concluded that closed laboratories had a positive effect on students' CS 1 performance.

There were some flaws in the way Thweatt (1994) implemented the study. The most serious one was the lack of randomization for group assignment during the spring semester. Because students were allowed to choose which class they wanted to take, it is likely that the two groups were unequal on important predictor variables. Only GPA and previous computer experience were tested to insure group equivalence. This type of group assignment limits the generalizability and internal validity of the results for the spring semester. During the fall semester, when random assignment of classes to groups was made, a different course instructor was used for each class. Although an effort was made to insure lecture material was the same, it is plausible to conclude that some of the differences in student performance were attributable to instructor differences. This limits the validity of the results for the fall semester. Mortality was another threat to the

internal validity of Thweatt's study. Thweatt reasoned that because the drop-out rate in all classes was approximately the same, mortality was not a threat. This is a concern because the drop-out rate was over 50% in all groups. The loss of this many students raises the question of how representative the sample was to the population from which it was selected.

Wu (1997) investigated the effect of closed laboratory sessions that incorporated two software packages, SimLIST and SimRECUR, which were researcher-designed software lab packages for linked lists and recursion, respectively. Twenty-four junior high school CS 2 teachers were randomly assigned to either a closed laboratory-based class ($n = 12$) or an open laboratory-based class ($n = 12$). Students in both classes attended lecture sessions on linked lists and recursion. After the linked list lecture, students were given two corresponding laboratory assignments. Students in the open-lab class completed these assignments outside of class; students in the closed-lab class completed the assignments during two 90-minute lab sessions using SimLIST with a lab instructor present. Similarly, after the recursion lecture, students were assigned one related assignment. Open-lab students completed the assignment outside of class; closed-lab students completed the assignment during a 90-minute lab session using SimRECUR with a lab instructor present. Effectiveness of the two laboratories was measured by a post-laboratory achievement test embedded in a regular class exam administered after the three laboratory assignments were completed. The SimLIST

component of the exam had a reliability coefficient alpha of .62 and the SimRECUR component had a coefficient alpha of .68. Results from a one-way ANCOVA in which final grades from CS 1 were treated as a covariate showed that students in the closed laboratory-based class had significantly higher post-lab achievement scores than students in the open laboratory-based class (ANCOVA, $F_{1,21} = 4.18, p < .05$). Wu (1997) concluded that these results demonstrated the effectiveness of using software packages in closed laboratories when teaching programming.

In reviewing Wu's (1997) study, it is important to note that a significant effect was found in the use of software packages incorporated within a closed-lab setting and not between open and closed labs. It also can be argued that the reason the closed-lab group had significantly higher achievement in the targeted concepts was because they received extra instruction time: Closed-lab students received three 90-minute lab sessions that extended beyond the lecture. To guard against this from occurring in the current study, all students (control and treatment) had exactly the same amount of instruction time. It also should be noted that Wu's sample was not representative of the general student population taking CS 2. Hence, results and conclusions might not be generalizable to the general student population.

A study by Corritore et al. (1999) compared two styles of laboratory in a computer science programming course. In addition, the closed laboratory classes were compared to a traditional lecture class. The purpose of the study was to

compare the effects of a traditional closed laboratory-based class to those of an interactive demonstration laboratory-based class on student performance, class attrition, and student perceptions toward the class. Corritore et al. developed the interactive demonstration closed laboratory-based class as an alternative to the more traditional closed laboratory-based class.

Both the traditional and interactive closed laboratory-based classes were separated into two components: lecture and laboratory, but the lecture-based class did not have a laboratory component. In the laboratory classes, course time was split evenly between lecture time and weekly laboratory time. The lecture part was held in a regular classroom; the same instructor taught both lecture classes. The laboratory for the traditional closed laboratory-based class was held in a computer laboratory with computers for individual students. Students worked at their own rate on assigned experiments with a laboratory instructor who was available to answer individual questions. Students were encouraged to discuss the assignment with their peers and work in pairs. The laboratory for the interactive laboratory-based class was held in a classroom with a computer and computer projection system. The same lab instructor led the group in an interactive discussion to complete the same laboratory assignments. Group interaction and student decision-making were encouraged during the interactive laboratory.

Two intact classes of CS 2, a 2nd semester programming course at the University of Nebraska, were used in the study. Data from a third lecture-only

class, not part of the study, was also gathered. At the beginning of the study there were 28 students in the hands-on closed laboratory and 31 in the interactive demonstration laboratory. To ensure that the classes were similar, demographic data on gender, age, GPA, access to home computer, and computer science preparation were collected at the beginning of the semester. Four indicators of student performance were used: scores on laboratory assignment, out of class programming assignment scores, final exams, and total course points (sum of all assignment, test and final exam scores). A third instructor scored all assignments, tests, and exams.

A *t* test was used to compare the two class means on each of the four chosen indicators of student performances. An open-ended survey was administered at the end of the semester to determine student attitudes toward the different class structures. The mean student ranking of each class structure from the open-ended survey was used to compare student attitudes toward different class structures. Course attrition rates for each class were computed as the percent of the number of students completing the course out of the number of students starting the course.

Preliminary analysis on the demographic data at the beginning of the course indicated that the students in each of the two classes were similar in gender, age, average GPA, access to home computers and computer science preparation. Post-analyses were conducted using the data from students remaining in the two classes

at the end of the semester. Only 14 of the 28 students remained in the traditional closed laboratory-based class and 13 of the 31 remained in the interactive laboratory-based class. Post-analysis was not done on the lecture-only class due to an extremely high drop-out rate, 14 out of 21. Results indicated that there were no significant differences in student performance on each of the four indicators: Mean class scores on laboratory assignments showed no significant difference ($t = .89$, $p < .38$); mean class scores on programming assignments showed no significant difference ($t = .18$, $p < .18$); mean class scores on final examinations showed no significant difference ($t = 1.74$, $p < .10$); and class means of total point showed no significant difference ($t = 3.72$, $p < .48$).

Students from each of the two laboratory classes preferred their own respective class structure. The mean rank of the traditional closed laboratory-based class by students in the closed-lab class was 5.8, and the mean rank of the interactive laboratory-based class by students in the interactive lab class was 4.8. These mean ranks were also compared to the results from a third lecture-only class, which preferred the traditional closed laboratory-based structure with a mean rank of 4.2. Student comments on the survey confirmed support for a laboratory component in the course but also expressed concern about the additional time required by such a component.

The attrition rates of the two laboratory-based classes were nearly the same with 53.6% for the hands-on closed laboratory-based class and 54.8% for the

interactive laboratory-based class. These percentages were compared to the attrition rate for a lecture-based class from the same semester of 66.7%. The attrition rates for female students for each class were also computed and compared. The hands-on closed laboratory-based class female attrition rate was 42.9%, the interactive laboratory-based class female attrition rate was 28.6%, and the female attrition rate for the lecture-based class was 80%.

Anecdotal instructor observations were reported on the advantages of the two laboratory-based classes. Both laboratory experiences encouraged a community of students and instructors not usually seen in a lecture-based class. A mentoring relationship between instructors and students also developed in both laboratory-based classes. The traditional closed laboratory-based class encouraged students to explore questions beyond those assigned and seemed to foster a greater sense of self-confidence than the interactive laboratory-based class. In the interactive laboratory-based class the instructor was in more control of the learning experience and could guide the discussion so that it remained on-topic. In this way the interactive laboratory-based class made better use of class time and provided a more uniform learning experience than the traditional closed laboratory-based class.

Corritore et al. (1999) indicated that all findings in this study should be considered preliminary due to the small size of the sample. They concluded that student performance and attrition in both laboratory-based classes was equivalent,

while female attrition was found to be slightly lower in the traditional closed laboratory-based class. Overall student attrition and female student attrition rates in both laboratory-based classes were lower than in an all lecture class. They also noted that the validity of this conclusion is limited because the laboratory instructor of the laboratory-based classes was female whereas the instructor of the lecture-based class was male. Overall the study indicated that the interactive laboratory-based class is an effective and economical alternative to the traditional closed laboratory-based class.

In addition to the sample size and mortality issues, the primary concern of Corritore et al.'s (1999) study was the use of multiple independent *t* tests, which increases the likelihood of inflated alpha levels. Given the study's four dependent measures, a multivariate analysis of variance strategy should have been employed. Additionally, the simple comparison of percent rate for attrition with no type of statistical test for significance of the difference makes it difficult to determine if the differences are large enough to be considered important. The statistics reported for student attitudes toward class structures were also not tested for significance leaving the conclusions made about those attitudes unreliable.

Kumar (2003) compared retention and achievement results for two intact classes of Computer Science I, which included an optional closed laboratory with results from two previous intact classes that did not include the optional closed laboratory. These classes were held at Ramapo College of New Jersey during the

fall semester of 1998 and the spring and fall semesters of 2001. One hundred thirty-five students were initially enrolled in the classes, but only 80 students completed the course. The purpose of Kumar's study was to determine the effectiveness of the closed lab in improving retention and increasing achievement in the CS 1 course. Three questions guided his research: (1) Do closed labs help improve retention in the course? (2) Do closed labs help improve student learning in the course? and (3) Do closed labs help students carry out programming projects in the course?

Kumar (2003) taught all four classes and each class consisted of one 150-minute class session per week. Additionally, the 2001 classes included an optional 45-minute lab period held at the end of each class. These classes were assigned lab projects and students were allowed to work on them during the last 45 minutes of the class period. Students also were encouraged to stay an additional 45 minutes after class and continue working on the projects. These projects were marked but not graded and were not used in calculating the final grade. Students in all four classes were assigned programming projects that they were expected to complete outside of class. Students were required to complete all programming projects in order to receive a passing grade; however, these projects also were not graded and were not used in calculating the final grade. Two tests and a final exam were used as assessment instruments.

Kumar (2003) found that retention, measured as the percentage of students taking the final exam, was about the same for all four class sections. One-way ANOVAs indicated a significant difference in the average between the closed and open class sections on the first test score but not the second or the final exam. Kumar noted that the open lab class sections took this first test using paper and pencil, but the closed lab class sections took the test online. He did not believe this was important to the results. Percentages were calculated for the number of programming projects completed and found to also be about the same for all four class sessions. Kumar concluded that retention and the number of programming projects completed by students were not affected by the optional closed labs. Closed laboratories did improve student performance, but were most effective on the first test and became less helpful on later tests. Kumar indicated that his future plans are to perform a larger, more controlled study that includes a treatment component with a required closed laboratory.

The results of Kumar's (2003) study were very relevant to the current study because of the similarities between the studies. Unfortunately, Kumar's study suffered from multiple threats that question the validity of its results and conclusions. Essentially his study did not use a control group. Classes in 2001 were assigned to treatment (optional closed lab) and then compared to classes he taught in 1998 (control). The 3-year time difference is problematic and the 1998 sections did not receive the same lab assignments or instructor time as those in

2001. Students also were not surveyed for characteristics that might have skewed the results, and instruments used to measure student performance were not tested for validity or reliability. Multiple inappropriate statistical tests were used to test the hypotheses which could have led to inflated alpha levels.

Allitt-Wheeler (2005) conducted a study of 1st-year students at Tabor School of Business, a part of Milikin University in Decatur, Illinois. The purpose of her study was to determine the value of closed computer laboratories in entry-level software application courses. The study had a quasi-experimental design with non-equivalent controls, and the sample consisted of students registered during the fall semester of 2002 in MS 120, a required computer applications course consisting of units on PowerPoint, Excel, and FrontPage. Of the 83 students who completed the study, 53 were male and 30 were female. The average age of the participants was 19.1.

Each of two class sections was divided into two types of laboratory sessions (closed and open). Allitt-Wheeler (2005) taught both class sections and laboratory sessions. Open lab students were required to attend a session explaining the laboratory rules, and two testing sessions. Closed lab students were required to attend additional laboratories of 2 hours per week. The instructor was available outside of class to answer students' questions.

The instruments Allitt-Wheeler (2005) used included a variety of multiple-choice written exams, hands-on practical exams, lab assignments, and previously

used quizzes that she had validated based on her experiences. The total number of points earned was the sum of the scores on these instruments. She also used a multiple-choice exam consisting of three sections on PowerPoint, Excel, and FrontPage. This exam was administered both as a pre- and posttest. The posttest included questions about lab preference as well. An instrument to determine self-efficacy by Casidy and Eachus (1998) was also administered during the 1st week of the semester. A Cronbach's alpha of .96 was computed for this instrument.

Independent variables were listed as: total points earned in the class, scores on pretest and posttest, scores earned on each of the posttest application components (PowerPoint, Excel, and FrontPage) and self-efficacy scores. Dependent variables were listed as lab session (closed or open) and gender. There were 12 hypothesis for the study, the most important of which dealt with the difference in points scored on the various achievement instruments by students in the closed labs (control group) and students in the open labs (experimental group). Pearson product correlation, multiple *t* tests, and one-way ANOVAs were used to test the various hypotheses. For each hypothesis the decision was fail to reject. Additionally, Allitt-Wheeler (2005) determined that the majority of students preferred an open lab and that students who preferred the open lab had higher mean total points than those who preferred the closed lab.

Allitt-Wheeler (2005) concluded that lab conditions and gender had no influence on achievement and there was no correlation between self-efficacy and

achievement. She noted that the interaction between gender and lab conditions on total points was close to significance. She postulated that the reason for this result was females completed their lab assignments more completely and promptly than males and thus had higher total points. Overall the study showed that the value of closed labs is diminishing.

The results of Allitt-Wheeler's (2005) study were informative; however, her study also suffered from many of the same concerns as those cited earlier from previous studies. Incorrect statistical tools were used, (e.g., MANOVA should have been used instead of ANOVA), multiple *t* tests led to inflated alpha levels, and the researcher-constructed instruments were not tested for reliability.

Studies on Self-Regulated Learning

Bergin, Reilly, and Taylor (2005) investigated the relationship between self-regulated learning (SRL) and programming performance. The study was conducted during the 1st semester of the 2004–2005 school year involving 35 students enrolled in a postsecondary introductory programming course. The independent variables were motivation and learning strategies, which were measured using four scales of the Motivated Strategies for Learning Questionnaire (MSLQ) by Pintrich, Smith, Garcia, and McKeachie (1991). The MSLQ is a self-reporting instrument with 17 different scales; Cronbach alpha coefficients range between .52 and .90. The scales used were value components, cognitive strategies, metacognitive strategies, and resource management strategies. The dependent variable was

programming performance, which was measured using a score consisting of class tests, lab tests, and assignments,

Bergin et al. (2005) tested five hypotheses related to the relationship between SRL and performance by calculating Pearson correlation coefficients and performing several one-way ANOVAs. They found significant correlations between performance and the use of metacognitive strategies ($r = .54, p < .01$), the use of resource management strategies ($r = .57, p < .05$), the intrinsic motivation scale ($r = .53, p < .01$), and the task value scale ($r = .54, p < .01$). No significant correlation was found between programming performance and the use of cognitive strategies. When students were classified according to their level of programming expertise (high, medium, or low), ANOVA results showed a significant difference in student scores based on their: (1) use of metacognitive strategies ($F_{2, 31} = 6.1, p = .006$); (2) use of resource management strategies ($F_{2, 31} = 5.1, p = .012$); (3) level of intrinsic motivation ($F_{2, 31} = 4.2, p = .025$); and (4) level of task value ($F_{2, 31} = 6.2, p = .006$).

A stepwise regression analysis was also performed using the subscales of three categories (cognitive, metacognitive, and resource management strategies). The resulting model was significant with an adjusted $R^2 = .45, F_{1, 32} = 14.3, p < .001$, and two variables—effort regulation and critical thinking—were also significant. Bergin et al. (2005) concluded that students who use high levels of metacognitive and resource management strategies, and who have high intrinsic

motivation and task value, perform better in a programming class than those who don't. In contrast, students who use high levels of cognitive strategies perform no better than those who don't. They also concluded that self-regulated learning provides a useful area of research in the quest to discover factors that influence programming success. Bergin et al. reported two limitations to the results of their study: The study was conducted only once and needs to be extended to other situations, and the MSLQ instrument required students to self-report their behaviors.

A two-group, posttest only, double-blinded randomized study was conducted by Kumar et al. (2005) to examine the effects of self-regulated learning on programming performance. The sample used in the study consisted of 40 student-volunteers with previous programming experience ranging from 2 to 5 years. The treatment group was assigned a program to write using the Self-Regulated Learning in Programming (SRLP) tool while completing a set of self-monitoring checklists. The SRLP provided prompts to guide students through the programming task. The control group was assigned the same program under identical conditions without the SRLP tool or the checklists. The time allowed for both groups to finish the program was 100 minutes. Participants were paid \$40 at the end of the study and all who successfully completed the program were entered into a drawing for three \$50 prizes.

The independent variable was group membership and the dependent variable was the assigned grade given to the program. Completed programs were analyzed using common software metrics and were graded by an independent evaluator with several years of programming experience. The software metrics revealed that the treatment group outperformed the control group in all categories except I/O. Scores assigned by the independent evaluator were higher for the treatment group ($M = 54.39$) than for the control group ($M = 43.41$). There was also a significant difference between the time each group spent for warm-up ($t_{1, 40} = -3.07, p < .05$) and the time each group spent in thinking ($t_{1, 40} = -2.10, p < .05$). Warm-up and thinking are activities performed before coding of program begins. Kumar et al. (2005) concluded that their experiment positively revealed the effects of SRL in programming and that enhancing the process of self-regulation while programming leads to better programming performance. Concerns about the Kumar et al. study included the use of volunteers, lack of any pre-assessments to show group equivalency, achievement based on the results of a single computer program, no report of inter-rater reliability for evaluations of the programs, and the use of a financial incentive. All render Kumar et al.'s study and corresponding results as preliminary in nature.

Summary and Study Implications

As noted earlier, learning theories provide a foundation on which to study and build on the current knowledge base in the area of computer science education.

Constructivism tells us that all new knowledge is built on previous knowledge, thus it becomes important to know student attributes to judge what experiences best improve student learning. Different class structures with or without laboratories also can be grounded in several theories of learning, including individual and social constructivism and metacognition. Finally, learning style theory tells us that students may learn differently under different experiences and thus it is important to investigate how to provide those different experiences.

After a review of studies that attempted to correlate student attributes, including learning styles with achievement in and attitudes toward computer courses, and the use of different class structures to improve achievement and attitudes, it is obvious that no general consensus exists in either area. Although some studies found math ability or background as a good predictor (Butcher & Muth, 1985; Wilson, 2002), other studies found computer programming background important (Taylor & Mounfield, 1991). Some evidence also exists that computer science is an area where students with the intuitive, cognitive learning style as defined by Meyers-Briggs (1962, 1975) are most successful (Crosby & Stelovsky 1995).

Of particular interest were studies of classroom structures that included various types of laboratories. With the increased use of classes structured to include laboratories in the computer science curriculum, it was important to investigate not only how these laboratories affect students' understanding of computer science

concepts and students' perceptions of computer science as a career, but also how to structure these laboratories as part of the class and what form these laboratories should take. Several studies have attempted to answer these questions and many other anecdotal articles have been written about authors' experiences with laboratories. Although anecdotal articles are almost unanimous in their praise for classes with closed laboratories, experimental studies gave mixed reviews. Additionally, in virtually all of the experimental studies reviewed, serious flaws existed that question the internal validity of the results and the generalizability of the conclusions.

Computer science departments are under continuing pressure to maintain graduation rates in the face of declining enrollment. In this climate retention becomes a major issue. Historically, students who have a good experience in the early computer science courses continue and are successful in the major. Thus, many schools are willing to make the financial commitment required by computer science laboratories similar to other science laboratories because they "make sense." In spite of the paucity of evidence that laboratories have real effect on achievement or attitude, many schools are including some type of closed laboratory experience as part of early computer science courses.

Although several articles presented research findings relative to classes being restructured to include laboratories, few were well-designed studies, as was noted through the literature review. The current study addressed many of the flaws

found in these previous studies. It also provided additional insight into the question, “Can a class structured to include an open laboratory provide the same benefits as a class structured to include a traditional, scheduled closed laboratory?”

CHAPTER 3

METHODOLOGY

Population and Sample

Population

The target population of this experimental study was all undergraduate students enrolled in a first-level computer science course (CS 1) in the United States. The accessible population was all such students attending two small Northern Alabama public universities. In this context, a small college or university was defined as one in which its total full-time undergraduate student enrollment is less than 6,000. A convenience sample was selected consisting of intact groups of students registered for a CS 1-equivalent course at the University of North Alabama (UNA) and Athens State University (ASU).

Offering undergraduate degrees in education, nursing, business, and arts and sciences with graduate programs in education, counseling, nursing, and business, UNA is a traditional regional university located in the extreme northwest corner of Alabama and draws most of its students from Alabama, Tennessee, and Mississippi. During the 2005–2006 academic year, 73% of all students were residents of Alabama, 8% were residents of Tennessee, and 6% were residents of Mississippi (University of North Alabama, 2007). The total number of students who attended UNA during the 2005–2006 academic year was 6,404 of which 5,446 were undergraduates: 4,650 (72%) were full-time, 1,765 (28%) were part-time,

43% were male, 57% were female, 72% were white, 9% were African-American, 7% were international students, and 12% were classified as “other.” Additionally, between 70 and 75 students were computer science (CS) majors.

Located in northern Alabama, ASU is an upper division university serving junior and senior level undergraduate student transfers from the state community college system and other 4-year universities. ASU is in close proximity to Huntsville, Alabama, Redstone Arsenal, and NASA’s Marshall Space Flight Center. The university draws a large number of full-time and part-time students from the area’s industries, and has a strong demand for evening classes.

Undergraduate student enrollment during the 2005–2006 academic year was 2,642 with 1,125 (43%) full-time students: 31% were male, 69% were female, 81% were white, 11% were African-American, less than 1% were international students, and 8% were classified as “other” (C. Brett, ASU Office of Institutional Research, Planning, and Effectiveness, personal communication, June, 2007). Additionally, there were 85 CS majors and 20 Computer Information Systems (CIS) majors.

When compared to the 2006 National Survey of Student Engagement of 557 U. S. colleges (Indiana University for Postsecondary Research, 2007) and the 2004 National Science Foundation’s (2005) survey of all U. S. degree granting institutions the UNA and ASU student profiles and computer science graduate profiles were representative of the target population. Finally, a survey of 30 randomly chosen U.S. colleges and universities with Web sites confirmed that the

profiles of students required to take CS 1 at UNA and ASU was consistent with those colleges and universities that offer a similar course.

Sample

A convenience sample was selected and consisted of all UNA students enrolled in CS 155, Computer Science 1, and all ASU students enrolled in CS 317, Computer Science 1, during the Fall and Spring semesters of the 2006–2007 academic year. CS 1 at both schools is a beginning course in programming that meets the requirements for CS majors, CS minors, pre-engineering majors, and mathematics majors. CIS majors at ASU are also required to take Computer Science 1 whereas UNA CIS majors have a different curriculum.

The overall initial sample size was $N = 92$ ($N_{UNA} = 61$, $N_{ASU} = 31$), but only 71 completed all study protocols ($N_{UNA} = 45$, $N_{ASU} = 26$). A summary of the salient student characteristics is provided in Table 3.1; additional student demographics also are reported and discussed in Chapter 4. As noted from this table, overall there were twice as many males as females ($n_M = 48$, $n_F = 23$); however, at UNA the male-female ratio was 3.5 to 1. There was nearly a 50-50 split between the number of students who had a previous computer programming course ($n = 38$) and those students who did not ($n = 32$); however, of the 26 ASU students, 20 reported having a previous computer programming course. This large discrepancy is because ASU's CS 317 course requires a programming course as a prerequisite. Relative to students' level of mathematics preparation, approximately one-third

Table 3.1
Summary of Student Characteristics

School/Term	Level of Math Background ^a				Prev Com Prog Ed ^b		Gender		N
	C	T	C1	AC1	No	Yes	M	F	
UNA									
Fall	7	5	7	7	13	13	19	7	26
Spring	9	6	1	3	14	5	16	3	19
Total	16	11	8	10	27	18	35	10	45
ASU									
Fall	4	3	1	7	4	11	8	7	15
Spring	4	0	3	4	2	9	5	6	11
Total	8	3	4	11	6	20	13	13	26
Overall									
Fall	11	8	8	14	17	24	27	14	41
Spring	13	6	4	7	16	14	21	9	30
Total	24	14	12	21	33	38	48	23	71

Note. ^aLevel of Math Background reflects the number of students who had taken as their highest math course, C = College Algebra, T = Trigonometry, C1 = Calculus I, AC1 = Above Calculus I. ^bPrev Com Prog Ed reflects the number of students who had or did not have a previous computer programming course.

($n = 24$) of the students listed college algebra as their highest level of math preparation, one-sixth ($n = 14$) listed trigonometry, one-sixth ($n = 12$) listed Calculus I, and one-third ($n = 21$) listed courses above Calculus I. Additional information that reflects fall and spring data is provided in Table 3.1.

Three teachers implemented the study protocols: one at UNA (female), who was the researcher, and two at ASU (one male and one female); the male teacher taught during the fall term. Both female teachers were full-time faculty members at their respective schools and the male teacher was a part-time adjunct. All three teachers were nearly the same age and the ASU teachers had approximately the same number of years teaching experience, while the UNA teacher had twice as

Table 3.2
Summary of Teacher Characteristics

School	N	Gender	Age	Yrs Teach ^a	Status ^b
UNA	1	Female	54	22 ^c	Full-time
ASU					
Fall	1	Male	53	10	Part-time
Spring	1	Female	43	8	Full-time

Note. ^aYrs Teach = total number of years teaching. ^bStatus = whether the teacher was employed full- or part-time at the school. ^c10 years teaching CS.

many years teaching as the other two. A summary of the teacher characteristics is provided in Table 3.2.

Power Analysis

According to Cohen, Cohen, West, and Aiken (2003, p. 92), “statistical power analysis is concerned with the special case of determining the probability that the sample value will be significantly different from some hypothesized value, typically one of no effect such as a zero R^2 .” The components of a power analysis include the population effect size (ES) for either the overall R^2 or for a partial coefficient (e.g., sr^2), the sample size, the number of independent variables (IVs), and the significance criterion. Given any three of these parameters, the fourth can be determined using a computer program, power tables, or by following Cohen et al.’s algorithm.

A power analysis was conducted from three aspects: (1) overall model, (2) research factor sets, and (2) single research factor. All calculations were done based on a preset $\alpha = .05$, and effect sizes were calculated using actual R^2 values. The

Table 3.3
Power Analysis and Calculated Powers for $\alpha = .05$

Variable	Actual Value	Actual Effect Size	Approximate Power for $N = 142$
X_1-X_{13} (Overall Model) ^a	$R^2 = .668$	2.012	> .99
Set A (Student Attributes Set) ^b	$R^2 = .193$	0.239	.99
Set B (Academic Set) ^c	$R^2 = .040$	0.042	.50
Set C (Treatment + Subjects Set) ^d	$R^2 = .668$	2.012	> .99
Treatment Only	$R^2 = .001$.001	.067
Any single research factor ^e	$sr_i^2 > .016$	$ES > 0.016$	> .33

Note. Total sample size = 71 but data were coded for two repeated measures using criterion scaling via a multiple regression analysis strategy; thus, $N = 142$.

^aThe overall model consisted of 13 independent variables that were separated into three functional sets A, B, and C. ^bThe student attributes set consisted of variables representing gender, previous computer programming education, math background, and learning style. ^cThe academic set consisted of variables representing semester, school, and the interaction between semester and school. ^dThe treatment + subjects set consisted of variables representing the classroom structure and the total points scored by each participant. ^eResearch factors consisted of gender, previous computer programming education, math background, and learning style.

results of this analysis are summarized in Table 3.3. As noted in this table, given the study's sample size of 71—which was doubled to accommodate two repeated measures using criterion scaling—and the actual R^2 and effect size values, the statistical power related to finding a significant overall model was greater than .99; this was also the case for Sets A and B. However, the statistical power of finding Set B (the set of academic variables) significant was equivalent to a coin toss (.50), the power of finding any single research factor significant was .33, and the power of finding the group membership variable significant was .067. These power parameters will be discussed further in Chapters 4 and 5.

Instrumentation

Data collection instruments used in this study included the following: An attitude scale, a learning style inventory, and researcher-developed assessments, including unit exams, laboratory assignments, and programming assignments. The attitude scale was administered during the 1st, 7th, and last week of classes; the learning styles inventory was administered during the 1st week of classes; and the researcher-developed assessments were administered continuously throughout the semester. A copy of the unit exams, the attitude scale, and a sample of the KLSI-3.1 are included in Appendix A (the KLSI is a copyrighted instrument and thus only a sample of its questions and contact information are provided); copies of the lab and programming assignments are provided in Appendix B. A description of each instrument follows.

Attitude towards Computers and Computer Courses

Newby and Fisher's (1997) Attitude towards Computers and Computer Courses (ACCC) instrument was used to assess students' attitudes toward computers, CS 1 as a course, and computer science as a career choice. Because the ACCC was developed to measure college-level computer science students' attitudes in courses that included a laboratory, it was appropriate for the study.

The ACCC has four subscales: Anxiety, Enjoyment, Perceived Usefulness of Computers, and Perceived Usefulness of Course. The first three subscales were derived from Loyd and Loyd's (1985) instrument; the last scale was created by

Newby to replace Loyd and Loyd's Confidence scale because it was difficult to distinguish between lack of confidence and anxiety. The Anxiety subscale measures how comfortable students feel using a computer; a sample statement is, "Working with a computer makes me very nervous." The Enjoyment subscale measures the extent to which students enjoy using a computer; a sample statement is, "I enjoy learning on a computer." The Usefulness of Computers subscale measures students' beliefs on the usefulness of computers; a typical statement is, "My future career will require knowledge of computers." The Usefulness of Course subscale measures how useful students find the course; a typical statement is, "I do not think I will ever use what I learned in this class."

Each subscale consists of seven items and responses are assessed on a 5-point Likert scale ranging from 1 = Strongly Disagree to 5 = Strongly Agree. Thus, scores can range from 28 to 140, with higher scores indicating more positive attitudes or perception. Eight of the 28 items are negatively worded statements and hence reversed scored. The four subscales of the ACCC have a reported Cronbach alpha that ranges from .64 to .90, and the subscales' discriminant validity mean correlations range from .28 to .49. Factor analysis confirmed that four distinct factors exist for the ACCC. In addition to reporting students' overall attitudes, summary analyses are provided separately for each subscale. Using sample data collected from the current study, the overall reliability alpha was .95, and the respective subscale alphas were .82, .94, .77, and .92.

Kolb's Learning Style Inventory

Kolb's Learning Style Inventory-Version 3.1 (KLSI-3.1) was used to classify study participants by their dominant learning style. The instrument is commercially available in printed or Web-based forms from the Training Resource Group, Hay/McBer (Kolb, 2005). The KLSI-3.1 consists of 12 incomplete statements with four possible endings for each statement. For each of the 12 statements, participants were asked to rank the four possible endings from best answer to worst answer according to their personal preference about how they like to learn. For example, one statement from the KLSI-3.1 is, "I learn by ...," with ending choices: "thinking," "watching," "learning," and "doing." Each possible ending categorizes test-takers into one of four learning styles as defined by Kolb: convergers, divergers, assimilators, and accommodators. Convergers learn through the use of active experimentation and abstract conceptualization. Divergers learn through reflective observation and concrete experience. Assimilators learn through reflective observation and abstract conceptualization. Accommodators learn through active experimentation and concrete experience.

Earlier versions of the KLSI were criticized for low test-retest reliability and internal consistency (Bostrom, Olfman, & Sein 1993). The inventory underwent several revisions since its initial offering to improve its scores and increase its practical uses. In the latest version, KLSI-3.1, the four possible ending choices of thinking, watching, learning, and doing are randomized to prevent test-

takers from selecting answers according to their position in the list. The results of several recent studies of the KLSI-3.1 suggest good internal consistency reliability across a variety of populations with Cronbach's alpha coefficients that ranged between .77 and .84 (Kayes, 2005; Kolb & Kolb, 2005; Wierstera & DeJong, 2002). Veres, Sims, and Locklear (1991) tested a randomized KLSI similar to the KLSI-3.1 and reported test-retest correlations above .90 in all cases. The inventory has been shown to be effective in characterizing students by their dominant learning style (Bostrom et al.). The KLSI has gained acceptance among educational researchers and has been widely studied in the areas of computer and information science. Studies have examined the relationships between learning style and end-user training and software use, problem solving and decision making, on-line search behavior, and performance in computer training and computer-assisted instruction (Kolb & Kolb, 2005).

Teacher-Based Assessments

As noted in the introductory paragraph of this section, several researcher-developed assessments were used to measure students' achievement, including laboratory assignments, programming assignments, and unit exams. The corresponding instruments for these assessments were pilot tested during the 2006 Summer term at UNA and ASU. A description of these assessments follows, and a summary of the type and number of assessments, as well as the point values assigned to each assessment is given in Table 3.4.

Table 3.4
Session Assessments

	Unit Exams	Laboratories	Programs	Total
Number of Assessments	2	15	3	
Points Per Assessment	100	2	15	
Total Points	200	30	45	275
Percent of session grade	72%	11%	17%	

Note. Sessions were 7 weeks in length and there were two sessions each semester with the same number and types of assessments.

Laboratory Assignments

Laboratory assignments consisted of a purpose statement, a reading assignment, a programming exercise, and a post-lab. The purpose statement specified the skills students would acquire by completing the lab. The reading assignment directed students to a specific textbook passage that corresponded to the lab. The programming exercise contained a detailed description of a specific programming problem relative to the purpose statement and included printouts of any code needed to complete the exercise. The post-lab was a form that students used to record program output and write any observations or discoveries they made while completing the exercise.

The lab programming exercises consisted of problems that required students to write short blocks of code to be used in a provided program, modify a previously written program, or write a function for a provided program. For example, the programming exercise in Laboratory Assignment 12 (see Appendix B) was to modify an existing program to find the minimum of two numbers using provided

psuedocode. Code to get the two numbers from the user and to print the minimum were provided in the assignment handout. The assignments were designed to be completed in less than 25 minutes. Laboratory assignments in general, and the lab programming exercises in particular, were used to supplement classroom lectures and prepare students for the longer, more complex programming assignments.

The design and structure of the laboratory assignments followed Walker's (2004) recommendations and tested students' knowledge at all six levels of Bloom's (1956) taxonomy of learning (see Figure 3.1). Each laboratory was worth up to 2 points for a maximum of 30 laboratory points for each session: Students in closed lab-based classes received 1 point based on their participation in the laboratory and 1 point for the timely return of complete post-labs. Students who did not finish a lab assignment during the class session or were absent from a closed lab class were also given the option to turn in the completed code along with their post-labs. Students in open lab-based classes received 1 point each for timely return of completed code and post-labs. Assignment of points for laboratory exercises was performed using a researcher-designed score card (see Appendix B). Because session grades were based on 275 points, the labs represented approximately 11% of students' session grades. Copies of laboratory assignments are located in Appendix B.

The laboratory exercises were assumed to be content valid because they were based on CS 1 textbook exercises, examples, and lab manuals. Additionally, I

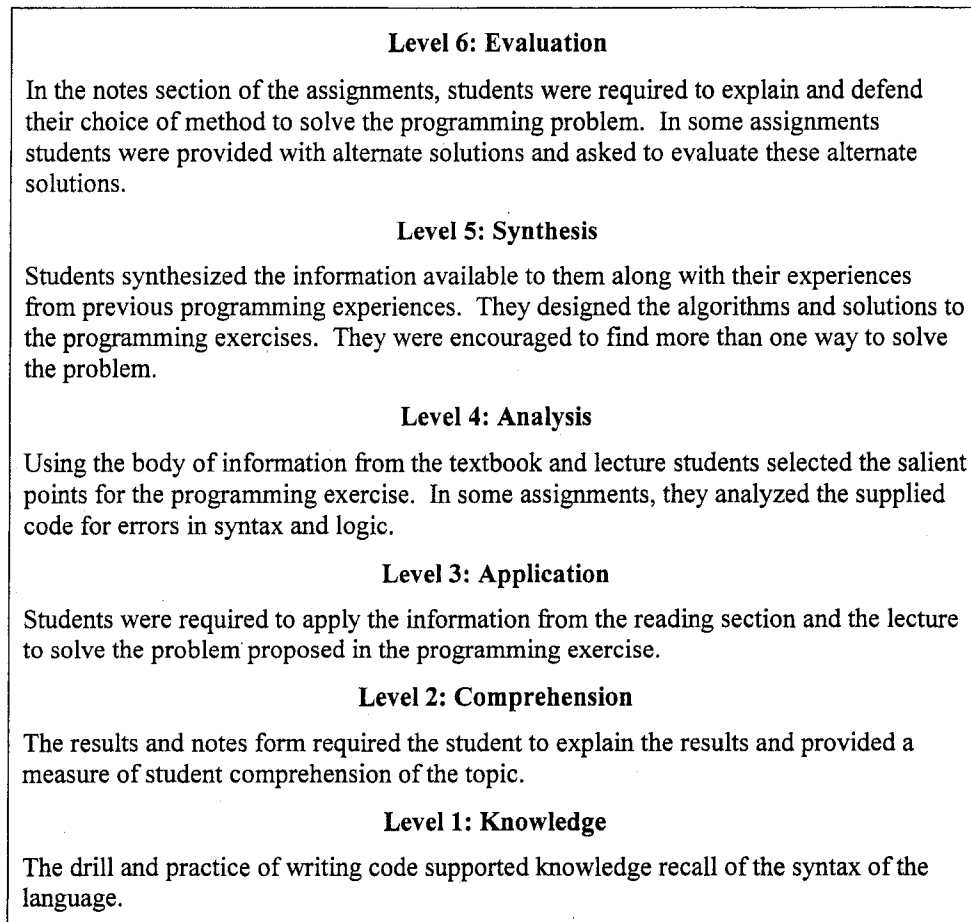


Figure 3.1. The relationship between the activities of students' laboratory assignments and the six levels of Bloom's (1956) taxonomy.

chose the assignments so they matched the course objectives and prepared students to write the longer programming assignments. Further content validity was provided by another CS 1 instructor who reviewed and approved of the laboratory assignments. An inter-rater reliability coefficient of .93 was calculated for the two instructors who graded the laboratory exercises and programming assignments.

Programming Assignments

The programming assignments were selected from various CS 1 textbooks, including Dale (2004), Dale and Weems (2004), Dietel and Dietel (2005), and Zak (2005). These assignments required students to use several topics from recent classes and were designed as an extension of the skills learned in the laboratories. Students were required to complete these assignments individually outside of class. For example, one assignment required students to write a program using functions that reads into an array the name and grade point average (GPA) of an unknown number of students (maximum of 100 students). The program finds the average GPA and then prints the names of all students whose GPA is greater than the average. Programming assignments were graded using a researcher-developed rubric (see Appendix B). Each programming assignment was worth up to 15 points for a maximum of 45 points per session, which represented approximately 16% of students' session grade. Copies of the programming assignments are also provided in Appendix B.

The programming assignments were assumed to be content valid because they were traditional in nature, based on CS 1 textbook programming exercises, and chosen to correspond with course objectives. Again, further content validity was provided by a second CS 1 instructor who reviewed the programming assignments. As was stated earlier, an inter-rater reliability coefficient of .93 was

calculated for the two instructors who graded the laboratory exercises and programming assignments.

Unit Exams

The 14-week semester was divided into two 7-week sessions. The topics covered in each session are summarized in Table 3.5. Two researcher-prepared unit exams were administered during the middle and last week of each session: the first unit exam was given during Week 4, the second during Week 7, the third during Week 11, and the last at the end of Week 14. The exams consisted of questions from test banks that accompanied the course textbook (Dietel & Dietel, 2005) as well as two other popular CS 1 textbooks (Dale & Weems, 2004; Zak, 2005). The exams consisted of multiple-choice questions and represented all levels of Bloom's (1956) taxonomy. The unit exams were graded manually by the researcher and all questions were dichotomously scored.

The questions were presumed to be content valid because they were derived from textbook test banks. Additional attention to content validity was provided in two ways. First, as I developed the unit exams, I carefully weighed each selected item from the test bank relative to the course objectives. Second, my dissertation committee members (two of whom were computer science professors) reviewed the unit exams as a part of the proposal for this dissertation. Reliability indices for the four unit exams based on Kuder-Richardson-21 were respectively .74, .83, .70, and .82. Copies of the unit exams are provided in Appendix A.

Table 3.5

Session Topics

Session	Topics
1	History of C++, C++.NET, intro to object technology, UML, parts of a C++ program, output statements, reserved words and identifiers, literals, constants, variables, integers and strings, input statements, assignment statements, order of operations, standard library functions, functions, parameters, classes, objects, member functions, data members, set and get functions, constructors, algorithms, pseudocode, float and double types, equality and relational operators, logical operators, and selection statements.
2	Repetition statements, nested blocks, abbreviated assignment operators, case statements, char, boolean, and string data type, break and continue statements, file I/O, functions with multiple parameters, scope, activation records and the call stack, one dimensional arrays, searching and sorting arrays, parallel arrays, two dimensional arrays.

Course Description

The ASU course, CS 317 (Computer Science 1 C++), and the UNA course, CS 155 (Computer Science I), were used in the study. These courses are representative of ACM's (1979) CS 1 introductory undergraduate computer science course, which was later updated in the 2001 curricula guide (ACM, 2001). They are gateway courses for CS majors. The fundamentals taught in these courses provide the foundation for later studies. Based on the recommendations of the joint ACM and IEEE-CS computing committee recommendations, the respective CS 1 courses at UNA and ASU provide students with an introduction to the theoretical foundations of computer science and to the constructs of the programming language C++. Topics covered in the courses include basic language constructs, procedural abstraction and structured programming, principles of software engineering, functions, structured data types, and an introduction to classes and

object oriented programming. A copy of the respective catalog descriptions for the two courses is provided in Appendix C.

The prescribed textbook at both schools was Dietel and Dietel's (2005), *Small C++: How to Program* (5th ed.), which was selected by faculty at each school and was in its 1st year of use at the colleges. The textbook's chapters (and sections) that corresponded to the topics in CS 1 were:

Chapter 1: Introduction to Computers (entire)

Chapter 2: Introduction to C++ Programming (entire)

Chapter 3: Introduction to Classes and Objects (3.1–3.7)

Chapter 4: Control Statements Part I (entire)

Chapter 5: Control Statements Part II (entire)

Chapter 6: Functions and Introduction to Recursion (6.1–6.10, 6.12)

Chapter 7: Arrays (7.1–7.9)

Chapter 17: Using Files (17.1–17.5)

Traditionally, the manner in which this course is taught is usually the prerogative of the instructor and has included many different implementations. For example, in past studies, this course has employed variations of class structures that include laboratory exercises as part of the class, independent of the class (e.g., a 4-hour course with 3 hours of class and a 1-hour lab), and as separate homework. In the current study, the manner in which the targeted classes were taught was specified by the research protocols.

Procedures

Research Methodology/Design

A counterbalanced repeated measures design involving intact classes of students was used in this study. In a repeated measures design, “all participants in a single group participate in all experimental treatments with each group becoming its own control. The researcher compares a group’s performance under one experimental treatment with its performance under another experimental treatment” (Creswell, 2005, p. 303). A repeated measures design eliminates the subject characteristics threat, which is the most difficult internal validity threat to control in education research studies, because students serve as their own control. It is also more powerful than completely randomized designs and far less subjects are needed for a study.

Repeated measures “designs also have potentially serious disadvantages, unless care is taken. When several treatments are involved, the order in which treatments are administered might make a difference in the subjects’ performance. Thus, it is important to counterbalance the order of the treatments” (Stevens, 2002, p. 495). When the order of treatments is counterbalanced, participants are placed in groups and each group experiences the treatments in a different order. This design was appropriate to the current study for three reasons: (1) there were a limited number of students; (2) only one section of the targeted course was offered per

term; and (3) the section could not be divided into experimental and control groups, thus rendering an experimental design inapplicable.

The 14-week Fall 2006 term at UNA was divided into two 7-week sessions. A single intact group of UNA students taking CS 155 during the term was administered two separate treatments, one each session. The first treatment involved restructuring the class to include an open laboratory; this was conducted during the first half of the term. The second treatment involved restructuring the class to include a closed laboratory; this was conducted during the last half of the term. This implementation protocol was repeated with a second intact group of UNA students during the Spring 2007 term. In this second implementation the class restructuring was reversed; that is, the closed laboratory-based class structure was conducted during the first half of the semester and the open laboratory-based class structure was conducted during the second half of the semester. During each session students were administered two unit exams, participated in 15 labs, and completed three programming assignments (see Table 3.4). This experiment was also conducted concurrently at ASU with intact groups of students taking CS 317 during the fall and spring terms, respectively. At ASU, however, the treatment protocols were reversed relative to the order in which they were implemented at UNA. A summary of the implementation schedule is given in Table 3.6.

Table 3.6
Treatment Implementation Schedule

Semester	UNA		ASU	
	Session 1	Session 2	Session 1	Session 2
Fall	Open Lab	Closed Lab	Closed Lab	Open Lab
Spring	Closed Lab	Open Lab	Open Lab	Closed Lab

Note. Each session was 7 weeks long.

Study Implementation

As noted previously, the purpose of this study was to assess the effect of classroom restructuring on student achievement in an introductory computer science course and on their attitudes toward computers and computer courses. Classroom restructuring involved incorporating a closed or open laboratory. These two restructuring formats represented the two treatment conditions and involved a repeated measures design in which a single group of students was exposed to both treatments. Prior to the implementing this study, pilot studies were conducted during Summer 2006 at UNA and at ASU. During the pilot studies all assessment protocols (i.e., unit exams, lab and programming assignments, and corresponding rubrics) were administered and reviewed for validity and reliability purposes as well as for implementation issues. Results from the pilot studies were used to make minor adjustments in the assessment protocols and the design and implementation of the final study. For example, the lab assignments were modified to include additional instructions because during the open lab-based class structure session, students needed very detailed instructions in order to complete the assignments.

As noted in the foregoing paragraph the final study was conducted during the 2006–2007 academic year and involved intact groups from UNA and ASU during the fall and spring terms. All treatment conditions were implemented in special classrooms that were designed to double as both a lecture room and a computer laboratory. The rooms contained a whiteboard, a teacher-controlled computer with a computer projection system, and individual student computers. At UNA there were 18 student computers, which were networked with the teacher-controlled computer. At ASU there were 24 student computers, however only 18 were functional. All programming was done in C++ on IBM compatible microcomputers using Microsoft's Windows XP operating system. This was the default language and hardware platform at the targeted universities. Diagrams of the respective classroom setups are located in Appendix C. The same classrooms were used for each of the two different classroom structures. Given the sample sizes of the UNA classes, a 1:1 student-computer ratio was not possible. This is noted below as part of the discussion.

Closed Laboratory-Based Class (Treatment)

The closed lab-based class structure consisted of two parts, lecture and laboratory. At UNA, three 50-minute class periods per week were equally divided between lecture and lab: the first 25 minutes consisted of lecture and the remaining 25 minutes consisted of lab. At ASU, a single 150-minute class period was equally divided between lecture and lab. The lecture consisted of a multimedia

presentation of the targeted concept. During the laboratory period, students worked individually or in pairs at their own pace on the assigned lab and the classroom instructor was available for assistance. A different lab was assigned to students during each class session except for review classes and days in which the assessment protocols were administered. Thus, there were a total of 15 lab assignments per session. Table C.1 in Appendix C contains the schedule in which these assignments were given.

As described earlier in the Instrumentation section, laboratory assignments consisted of a purpose statement, a reading component, a lab programming exercise that involved students writing a short piece of code, and a post-lab. The labs were available online and were provided to students in hard-copy form. Students were expected to complete the reading component prior to the lab period, the lab programming exercise during the lab period, and the post-lab before the next class period. Students were not penalized if they did not complete the lab programming exercise during the lab period but were required to provide copies of completed code at the next class period. Appendix B contains copies of the actual lab assignments.

Open Laboratory-Based Class (Control)

The open lab-based class structure was a traditional lecture class with outside laboratory assignments. Thus, unlike the closed lab-based treatment condition, there was no separate lab component integrated into the instructional

period. Instead, students completed the laboratory assignments outside of class at their convenience with no instructor supervision. These laboratory assignments were exactly the same as those given in the closed lab structure and were made available to the students in exactly the same manner. A final difference between the class structures was that students in the open lab-based class were permitted to use their own personal computers or those provided by the university in on-campus computer laboratories.

As was the case with the closed lab structure, there were three 50-minute class periods per week at UNA and one 150-minute class meeting per week at ASU. Time was allotted at the beginning of each class period to discuss the laboratory assignment from the previous class and the reading assignment for the current class. This was then followed by the lecture, which consisted of the same multimedia presentation used in the closed lab structure. Any remaining class time was devoted to a question and answer session. Students were required to submit, by the beginning of the next class period, hard or soft copies of revised or completed lab program exercises along with post-labs.

Human Subjects Research Issues

The implementation of the study involved human subjects. As such, I followed the ethical principles of human subject research, including providing subjects with informed consent forms, ensuring that participants remained anonymous, and maintaining the confidentiality of the research data. As part of

this effort, I received approval from the Institutional Review Board for Human Subjects (IRB) at UNA, ASU, and Florida Institute of Technology. Copies of the applications, consent form, and approval notices are given in Appendix D.

Threats to Internal Validity

Threats to internal validity are an important consideration in experimental studies because the presence of such threats may lead to study results being influenced by unknown variables other than those targeted by the researcher. Twelve threats to internal validity are described by Fraenkel and Wallen (2003) and well-designed studies should take these threats into consideration. Although a repeated measures design study provides control over many of these threats, it is still important to consider all of them. As a result, a description of each threat, the manner in which each threat might have affected the study, and what was done to minimize or control the effects of applicable threats is discussed.

Subject Characteristics

The possibility that subjects in a study might differ in unknown and unintended ways that are related to the study's dependent variables is called a subject characteristic threat or selection bias. Subject characteristics related to this study included gender, mathematics background, previous computer programming education, learning styles, and attitudes toward computers and computer courses. Given the research design of this study, namely, repeated measures, this threat was not applicable because subjects served as their own control (Creswell, 2005).

Nevertheless, data were collected on these student attributes and were incorporated into the study as predictor variables.

Mortality

The loss of subjects during the implementation of a study is called a mortality threat. This occurs when participants drop out of a study or when they fail to complete all of the study's assessment protocols. This loss of data can lead to bias in the results, limited generalizability, and a reduction in statistical power. Prior to implementing the study, mortality was initially a concern because of the traditionally high dropout rate for CS 1; however, this threat was not applicable because a repeated measures design was used. Nevertheless, two actions were taken to mitigate the mortality threat: I acquired as large a sample as possible by using two universities, and I reported the dropout rate.

Location

When the location at which a study takes place affects the results of the study, a location threat occurs. Differences in the facilities and resources used in a study can affect the results. For example, if one group is administered a test at home over a computer network while another group takes the same test with pencil and paper in a classroom, a significant portion of the difference in test scores could be due to location. In this study there was a potential for a location threat because two different schools were used. This was partially mitigated by using nearly identical classrooms at both schools, and by assigning a variable to represent school

location to statistically measure any location effect. A separate issue was the problematic nature of the open lab-based class structure because students were free to complete assignments at different locations. However, this was not considered a threat because this classroom structure is the traditional way in which labs are assigned and was regarded as the control condition.

Instrumentation

When there is a change in the instruments used in the study or the reliability of an instrument is in question an instrumentation threat exists. An instrumentation threat can be the result of instrument decay, data collector characteristics, or data collector bias.

Instrument decay. Instrument decay occurs when an instrument is modified during a study or there is a change in scoring during the study. Change in scoring often exists when the instrument is long or difficult to score, which can result in scorer fatigue. Decay was not applicable in this study because there was no modification of the instruments and a scoring rubric was used for all the assessment protocols. Furthermore, the main achievement instrument was dichotomously scored, and the lab and programming assignments were scored by two people and had an inter-scorer reliability of .93.

Data collector characteristics. Data collector characteristics include age, gender, nationality, and research experience. From a procedural perspective, if two or more data collectors are used it is possible for their characteristics to affect study

results. For example, student participants are usually more comfortable when interviewed by a person of their gender and may give more information. In this study, although there were three different instructors, only two were involved in data collection procedures. I collected all data from UNA as well as the data from the fall term at ASU; the ASU instructor for the spring term, however, collected the data herself. Thus, a data collector characteristic threat was possible. To control for this threat, data collection procedures were monitored by the researcher. This monitoring included email communications between the spring term ASU instructor and me about proper data collection procedures; I also personally visited the ASU campus twice to administer study instruments and to insure that proper protocols were followed. This threat was further mitigated because the ASU instructor and I shared similar personological and academic characteristics, including gender, age, level of education, and employment status at our respective schools.

Data collector bias. Data collector bias refers to data collectors' or scorers' approaches or attitudes toward the research due to their knowledge about the study's hypotheses. Data collector bias can affect results by the unconscious attitudes of the collector. In this study there was a potential for data collector bias on the researcher's part only because the fall term ASU instructor did not collect data and the spring term ASU instructor did not have enough information about the

research to form a bias. This threat was partially controlled by following standardized data collection procedures and scoring.

Testing

When a study includes the use of a pre-assessment, a testing threat exists if the results on the post-assessment are affected by the practice of taking the pre-assessment. In this study, no pre-assessments were used and hence, there was no testing threat.

History

When the results of a study are influenced by unexpected events that occur during the course of the study, a history threat exists. For example, a study on attitudes toward science could be impacted if during the intervention a NASA mission to the moon occurred. The only potential history threat in the current study was related to the health of two instructors. The fall term ASU instructor had a heart attack during the 12th week of classes, and I had pneumonia during the 6th week of the spring term. This resulted in the ASU instructor missing one class and I missing two classes. This was not considered a concern because all students during the same semester at the same school experienced the same history effects, which helped control the impact of this threat.

Maturation

When subjects change during a study due to the passage of time a maturation threat exists. Due to the age of the students and the short duration of the

study, 1 semester, this threat was not a concern. Furthermore, the repeated measures design is not affected by this threat (Creswell, 2005).

Subject Attitude

The attitudes of subjects toward their participation in a study can influence the results of a study. This threat can be characterized in two ways: when the reaction of subjects in a study is different to those who are not a part of the study (called the Hawthorne effect), or when there is a difference in reaction between groups in a study (called the novelty effect). To control for this threat, attitudes were assessed as part of the study. Furthermore, the nature of the research design (repeated measures) enabled all subjects to experience both treatments, which also helped control for this threat.

Regression

When the students used in a study have extremely low or high pre-treatment performance, there exists the possibility that post-treatment scores will change due to the tendency of scores to regress toward the mean rather than the effects of the study treatments. Thus, a group scoring extremely low or high on a pretest will likely score closer to the class mean on the posttest. The repeated measures design is not affected by this threat (Creswell, 2005).

Implementation

When one group (treatment or control) is treated in ways that are not part of the study but may be advantageous to that group, an implementation threat exists.

Because only a single group was used and all participants were exposed to all treatments, this threat was not applicable. However, because of this study's design (i.e., repeated measures), it was possible for a context effect to be present. A context effect refers to the degree to which participants' perception of a particular phenomenon in a given context is altered in another context. Greenwald (1976) identified three context effects related to a study's implementation: practice, sensitization, carry-over. Practice effects occur when performance on a skill improves with practice. This can be controlled by providing extensive practice prior to the beginning of the experiment. Sensitization effects occur when participants, exposed to several treatments, form hypotheses about treatment effects and respond to those hypotheses. This can be controlled by camouflaging the differences in treatments. Carry-over effects can occur when one treatment influences a subsequent treatment. This can be controlled or minimized by counterbalancing the sequence of treatments and providing sufficient time for the effects of one treatment to be reduced prior to beginning the next treatment. Given the nature of the study, all of the suggested approaches were not possible, however the sequence of treatments was counterbalanced and a 1-week review and testing period was placed between sessions, which helped reduce carry-over effects.

Treatment Verification and Fidelity

Treatment verification consists of the steps taken by a researcher to confirm that a study was implemented as planned. It is needed in order to provide for

validity and generalizability of study results and is the responsibility of the researcher. In the context of this study, treatment verification involved insuring that instructors implemented the correct class structures (i.e., open or closed lab-based) according to schedule, and that all lab assignments, programming assignments, and unit exams were administered. The following verification methods were used to ensure treatment fidelity: (1) I prepared a lesson schedule for each semester and school, (2) I prepared PowerPoint presentations on which all lectures were based, and (3) I conducted systematic classroom observations to confirm treatment fidelity. Because of concern over possible digressions from study protocols at ASU, I also performed statistical analyses using the data from both schools and on the data from UNA only. Identical conclusions resulted from both sets of analyses and thus the decision was made to include the ASU data. This is discussed further in Chapter 4 (see Table 4.14).

Data Analysis

There were three sets of independent variables for this study: Set A, student attributes, consisted of gender, previous computer programming education, math background, and learning style; Set B, academic attributes, consisted of school and semester variables, which also captured instructor effects; and Set C, classroom treatment + subjects, consisted of the classroom treatment variable (i.e., open or closed lab-based class structure) and two variables that were used to accommodate the repeated measures analysis: the sum of each subject's achievement on the

closed lab-based class and the open lab-based class (i.e., the sum of scores acquired from each repeated performance), and the sum of each subject's attitude at the conclusion of the closed lab-based class and the open lab-based class. The dependent measures were students' attitudes and achievement. The variables' data type and description are summarized in Table 3.7, and descriptive statistics involving these variables are summarized and discussed in table and narrative forms in Chapter 4.

The primary inferential statistical strategy used to test the study's hypotheses was multivariate analysis of variance (MANOVA) via hierarchical multiple regression. MANOVA is an omnibus strategy used for analyzing data involving multiple dependent measures. It provides a means for controlling Type I and Type II error rates for studies with multiple independent and multiple dependent variables. The results of inferential statistics are provided in Chapter 4.

Table 3.7
Characteristics of the Variables

Variable	Measurement/Meaning
Set D (Dependent Variables)	
Y_1 = Session Achievement	Total achievement points at end of Session
Y_2 = Session Attitude	Scores on ACCC at end of Session
Set A (Student Attributes)	
X_1 = Gender ^a	
X_2 = Previous Computer Programming Education ^b	Previous computer programming course
M_1 = Math Background-1 ^c	Trigonometry vs. College Algebra
M_2 = Math Background-2 ^c	Calculus I vs. College Algebra
M_3 = Math Background-3 ^c	Above Calculus I vs. College Algebra
L_1 = Learning Style-1 ^d	Accommodators vs. Convergers
L_2 = Learning Style-2 ^d	Assimilators vs. Convergers
L_3 = Learning Style-3 ^d	Divergers vs. Convergers
Set B (Academic Attributes)	
S_1 = School/Semester-1 ^e	UNA vs. ASU
S_2 = School/Semester-2 ^e	Fall vs. Spring Semester
S_3 = School/Semester-3 ^e	Interaction of School and Semester
Set C (Treatment + Subjects)	
X_3 = Treatment ^f	Closed lab structure vs. Open lab structure
ST_1 = Subjects Total-1 ^g	Sum of points for both sessions
ST_2 = Subjects Total-2 ^h	Sum of scores on ACCC for both sessions

Note. ^aGender was dummy coded with females as the reference group. ^bPrevious Computer Programming Education was dummy coded with no previous computer programming course as the reference group. ^cMath Background-1, Math Background-2, and Math Background-3 = dummy-coded factors representing respectively, Trigonometry, Calculus I, or Above Calculus I as the highest level of math course attained, with College Algebra (or its equivalent) as the reference group. ^dLearning Style-1, Learning Style-2, and Learning Style-3 = dummy-coded factors representing respectively Kolb's (2005) learning styles of Accommodators, Assimilators, and Divergers, with Convergers as the reference group. ^eSchool/Semester-1, School/Semester-2, and School/Semester-3 are contrast coded variables representing respectively UNA vs. ASU, Fall vs. Spring, and the interaction between school and term. ^fTreatment = dummy-coded factors representing the two class structures, Open lab vs. Closed lab, with Open lab as the reference group. ^g ST_1 = the sum of each subject's achievement on the closed lab-based class and the open lab-based class (i.e., the sum of scores acquired from each repeated performance). ^h ST_2 = The sum of each subject's attitude at the conclusion of the closed lab-based class and the open lab-based class.

CHAPTER 4

RESULTS

Introduction

This chapter is presented in three sections. The first section provides descriptive statistics for the sample relative to data acquired from the study's assessment protocols, including the achievement instruments (lab assignments, programming assignments, and unit exams), Newby and Fisher's (1997) Attitudes toward Computers and Computer Courses (ACCC) instrument, and Kolb's (2005) Learning Style Inventory-Version 3.1. The second section contains inferential statistics results from data analysis corresponding to the study's design (i.e., repeated measures). The final section contains results from testing the study's research hypotheses.

Descriptive Statistics

Overview

In this section, a summary of the data corresponding to the study's instruments is reported. This includes the results of students' learning styles, achievement, and attitudes toward computers and computer courses. Other student demographics such as gender, previous math experience, and previous computer programming education were provided in Chapter 3 (see Table 3.1). Teacher attribute data such as gender, age, years teaching, and employment status (i.e., full-time vs. part-time) also were provided in Chapter 3 (see Table 3.2).

Kolb's Learning Style Inventory

As discussed in Chapter 3, Kolb's (2005) Learning Style Inventory-Version 3.1 (KLSI-3.1) was used to determine participants' learning styles. The KLSI-3.1 contains a series of 12 incomplete sentences followed by four possible ending phrases, and responses are used to place students into one of four categories of basic learning styles: accommodators, assimilators, convergers, or divergers. Accommodators rely on active experimentation and concrete experience to learn, assimilators learn through reflective observation and abstract conceptualization, convergers learn through the use of active experimentation and abstract conceptualization, and divergers learn primarily through reflective observation and concrete experience. A summary of students' learning styles is given in Table 4.1.

As can be seen from Table 4.1, of the 71 students who participated in the study, the majority were either convergers ($n = 31$) or assimilators ($n = 22$). This trend was consistent across both schools and for both semesters (UNA: $n_{\text{Total}} = 45$, $n_{\text{Con}} = 20$, $n_{\text{Assim}} = 12$; ASU: $n_{\text{Total}} = 26$, $n_{\text{Con}} = 11$, $n_{\text{Assim}} = 10$). Relative to the other two learning styles, the number of divergers ($n = 10$) and accommodators ($n = 8$) was nearly the same. Interestingly, there were twice as many divergers as accommodators during the fall term at each school but this trend was reversed during the spring term (UNA: $n_{\text{Div-Fall}} = 5$, $n_{\text{Con-Fall}} = 2$; $n_{\text{Div-Spring}} = 2$, $n_{\text{Con-Spring}} = 4$; ASU: $n_{\text{Div-Fall}} = 2$, $n_{\text{Con-Fall}} = 1$; $n_{\text{Div-Spring}} = 1$, $n_{\text{Con-Spring}} = 1$). However, given the

Table 4.1
*Summary of Students' Learning Styles from Kolb's (2005) Learning Styles Inventory-
 Version 3.1*

School/Term	Learning Style ^a								Overall
	ACC		ASSIM		CON		DIV		
	N	%	N	%	N	%	N	%	
UNA									
Fall	2	7.7	8	30.8	11	42.3	5	19.2	26
Spring	4	21.1	4	21.1	9	47.3	2	10.5	19
Total	6	13.3	12	26.7	20	44.4	7	15.6	45
ASU									
Fall	1	6.7	8	53.3	4	26.7	2	13.3	15
Spring	1	9.1	2	18.2	7	63.6	1	9.1	11
Total	2	7.7	10	26.7	11	42.3	3	11.5	26
Overall									
Fall	3	7.3	16	39.0	15	36.6	7	17.1	41
Spring	5	16.7	6	20.0	16	53.3	3	10.0	30
Total	8	11.3	22	31.3	31	43.7	10	14.1	71

Note. N = 71.

^aThe four learning styles are: ACC = Accommodators, who rely on active experimentation and concrete experience to learn; ASSIM = Assimilators, who learn through reflective observation and abstract conceptualization; CON = Convergers, who learn through the use of active experimentation and abstract conceptualization; and DIV = Divergers, who learn primarily through reflective observation and concrete experience.

relatively small sample sizes for these groups, any conclusions about this trend should be made with prudence.

Achievement

Student achievement scores were based on several researcher-developed assessments, including laboratory assignments, programming assignments, and unit exams. There were 15 laboratory assignments per session (open vs. closed lab-based) for a total of 30 overall. Each laboratory assignment was scored using the score card located in Appendix B. The three programming assignments per session were scored using the rubric in Appendix B. Two unit exams per session contained

a varying number of dichotomously scored questions. Points per session were the sum of laboratory points (maximum of 30), programming assignment points (maximum of 45) and points scored on each of the two unit exams (maximum of 200). The upper limit of points per session was 275 with upper scores indicating higher achievement. Tables 4.2A and 4.2B contain a summary of achievement means and ranges by school (Table 4.2A) and semester (Table 4.2B). As can be seen from Table 4.2A, students' overall mean achievement in the closed lab-based class ($M = 207.3$) was slightly higher than that of the students in the open lab-based class ($M = 204.6$), and there was considerably less variability in the scores for the closed lab-based class ($Range = 93-275$) compared to the open lab-based class ($Range = 73-275$). Comparing schools overall, UNA students' mean achievement was slightly higher than ASU students' mean achievement, but there was much greater variability in the UNA students' achievement scores (UNA: $M = 207.4$, $Range = 73-275$; ASU: $M = 203.4$, $Range = 126-271$). Comparing semesters overall (Table 4.2B), students in the fall term had a slightly higher mean achievement than those in the spring term, and there was considerably less variability in their scores (Fall: $M = 208.1$, $Range = 93-275$; Spring: $M = 202.1$, $Range = 73-275$). As shown in Table 4.3, these achievement differences between schools and between semesters were not significant. These results confirmed group equivalency relative to the study's primary dependent measure (i.e., achievement).

Table 4.2A

Achievement Results by School Location

Session	UNA			ASU			Overall		
	<i>N</i>	<i>M</i>	<i>Range</i>	<i>N</i>	<i>M</i>	<i>Range</i>	<i>N</i>	<i>M</i>	<i>Range</i>
Open	45	202.1	73–275	26	208.9	160–271	71	204.6	73–275
Closed	45	212.6	93–275	26	198.0	126–268	71	207.3	93–275
Overall	90	207.4	73–275	52	203.4	126–271	142	205.9	73–275

Note. Total sample size = 71 but data were coded for two repeated measures using criterion scaling via a multiple regression analysis strategy; thus, $N = 142$. Achievement was based on three separate researcher-constructed instruments. Scores reported reflect the total number of points earned; total possible points were 275.

Table 4.2B

Achievement Results by Semester

Session	Fall			Spring			Overall		
	<i>N</i>	<i>M</i>	<i>Range</i>	<i>N</i>	<i>M</i>	<i>Range</i>	<i>N</i>	<i>M</i>	<i>Range</i>
Open	41	213.2	125–275	30	192.9	73–271	71	204.6	73–275
Closed	41	204.3	93–275	30	211.3	126–275	71	207.3	93–275
Overall	82	208.1	93–275	60	202.1	73–275	142	205.9	73–275

Note. Total sample size = 71 but data were coded for two repeated measures using criterion scaling via a multiple regression analysis strategy; thus, $N = 142$. Achievement was based on three separate researcher-constructed instruments. Scores reported reflect the total number of points earned; total possible points were 275.

Table 4.3

t Test Comparisons of UNA vs. ASU and Fall vs. Spring Relative to Achievement and Attitude

	Achievement		Attitude ^a	
	<i>t</i> (140)	<i>p</i>	<i>t</i> (140)	<i>p</i>
UNA ($N = 90$) vs. ASU ($N = 52$)	-0.478	.6334	0.782	.4356
Fall ($N = 82$) vs. Spring ($N = 60$)	-0.822	.4123	-1.818	.0712

Note. ^aAttitude data had one outlier that was replaced with overall attitude mean.

Attitude

Attitude scores were acquired from Newby and Fisher's (1991) Attitude towards Computers and Computer Courses. There were 28 items scored on a 5-point Likert scale ranging from 1 = Strongly Disagree to 5 = Strongly Agree. Thus, scores could range from 28 to 140, with higher scores indicating more positive attitudes or perception. A summary of the attitude data is provided in Tables 4.4A, 4.4B, 4.5, and 4.6.

As can be seen from Table 4.4A, students' overall mean attitudes in the open lab-based class was slightly higher than the mean attitudes of the students in the closed lab-based class, but the variability of these scores was also higher for students in the open lab-based class than for the closed lab-based class (Open: $M = 114.3$, $Range = 52-140$; Closed: $M = 113.5$, $Range = 83-140$). Comparing schools overall, ASU students' mean attitude was slightly higher than UNA students' mean attitude, and the variability was also lower (ASU: $M = 115.6$, $Range = 81-140$; UNA: $M = 113.3$, $Range = 52-140$). Comparing semesters overall (Table 4.4B), students in the fall term had a more positive attitude than those in the spring term, and there was less variability in their attitude scores (Fall: $M = 116.8$, $Range = 83-140$; Spring: $M = 111.4$, $Range = 52-140$). As noted in Table 4.3 above, when a single outlier from the spring UNA group was either removed or replaced with the overall mean, differences in attitude scores between schools and between terms

Table 4.4A

Attitude Results by School Location

Session	UNA			ASU			Overall		
	<i>N</i>	<i>M</i>	<i>Range</i>	<i>N</i>	<i>M</i>	<i>Range</i>	<i>N</i>	<i>M</i>	<i>Range</i>
Open	45	111.8	52–140	26	116.5	81–140	71	114.3	52–140
Closed	45	114.9	83–140	26	114.9	84–139	71	113.5	83–140
Overall	90	113.3	52–140	52	115.6	81–140	142	113.5	52–140

Note. Total sample size = 71 but data were coded for two repeated measures using criterion scaling via a multiple regression analysis strategy; thus, $N = 142$. Attitudes were assessed by the Attitude towards Computers and Computer Courses instrument (Newby & Fisher, 1997), which is a 28-item Likert-scale. Scores could range from 28 to 140; higher scores reflect more positive attitudes.

Table 4.4B

Attitude Results by Semester

Session	Fall			Spring			Overall		
	<i>N</i>	<i>M</i>	<i>Range</i>	<i>N</i>	<i>M</i>	<i>Range</i>	<i>N</i>	<i>M</i>	<i>Range</i>
Open	41	117.3	83–140	30	111.4	52–140	71	114.3	52–140
Closed	41	116.2	83–140	30	111.3	84–140	71	113.5	83–140
Overall	82	116.8	83–140	60	111.4	52–140	142	113.5	52–140

Note. Total sample size = 71 but data were coded for two repeated measures using criterion scaling via a multiple regression analysis strategy; thus, $N = 142$. Attitudes were assessed by the Attitude towards Computers and Computer Courses instrument (Newby & Fisher, 1997), which is a 28-item Likert-scale. Scores could range from 28 to 140; higher scores reflect more positive attitudes.

were not significant. These results confirmed group equivalency relative to the study's second dependent measure (i.e., attitudes).

Table 4.5 contains an item analysis of students' responses to the 28 items from the ACCC. The mean and standard deviations are based on overall responses for the items, which were scored on a Likert scale ranging from 1 = Strongly Disagree to 5 = Strongly Agree. As can be seen from this table there was very little

Table 4.5

Summary of Results of Student Attitude towards Computers and Computer Courses

Statement ^a	Open ^b		Closed ^c	
	<i>M</i>	<i>SD</i>	<i>M</i>	<i>SD</i>
1. I do not think I will ever use what I learned in this class.	2.1	1.2	2.1	1.1
2. I feel comfortable when a conversation turns to computers.	3.9	1.0	3.8	1.0
3. Studying about computers is a waste of time.	1.4	0.7	1.5	0.8
4. It is fun to find out how computer systems work.	3.9	1.2	3.8	1.1
5. This class is providing me with skills I expect to use in the future.	4.0	1.1	4.0	1.0
6. I feel at ease when I am around computers.	4.0	1.1	4.0	1.0
7. My future career will require a knowledge of computers.	4.4	0.8	4.3	0.9
8. I enjoy using a computer.	4.4	0.9	4.4	0.8
9. This class is increasing my technical skills.	4.2	0.8	4.2	0.7
10. Working with a computer makes me very nervous.	1.9	1.1	1.9	1.0
11. I cannot imagine getting a job that does not involve using computers.	4.0	1.2	3.9	1.1
12. I think working with computers would be enjoyable and stimulating.	3.8	1.1	3.9	1.0
13. I am gaining few useful skills from this class.	2.8	1.3	2.9	1.3
14. I get a sinking feeling when I think about trying to use a computer.	1.8	1.0	1.8	1.0
15. Computers are an important factor in the success of a business.	4.4	0.9	4.4	0.8
16. The challenge of solving problems using a computer does not appeal to me.	2.3	1.1	2.4	1.1
17. The skills gained in this class are too specific to be generally useful in the future.	2.4	1.1	2.5	1.1
18. Computers make me feel uncomfortable.	1.7	0.9	1.8	1.1
19. The use of computers will increase in the future.	4.7	0.5	4.7	0.6
20. I would like to work with computers.	4.1	1.0	4.0	0.9
21. This class is helping develop my problem-solving skills.	4.0	1.0	4.0	0.8
22. Computers make me feel uneasy and confused.	1.8	1.0	1.9	1.0
23. All college students need a course about using computers.	4.2	0.8	4.3	0.8
24. I enjoy learning on a computer.	4.1	0.9	4.1	0.9
25. As a result of this class I feel confident about tackling unfamiliar problems involving computers.	3.6	1.1	3.7	0.9
26. I feel aggressive and hostile towards computers.	1.9	1.0	2.0	1.1
27. Knowledge of the use of computers will help me get a job.	4.4	0.6	4.4	0.7
28. Learning about computers is boring.	1.8	0.9	1.7	1.0

Note. $N = 71$. Attitudes were measured using Newby and Fisher's (1997) Attitude towards Computers and Computer Courses (ACCC) instrument, which used a Likert-scale of 1 = Strongly Disagree to 5 = Strongly Agree. Attitude scores ranged from 28 to 140 with higher numbers reflecting more positive attitudes.

^aStatements 2, 6, 10, 14, 18, 22, 26 measured "anxiety"; Statements 4, 8, 12, 16, 20, 24, 28 measured "enjoyment"; Statements 3, 7, 11, 15, 19, 23, 27 measured "perceived usefulness of computers"; and Statements 1, 5, 9, 13, 17, 21, 25 measured "perceived usefulness of course." Statements 1, 3, 10, 13, 14, 16-18, 22, 26, 28 were negatively worded but results shown in table do not reflect reverse scoring. ^bOpen refers to students' attitudes assessed at the end of the open lab-based class. ^cClosed refers to students' attitudes assessed at the end of the closed lab-based class.

difference between the mean attitude scores for the open and closed lab-based classes relative to each item. Students' overall attitudes did not change during the course of this study: they had generally positive attitudes toward computers and computer courses and with few exceptions, their attitudes were positive over the entire term irrespective of the type of class structure or their success or failure in the course.

The highest positive score on the attitude scale, on average, was in agreement with Statement 19: "The use of computers will increase in the future" ($M_{\text{both}} = 4.7$). Students also were inclined to agree with Statements 7, 8, 15, and 27: "My future career will require a knowledge of computers" ($M_{\text{Open}} = 4.4$, $M_{\text{Closed}} = 4.3$), "I enjoy using a computer" ($M_{\text{both}} = 4.4$), "Computers are an important factor in the success of a business" ($M_{\text{both}} = 4.4$), and "Knowledge of the use of computers will help me get a job" ($M_{\text{both}} = 4.4$). Students tended to strongly disagree with Statements 3, 22, and 28: "Studying about computers is a waste of time" ($M_{\text{Open}} = 1.4$, $M_{\text{Closed}} = 1.5$), "Computers make me feel uncomfortable" ($M_{\text{Open}} = 1.8$, $M_{\text{Closed}} = 1.9$), and "Learning about computers is boring" ($M_{\text{Open}} = 1.8$, $M_{\text{Closed}} = 1.7$). However, because these statements were negatively worded, these low scores actually reflect a higher positive attitude when they are reversed scored for analysis purposes. It is also interesting to note that there were very few items on which students were "neutral" (Likert score of 3). The only item that stands out is Statement 13, "I am gaining few useful skills from this class" ($M_{\text{Open}} = 2.8$, $M_{\text{Closed}} =$

2.9). Even though this was a negatively worded question, when reversed score the mean response was still around 3, indicating a neutral attitude.

Table 4.6 provides additional information relative to the four subscales of the ACCC: anxiety, enjoyment, perceived usefulness of computers, and perceived usefulness of course. As can be seen from this table the mean attitude score on each of the subscales was identical for the open lab-based class session and the closed lab-based session. An interesting result was the comparison of the scores between the two perceived usefulness subscales. Although the perceived usefulness of computers subscale had the highest mean of the four subscales ($M = 4.4$), the perceived usefulness of course subscale had the lowest mean ($M = 3.8$) among the four subscales. Nonetheless, the means of all four subscales were near 4, indicating that students had a highly positive attitude relative to each subscale irrespective of the type of class structure (open lab-based vs. closed lab-based).

Inferential Statistics

Overview

The purpose of this study was to assess the effect of classroom restructuring on student achievement in a 1st-year computer science course (CS1) and on their attitudes toward computers and computer courses. Functional sets were used to represent the multiple dependent and independent variables of this study. Set A was the set of student attributes and consisted of dummy-coded variables $X_1 = \text{gender}$

Table 4.6
*Students' Mean Subscales Scores for Attitude towards
 Computers and Computer Courses*

Group	Subscales ^a				Overall
	A	E	PU-Comp	PU-Crs	
Open lab	4.1	4.0	4.4	3.8	4.1
Closed lab	4.1	4.0	4.4	3.8	4.1
Overall	4.1	4.0	4.4	3.8	4.1

Note. $N = 71$. Attitudes were measured using Newby and Fisher's (1997) Attitude towards Computers and Computer Courses (ACCC) instrument, which was based on a Likert-scale of 1 = Strongly Disagree to 5 = Strongly Agree.

^aThe subscales consisted of A = anxiety, E = enjoyment, PU-Comp = perceived usefulness of computers, PU-Crs = perceived usefulness of course.

and X_2 = previous computer programming course; the dummy-coded math background variables M_1 = Math-1 (Trigonometry), M_2 = Math-2 (Calculus I), M_3 = Math-3 (Above Calculus I), with College Algebra (or its equivalent) as the reference group; and the dummy-coded learning styles variables L_1 = Learning Style-1 (Accommodators), L_2 = Learning Style-2 (Assimilators), and L_3 = Learning Style-3 (Divergers), with Convergents as the reference group. Set B contained contrast coded variables for academic characteristics: S_1 = School, S_2 = Term, and S_3 = the interaction between Term and School. Set C was the set of subject and treatment factors and consisted of X_3 = treatment, and the two criterion-scaled variables: ST_1 = sum of each student's achievement scores and ST_2 = sum of each student's attitude scores, which were used as part of the repeated measures analysis. Finally, Set D, the dependent variable set, consisted of Y_1 = achievement scores and Y_2 = attitude scores. Set membership is summarized in Chapter 3 (see Table 3.7).

The statistical strategy used to analyze the study's data was multivariate analysis of variance (MANOVA) via a hierarchical multiple regression approach. MANOVA is an omnibus strategy used for analyzing data involving multiple dependent measures. Given the nature of this study's research design (i.e., within-subjects repeated measures), criterion scaling was used to facilitate data analysis. This involved maintaining a separate set of data for each repeated performance for each student. According to Cohen and Cohen (1983, p. 428):

There is another class of research designs in which each of the n subjects gives rise to a Y observation under each of c (> 1) different conditions (C). The data layout has n rows, c columns, and hence a total of $N = nc$ Y observations.

As a result, given $n = 71$ subjects and $c = 2$ treatments, data analysis for this study was based on a total of $N = 71 \times 2 = 142$ observations.

Preliminary Analysis

Several preliminary analyses were conducted prior to testing the study's hypotheses to make certain the data set was proper and "clean." These analyses included missing data analysis, outlier analysis, and regression assumptions analysis.

Missing Data Analysis

Missing data, an essential consideration in any research study, can arise in two ways: missing values on variables resulting from mortality, and missing values on independent variables due to incomplete or contradictory responses on data

collection instruments. At the beginning of the fall and spring terms, data were collected from a total of 91 students when the ACCC instrument was administered. At the end of the fall and spring terms a total of 71 students completed all study protocols; the remaining 20 students (22%) either dropped the course or did not complete all instruments. According to Cohen et al. (2003, p. 433):

When the pattern of missing data is such that they occur exclusively for a proportion of subjects (P_a), the analyst may opt to drop these subjects and perform the analysis on the remainder of the sample, a practice called *list-wise deletion* in computer programs. If P_a is small enough and the n is large enough there can hardly be a material difference between the results obtained with these subjects dropped and those which would have been obtained from all cases.

In the current study, the percentage of students with missing data was high (22%) and the overall initial sample size was relatively small ($N = 91$) and hence, Cohen et al.'s (2003) guidelines for implementing a list-wise deletion strategy were not met. However, because these 20 students either dropped the course or did not complete all assessment protocols, maintaining their presence in the data set or imputing missing data values was also not appropriate. To help inform the decision of whether to drop or not drop the 20 students from whom data were missing, two additional considerations were made. The first consideration was to investigate how the missing data affected group membership. Given the nature of this study (i.e., repeated measures), all students experienced both treatments and hence there was

Table 4.7
Missing Data Results

	UNA			ASU			Overall		
	<i>N</i>	<i>n</i>	Loss	<i>N</i>	<i>n</i>	Loss	<i>N</i>	<i>n</i>	Loss
Fall	36	26	28%	18	15	17%	54	41	24%
Spring	25	19	24%	12	11	8%	37	30	23%
Overall	61	45	26%	30	26	13%	91	71	22%

an equal number of students on whom data were missing from each group. The second consideration was to examine the effect dropping 20 students would have on power. Although reducing the sample size also reduces power, a reduction in *N* from 91 to 71 still resulted in an acceptable level of power (see Table 3.3). As a result, the 20 students from whom data were missing were dropped from the study and final analyses were conducted using a sample size of 71 participants. Further details on missing data are provided in Table 4.7.

Outlier Analysis

Outliers are unusually low or high data values in comparison to the rest of the data set. Outliers can occur in two ways: contamination of the data set by data coding errors or from valid but rare cases that are unusual observations in the population (e.g., an 80-year old student). Outliers can produce results in a regression analysis that are skewed toward the unlikely cases. Outlier analysis provides for the identification and possible removal of these rare cases and also provides for recognition and correction of contaminated observations. According to Stevens (2002), outliers that are valid cases should not be dropped but two

analyses should be reported: one that includes the outliers and one that does not.

This process makes for a more valid analysis.

There are several strategies available for detecting outliers that often give conflicting results. Five possible outliers in achievement and one in attitude were detected using four of these strategies: Mahalanobis Distance (Cohen et al., 2003, p. 398), Jackknife Distance (Faraway, 2005, p. 125), ± 2.65 standard deviations from the mean, and residual analysis. The possible achievement outliers were examined and it was determined that they were true cases representing students who remained in the class but scored extremely low on some of the assessments. The attitude outlier was examined and it was determined that it also represented a true case. Analyses were performed with and without the outliers and the results were almost identical (i.e., removal of the outliers did not significantly change the results). Thus, the decision was made to keep the outliers.

Regression Assumptions

Multivariate analysis of variance (MANOVA) via hierarchical multiple regression was used for the inferential statistical analyses of the study's data. This statistical strategy has several underlying assumptions that must be satisfied (Cohen et al., 2003): linearity of the relationship, relevant choice of IVs, reliable measurement of IVs, equality of variance of the residuals (homoscedasticity), independence of residuals, and normality of their distribution. If any of these assumptions is violated, the end result could be an incorrect statistical analysis

based on a sample that did not accurately embody the characteristics of the population. Violations of these assumptions can also yield distorted estimates of regression coefficients or of the standard error. In the latter case the regression coefficients would be correct but the confidence intervals for hypothesis testing would be incorrect. Further explanation of these assumptions and the techniques used to certify their compliance in this study follows.

Linearity. Regression analysis is a procedure used to estimate the linear relationships between variables. If the data are related in a nonlinear manner, conventional univariate regression analysis will underestimate these relationships and multivariate regression analysis will produce unusable results. A residual analysis was conducted in which the residuals were plotted against the predicted values to confirm the multivariate linearity of the dataset. No discernable nonlinear pattern was displayed on the resulting graphs, one for each dependent measure demonstrated. Thus, the multivariate linearity assumption was fulfilled.

Correct specification of the IVs. The correct specification of the IVs assumption implies that the independent variables of the study are correctly contained in the model and the independent variables and residuals are in fact independent in the population. When a regression model is specified, the researcher must attempt to incorporate all the IVs that correlate to the study's hypotheses and use theory and previous research as guidelines to ascertain which IVs to include in

the model. If the correct specification of the IVs assumption is violated, it can produce significant misinterpretations of results.

The initial choice of IVs was based on extensive review of the literature. Additionally, added variable plots (AVPs) for each independent variable were investigated to verify this assumption. Trend lines from the AVPs were used to classify the variables as correctly specified in the model (i.e., with a non-zero slope line) or not (i.e., with a slope of zero). Trend lines from this analysis indicated that Previous Computer Programming Course, Math Experience-1 (Trigonometry), Math Experience 2 (Calculus I), Math Experience-3 (above Calculus I), Learning Style-2 (accommodators), Learning Style-3 (Divergers), and Academic-1 (school) had positive slopes. Gender and Academic-2 (Term) had negative slopes. Learning Style-1 (Assimilators) had a zero slope and Academic-3 (the interaction between school and semester) had a near-zero (slightly positive) slope. Even though the mean independence assumption was not fully satisfied for the last two variables, they remained in the overall model. The inclusion of these variables will be further discussed in Chapter 5.

Measurement error specification. Multiple linear regression assumes that each independent variable in the regression equation has been measured without error. When this assumption of perfect reliability is violated, statistics from the analysis will be biased and it is impossible to know if the values of a particular measure are too low, too high, or even just right. Reliability of a measurement tool

is an indication of the measurement error of the variables measured by that tool. While in practice there is no perfect reliability, Cohen et al. (2003) recommend a reliability coefficient greater than .66.

In the current study, the measurement tools included a researcher-constructed student survey that was used to collect student attribute data (i.e., gender, computer experience, and math background), student learning styles, an attitude scale, and researcher-constructed achievement instruments (i.e., laboratory assignments, programming assignments, and unit exams). The self-reported student attribute data were checked against university records and found to be consistent. Thus, the variables corresponding to these data had near perfect reliability. Kolb's (2005) Learning Styles Inventory-Version 3.1 has a reported reliability alpha of between .77 and .90. Thus, the variables corresponding to learning styles were measured with acceptable measures of reliability. Sample attitude data collected from the current study had an overall reliability alpha of .95, and the respective subscale alphas were .82, .94, .77, and .92. Thus, the corresponding attitude variable was measured with an acceptable reliability coefficient. Reliability indices for the four unit exams were respectively .74, .83, .70, and .82, and an inter-rater reliability of .93 was maintained for the lab and programming assignments. Thus, the corresponding achievement variable had acceptable measures of reliability. As a result, the measurement error specification assumption was satisfied.

Homoscedasticity of residuals. This assumption means that the variance of the residuals around the calculated regression line remains constant regardless of the value of the independent variable X . If this assumption is violated, then the statistics from the regression analysis will be incorrect. If a non-constant variance is indicated, separate models for different ranges of X might be necessary. Violations of this assumption can be detected using the same scatterplot used for the linearity assumption (i.e., residual vs. fitted). Further inspection of this plot revealed no obvious pattern, which implies that the homoscedasticity of the residuals assumption was satisfied.

Independence of residuals. Multiple regression assumes that the residuals of the observations are independent. Violations of this assumption often occur when data are collected from groups of individuals in which the participants from each group are more similar to each other than to members of other groups. It can also occur when multiple observations are made of a single subject or a small group of subjects over time and the change in the observations is systematic. When residuals are dependent, significance tests and standard errors are incorrect. The nature of this research (i.e., repeated measures) could lend itself to dependence of the residuals. A plot of the residuals versus the case numbers was used to detect violations of this assumption. The plot was inspected for any systematic pattern that might indicate dependence. No such pattern was found and thus, the residuals were assumed to be independent.

Normality. In order to meet this assumption it is necessary that the residuals have a normal distribution for any value of the independent variables. Three procedures were performed to determine if this study's data were compliant: A histogram of the standardized residuals was plotted and the result was a slightly skewed normal distribution; a univariate plot of the residuals was overlaid with a kernel density line in which the residuals adhered closely to the superimposed line; and a normal $q-q$ plot of the residuals with a superimposed straight line and a 95% confidence band was examined and the residuals closely followed the superimposed line and all of the data fell within the confidence band. The results of these three tests imply the normality of residuals assumption was met for the data set of this study.

In summary, as a product of the aforementioned results, it was established that the study data were compliant with the all assumptions required for multiple linear regression models. This made it possible to advance to the next step, analyzing the data via a multiple regression strategy.

MANOVA Analysis

Overview

Inferential statistical analysis was performed using multivariate analysis of variance (MANOVA) to test the null hypotheses outlined in Chapter 1. MANOVA is an adaptation of the univariate analysis of variance (ANOVA) designed for studies that contain sets of multiple dependent variables. This statistical strategy

was appropriate because the study's data set involved more than one dependent measure and multiple independent variables. By employing a MANOVA, all of these issues were addressed while protecting against inflated Type I and Type II error rates. In association with MANOVA, a hierarchical multiple regression strategy was employed using sets of independent variables.

A coding scheme known as criterion scaling also was used to accommodate the repeated measures aspect of the analysis. This coding scheme uses each student's total score on a dependent measure (i.e., the sum of each subject's score from both treatments) to code subjects. For example, if a student's achievement score was 150 from participating in the open lab-based class and 210 from participating in the closed lab-based class, then the data point $150 + 210 = 360$ appears twice in the data set to reflect when the student participated in each treatment. These variables were labeled ST_1 for achievement and ST_2 for attitude (see Table 3.7).

Testing the MANOVA Model for Significance

The MANOVA model was tested for significance using JMP Version 6. As reported in Table 4.8, the full model was significant via each of the four test statistics commonly used for MANOVA: Wilks' $\Lambda = .27$, $F_{26, 254} = 9.02$, $p < .0001$; Pillai's Trace = .84, $F_{26, 256} = 7.19$, $p < .0001$; Hotelling-Lawley = 2.28, $F_{26, 252} = 11.04$, $p < .0001$; and Roy's Max Root = 12.07, $F_{13, 128} = 20.42$, $p < .0001$. (Throughout the remaining narrative, only Wilks' Lambda will be reported.)

Table 4.8

Overall MANOVA Results

Test	Value	Approx. <i>F</i>	<i>df</i>	<i>p</i>
Wilks' Lambda	.27	9.02	26, 254	< .0001**
Pillai's Trace	.84	7.19	26, 256	< .0001**
Hotelling-Lawley	2.28	11.04	26, 252	< .0001**
Roy's Max Root	2.07	20.42	13, 128	< .0001**

Note. $N = 142$: total sample size = 71 with data coded for two repeated measures using criterion scaling via a multiple regression analysis strategy.

** $p < .01$.

Table 4.9 contains a summary of the MANOVA results for each independent variable. Observe from Table 4.9 that the overall MANOVA resulted in two significant variables, ST_1 ($\Lambda = 1.342$, $F_{2, 126} = 84.58$, $p < .0001$) and ST_2 ($\Lambda = .584$, $F_{2, 126} = 36.79$, $p < .0001$). As noted above, these two variables represented subject codes and consisted of the sum of each subject's achievement on the closed lab based class and the open lab-based class (i.e. the sum of scores acquired from each repeated performance) and the sum of each subject's attitude at the conclusion of the closed lab-based class and the open lab-based class, respectively. Although none of the primary factors were significant when assessed relative to the two dependent measures simultaneously, the subject code variables were, thus follow-up analyses were performed.

According to Ellis and Haas (1987), the protected univariate F test is the most commonly used MANOVA follow-up procedure. Based on the logic of Fisher's protected t concept to control for inflated Type I error rates, this procedure

Table 4.9

MANOVA Results for Independent Variables

Source Variance	Wilks' Λ	df	F	p
Set A (Student Attributes)				
X_1 = Gender ^a	0.000	2, 126	0.01	.99
X_2 = Previous Computer Programming Course ^b	0.020	2, 126	1.28	.28
M_1 = Math Background-1 ^c	0.000	2, 126	0.01	.98
M_2 = Math Background-2 ^c	0.035	2, 126	2.20	.12
M_3 = Math Background-3 ^c	0.023	2, 126	1.42	.25
L_1 = Learning Style-1 ^d	0.012	2, 126	0.74	.48
L_2 = Learning Style-2 ^d	0.005	2, 126	0.32	.73
L_3 = Learning Style-3 ^d	0.003	2, 126	0.17	.85
Set B(Academic Attributes)				
S_1 = School/Semester-1 ^e	0.003	2, 126	0.19	.83
S_2 = School/Semester-2 ^e	0.023	2, 126	1.55	.22
S_3 = School/Semester-3 ^e	0.014	2, 126	0.87	.42
Set C(Treatment + Subjects)				
X_3 = Treatment ^f	0.005	2, 126	0.31	.73
ST_1 = Subjects Total-Achievement ^g	1.342	2, 126	84.58	< .0001**
ST_2 = Subjects Total-Attitude ^h	0.584	2, 126	36.79	< .0001**

Note. $N = 142$: total sample size = 71 with data coded for two repeated measures using criterion scaling via a multiple regression analysis strategy.

^aGender was dummy coded with females as the reference group. ^bPrevious Computer Programming Course was dummy coded with no previous computer programming course as the reference group.

^cMath Background-1, Math Background-2, and Math Background-3 = dummy-coded factors representing respectively, Trigonometry, Calculus I, or Above Calculus I as the highest level of math course attained, with College Algebra as the reference group. ^dLearning Style-1, Learning Style-2, and Learning Style-3 = dummy-coded factors representing respectively Kolb's (2205) learning styles, Accommodators, Assimilators, and Divergers, with Convergents as the reference group.

^eSchool/Semester-1, School/Semester-2, and School/Semester-3 are contrast coded variables representing respectively UNA vs. ASU, Fall vs. Spring, and the interaction between school and term. ^fTreatment = dummy-coded factors representing the two class structures, Open-lab vs. Closed lab, with Open lab as the reference group. ^g ST_1 = the sum of each subject's achievement on the closed lab-based class and the open lab-based class (i.e. the sum of scores acquired from each repeated performance). ^h ST_2 = The sum of each subject's attitude at the conclusion of the closed lab-based class and the open lab-based class.

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

allows the analysis of a separate, univariate ANOVA for each dependent variable only if a significant MANOVA model has been observed. One drawback of this type of follow-up procedure is the loss of capacity to examine the links between the

dependent variables. Although the two dependent variables in this study, achievement and attitude, were somewhat correlated, $r = .377$, this was expected and is reasonable because computer science students who display higher achievement in CS 1 are expected to have positive attitudes toward computers and computer science courses. As a result, there was no need to examine this relationship between the two dependent measures and hence separate univariate ANOVAs, one each for achievement and attitude, were conducted via a hierarchical multiple regression correlation (MRC) as the follow-up course of action. A brief discussion for each follow-up analysis ensues.

Y₁ (Achievement) Follow-Up

A follow-up univariate hierarchical regression analysis was performed for achievement with the set order entry of A–B–C. This analysis yielded an overall R^2 of .668, which indicates that 66.8% of the variability in achievement scores was explained by collective variables contained in Sets A, B, and C; this was significant, $F_{13, 128} = 19.8, p < .0001$. Based on the logic of Fisher's protected t test, which guards against inflated Type I error rates, follow-up analyses of the individual sets were appropriate because the overall model was significant. Table 4.10 contains a summary of those analyses. Note from Table 4.10 that the contribution Set A made was significant, $R^2 = .193, F_{8, 133} = 3.96, p = .0003$, as was the unique contribution Set C made in the presence of Set A and Set B, $sR^2 = .434, F_{2, 128} = 83.66, p < .05$. Thus, inspection of the individual factors within Set A and

Table 4.10

Summary of Follow-up Univariate Hierarchical Regression Analysis for Achievement

Set in Model	Cumulative R^2	I	df_1	F_1	p_1
Set A	.193	.193	8, 133	3.96	.0003**
Set B	.234	.041	3, 130	2.34	> .05
Set C	.668	.434	2, 128	83.66	< .05*

Note. $N = 142$: total sample size = 71 with data coded for two repeated measures using criterion scaling via a multiple regression analysis strategy. I = Increment in R^2 . Set A = Student attributes (gender, previous computer programming education, math background, and learning styles). Set B = Academic attributes (school, term, school-term interaction). Set C = Subjects Total (treatment vs. control and the sum of each subject's achievement on the closed lab based class and the open lab based class).

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

Set C for statistical significance was allowed. The unique contribution of Set B in the presence of set A was not significant, $sR^2 = .041$, $F_{3, 130} = 2.34$, $p > .05$, and thus no further analysis of the individual factors within Set B was conducted relative to achievement. This implies that none of the academic attributes (UNA vs. ASU, Fall vs. Spring, the interaction between school and term) accounted for a significant amount of variability in achievement scores.

A summary of the follow-up analyses of Set A and Set C relative to achievement is given in Table 4.11. Only one research factor in Set A, Math Background, which represented the highest level of math courses taken by students, made a significant contribution to the variability of achievement scores, $sR^2 = .18$, $F_{3, 136} = 8.84$, $p < .01$. This factor was comprised of the variables M_1 , M_2 , and M_3 , which respectively represented the pairwise comparisons of Trigonometry vs. College Algebra, Calculus I vs. College Algebra, and Above Calculus I vs. College

Table 4.11
 Summary of Follow-up Tests Relative to $Y_1 = \text{Achievement}^a$

X_i	B_i	Error ^b	Cum. R^2	I	df	F_I
Set A^c						
Gender ^d	-.705	8.53	.016	.016	1, 140	0.68
Prev Comp Prog Cr ^e	12.23	7.50	.020	.004	2, 139	2.66
Math Bkgd ^f			.183	.163	3, 136	8.84**
M_1	13.55	10.98				1.52
M_2	50.02	11.44				19.13**
M_3	32.73	9.91				10.91**
Learning Style ^g			.193	.010	3, 133	0.55
Set C^h						
Treatment	2.69	8.00	.001	.001	1, 70	0.17
Subjects	0.50	0.30	.668	.667	70, 140	280.33**

Note. $N = 142$: total sample size = 71 with data coded for two repeated measures using criterion scaling via a multiple regression analysis strategy. I = increment.

^aOnly sets A and C are reflected in this table because those were the only sets found to be significant in the univariate analysis relative to achievement (see Table 4.10). ^bStandard error of the mean ^cSet A = Student Attributes. ^dGender = dummy coded variable X_1 with females as the reference group. ^ePrev Comp Prog Cr = Previous Computer Programming Course, which was a dummy coded variable, X_2 , with no previous computer programming course as the reference group. ^fMath Bkgd = dummy-coded variables M_1 , M_2 , and M_3 representing the highest level of math course attained: College Algebra, Trigonometry, Calculus I, or Above Calculus I; the reference group was College Algebra. M_1 reflects the Trig-College Algebra comparison. M_2 reflects the Calc I- College Algebra comparison. M_3 reflects the Above Calc I-College Algebra comparison. ^gLearning Style= dummy-coded variables L_1 , L_2 , and L_3 representing learning styles as classified by Kolb's (2005) Learning Style Inventory-3.1: Convergents, Accommodators, Assimilators, and Divergers; the reference group was Convergents. L_1 reflects Convergents-Accommodators comparison. L_2 reflects Convergents-Assimilators comparison. L_3 reflects Convergents-Divergers comparison. ^hSet C = Subject + Treatment effect. The subject effect is the sum of each subject's achievement at the conclusion of the closed lab-based class and open lab-based class. The treatment effect is the comparison between open lab-based class and closed lab-based class.

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

Algebra. In light of the significance of the contribution of Math Background, a follow-up analysis was warranted. As shown in Table 4.11, the unique contributions of $M_2 = \text{Calculus I vs. College Algebra}$ ($F = 19.13, p < .01$) and $M_3 = \text{Above Calculus I vs. College Algebra}$ ($F = 10.91, p < .01$) accounted for a

Table 4.12

Summary of Follow-Up of the Independent Variable "Math Background"

Variable	<i>B</i>	<i>t</i>	<i>p</i>
M_1 = Math Background-1 ^a	16.5	1.584	.1155
M_2 = Math Background-2 ^a	51.5	4.690	< .0001**
M_3 = Math Background-3 ^a	35.1	3.779	.0002*

Note. $N = 142$.

^aMath Background-1, Math Background-2, and Math Background-3 = dummy-coded factors representing respectively, Trigonometry, Calculus I, or Above Calculus I as the highest level of math course attained, with College Algebra as the reference group.

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

significant amount of variability in achievement scores. More specifically, students whose math background included up to Calculus I averaged 50 points more on their overall achievement scores than those students with only a College Algebra background. Similarly, students whose math background included courses above Calculus I averaged 33 points more on their overall achievement scores than those with only a College Algebra background. The Math Background factor was also examined in the absence of all other attributes. The results of this analysis are given in Table 4.12. As can be seen from Table 4.12, M_2 and M_3 were still the only Math Background variables that made a significant contribution to explaining the variability in achievement scores, and their respective regression coefficients (B_i) corresponded to those reported above in the presence of other attributes.

In Set C only the Subjects variable (ST_1) was significant relative to achievement, $sr^2 = .667$, $F_{70, 140} = 280.33$, $p < .0001$ (see Table 4.11). As noted

earlier, this variable was the sum of each subject's achievement at the conclusion of the closed lab-based class and open lab-based class and was created to facilitate the regression analysis of the study's repeated measures design. More importantly, though, after the effect of the Subjects variable was removed, the treatment variable, also a member of Set C, was not significant ($F_{1, 70} = .17, p > .05$). The combined effect of these two results reflects the extent to which subject differences accounted for a significant amount of variability in a dependent measure (in this case, achievement scores). Because of the way these variables were coded, the repeated measures analysis via a multiple regression strategy partialled subject differences from the treatment effect. In so doing, results indicated that group membership (open vs. closed lab-based class structure) was not significant relative to achievement but subject differences accounted for a large percentage of the variability in achievement scores. This speaks to the robustness of the repeated measures design: it enables researchers to separate subject differences from treatment effects. If this had not been done, the treatment and subject differences would have been combined as a single treatment effect and the analysis would have revealed that "treatment" was significant. In summary, the only research factor that had a significant effect on achievement was students' math background.

Y₂ (Attitude) Follow-Up

A follow-up univariate hierarchical regression analysis was conducted for attitude with the set order entry of A-B-C. This analysis yielded an overall R^2 of

Table 4.13

Summary of Follow-up Univariate Hierarchical Regression Analysis for Attitude

Set in Model	Cumulative R^2	I	df_1	F_1	p_1
Set A ^a	.09	.09	8, 133	1.67	.11
Set B ^b	.11	.02	3, 130	0.97	> .05
Set C ^c	.44	.33	2, 128	37.71	< .05*

Note. $N = 142$: total sample size = 71 with data coded for two repeated measures using criterion scaling via a multiple regression analysis strategy. I = Increment in R^2 .

^aSet A = Student attributes (gender, programming experience, math background, and learning styles). ^bSet B = Academic attributes (school, term, school-term interaction). ^cSet C = Treatment + Subjects (treatment vs. control and the sum of each subject's attitude at the conclusion of the closed lab-based class and the open lab-based class).

* $p < .05$.

.44, which indicates that 44% of the variability in attitude scores was explained by the collective variables contained in Sets A, B, and C; this was significant, $F_{13, 128} = 7.72, p < .0001$. Because the overall attitude model was significant, follow-up analyses of the individual sets within the model were warranted based on the Fisher protected t test concept. Table 4.13 contains a summary of this analysis. Observe from Table 4.13 that the contributions of Sets A and B were not significant in explaining the variability in students' attitude scores (Set A: $R^2 = .09, F_{8, 133} = 1.67, p = .11$; Set B: $sR^2 = .02, F_{3, 130} = .97, p > .05$). Thus, no further analyses of the individual factors within Set A or Set B were warranted. The contribution Set C made in the presence of Set A and Set B was significant, $sR^2 = .44, F_{2, 128} = 37.71, p < .05$. Thus, an analysis of the individual research factors within Set C for statistical significance was performed.

Table 4.14
Summary of Follow-up Tests Relative to $Y_2 = \text{Attitude}^a$

X_i	Coefficient	Error ^b	Cum. R^2	I	df	F_1
Set C^c			.378	.378	2, 139	42.26**
Treatment	1.34	2.90	.001	.001	1, 70	0.17
Subjects	0.33	0.04	.378	.377	70, 140	84.59**

Note. $N = 142$: total sample size = 71 with data coded for two repeated measures using criterion scaling via a multiple regression analysis strategy. I = increment.

^aOnly Set C is reflected in this table because it was the only set found to be significant in the univariate analysis relative to attitude (see Table 4.13). ^bError = standard error of the mean.

^cSet C = Subject + Treatment effect. The subject effect is the sum of each subject's attitude at the conclusion of the closed lab-based class and open lab-based class. The treatment effect is the comparison between open lab-based class and closed lab-based class.

** $p < .01$.

A summary of the follow-up analysis of Set C relative to attitude is located in Table 4.14. Similar to the achievement model, Set C represented the coding for treatment and subjects. The Treatment variable represented group membership (open vs. closed lab-based class), and the Subjects variable was the sum of each subject's attitude scores at the conclusion of the closed lab-based class and open lab-based class. Observe from Table 4.14 that the Subjects variable was significant relative to attitude, $sr^2 = .377$, $F_{70, 140} = 84.59$, $p < .01$, but the Treatment variable was not significant, $sr^2 = .001$, $F_{1, 70} = 0.17$, $p > .05$, after the effect of Subjects was removed. As noted earlier, the combined effect of these two results reflects the extent to which subject differences accounted for a significant amount of variability in a dependent measure (in this case, attitude scores). Because of the way these variables were coded, the repeated measures analysis via a multiple regression strategy partialled subject differences from the treatment effect. In so doing, results showed that group membership was not significant relative to attitudes but subject

differences accounted for a large percentage of variability in attitude scores. In summary, none of the study's research factors had a significant effect on students' attitudes toward computers and computer courses.

Attribute-Treatment Interaction (ATI) Analyses

Two separate attribute-treatment interaction (ATI) analyses involving the interaction between student attributes and treatment were conducted: one was relative to achievement and the second was relative to attitudes. These analyses were done independently of the overall MANOVA model, or its MANCOVA counterpart, because it involved the interactions of research factors that were not considered covariates. Both analyses were performed using hierarchical multiple regression with the entry order of student attributes, treatment, and the set of variables representing the interaction between these factors. The results of the ATI achievement model was significant, $R^2 = .274$, $F_{17, 124} = 2.75$, $p = .0007$, but a follow-up analysis revealed that the increment the interaction set made to explaining the variability in achievement scores was not significant, $sr^2 = .081$, $F_{8, 124} = 1.73$, $p > .05$. The results of the ATI attitude model also was not significant, $R^2 = .123$, $F_{17, 124} = 1.03$, $p = .4342$, and hence no follow-up analyses were conducted relative to this model.

Supplemental Analysis

Due to concerns about treatment fidelity at ASU, a supplemental analysis was performed using only data from UNA, which was the author's home school;

the results were then compared to those of the full data set. This analysis was conducted independently of the study's primary analyses and was done so for personal interest and information purposes only. The reader is cautioned that any results from this analysis are subject to inflated alpha levels and hence should be interpreted accordingly.

A summary of this comparison is given in Table 4.15. Note from this table that the UNA-only results were nearly identical to those from the full model:

1. The UNA-only MANOVA was significant, Wilks' $\Lambda = .07$, $F_{24, 152} = 18.47$, $p < .0001$, and the significance was relative to the same two variables of the full model, namely, ST_1 ($\Lambda = 1.383$, $F_{2, 76} = 52.55$, $p < .0001$) and ST_2 ($\Lambda = 2.45$, $F_{2, 76} = 93.19$, $p < .0001$).
2. The univariate analyses of the UNA-only data for achievement yielded a significant overall model and follow-up analyses confirmed that Sets A and C were significant. Furthermore, Set A's significance was due to the same Math Background factor as in the full data set (Calculus I vs. College Algebra and Above Calculus I vs. College Algebra), and Set C's significance was due to subject differences, not treatment. The corresponding regression coefficients for the math background comparisons were also nearly the same for both the full model and the UNA-only model (Full model: $B_{M_2} = 52.02$, $B_{M_3} = 32.73$; UNA-only model: $B_{M_2} = 62.7$, $B_{M_3} = 43.3$).

Table 4.15

Summary of Results Comparing the Full Dataset (UNA and ASU) to UNA Data Only

Model	Full Dataset			UNA Data Only				
	Statistic	Value	df	F	Statistic	Value	df	F
MANOVA	Λ	.27	26, 254	9.02**	Λ	.070	24, 152	18.47**
Univariate Achievement	R^2	.668	13, 128	19.82**	R^2	.725	11, 78	18.65**
<u>Set A^a</u>	R^2	.193	8, 133	3.96**	R^2	.268	8, 81	3.70**
• Math Background ^b	sR^2	.163	3, 136	8.84*	sR^2	.229		8.66*
– M_2^c	B	50.02 ^d		19.13**	B	62.65 ^d		16.97**
– M_3^e	B	32.73 ^d		10.91**	B	43.26 ^d	3, 84	8.70**
<u>Set B^f</u>	sR^2	.041	3, 130	2.34	sR^2	.003	1, 80	0.33
<u>Set C^g</u>	sR^2	.434	2, 128	83.66*	sR^2	.454	2, 78	64.40*
• ST_1^h	sr^2	.667	70, 140	280.33**	sr^2	.714	44, 88	219.34**
• Treatment ⁱ	sr^2	.001	1, 70	0.17	sr^2	.010	1, 44	1.58
Univariate Attitude	R^2	.439	13, 128	7.72**	R^2	.859	11, 78	43.29**
<u>Set A^a</u>	R^2	.091	8, 133	1.67	R^2	.118	8, 81	1.35
<u>Set B^f</u>	sR^2	.021	3, 130	0.97	sR^2	.009	1, 80	0.92
<u>Set C^g</u>	sR^2	.329	2, 128	37.71**	sR^2	.732	2, 78	202.47**
• ST_2^j	sr^2	.377	70, 140	84.59**	sr^2	.852	44, 88	507.37**
• Treatment ⁱ	sr^2	.001	1, 70	0.17	sr^2	.007	1, 44	2.14

Note. Full Dataset: $N = 142$, total sample size = 71; UNA Data Only: $N = 90$, total sample size = 45 with data coded for two repeated measures using criterion scaling via a multiple regression analysis strategy.

^aSet A = Student attributes (gender, programming experience, math background, and learning styles). ^bMath Background = dummy-coded variables M_1 , M_2 , and M_3 representing the highest level of math course attained: College Algebra, Trigonometry, Calculus I, or Above Calculus I; the reference group was College Algebra. ^c M_2 reflects the Calc I- College Algebra comparison. ^d B = the coefficient of the M_i variable in the regression equation. ^e M_3 reflects the Above Calc I-College Algebra comparison. ^fSet B = Academic attributes (school, term, school-term interaction for the full data set, term for the UNA data set). ^gSet C = Treatment + Subjects (treatment vs. control and the sum of each subject's attitude at the conclusion of the closed lab based class and the open lab based class). ^h ST_1 = the sum of each subject's achievement at the conclusion of the closed lab based class and open lab based class. ⁱTreatment = dummy coded variable representing class structure (open lab-based class or closed lab-based) with open lab-based class as the reference group. ^j ST_2 = the sum of each subject's attitude at the conclusion of the closed lab-based class and open lab-based class.

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

3. The univariate analyses of the UNA-only data for attitude yielded a significant overall model with Set C being significant. Similar to the full data set, subject differences were significant but treatment was not. Consequently, the results using only the data from UNA paralleled the results from the full data set.

Results of Hypothesis Testing

This section contains a summary of the results from testing the study's hypotheses. The decisions to reject or fail to reject these hypotheses were based on the statistical results presented in the previous section. Please note that the research hypotheses from Chapter 1 have been restated in null form here.

Hypothesis 1

There will be no significant difference in achievement between students in a restructured undergraduate introductory computer science class that incorporates a closed laboratory and students in a restructured undergraduate introductory computer science class that incorporates an open lab.

MANOVA results indicated that the overall regression model was significant when the two dependent measures were regressed simultaneously on the research factors, $\Lambda = .27$, $F_{26, 254} = 9.02$, $p < .0001$ (see Table 4.8). Univariate follow-up tests for achievement showed that Set C (Subject+Treatment) uniquely accounted for a significant amount of variability in achievement scores in the presence of the other research factors, $sR^2 = .434$, $F_{2, 128} = 83.66$, $p < .05$ (see Table

4.10). However, based on the preset alpha level of .05, the variable representing treatment was not statistically significant, $sr^2 = .001$, $F_{1, 170} = 0.17$, $p > .05$ (see Table 4.11). Consequently, this hypothesis was not rejected: There was no statistical difference in achievement between students in a restructured undergraduate introductory computer science class that incorporated a closed laboratory and those in a restructured undergraduate introductory computer science class that incorporated an open lab setting.

Hypothesis 2

There will be no significant difference in attitudes toward computers and computer courses between students in a restructured undergraduate introductory computer science class that incorporates a closed laboratory and students in a restructured undergraduate introductory computer science class that incorporates an open lab.

MANOVA results indicated that the overall regression model was significant when the two dependent measures were regressed simultaneously on the research factors, $\Lambda = .27$, $F_{26, 254} = 9.02$, $p < .0001$ (see Table 4.8). Univariate follow-up tests for attitude showed that Set C (Subject+Treatment) uniquely accounted for a significant amount of variability in attitude scores in the presence of the other research factors, $sR^2 = .33$, $F_{2, 128} = 37.71$, $p < .05$ (see Table 4.13). However, based on the preset alpha level of .05, the variable representing treatment was not statistically significant, $sr^2 = .001$, $F_{1, 170} = 0.17$, $p > .05$ (see Table 4.14).

Consequently, this hypothesis was not rejected: There was no statistical difference in attitudes toward computers and computer courses between students in a restructured undergraduate introductory computer science class that incorporated a closed laboratory and those in a restructured undergraduate introductory computer science class that incorporated an open lab setting.

Hypothesis 3

There will be no statistical significance in student achievement relative to the targeted student attributes of gender, previous computer programming education, math background, and learning styles.

MANOVA results indicated that the overall regression model was significant when the two dependent measures were regressed simultaneously on the research factors, $\Lambda = .27$, $F_{26, 254} = 9.02$, $p < .0001$ (see Table 4.8). Univariate follow-up tests for achievement showed that Set A (student attributes) uniquely accounted for a significant amount of variability in achievement scores, $R^2 = .193$, $F_{8, 133} = 3.96$, $p = .0003$ (see Table 4.10). When Set A was further investigated, only the math background factor, which examined students' highest level of mathematics by comparing the courses of Trigonometry, Calculus I, and above Calculus I to College Algebra, was significant, $sR^2 = .163$, $F_{3, 136} = 8.84$, $p > .05$ (see Table 4.11). In other words, the overall attribute set accounted for 19.3% of the variability in student achievement, and of this, 16.3% was uniquely attributed to students' math background. Further analysis showed that the Calculus I–College

Algebra comparison ($F = 19.13, p < .01$) and the Above Calculus I–College Algebra comparison ($F = 10.91, p < .01$) were significant. More specifically, students whose highest math background was at the Calculus I or Above Calculus I levels respectively averaged 51.5 and 35.1 points higher on their overall achievement than those students whose highest math background was College Algebra. In summary, although the collective influence of gender, previous computer programming education, math background, and learning styles had a significant effect on achievement, this effect was relative only to students' math background. As a result, Hypothesis 3 was rejected from the perspective of math background only: Given the targeted student attributes, only students' mathematics background had a significant effect on student achievement.

Hypothesis 4

There will be no statistical significance in students' attitudes toward computers and computer courses relative to the targeted student attributes of gender, previous computer programming education, math background, and learning styles.

MANOVA results indicated that the overall regression model was significant when the two dependent measures were regressed simultaneously on the research factors, $\Lambda = .27, F_{26, 254} = 9.02, p < .0001$ (see Table 4.8). Univariate follow-up tests for attitude showed that Set A (student attributes) did not uniquely account for a significant amount of variability in attitude scores, $R^2 = .09, F_{8, 133} =$

1.67, $p = .11$ (see Table 4.13). As a result, an examination of the research factors within Set A for significance relative to this dependent measure was not warranted because of inflated Type I and Type II error rates. Consequently, Hypothesis 4 was not rejected: The targeted student attributes of previous computer programming education, math background, gender, and learning styles did not have a significant effect on students' attitudes toward computers and computer courses.

Hypothesis 5

There will be no significant interaction effect between the targeted student attributes (gender, previous computer programming education, math background, and learning styles) and treatment (open lab-based vs. closed lab-based) relative to student achievement.

Independent of MANOVA, the hierarchical multiple regression ATI analysis between Set A = student attributes, X_3 = treatment, and the interaction between Set A factors and X_3 relative to achievement, resulted in a significant model, $R^2 = .274$, $F_{17, 124} = 2.754$, $p = .0007$. However, the increment the interaction set made to explaining the variability in achievement was not significant, $sR^2 = .081$, $F_{8, 124} = 1.73$, $p > .05$. As a result, Hypothesis 5 was not rejected: There was no significant interaction effect between the targeted student attributes (gender, previous computer programming education, math background, and learning styles) and treatment (open lab-based vs. closed lab-based) relative to student achievement.

Hypothesis 6

There will be no significant interaction effect between the targeted student attributes (gender, previous computer programming education, math background, and learning styles) and treatment (open lab-based vs. closed lab-based) relative to students' attitudes toward computers and computer courses.

Independent of MANOVA, the hierarchical multiple regression ATI analysis between Set A = student attributes, X_3 = treatment, and the interaction of Set A factors and X_3 relative to attitude, was not significant, $R^2 = .123$, $F_{17, 124} = 1.027$, $p = .4342$. As a result, Hypothesis 6 was not rejected: There was no significant interaction effect between the targeted student attributes (gender, previous computer programming education, math background, and learning styles) and treatment (open lab-based vs. closed lab-based) relative to students' attitudes toward computers and computer courses.

The findings of these analyses will be discussed in Chapter 5 with regards to the theoretical and empirical results acquired from previous research on computer science classes restructured to include closed laboratories. The next chapter will also introduce the pedagogical implications of the conclusions of the present study, discuss the limitations of the study's research design, and present suggestions for future research on the restructuring of computer science classes to include closed laboratories.

CHAPTER 5

CONCLUSIONS, IMPLICATIONS, AND RECOMMENDATIONS

Summary of Study

The purpose of this study was to investigate the relationships among various student attributes and different classroom structures involving computer laboratories on student achievement in and attitudes toward computer science and computer science courses. Two types of classroom structures were examined: open laboratory-based and closed laboratory-based. The open lab structure was a traditional lecture class in which student laboratory assignments were completed outside of class at the convenience of the students with no instructor supervision or assistance. The closed lab structure was a traditional lecture class that was restructured to include a laboratory component. The first half of the class was lecture and the second half of the class was the laboratory period where students worked individually or in pairs at their own pace on targeted laboratory assignments. During the lab period, a lab instructor was available for assistance and students were expected to complete the assignments during the laboratory period. The study examined the effectiveness of both structures relative to the targeted dependent measures. The study was grounded in the constructivist philosophy of cognitive learning theory: the open lab structure was tied to individual constructivism and the closed lab structure was tied to social constructivism.

The research factors were represented as sets of variables. Set A consisted of student attributes: gender, previous computer programming education, math background, and learning style. Set B consisted of school-related attributes: school site, term, and the interaction between school site and term. Set C consisted of the group membership variable (i.e., classroom structure) and two additional variables that were incorporated into the study to support the data-analytic method, namely, repeated measures using criterion scaling. The study's primary focus was to examine the effects of these research factors on the two dependent measures, student achievement in and student attitudes toward computers and computer courses.

The sample consisted of four intact classes of students from two different schools across 2 semesters; one class from each school was used in the fall term, and one class from each school was used in the spring term. The research methodology was a counterbalanced repeated measures experimental research design. In the course of each semester, two schedules of treatment implementation were used, one at each school. For the first schedule, students experienced the open lab-based class the first half of the term and the closed lab-based class the second half of the term. In the second schedule, the order of treatments was reversed. During the first semester, all the students within the class at the first school were randomly assigned to one of the two treatment schedules and all of the students within the class at the second school were randomly assigned the opposite

treatment schedule. During the following semester, the schedules assigned to each school were exchanged (see Table 3.6). Thus, all students experienced both types of classroom structures (repeated measure design), and the order in which the treatments were presented was rotated among the intact groups (counterbalanced design).

The target population was students in an introductory computer science course such as CS 1 as described by the ACM (2001), and the accessible population was students in a CS 1 type course at the University of North Alabama (UNA) and Athens State University (ASU); both schools are located in Northern Alabama. The initial sample of 91 students was selected from students registered for Computer Science I during the fall and spring semesters of the 2006–2007 school year. The final sample consisted of the 71 students who completed the course.

Each 14-week semester was divided into two 7-week sessions. During each session one type of class structure (open lab-based or closed lab-based) was used. The open laboratory was the control and the closed laboratory was the experimental treatment. Classes during the open lab-based session consisted of group discussions of the previous laboratory assignments, lecture, and a question and answer session. Students then received that day's laboratory assignment, which they were expected to complete outside of class time. Classes during the closed lab-based session consisted of lecture and lab time during which students were given that day's laboratory assignment and were expected to complete the

assignment during class time. Classes for both structures were identical in length with three 50-minute class sessions per week at UNA and one 150-minute class session per week at ASU.

The primary data collection instruments were the Attitude towards Computers and Computer Courses (ACCC; Newby & Fisher, 1997), Kolb's (2005) Learning Styles Inventory-Version 3.1 (KLSI-3.1), and three researcher-developed assessments. The ACCC measured the effect the different class structures had on students' attitudes toward computers and the CS 1 course. This 28-item instrument incorporated a 5-point Likert scale and comprised four subscales that represented students' anxiety, enjoyment, perception of computers' usefulness, and perception of course's usefulness. I administered the ACCC during the 1st, 7th, and last weeks of the fall semester at both schools, and the ASU CS 1 instructor and I administered the ACCC during the 1st, 7th, and last weeks of the spring semester at ASU. Reliability coefficients based on Cronbach's alpha were .95 for the overall instrument and between .77 and .94 for each of its subscales.

The KLSI-3.1 was used to classify students' dominant learning style. Based on responses to a set of 12 questions that targeted specific methods of learning, students were ultimately classified as being convergers (those who learn through active experimentation and abstract conceptualization), divergers (those who learn primarily through reflective observation and concrete experience), assimilators (those who learn through reflective observation and abstract conceptualization), or

accommodators (those who learn by relying on active experimentation and concrete experience). The KLSI-3.1 has a reported Cronbach alpha of between .77 and .84 (Kayes, 2005; Kolb & Kolb, 2005; Wierstera & DeJong, 2002). I administered the KLSI-3.1 during the 1st week of each semester at both schools.

Achievement was measured using three researcher-constructed assessments: laboratory assignments, programming assignments, and unit tests: 15 laboratory assignments, three programming assignments, and two unit tests were given during each session. Each laboratory assignment consisted of a purpose statement, a reading assignment, a short programming exercise, and a post-lab. Laboratory assignments were graded with one point for successful completion of the programming exercise and one point for timely submission of the post-lab. Programming assignments were longer programming exercises and were selected from popular CS 1 textbooks. These assignments were graded using a researcher-designed rubric. Unit exams consisted of 31 to 43 questions that were dichotomously scored. I selected the questions for the unit exams from various CS 1 textbook test generators. Based on my 10 years experience in teaching CS 1 courses, I reviewed these questions carefully and confirmed that they were content valid relative to the courses taught at both schools. The reliability coefficients of the unit exams were calculated using Cronbach's alpha and ranged between .70 and .83, and an inter-rater reliability of .93 was achieved for the lab and programming assignments, which were graded by one of the project faculty members from ASU

and me. Achievement data were collected continuously throughout each semester. I collected these data at both schools during the fall term and at UNA during the spring term; the ASU instructor collected these data at ASU during the spring term.

A copy of the ACCC (Newby & Fisher, 1997), the KLSI-3.1 (Kolb, 2005), and the unit exams are included in Appendix A. Copies of the lab and programming assignments and their rubrics are provided in Appendix B. In addition to these instruments, student demographic data (i.e., gender, math background, and previous computer programming education) were self-reported by students using a researcher-prepared and administered questionnaire during the 1st week of each semester at both schools. I then checked these data for accuracy via university records.

Summary of Findings

The study's data were analyzed using multivariate analysis of variance (MANOVA) via a hierarchical multiple regression approach with criterion scaling used to facilitate the repeated measures aspect of the analysis (Cohen et al., 2003). Functional sets were used to represent the variables. Set A was the set of student attributes, Set B was the set of academic characteristics, Set C was the set of subject and treatment factors, and Set D consisted of the two dependent measures (achievement and attitude). Several preliminary data analyses were performed prior to formal analysis. As part of these preliminary analyses, missing data issues were examined and resolved based on suggestions from Cohen et al., outliers were

examined and remedial actions were taken according to Cohen et al., and the data were examined relative to ordinary least squares (OLS) regression assumptions to ensure they were compliant with these assumptions. The final data set was then analyzed via MANOVA.

Data analysis revealed that the overall MANOVA model and the two corresponding follow-up univariate models (one for each dependent measure) were significant. These results indicated that (1) the collective influence of the three research factor sets on the two dependent measures simultaneously was nonzero, (2) the collective influence of the three research factor sets on achievement was nonzero, and (3) the collective influence of the three research factor sets on attitudes toward computers and the CS 1 course was nonzero. Subsequent analyses relative to the univariate achievement model indicated that the only targeted research factor that accounted for a significant portion of the variance in achievement scores was students' math background. More specifically, students who reported having either a Calculus I or above Calculus I background had, on average, significantly higher achievement scores than students who reported having a college algebra background. Subsequent analyses relative to the univariate attitude model indicated that none of the targeted research factors had a significant effect on students' attitudes toward computers and the CS 1 course. In both models, though, subject characteristics, which were separated from treatment effects as a

result of the repeated measures design, were significant, but the group membership variable (open lab-based vs. closed lab-based) was not.

Independent of the primary analyses, additional analyses were conducted to examine student attributes–treatment interactions relative to achievement and attitude. In both interaction models, the increment in the proportion of variance in achievement scores and in attitude scores accounted for by the set of interaction variables was not significant. Thus, there were no significant interaction effects between any of the targeted variables and group membership relative to achievement and attitude. A summary of these results applied to testing the study’s six null hypothesis is provided in Table 5.1.

Conclusions and Inferences

This section discusses the research questions corresponding to the study’s six hypotheses. The discussion includes a summary of the findings in response to each question, interpretations of the results, and some plausible explanations for the results obtained.

Research Question 1

***What is the effect of classroom restructuring
(open lab-based vs closed lab-based) of an undergraduate
introductory computer science class on student achievement?***

Student achievement was defined as the aggregate number of points earned on each session’s laboratory exercises, programming exercises, and unit exams.

Table 5.1
Summary of Hypotheses Tests Results

Null Hypothesis	Decision
1. $H_0: sR_{Y_1 \cdot X_3}^2 = 0^a$	Fail to Reject
2. $H_0: sR_{Y_2 \cdot X_3}^2 = 0^b$	Fail to Reject
3. $H_0: R_{Y_1 \cdot A}^2 = 0^c$	Reject
4. $H_0: R_{Y_2 \cdot A}^2 = 0^d$	Fail to Reject
5. $H_0: sR_{Y_1 \cdot \text{interactions}}^2 = 0^e$	Fail to Reject
6. $H_0: sR_{Y_2 \cdot \text{interactions}}^2 = 0^f$	Fail to Reject

Note. ^aThe effect of class structure (open vs. closed lab) on achievement after within subject characteristics were partialled out. ^bThe effect of class structure (open vs. closed lab) on attitude after within subject characteristics were partialled out. ^cThe effect of student attributes (gender, previous computer programming education, math background, learning style) on achievement; significance was found relative to math background. ^dThe effect of student attributes (gender, previous computer programming education, math background, learning style) on attitude. ^eThe interaction effect between student attributes and class structure on achievement after student attributes and class structure were partialled out. ^fThe interaction effect between student attributes and class structure on attitude after student attributes and class structure were partialled out.

Although the overall MANOVA model was significant, $\Lambda = .27$, $F_{26, 254} = 9.02$, $p < .0001$ (see Table 4.8), the MANOVA results of the group membership factor (i.e., classroom structure) were not significant, $\Lambda = 0.005$, $F_{2, 126} = 0.31$, $p = .73$ (see Table 4.9). Thus, the effect of classroom structure on achievement and attitude simultaneously was not significant.

When this was examined from a univariate perspective, a follow-up analysis for achievement revealed that Set C, which contained the subject and treatment

effects, was significant, $sR^2 = .434$, $F_{2, 128} = 83.66$, $p < .05$ (see Table 4.10).

However, when treatment effects were partialled from the corresponding subjects' effect as a result of the repeated measures aspect of the analysis, treatment was not significant. More specifically, when students were in the closed lab-based class structure their aggregate achievement scores averaged 2.7 points higher than their aggregate achievement scores from the open lab-based class structure. This mean difference was not significant, $sr^2 = .001$ $F_{1, 170} = 0.17$, $p > .05$ (see Table 4.11). Accordingly, the answer to Research Question 1 is that classroom restructuring (open lab-based vs. closed lab-based) of an undergraduate CS 1 course does not have an effect on student achievement.

There are several possible reasons for this result. One plausible explanation is that the two instructional methodologies were equally successful in facilitating students' comprehension of CS 1 material. This is a reasonable explanation when one considers the two instructional methodologies from an educational theory perspective. One strategy, the open lab-based class, was related to individual constructivism and the other, the closed lab-based class was related to social constructivism. Thus, both methodologies were grounded in similar theories and both should have a similar effect on student achievement.

Another plausible explanation has to do with the study's design, which was a counterbalanced repeated measures design. This design suffers from several unique context threats to internal validity as described in Chapter 3. One type of

context effect, *practice*, occurs when performance on a skill improves with practice. For example, students' ability to program improves as they practice doing so. As a result, we can surmise that student achievement improved as they progressed from the first session to the second session. A second context effect, *sensitization*, occurs when participants, exposed to several treatments, form hypotheses about treatment effects and respond to those hypotheses. For example, some participants in this study may have hypothesized that the closed lab was designed to improve achievement and thus worked harder during the closed lab-based session. A final context effect, *carry-over*, can occur when one treatment influences a subsequent treatment. For example, students who participated in the closed lab-based session first (completing the labs during class time) may have found it difficult to schedule time to complete their labs outside of class during the open lab-based session. Although measures were taken to lessen the impact of these threats, it is possible that they may have affected the results of this study in such a way as to mask the true effect of classroom structure on achievement.

Additional evidence supporting the finding of no difference between the two classroom structures is related to power and sample size. Before the start of this study, a sample size of at least $n = 70$ was calculated using a power analysis based on $\alpha = .05$, power = .80, and effect size = .25. The choice of α and power were conventional for this type of study. The medium-to-large effect size was chosen due to the lack of previous information about effect sizes in similar studies

and also due to the considerable investment of time and money necessary when changing the structure of a class. Thus, there was a $\beta = 20\%$ chance of committing a Type II error (i.e., claiming that no treatment effect exists in the population based on sample data when in fact such a treatment effect really does exist in the population), but only an $\alpha = 5\%$ chance of committing a Type I error (i.e., claiming an effect exists in the population based on sample data when in fact such an effect really does not exist in the population). This β - α ratio of .20 to .05 placed a greater emphasis on *not* committing a Type I error (i.e., the probability of committing a Type II error was four times greater than committing a Type I error). Thus, it is conceivable that this finding of no significant difference between class structures relative to achievement was a victim of these conventional preset power parameters. If β were less than .20 (and hence power greater than .8), it is possible that a significant effect might have been detected. However, this increase in power would also constitute an increase in sample size. Supporting this explanation, note from Table 3.3 that the actual treatment effect size was .001, which was much smaller than the hypothesized effect size of .25. As a consequence, the size of the sample, $N = 71$, was not sufficient to find a true effect. In order to find this small magnitude of an effect, it would have been necessary to increase the sample size considerably, which was not practical for this study.

Finally, the no significant difference finding could be related to the type of students in the classes. Typical CS 1 students are not prepared for the rigor of the

course and do not perform well in the course. In the current study, this was evidenced by the high percentage of students who either dropped (22%, see Table 4.7) or received a grade of F in the course. Poor preparation and lack of aptitude could possibly lead to poor performances regardless of the type of class structure.

Research Question 2

What is the effect of classroom restructuring (open lab-based vs. closed lab-based) of an undergraduate introductory computer science class on students' attitudes toward computers and computer courses?

Students' attitudes were defined as their score on the ACCC (Newby & Fisher, 1997). Although the overall MANOVA model was significant, $\Lambda = .27$, $F_{26, 254} = 9.02$, $p < .0001$ (see Table 4.8), the MANOVA results related to classroom structure (Table 4.9) were not significant, $\Lambda = 0.005$, $F_{2, 126} = 0.31$, $p = .73$. Thus, the effect of classroom structure on achievement and attitude simultaneously was not significant. When the overall significant MANOVA model was examined via a follow-up univariate hierarchical analysis of attitude with respect to classroom structure (Table 4.13) Set C, which contained the treatment and subjects variables, was significant, $sR^2 = .44$, $F_{2, 128} = 37.71$, $p < .05$. However, when the factors of Set C were examined individually, the treatment variable, which represented group membership (i.e., classroom structure: open lab-based vs. closed lab-based) was not significant, $sr^2 = .001$, $F_{1, 70} = 0.17$, $p > .05$ (see Table 4.14). Accordingly, the answer to Research Question 2 is that classroom

restructuring (open lab-based vs. closed lab-based) of an undergraduate CS 1 course does not have an effect on students' attitudes toward computers and computer courses.

There are several possible reasons for this result. First, as an affective domain variable, attitudes are very difficult to alter, particularly in the relatively short time span (7 weeks) of this study. This was further impacted by the relatively positive attitudes students in both groups had toward computers and computer courses at the beginning of the study (see Tables 4.4, 4.5, and 4.6). With few exceptions, those attitudes did not substantially change throughout the study. This result is also logical because students who take CS 1 often do so because they are interested in computer science and correspondingly have positive attitudes toward the discipline in general.

A second plausible explanation has to do with students' perceptions of the instrument used to measure attitudes. It was my perception that students did not believe they were to receive any benefit from completing the instrument and I suspect that some students did not take the attitude survey seriously. Although I observed some students reading each statement and carefully considering their responses, others simply marked the middle answer without reading the questions, while still others scanned the questions and either marked 5 for the positively worded questions or 1 for the negatively worded questions. Thus, although these students' responses were consistent, the results were not an accurate measure of

their attitudes. This issue of participant honesty when responding to survey items is a limitation in all studies in which participants self-report their responses.

A third plausible explanation is that student attitudes are truly not affected by class structure. In other words, it is conceivable that the two instructional methodologies are equally successful in maintaining students' attitudes toward computers and computer courses. Finally, the results also could have been a victim of the β - α ratio, which was discussed earlier.

Research Question 3

What is the effect of the targeted student attributes (gender, previous computer programming education, math background, and learning styles) on student achievement?

Given both an overall significant MANOVA model, $\Lambda = .27$, $F_{26, 254} = 9.02$, $p < .0001$ (see Table 4.8), and a significant follow-up univariate achievement model, $R^2 = .668$, $F_{13, 128} = 19.8$, $p < .0001$, the research factor sets were examined hierarchically relative to achievement using the set entry order of A-B-C (see Table 4.10). The results of this analysis showed that Set A, student attributes, contributed 19.3% of the variability in achievement scores, which was significant, $F_{8, 133} = 3.96$, $p = .0003$. When the individual variables of Set A were examined (see Table 4.11), only the math background factor was significant. Specifically, students whose highest level math background was Calculus I averaged 50 points more on their achievement scores than those with only a College Algebra background

($F_{M_2} = 19.13, p < .0001$), and students with above Calculus I math background averaged 33 points more than students with only a College Algebra background ($F_{M_3} = 10.91, p < .0001$). Accordingly, the answer to Research Question 3 is that students' math background has a significant effect on student achievement, but gender, previous computer programming education, and learning styles do not. Furthermore, the math background effect was relative only to the Calculus I–college algebra and the Above Calculus I–college algebra comparisons.

The significant effect of students' math background on achievement was expected and reasonable given the historical relationship between mathematics and computer science. It was somewhat surprising, though, that none of the other targeted student attributes were significant, particularly because those attributes were purposively selected based on my personal experience teaching CS 1 courses and from prior studies. A brief discussion of the math background effect, as well as a discussion about the lack of effect for the other attributes follows:

Mathematics Background

Students' math backgrounds were examined via three comparisons: trigonometry–college algebra, Calculus I–college algebra, and above Calculus I–college algebra. As noted above, only the latter two were significant. The significance of the two groups with high levels of math preparation is reasonable in that students with a higher level of math preparation should have higher achievement in computer science because the two fields are closely related in much

the same way as other sciences are closely related to mathematics. The non-significance of the trigonometry-college algebra comparison could be related to the sample size of this study. Although there were 71 total participants, only 14 reported trigonometry as their highest level of math preparation. A larger sample that included more participants in each group might produce significant results for all levels of math preparation. Another plausible reason for the results in the trig-college algebra comparison deals with the way the math background variable was measured. Only students who took a *college* trigonometry course as their highest level of math preparation were included in the trigonometry group. It is likely that many of the students in the group with college algebra as their highest level of preparation took a *high school* trigonometry course. On the other hand, it is unlikely that those same students took a high school calculus course. A high school trigonometry course background existent in the college algebra only students could have provided a sufficient math background to mitigate the significant effect by the college level trigonometry course. Finally, both of the schools used in this study offer a course that combines college algebra and trigonometry. As a result, one possible reason why the trig-college comparison was not significant might be because students did not realize that they took a trig course if they took this or another such combined course. This makes the comparison problematic because of the potential ambiguity in the way students interpreted “trig background.”

College algebra was treated as the comparison group because it is considered a foundation course for higher-level mathematics. It is also the default prerequisite math course for CS 1 type courses at many institutions. Independent of the college algebra comparisons, additional statistical analyses were performed using trigonometry as the comparison group and then again using Calculus I as the comparison group. When trigonometry was used as the comparison group, the Calc I–Trig comparison was significant ($t = 4.223, p < .0001$) as was the Above Calc I–Trig comparison ($t = 3.225, p = .0016$). When the Calculus I group was used as the comparison group, the Above Calc I–Calc I comparison was not significant ($t = -1.461, p = .1462$). In summary, the data confirm that there is a strong and significant relationship between CS 1 course achievement and a strong mathematics background (i.e., at the level of Calculus I or greater).

Student Gender

In the case of student gender, although twice as many male students ($n_{\text{male}} = 48$) than female students ($n_{\text{female}} = 23$) participated in the study, gender did not have a significant effect on achievement. Given this large difference between the number of male and female students, it was surprising that a gender effect in the direction of males was not detected. One possible explanation for this result might be related to the gender of the instructors. Although a male teacher was the instructor for only 7 of the 23 female students, a female teacher was the instructor for 40 of the 48 male students. This means that a female teacher taught 56 of the 71 students—

nearly 80%. As a result, it is conceivable that teacher gender might have mitigated any possible student gender effect relative to achievement.

A second plausible explanation for the lack of a gender effect is the nature of the research design (repeated measures) and the manner in which the data were analyzed. One of the advantages of a repeated-measures design is it gives the researcher control for subject differences, which represents one of the largest sources of variation in most research studies. When left uncontrolled, as in a completely randomized design, subject differences comprise part of the error term. In a repeated-measures design, though, participants serve as their own control and hence the variance due to individual differences is separated from the error term. Although this leads to a more precise analysis, it also could have masked any potential gender effect.

A third plausible explanation is also related to the repeated measures design. Each of the study's participants experienced both treatments and hence any bias in the personal characteristics of the participants was equally distributed between both treatment groups. Thus, it is reasonable to conjecture that the assignment of all subjects to both treatment groups controlled the possible influence of any gender differences among the participants. Another possible explanation is grounded in the position that we are now at a state in computer science instruction (as well as in society) where gender differences are no longer an

issue. Lastly, the lack of a gender effect could be related to insufficient power as discussed previously.

Previous Computer Programming Education

In the case of previous computer programming education, a plausible explanation as to why it did not have a significant effect on achievement lies in the manner in which the factor was defined and measured: Students self-reported whether or not they successfully passed (with a C or better) a previous computer programming course, and based on their response were placed into either a “yes” or “no” group. This factor did not take into account students who self-taught themselves a programming language. Based on my personal experiences teaching CS 1 courses, I suspect that some of the students who reported “no” should have been included in the “yes” previous computer programming education group. Thus, it is possible that if students had accurately reported their past computer programming education or if data for this factor were collected in another form other than as a dichotomous question, then significance might have been achieved. Other possible explanations for the lack of a previous computer programming education effect include the impacts of a repeated measures design and insufficient power as discussed previously.

Learning Styles

The lack of a learning styles effect was anticipated. This is because learning styles was flagged during preliminary analysis as an incorrectly specified variable

when the data set was examined for compliance relative to the underlying assumptions of regression: As reported in Chapter 4, it yielded a zero slope during an AVP plot involving the dependent measures. The most plausible explanation for this result is that the sample sizes for each learning style were too small to detect differences in achievement. Although there were 31 convergers and 22 assimilators, there were only 10 divergers and 8 accommodators. These small learning style groups, especially for divergers and accommodators, would make it difficult to discover all but a very large effect on achievement. Other reasonable explanations include those associated with the repeated measures design as discussed in the foregoing paragraphs.

Research Question 4

What is the effect of the targeted student attributes

(gender, previous computer programming education, math background, and learning styles) on student attitudes toward computers and computer courses?

Given both an overall significant MANOVA model, $\Lambda = .27$, $F_{26, 254} = 9.02$, $p < .0001$ (see Table 4.8) and a significant follow-up univariate attitude model, $R^2 = .44$, $F_{13, 128} = 7.72$, $p < .0001$, the research factor sets were examined hierarchically relative to attitude using the set entry order of A-B-C (see Table 4.13). The results of this analysis showed that Set A, student attributes, contributed 9% of the variability in attitude scores, which was not significant, $F_{8, 133} = 1.67$, $p = .11$. Accordingly, the answer to Research Question 4 is that the student attributes

of previous computer programming education, math background, gender, and learning styles do not have any effect on students' attitudes toward computers and computer courses.

Again, several explanations are offered for this result. Similar to the discussion of Research Question 2, the most logical explanation is that students' previous computer programming education, gender, learning styles, and math background are truly not important factors relative to attitude in an introductory computer programming course. This implies that other attributes that were not targeted might impact their attitudes. This is discussed later in the chapter.

Another possible explanation has to do with the type of student taking this course. As noted earlier, initially the students in this study had very positive attitudes toward computers and computer courses. With few exceptions, those attitudes did not substantially change throughout the study. Their positive attitudes did not appear to be affected by anything, including the targeted attributes.

Finally, two other possibilities for not finding an attributes effect on attitudes are the small sample size and the Type I–Type II error ratio, both of which were as discussed in Research Question 1. Thus, it is possible that by placing less emphasis on committing a Type II error (i.e., by increasing power, which results in an increase in sample size), the targeted student attributes' effect on students' attitudes toward computers and the computer course might be detected.

Research Question 5

What is the interaction effect between the targeted student attributes (gender, previous computer programming education, math background, and learning styles) and treatment (open lab-based vs. closed lab-based) relative to student achievement?

To answer this research question, an attributes-treatment interaction (ATI) model was examined via a univariate hierarchical multiple regression analysis with the variable entry order of Set A (student attributes), treatment (i.e., classroom structure), and the set of variables representing the interaction between attributes and treatment. This model explained 27.4% of the variability in achievement, which was significant, $R^2 = .274$, $F_{17, 124} = 2.75$, $p = .0007$. However, the increment the interaction set made to explaining the variability in achievement scores was 8.1%, but this was not significant, $sr^2 = .081$, $F_{8, 124} = 1.73$, $p > .05$. Accordingly, the answer to Research Question 5 is that the interaction between the targeted student attributes (gender, previous computer programming education, math background, and learning styles) and treatment (open lab-based vs. closed lab-based) has no effect on student achievement.

As with the previous research questions, two plausible explanations for the results of Research Question 5 are statistically oriented: sample size and power. This study's sample size was chosen to recognize a medium-to-large effect size 80% of the time. However, if the true effect is small, it would be found in a much

smaller percentage of samples. Thus, the true effect might be too small for this study's sample to recognize. Furthermore, by increasing power from .8 to .9 or .95, which also has a concomitant increase in sample size, it is possible that an interaction effect might have been detected.

Other possible explanations, which also have been noted earlier, include: (1) the assignment of all participants to both groups, which distributes individual differences equally between groups; (2) the nature of the research design, which separates subject differences from treatment; (3) the length of each treatment (7 weeks), which might not have been sufficient time to fully develop an attribute-treatment interaction effect relative to achievement, and (4) although counter-intuitive, there might truly be no interaction effect between the targeted attributes and treatment relative to student achievement.

Research Question 6

What is the interaction effect between the targeted student attributes (gender, previous computer programming education, math background, and learning styles) and treatment (open lab-based vs. closed lab-based) relative to students' attitudes toward computers and computer courses?

Similar to Research Question 5, this research question was answered by examining an attributes-treatment interaction (ATI) model via a univariate hierarchical multiple regression analysis with the variable entry order of Set A (student attributes), treatment (i.e., classroom structure), and the set of variables

representing the interaction between attributes and treatment. This ATI attitude model explained 12.3% of the variability in attitude scores, but it was not significant, $R^2 = .123$, $F_{17, 124} = 1.03$, $p = .4342$, and hence no follow-up analyses were conducted. Accordingly, the answer to Research Question 6 is that the interaction between the targeted student attributes (gender, previous computer programming education, math background, and learning styles) and treatment (open lab-based vs. closed lab-based) does not have any effect on students' attitudes toward computers and computer courses.

Plausible explanations for the results of Research Question 6 parallel those of the Research Question 5: (1) sample size and power were inadequate, (2) the assignment of all participants to both groups distributed individual differences equally between groups, (3) the repeated measures separated subject differences from treatment, (4) the 7-week length of each treatment was too short of a time period, and (5) there truly is no interaction effect between the targeted attributes and treatment relative to students' attitudes toward computers and computer courses.

Implications Relative to Prior Theory and Research

In this section, implications from this study relative to educational theory and previous research are examined. The theoretical foundations and research studies presented in Chapter 2 are used to guide the discussion.

Theoretical Implications

Constructivism

The theoretical foundation for this study was the constructivist perspective of cognitive learning theory, which states that students learn better if they construct their own knowledge. Such knowledge construction can be done either individually (individual constructivism) or as part of a group (social constructivism). In this study, the open laboratory class structure was grounded in individual constructivism and the closed laboratory class structure was grounded in social constructivism. In the former, students completed lab assignments on their own time outside of class and used notes from classroom discussions, reference books, and compiler-generated error messages to construct both an understanding of the lab's topic and how the topic can be used to solve problems. In the latter, classes were partitioned into lecture and lab components and during the laboratory sessions, students worked on their lab assignments as part of a classroom community by collaborating and sharing with each other their discoveries and solutions to the assigned problems.

Additionally, Vygotsky's (1996) zone of proximal development (ZPD) concept was manifested in the closed laboratory-based class by having an instructor provide guidance as students worked alone or in pairs to complete the laboratory assignments. Vygotsky believed that complex mental processes begin as social activities. He posited that children learn through cultural interactions, and that such

interactions are essential for cognitive development. As was expected, there were considerable instructor-student and student-student interactions throughout the closed lab-based classes: students asked questions and exchanged ideas with their classmates, and they utilized the instructor as a mentor or guide by asking him or her to clarify concepts or answer questions about various aspects of the assignments. This practice of promoting peer and teacher-student interactions is consistent with social constructivism's basic foundation. In summary, it is reasonable to surmise that achievement was "equivalent" between the class structures because both were manifestations of the constructivist perspective of cognitive learning theory.

Learning Styles

One of the basic tenets of learning theory is that students have individual preferences or methods for how they perceive they learn best. For example, some students might be visual learners where they favor reading about a topic or watching a demonstration, other students might be auditory learners and prefer listening to an instructor lecture about a topic, and still others might prefer to "learn by doing" (i.e., take a hands-on approach to a particular learning task). As explained earlier, Kolb's (2005) learning style inventory (KLSI-3.1) was used to identify participants' as convergers (those who learn through active experimentation and abstract conceptualization), divergers (those who learn primarily through reflective observation and concrete experience), assimilators

(those who learn through reflective observation and abstract conceptualization), or accommodators (those who learn by relying on active experimentation and concrete experience). The results of the study showed no significant learning style effect. On reflection, this result is not surprising because both class structures provided students with multiple opportunities to pursue a learning task that was commensurate with their learning preference. All students were assigned sections to read from the text, they received the same lectures, which were supplemented by PowerPoint presentations, they were required to practice the day's topic via the laboratory exercises, and they were expected to ask questions and participate in discussions with the instructor and their peers. Thus, it did not matter if students were accommodators, assimilators, convergers, or divergers because they all rotated between both class structures and they all had the same learning opportunities that offered a variety of different learning environments that were congruent with the four learning styles. An interesting follow-up question is, if students were not provided with multiple learning environments, would we get the same or different results? This question is considered later in the recommendations section of this chapter.

Implications Relative to Prior Research

Prior Studies on Class Structure and Laboratories

As noted earlier in this chapter, my study found that classroom restructuring (open lab-based vs. closed lab-based) of an undergraduate CS 1 course did not have

an effect on student achievement. This finding was mostly inconsistent with those of Thweatt's (1994) who also compared, over 2 semesters, student performance in a CS1 class that included closed lab experiences with student performance in a class that included open laboratory experiences. Thweatt reported that (1) the spring closed lab-based group had significantly higher achievement than the spring open lab-based group, (2) the combined fall and spring closed lab-based groups had significantly higher achievement than the combined fall and spring open lab-based groups, and (3) although the fall closed lab-based group outperformed the fall open lab-based group, the difference in achievement scores was not significant.

There are several possible reasons for the difference in the results found in Thweatt's (1994) spring and combined classes. First, Thweatt used GPA as a covariant and investigated previous computer experience as the only other student attribute of interest when he analyzed the spring data. It is reasonable to assume that students in the open lab class differed from those in the closed lab class on other important characteristics, including mathematics background, because students self-assigned themselves to class. This could have contributed to the achievement differences of the two classes. This speculation that the groups might have been different to begin with is supported in part from the results of Thweatt's findings from the fall sample. During the fall semester, Thweatt matched students on GPA and ACT/Scores and then randomly assigned them to a particular group. As noted in (3) above, Thweatt found no significant achievement effect for the fall

sample. This suggests that when the groups are similar, which was the case in my study (actually, in my study the groups were identical), there is no achievement effect, which is exactly what I found. However, when students have the opportunity to self-assign themselves to a particular class structure and there is little-to-no control to account for student differences, then an achievement effect is present.

A second plausible explanation for our different findings relative to achievement is that Thweatt's (1994) treatment and control groups experienced a marked difference in time with the instructor. The closed lab-based classes (treatment) benefited from additional instructor time during the weekly scheduled 2-hour laboratory sessions, which were separate from the lecture portion of the class. My classes experienced the same amount of instructor time, and the laboratory time for the closed lab-based classes was approximately 75 minutes per week. Thus, the achievement effect he found could be attributed to instructor time differences and not directly due to treatment.

Finally, a third plausible explanation for the differences in our findings could be related to differences in the respective data collection instruments we used. In Thweatt's (1994) study, student performance was assessed by a comprehensive final exam developed by a departmental committee. In my study, achievement was measured using the aggregate scores from several different types of assessments, one of which was an instructor-prepared comprehensive final exam. It is conceivable that if I had relied only on a single instrument to measure

achievement, then I too might have found a significant achievement effect. In summary, although my results were not entirely congruent with those of Thweatt's, I believe my results are more credible because of the controls I instituted to minimize internal validity threats, particularly those related to differences in subject characteristics, and my method of assessing achievement was inclusive.

The results of my study also were not consistent with those of Kumar (2003). In his study, Kumar compared the performance of students in two classes that included an optional closed laboratory (treatment) with the performance of students of two classes from a previous semester that did not include such a laboratory (control). He found that student performance improved with closed labs. His results also indicated that the treatment group performed significantly better than the control group only on the first test. They did no better on the second test or the final exam. Again, my study did not show a significant improvement in achievement with class structures that included a closed laboratory.

There are many credible reasons for the different results in our two studies. The most likely deals with the way in which achievement was measured. Kumar's (2003) student performance was based on students' final grade, which was expressed as a letter grade (e.g., A, B, C, D, F) and not numerically. He constructed a histogram that indicated the largest letter grade group for the control classes was a B, while the largest letter grade group for the treatment classes was an A. He concluded that this indicated more learning took place in the treatment group than

in the control group. As detailed in Chapter 4, in my study students' performance was measured as the sum of points scored on all achievement instruments. I based my conclusions on a multivariate analysis of variance (MANOVA) via hierarchical multiple regression. Thus, it is possible that had Kumar used more acceptable statistical techniques in analyzing his overall measure of student performance, he might have obtained different results. Moreover, Kumar did not perform any statistical calculations to show that the change in student performance was significant. Thus, he under-analyzed his findings, which infers that he might have paid little attention to treatment verification (Shaver, 1984).

A second possible reason for our different results deals with the instruments Kumar (2003) used in his study. Kumar did not report giving any attention to instrumentation validity and reliability. Furthermore, control students handwrote their programming exams while students in the treatment group used a computer compiler. According to Fraenkel and Wallen (2003), this change in instrumentation is considered an instrumentation threat to internal validity and can create problems in the interpretation of the results of a study. A third possible reason why our results were different could be related to Kumar's lack of random assignment. In his study, neither students nor intact classes were randomly assignment to treatment or control groups. Thus, it is unlikely that the two groups were equivalent on key significant student characteristics. Problems with implementation provide a fourth plausible explanation for our different findings. Kumar was the sole implementer.

As a result, it is conceivable that he might have intentionally biased the results. This might be especially true if he did not take any measures to verify treatment fidelity (Shaver, 1984). Additionally in Kumar's study, the closed laboratory classes had 45 minutes per week of extra instructor time. Thus, as was the case with Thweatt (1994), students in this group received something extra, which can easily account for improved performance. This is similar to students who receive after school tutoring sessions and have higher achievement than their non-tutored classmates. Finally, the open lab classes from a previous semester did not include any kind of laboratory and students did not receive laboratory assignments. Thus, as stated above, the students in the treatment group received something extra, beyond what students in the control group received. This could have provided another justification for improved performance in the treatment group.

In conclusion, Kumar's (2003) study suffered from several internal validity threats, including subject characteristics, instrumentation, and implementation. In contrast, I used multiple controls to restrict the influence of such internal validity threats, and my method of evaluating achievement was more reasonable. For these reasons, I believe my results are more realistic and credible.

The results of my study also were not consistent with those of Wu's (1997), who examined the effects of open vs. closed laboratories in a CS 2 course on student achievement. Wu found that students in closed laboratories had significantly higher achievement than students in open laboratories. The most

plausible and logical explanation why our findings are different is that students in Wu's closed lab class used specially designed software that Wu developed to help students. Students in the open lab class did not have access to this software. Thus, as was the case with Kumar (2003), Wu's treatment group received something extra. If Wu had administered exactly the same instructional strategy and resources to both treatment and control group students, which is what I did in my study, then it is possible that he also would not have found any significant differences between the two class structures relative to achievement.

In contrast to Thweatt's (1994), Kumar's (2003), and Wu's (1997) results, the findings of my study were congruent with those of Duplass (1995) and Altitt-Wheeler (2005), who independently investigated the effects of closed and open laboratories on student performance using students enrolled in computer application courses. Although both of these studies were performed on students enrolled in non-programming Microsoft Office application-oriented courses, some correlations still exist between their studies and mine. Both types of courses, applications courses and CS 1 courses, are considered introductory, computer-related courses. Although both are generally considered freshman level courses, in reality, the makeup of the class usually consists of all levels of student classifications. Finally, as an instructor of both types of courses I have found that the level of previous experience with computers often is also about the same: word processing, surfing the Internet, game playing, and use of email. Duplass and Altitt-

Wheeler, as did I, concluded that the closed laboratories did not have a significant positive effect. Thus, it appears that classes structured to include a closed lab may not provide improved performance over classes structured to include an open lab in first-level introductory computer-related courses, regardless of whether those courses are applications- or programming-oriented.

Prior Studies on Student Characteristics

This study used a repeated measures design in which subjects participated in both classroom structures for 7 weeks and their performance on the dependent variable was measured throughout and at the end of each 7-week period. Perhaps the single most important advantage of repeated measures designs is they enable the researcher to control for individual subjects' differences, which are probably the largest source of variation in most research studies. When left uncontrolled, as in a completely randomized design, individual differences comprise part of the error term. In a repeated measures design, though, participants serve as their own control and hence it is possible to identify the variance due to individual differences and separate it from the error term. In this study, student differences accounted for more than 66% of the variability in achievement scores (see Table 4.11) and 37% of attitude scores (see Table 4.14). However, students' personological and academic characteristics were not key research interests of this study. The targeted characteristics (i.e., gender, previous computer programming education, math background, and learning style) were chosen based on previous studies and were

incorporated as secondary factors for statistical control purposes. These variables were included in the statistical model, because past studies had reported their ability to account for some of the variability in achievement. As reported in Chapter 2, the results of past studies on student characteristics important to student success have been mixed and therefore it is no surprise that this study's results supported some of the conclusions of those studies and refuted others. A discussion of the general and direct implications of the current study's findings relative to prior research on student characteristics follows.

Previous computer programming education. As discussed in Chapter 4, students who self-reported that they took a previous computer programming course prior to enrolling in CS 1 did not have significantly higher achievement or more positive attitudes, regardless of the class structure, than those students who reported not having taken a previous computer programming course. This finding was inconsistent with those of (1) Taylor and Mounfield (1991), who found a previous high school computer programming course had a significant effect on achievement; (2) Hagen and Markham (2000), who found previous programming experience had a significant effect on achievement; and (3) Byrne and Lyons (2001), who found that students with previous programming experience performed better than those without. In contrast, my result of no significant previous computer programming education effect was consistent with those of Butcher and Muth (1985), who found no correlation between achievement and a previous high school computer course

and Wilson (2002), who reported no significant effect on achievement by previous programming experience.

One possible explanation for the different findings among these studies, including the current one, is the different terms used for “previous computer programming education.” Some researchers defined it as previous computer programming course, others defined it as previous computer course, and still others defined it as previous programming experience. The differences in wording and the corresponding differences in definition make comparing results relative to students’ prior computer experience problematic.

Another possible explanation for the different results might be chronologically based. For example, Butcher and Muth (1985) conducted their study in the early-to-mid 1980s when it was less likely for students to have taken a high school computer course prior to entering a CS 1 course than it is today. Thus, it is not surprising that there was no significant correlation between having taken a high school computer course and achievement. On the other hand, Taylor and Mounfield (1991), Hagen and Markham (2000), and Byrne and Lyons (2001) conducted their respective studies during the 1990s and early 2000s, which was a period marked by considerable interest in computer-related courses. Thus, it is reasonable to expect students with prior computer experience during this period to have significantly higher achievement in CS 1. The last and most recent study I reviewed, however (Wilson, 2002), did not find such an effect, and now my study,

which was conducted in 2006–2007, also did not find an effect. The reason for this might be related to our current state of affairs where nearly all students have prior computer experience thanks to the Internet and various convergence-based products such as the Ipod or Iphone. The reader might recall from the Research Question 3 discussion presented earlier in this chapter that students self-reported whether or not they successfully passed (with a C or better) a previous programming course. This did not take into account students who self-taught themselves a programming language. Thus, it is possible that we are beginning to see a shift from a time when previous “formal” computer education was important to achievement to where it is no longer relevant.

Concomitant with the chronology discussion in the previous paragraph are changes in the CS 1 curriculum. The curriculum for the courses used in this study was based on guidelines published by ACM, one of the accrediting agencies for computer science programs. Those guidelines have changed several times. For example, ACM published its computer science curriculum guidelines in 1968, modified them in 1978, and then modified the guidelines again in 1991. The current set of guidelines was published in 2001. Thus, it is possible that the mixed results of the impact previous computer programming education has on achievement might be related to the changes in ACM’s recommended CS 1 curriculum.

Gender. The declining female enrollment in computer science (Zweben, 2007) is troublesome especially given reports that the percent of female enrollment

in colleges is on the rise (Mather & Adams, 2007) as well as female undergraduate degrees awarded in math, engineering, and biological and physical sciences (NSF, 2004). One possible reason for this decline is that females don't like computer science. The results of my study refute this argument: There was no significant difference in attitude between male and female students. Another possible explanation is that females do not perform as well in computer science. The results of my study also refute this argument: There was no significant student gender effect on achievement. My findings are consistent with those of Wilson (2002), Byrne and Lyons (2001), and Pillay and Jugoo (2005); they were not consistent with those of Bergin and Reilly (2005), however, who found that gender had a significant effect on achievement.

There are several plausible reasons for the difference in my results and those of Bergin and Reilly (2005). The first concerns locale. Bergin and Reilly conducted their study at the University of Ireland whereas I conducted mine at two universities in Alabama. It is conceivable that students in Ireland have not progressed beyond gender differences as well as we have in the U.S. Additionally, Bergin and Reilly's sample consisted of only 30 students, which was much smaller than my sample of 71. Sample size notwithstanding, though, our two studies involved different target populations, and hence the results of my study are more credible for U.S. students in CS 1 type courses.

Math Background. My study's result of a significant previous mathematics background effect on achievement was consistent with the findings of Butcher and Muth (1985), Byrne and Lyons (2001), Wilson (2002), and Bergin and Reilly (2005). In each of these studies, the method used to measure math background was different: Butcher and Muth used students' ACT math scores, Byrne and Lyons used scores on the Irish Leaving Certificate examinations, Wilson used the number of semesters of math courses taken in high school, Bergin and Reilly again used the Irish LC scores, and I used the highest level of college math course taken. It is interesting to note that in spite of these disparate methods of measuring math background, all the studies reported a significant effect by math background on achievement. This gives greater testimony to the importance of a proper mathematics background for success in CS 1.

Studies on Learning Styles. As noted earlier, I was unable to find a significant learning styles effect. This result was inconsistent with the findings of Davidson and Savenye (1992) and Crosby and Stelovsky (1995). However, Avitable's (1998) attempt to recreate the Crosby and Stelovsky's study also resulted in a finding of no significant effect by learning styles. Moreover, Byrne and Lyons (2001) found no significant effect in their study of learning styles and their effect on CS 1 students. Comparing the results of these studies is problematic because of the different learning style scales used in each study. Crosby and Stelovsky (1995) and Avitable (1998) used the Myers-Briggs Type indicator

(Meyers, 1962, 1975), Davidson and Savenye (1992) used the Gregorc (1984) Style Delineator, and Byrne and Lyons (2001) and I used Kolb's (2005) Learning Style Indicator (KLSI). Suffice it to say that the two studies that used the KLSI (mine and Byrne and Lyons) were consistent in their findings.

Implications for Educational Practice

The findings of this study suggest that both types of class structures (open lab-based and closed lab-based) equally prepared students for the programming assignments and unit tests. It also appeared that neither affected their attitudes toward computers and computer courses. In the current atmosphere of declining student interest in the Computer Science major, it is important for educators to carefully consider where to allocate resources such as computers and classroom space. Additionally, courses must be made as attractive and accessible to today's student as possible. Based on the results of this study, educators who are faced with a decision between equipping or upgrading classrooms as computer laboratories might opt for less expensive, more pragmatic open lab-based class structures for their Computer Science I classes without jeopardizing the achievement or attitude of their students. The reality is that many schools already have up-to-date classrooms equipped as computer laboratories for their computer science classes. Instructors in this situation can (1) continue to use closed lab-based class structures (2) change over to open lab-based class structures allowing more time for student questions and discussion, or (3) use a mixture of the two

allowing students to work on the laboratory exercises in-class when time allows, sending them home to complete the exercise when class time is short.

Computer science instructors are interested in identifying students who may have trouble in their classes and finding ways to help such students. This is the fundamental rationale behind studies that attempt to discover student characteristics that significantly affect student performance. As noted earlier, the findings of this study stress the importance of math background and relative unimportance of gender, learning style, and previous computer programming education.

Furthermore, when the results of this study were compared to previous studies relative to math background, the preponderance of evidence supports the claim that math background is an important, robust characteristic for computer science students. This implies that students with a poor math background (in the context of the current study, this infers less than a Calculus I background) should be advised to improve their math skills before attempting computer science courses.

Additionally, instructors can use math background as a litmus test to determine students who need extra help.

The relative unimportance of both gender and previous computer programming education is an important implication for practice. In the case of gender, although concern over the shrinking number of female computer science majors and students is well founded, it appears that those females who are taking CS 1 are doing as well in the course as their male counterparts. This implies that

the direction computer science educators should take to increase female participation in CS 1's recruitment. Efforts toward retention should not be gender specific. In the case of the student attribute previous computer programming education, it appears that preparatory computer programming courses do not significantly increase student performance in CS 1. Thus, such courses as the currently popular CS 0 course that many schools are adding to their curriculum as a prerequisite for CS 1, need to be studied to ensure that the expense, time, and effort spent on such courses is significantly beneficial to students.

Generalizability, Limitations, and Delimitations

Generalizability

Generalizability (also called external validity) has two kinds, population generalizability and ecological generalizability. Population generalizability is the extent to which a study's sample represents the target population. Ecological generalizability is the extent to which the study's results can be applied to other settings (Fraenkel & Wallen, 2003). With regard to population generalizability, the content and objectives of the CS 1 course used in this study were standard to CS 1 type courses in colleges across the United States, the target population for this study. Additionally, the students of this study were found to be representative of students in other CS 1 classes across the U.S. in academic major, gender, race, and national origin (see Chapter 3). Conversely, it is difficult to ensure that the sample used in this study was representative on all important characteristics. For example,

it is difficult to be certain that 4-year public college students are similar to 4-year private college/university students, or that students attending an Alabama university are the same as students in a California university. In spite of these concerns, it is likely that this study has a high degree of population generalizability.

In regard to ecological generalizability, it is probable that the results of this study could be broadened to other CS courses where laboratories have been used, such as CS 2 or Data Structures. Students in these early courses are similar to those in CS 1 courses and their content is a continuation of the content in CS 1. To help facilitate ecological external validity, all aspects of this study are fully detailed in this dissertation. However, it is the reader's responsibility to determine whether this study's results are pertinent to other particular settings. Finally, given the relative dearth of such studies, it is my hope that this study will encourage other researchers to examine the effects of classroom restructuring (open lab vs. closed lab) on achievement and attitude to help enhance the current body of literature in this area.

Study Limitations and Delimitations

The results of all educational research are restricted by the limitations and delimitations associated with a study. As noted in Chapter 1, limitations are events or conditions outside the control of the researcher that limit the generalizations of study results; delimitations are additional restrictions imposed by the researcher that further limit the study. A discussion of this study's 3 limitations and 11 delimitations, which were presented in Chapter 1, are duplicated here for the

reader's convenience. Any interpretations of the study's results should be considered relative to them. Following this listing, a set of recommendations for future research relative to the study's limitations and delimitations is provided and discussed.

Limitations

1. Sample selection and participant assignment. A sample of convenience was used for this study and consisted of intact classes of students who self-enrolled in the targeted course, Computer Science 1. Neither random selection of participants nor random assignment of individual participants to treatment order was possible. This limitation was mitigated somewhat by the initial random assignment of intact classes to treatment order coupled by the rotation between schools and semesters. Similar studies should take into account the self-enrollment nature of this sample as well as the use of intact classes.
2. Course Curriculum. The content of the targeted course, Computer Science 1, was designed to match the recommendations of the ACM curricular guide (ACM, 2001). This is the accepted standard for CS 1 type courses throughout the United States. Additionally, all programming was done in C++ on IBM compatible microcomputers using Microsoft Windows XP as the operating system. This was the default language and hardware platform at the targeted universities. The reader should note that many schools use different languages

and operating systems. Thus, any generalization of these results should be considered relative to the language and operating system used.

3. Student Demographics. The demographics of the sample used in this study, which affected its results, were beyond my control. Students who comprised this study were a mixture of freshmen, sophomores, and community college transfers with junior status. The majority of the participants were computer science, computer information system, and math majors. The targeted student characteristics of gender, previous computer programming education, and learning styles had no significant effect on achievement or attitude, but math background was found to have a significant effect on achievement. Similar studies with different sample demographic conditions might result in different findings.

Delimitations

1. Study location. The sample was comprised of students who were attending one of two public regional liberal arts universities located in northern Alabama. These schools were selected due to their convenience for the researcher. Although a comparison analysis of available schools' and U.S. colleges' undergraduate computer science student demographic data indicated that the sample from these two schools was reasonably representative of all U.S. colleges, it is possible that studies using other locales could have different results.

2. Time of study. The sample consisted of students who self-enrolled for credit in an introductory computer science course during one of two 14-week semesters: Fall 2006 and Spring 2007. Computer science is a rapidly changing field. Thus, previous or later studies, or studies conducted during the summer term, could result in different findings.
3. Duration of the study. This study was implemented in 2 consecutive semesters. This was necessary because a sufficiently large sample size based on power analysis was not feasible in a single term due to historically low per-term enrollment in the introductory computer science course at the targeted universities ($M = 20$ per school). Semester and school variables were included in the statistical analysis to control for potential semester and school effects. If the study were limited to 1 semester or extended over a greater time, it might have provided different results. Only studies with similar durations should be compared to this study.
4. Study design. A counter-balanced repeated measures design was used for this study. Each semester was divided into two equal sessions and participants from four intact classes (two classes per semester) experienced treatment and control (i.e., class structures), one each session. This design was chosen due to circumstances at the participating institutions. Other study designs could lead to different results.

5. Achievement instruments. This study measured student achievement using researcher-prepared laboratory activities, programming assignments, and multi-chapter unit tests based on traditional introductory computer science textbooks, lab manuals, and textbook-provided chapter tests. These instruments were reviewed for content validity by the researcher. They also had a calculated reliability coefficient between .74 and .83, which exceeded the generally accepted minimum measure for deriving inferences within educational research (Cohen et al., 2003). It is possible that different instruments could yield different results.
6. Instrument scoring. Researcher-designed rubrics were used to score programming and laboratory assignments. Two of the participating instructors scored the instruments. The inter-rater reliability of .93 indicated acceptable consistency between scorers. Because student achievement results were directly related to these rubrics and by the scorers' interpretations of how to use them, different results might be achieved with different rubrics or scorers.
7. Participating instructors. The researcher, a female assistant professor with 20 years teaching experience (10 years teaching computer science courses), was the class and lab instructor for the classes at UNA. A male adjunct professor with 10 years experience teaching computer science courses was the class and lab instructor for the fall class at ASU. A female assistant professor with 8 years experience teaching computer science was the class and lab instructor for

the spring class at ASU. Although instructor differences were examined indirectly via the school/semester variables of Set B, it is still conceivable that they may have contributed in part to this study's results. The reader is cautioned not to ignore this potential instructor effect in comparison studies.

8. Class design. All groups received 150 minutes of classroom time per week. The control group had 150 minutes of lecture plus class discussion, and the experimental group had their instruction time divided between lecture and laboratory. This design of integrating the closed lab into the regularly scheduled class time was chosen due to restrictions at the targeted schools. Readers should take this into account when comparing results from other studies that use a longer laboratory scheduled separately from class time.
9. Attitude instrument. Attitudes were assessed using Newby and Fisher's (1997) Attitude towards Computers and Computer Courses instrument, which was designed to measure college level students' attitudes taking a computer science course with a laboratory. Although the reliability coefficient of this instrument based on sample data was .95, different results might be found using a different attitude scale.
10. Learning styles. Students' learning styles were assessed using Kolb's (2005) Learning Style Inventory-Version 3.1 (KLSI-3.1). As noted earlier in this chapter, there are several different methods to categorize students' learning styles. Thus, it is conceivable that studies using a different learning styles

instrument [e.g., the Myers-Briggs Type indicator (Meyers, 1962, 1975) or the Gregorc (1984) Style Delineator] most likely will get different results.

11. Achievement measure. Achievement was measured using the scores obtained from unit exams, laboratory assignments, and programming assignments. The weighting of these different assessments relative to students' final achievement scores was researcher-determined and based on her teaching experience. Other studies that use different assessment methods or weights to measure achievement might obtain different results.

Recommendations for Research

The primary purpose of this study was to determine the effect of two different class structures (open lab-based vs. closed lab-based) on students' achievement and attitudes toward computers and computer courses in an introductory computer science course (CS 1). In previous sections of this chapter, inferences and implications about the findings were discussed, and a discussion about the study's generalizability and limitations and delimitations were provided. In this section, three different but related sets of recommendations are presented: recommendations for future research relative to the study's limitations, recommendations for future research relative to the study's implications, and recommendations for practice relative to these implications.

Recommendations for Future Research

Relative to Study Limitations and Delimitations

1. Students were not randomly selected and intact classes were used for this study. This is a common situation in educational research. The reality of performing a study in an academic setting limits the choices of the researcher. Thus, to validate or refute the findings of this study, it is recommended that it be replicated at other colleges.
2. All programming was done in C++ on IBM compatible microcomputers using Microsoft Windows XP as the operating system. Many schools now use other languages such as Java in the introductory programming course and encourage the use of other platforms, especially the use of Linux as the default operating system. Using a different language or a different operating system could result in different conclusions. Thus, it is recommended that this study be replicated under these differing conditions.
3. The majority of the participants in this study were computer science, computer information system, and math majors. Although CS, CIS, and mathematics represent the most common majors of students who take CS 1, it is not uncommon for students of other majors to also take CS 1, including students majoring in engineering, information technology (IT), business, or software engineering. As a result, if appropriate sample sizes are available,

it is recommended that this study be replicated using students of these and other majors.

4. This study was performed at two public regional liberal arts universities located in northern Alabama. In order to validate its findings and its ecological generalizability, it is recommended that similar studies be conducted at other universities located on other parts of Alabama and throughout the nation.
5. This study was implemented during the 2006–2007 school year. Due to the rapid, ever-changing nature of students in computer science courses and the field itself, it is recommended that similar studies be conducted on a fairly regular basis to capture any changes that might be contrary to those of the current study.
6. In an effort to obtain a larger sample, this study was conducted over 2 consecutive semesters, fall and spring of the 2006–2007 school year. Although there were differences between each semester's schedule (e.g., number of class meetings and observed holidays), as well as differences between the students in each group, a variable that was used to capture a possible semester effect showed no significant differences between the achievement and attitude measures. Because of the potential for a semester effect, it is recommended that future studies similar to the current one be

conducted over a single semester to mitigate or eliminate the potential for confounding results from different semester schedules.

7. This study had a counterbalanced repeated measures design. A repeated measures design is usually not used in situations where there is a significant building effect, for instance, the one inherent in a programming course such as CS 1. Counter balancing the order of treatment was used to mitigate this building effect. Future studies to replicate this experiment should consider using an experimental design or a qualitative research method or complementing the quantitative results with additional qualitative data such as interviews, observations, and open-ended questions on surveys.

An associated problem was the manner in which participants were assigned to treatment or control groups: Intact classes were used with random assignment of the order of treatment (i.e., open lab-based first or closed lab-based first) using a repeated measures subject design with counterbalancing. Obviously, random assignment of participants to groups would have been preferred. In reality, this is often not feasible within the constraints of an educational setting. Although independent variables of semester, school, and the interaction between semester and school captured the effect of the treatment/control order and were not significant, it is recommended that this study be replicated using random assignment to treatment order to see if the results differ from those of the current study. It

would also be interesting to see if the results would be similar to an effects-type causal-comparative design. Thus, a related recommendation is to examine a set of data from this design perspective.

8. Achievement results were based on laboratory exercises, programming exercises, and unit exams constructed by the researcher. Scores on unit exams made up approximately 72% of the achievement score. Because the majority of similar studies conducted to date have relied on only a single achievement measure (e.g., final exam score), it is recommended that future studies similar to the current one use multiple achievement measures.
9. Laboratory and programming exercises were scored using two researcher-constructed rubrics. These rubrics were validated by the researcher as an experienced computer science teacher. Further studies should improve on these rubrics or use them and build on their reputation as valid and reliable instruments. Additionally, the unit exams contained dichotomous questions and were manually scored. Manual scoring can lead to errors. It is recommended that machine scoring be used for future studies for dichotomous questions and that future researchers include other types of questions for the unit exams.
10. Three instructors implemented this study: I implemented it during the fall and spring terms at UNA, a second instructor implemented it during the fall term at ASU, and a third instructor implemented it during the spring term at

ASU. Although no instructor differences were found, it is important to note that instructor differences were not directly measured but were instead measured indirectly via the school/semester variables. As a result, future studies similar to this one should incorporate a method to examine a potential instructor effect directly and not indirectly. Also, because it is possible that my expectations and biases might have influenced my performance as an instructor or the performance of my students, a related recommendation for future studies similar to the current one is to have the researcher not participate as an instructor but instead serve as an observer.

11. In this study students received the same amount of instructional time, 150 minutes per week. Each treatment also received approximately the same amount of lecture time. In the open lab-based class, time was allotted at the beginning of the class for discussion on the previous lab assignment and at the end of class for questions about the days lecture. In the closed lab-based class time was allotted at the end of each class to work on the laboratory assignment. It is possible that the discussion and question time allotted in the open lab-based class but not in the closed lab-based class may have affected the results of this study. Similar future studies should use other activities during the extra instructor time of the open lab-based class.
12. This study measured students' attitudes using Newby and Fisher's (1997) Attitude towards Computers and Computer Courses (ACCC). The ACCC is

is a fairly long instrument and other researchers may find using one or two of the subscales more suitable. Alternatively, researchers of future similar studies should consider whether the ACCC is appropriate for their purposes and might consider using a different attitude scale or constructing their own. Researchers might also want to consider complementing attitude measures with classroom observations and student interviews.

13. Kolb's (2005) Learning Style Inventory-Version 3.1 (KLSI-3.1) was used in this study to assess students' learning styles and two dominant learning styles were detected: assimilators and convergers. Given this result, researchers of future studies similar to the current one might consider a purposive sample that ensures all learning styles are well represented. A related recommendation for future research is to replicate this study using a different learning style assessment and see if the results are similar to those found in this study.
14. This research used a combination of laboratory assignments, programming assignments, and unit test scores to measure achievement. There is no accepted standard way to measure achievement in computer science courses. New research should be designed to investigate differences or similarities in common methods of measuring achievement in computer science courses.

Recommendations for Future Research Relative to Implications

1. A power analysis performed in preparation for this study resulted in a minimum sample size of 70. For this analysis, alpha was set at .05 and power at .80. Because no information was available on which to judge the actual effect size, an effect size of .25 was chosen. This was a generic effect size often used in such situations and recommended by Cohen et al. (2003). This choice of a medium-to-large effect size was also justified by the considerable effort and expense necessary to implement and maintain classes structured to include a closed laboratory. The final sample size of 71, doubled to 142 due to the repeated measures design, was larger than the desired one, however, the final effect size was much less than the hypothesized .25. As was stated earlier, a possible explanation for not finding a significant effect could be associated with the alpha, power, and sample size used in this study. Consequently, it is recommended that future similar studies use a larger sample size. A larger sample size will increase power and the chance of detecting a small effect. There should be a balance, however, among the three parameters of alpha, power, and sample size. A very large sample increases the probability of finding a statistically significant effect that is so small that it has no practical significance.
2. A second recommendation for future research is relative to some of the student characteristic variables used in this study. The choice of student

characteristics included was based on extensive literature review and was an attempt to remove the effects of subject characteristics on attitude and achievement prior to investigating the effects of treatment. The targeted student characteristics of gender, previous computer programming education, and learning style, were not significantly related to either of the dependent measures. This could have been due to the small size or homogeneous nature relative to these characteristics of the sample. Therefore, a recommendation for future research is to replicate this study using a larger more diverse sample, or to conduct other similar computer science-related studies that involve a more diverse sample of college students. Alternately, it could be that these subject characteristics are not important to student achievement or attitude. Thus, future studies should consider not including gender, previous computer programming education, or learning styles as variables. The use of fewer independent variables would allow for higher power with the same sample size.

3. As noted above, learning style was not found to significantly affect achievement or attitude. The design of the two class structures (open lab bases and closed lab-based) provided a variety of different learning environments that were congruent with the four learning styles. A suggestion for future research is to investigate what would happen if these multiple learning environments were not provided.

4. This study, in agreement with past studies, found that math background was significantly related to achievement. Thus, future research studies involving similar participants, with achievement as a dependent measure, should include math background as an independent measure whose influence should be removed before calculating the significance of the study's treatment.
5. This study found there was a significant difference between the achievement scores of students with a just a College Algebra course and students with Calculus I as their highest level of math preparation and between students with just a College Algebra course and students with math courses above Calculus I. Students with Trigonometry as their highest math did not perform significantly better than those with only a College Algebra course. This breakdown of math background into specific course categories needs further study to determine which math courses most benefit students in early computer science courses. For example, it is possible that students who took a precalculus-trig course did not realize that they had taken a trigonometry "course" and this might explain why the trig-college algebra comparison was not significant. It is also important to consider whether high school math courses produce the same effect as college math courses. Finally, other studies should consider whether the *type* of calculus course (e.g., business calculus, Calculus I with early trig, or Calculus I offered

during a 10-week term vs. a 15-week term) affects the level of significance of the math background results.

6. Of the targeted student characteristics, only math background was found to have a significant effect on student achievement. Thus, one suggestion for future research is to investigate other student characteristics that were not targeted, such as student age and major. Additionally for CS and CIS majors this is a gateway course, while for other majors this is a terminal course. This situation is shared by other courses such as college algebra. It is either a terminal course or a gateway course depending on the major of the student. Thus, like CS 1, not only does it consist of students who have a wide diversity in math ability and background, but students view the purpose of the course differently. This makes it difficult for teachers to teach such courses. It is possible that the student characteristics of age and major would have a significant effect and future studies should consider using them as variables.
7. The student characteristic of previous computer programming education did not have a significant effect on either student achievement or attitude. At both of this study's targeted schools, a pre-CS 1 programming course is offered. As evidenced by several articles and conference presentations (e.g., Bruce, Fowler, Guzdial, King, & Woszynski, 2005; Cook, 1997; Dierbach, Taylor, Zhou, & Zimand, 2005; McFarland, 2004; Mitchell,

2001) this “CS 0” course is gaining in popularity. Controlled studies need to be performed to discover the benefits of such a course and to determine if it is worth the time, effort, and expense for both students and schools.

8. Finally, this study included two dependent variables, achievement and attitude. The attitudes toward computers and computer courses of a large percentage of the students in this study were initially very positive and did not change significantly. Consequently, it is recommended that future studies do not include attitude as a dependent measure.

Recommendations for Practice Relative to Implications

This study was one of a very few studies that compared the effects of class structure (open lab-based vs. closed lab-based) on achievement and attitudes of computer science students. As a consequence, recommendations for practice should be viewed with caution. However, the implications of this study warrant several recommendations.

1. Instructors in CS 1 type courses cover many different topics during the semester. Some of these topics are straightforward and fairly easy to explain, allowing time for a closed laboratory exercise. Other topics, however, are more difficult and time consuming. It is unlikely that instructors would wish to shorten lecture and class discussion time on these difficult topics in order to incorporate a closed laboratory exercise. Golden (1990) advocated balancing the constructivist approach with more

traditional approaches. The results of this study imply that both class structures (open and closed lab-based) are equally beneficial. Thus, one recommendation for practice relative to this study's implications is instructors of CS 1 type courses should feel free to encourage their students to complete short laboratory assignments during class, time permitting, or as an alternative, as homework assignments.

2. An implication of this study was the role of math background in determining student achievement. Instructors are encouraged to survey their students to determine their math background and use this information to identify students who may need extra attention.
3. At the researcher's home school, several students who took part in this study continued in the computer science major and took the CS 2 course the following semester. These students were disappointed to discover that the CS 2 course did not include regular laboratory activities. They commented that they found that the labs (1) made it easier to understand the topic, (2) helped them prepare for the longer programming assignments, and (3) kept them working each class session so they didn't get behind. These comments suggested that the laboratory activities were an important contribution to their meaningful learning set (Ausubel, 1960). Thus, instructors are encouraged to develop very short laboratory assignments similar to those used in this study for other computer science programming courses.

4. During the course of this study, students who missed classes and thus did not turn in laboratory assignments on time were penalized. This procedure created problems with the instructors at ASU and the students at both schools. It is therefore recommended that instructors who intend to use the class structures of this study allow ample time for laboratory assignment completion and make-up in the event of extended student absences.
5. In preparing for this study, I was surprised to find few studies conclusively indicating that closed laboratory-based classes were superior to open laboratory-based classes. It is my perception that the prevailing belief of computer science instructors is in favor of classes structured to include closed laboratories. This is not astounding because it is a reasonable assumption, albeit not one supported by research. There is a need for teacher professional development at the college level to inform teachers about the value of research and the need to stay current not only in their field of expertise, but also in the methods of teaching their subject. Therefore, a final recommendation for practice is that schools and universities should institute an on-campus schedule of methods related professional development opportunities and should encourage its instructors and professors to attend off-campus conferences featuring similar topics.

REFERENCES

- ACM. (1979). Curriculum 78: Recommendations for the undergraduate program in computer science. *Communications of the ACM*, 22(3), 147–166.
- ACM. (2001). *Computing curricula*. Retrieved April 3, 2004, from <http://www.computer.org/education/cc2001/final/index.htm>
- Allitt-Wheeler, S. (2005). Comparison of student achievement in open versus closed computer laboratories (Doctoral dissertation, Illinois State University, 2005). *Dissertation Abstracts International*, 66(11), 3988.
- Ausubel, D. P. (1968). *Educational psychology: A cognitive view*. New York: Grune & Stratton.
- Avitabile, J. (1998). Interaction of presentation mode and learning style in computer science. *Proceeding of the National Educating Computing Conference*. 1–20.
- Beckenstein, W., & Staunton, M. (1998). *The effects of computer location on first graders' usage and enjoyment of computers*. (ERIC Document Reproduction Service No. ED419523)
- Bergin, S., & Reilly, R. (2005). Programming: Factors that influence success. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*. 411–415.

- Bergin, S., Reilly, R., & Traynor, D. (2005). Examining the role of self-regulated learning on introductory programming performance. *Proceedings of the 2005 International Workshop on Computing Education Research*, 81–86.
- Bloom, B. S. (1956). *Taxonomy of educational objectives handbook I: The cognitive domain*. New York: David McKay Co Inc.
- Borkowski, J., & Muthukrishna, N. (1992). Moving metacognition into the classroom: “Working models” and effective strategy teaching. In E. McIntyre & M. Pressley (Eds.), *Balanced instruction: Strategies and skills in whole language*. Norwood, MA: Christopher-Gordon.
- Borkowski, J. G., & Thorp, P. K. (1996). In D. H. Schunk & B. J. Zimmerman (Eds.), *Self-regulation of learning and performance*. Hillsdale, NJ: Lawrence Erlbaum.
- Bostrom, R., Olfman, L., & Sein, M. (1993). Learning styles and end-user training: A first step. *MIS Quarterly*, 118–120.
- Brooks, J. G., & Brooks, M. G. (1993). *In search of understanding: The case for constructivist classrooms*. Alexandria, Virginia: Association for Supervision and Curriculum Development.
- Bruce, R., Fowler, C., Guzdial, M., King, M. S., & Woszynski, A. (2005). CS0/CS1: Filter or funnel: Recruitment, retention and student success. *Proceedings of the 43rd Annual ACM Southeast Regional Conference*, 1, 29–30.

- Butcher, D. F., & Muth, W. A. (1985). Predicting performance in an introductory computer science course. *Communications of the ACM*, 28(3), 263–268.
- Burton, D. (1992). *The effect of closed laboratory activities on the comprehension of five concepts and the perception of effectiveness of the course in a second-semester computer science course*. Unpublished doctoral dissertation, The University of Texas at Austin.
- Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*. 49–52.
- Byrnes, J. P. (1996). *Cognitive development and learning in instructional contexts*. Boston: Allyn & Bacon.
- Campbell, P., & McCabe, G. (1984). Predicting the success of freshmen in a computer science major. *Communications of the ACM*, 27(11), 1108–1113.
- Chavey, D. (1991). A structured laboratory component for the introductory programming course. *The Papers of the Twenty-Second SIGCSE Technical Symposium on Computer Science Education*, 23, 287–295.
- Cohen, J., Cohen, P., West, S., & Aiken, L. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences* (3rd ed.). Mahwah, NJ: Lawrence Erlbaum.

- Cook, C. (1997). CS0: Computer science orientation course. *ACM SIGCSE Bulletin, Proceedings of the Twenty-Eighth SIGCSE Technical Symposium on Computer Science Education*, 29(1), 87–91.
- Corritore, C. L., Hickman, B.I., Grandgenett, N., & Hitchcock, R. (1999). A comparison of two laboratory-based approaches for teaching introductory computer science. *Innovations in Education and Training International* 36(4), 292–301.
- Creswell, J. W. (2005). *Educational research: Planning, conducting, and evaluating quantitative and qualitative research* (2nd ed.). Upper Saddle River, NJ: Prentice-Hall.
- Crosby, M., & Stelovsky, J. (1995). From multimedia instruction to multimedia evaluation. *Journal of Educational Multimedia and Hypermedia*, 4(2–3), 147–162.
- Dale, N. (2004). *A laboratory course in C++* (4th ed.). Sudbury, MA: Jones and Bartlett.
- Dale, N. (2005). Content and emphasis in CS1. *ACM SIGCSE Bulletin*, 37(4), 69–73.
- Dale, N., & Weems, C. (2004). *Programming and problem solving with C++* (4th ed.). Sudbury, MA: Jones and Bartlett.
- Davidson, G., Savenye, W., & Orr, K. (1992). How do learning styles relate to performance in a computer applications course? *Journal of Research on Computing in Education*, 24, 348–358.

- Davidson, G. V. (1990). Matching learning styles with teaching styles: Is it a useful concept in instruction? *Performance and Instruction*, 29(4), 36–38.
- Deitel, H. M., & Deitel, P. J. (2005). *Small C++: How to program*. Upper Saddle River, NJ: Prentice-Hall.
- Dewey, J. (1897). My pedagogic creed. *School Journal* 54, 77–80. Retrieved January 20, 2006 from <http://dewey.pragmatism.org/creed.htm>
- Dierbach, C., Taylor, B., Zhou, H., & Zimand, I. (2005). Experiences with a CS0 course targeted for CS1 success. *ACM SIGCSE Bulletin, Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, 37(1), 317–320.
- Driver, R., & Scanlon, E. (1988). Conceptual change in science. *Journal of Computer Assisted Learning* 5, 25–36.
- Duplass, J. A. (1995). Teaching software: Is the supervised laboratory effective? *Computers & Education*, 24(4), 287–291.
- Edwards, T. G. (1994). A reflective cycle: The evolution of a model of teacher change. Paper presented to the Sixteenth Annual Conference of the North American Chapter of the International Group for the Psychology of Mathematics Education. Baton Rouge, LA.
- Engelmann, S., & Carnine, D. (1991). *Theory of instruction*. Eugene, OR: ADI Press.

- Evans, G. E., & Simkin, M.G. (1989). What best predicts computer proficiency?
Communications of the ACM, 32(11), 1322–1328.
- Faraway, F. F., (2005). *Extending the linear model with R: Generalized linear, mixed effects and nonparametric regression models*. Boca Raton, FL: CRC Press.
- Fraenkel, J. R., & Wallen, N. E. (2003). *How to design and evaluate research in education* (5th ed.). Boston: McGraw Hill.
- Geitz, R. (1994). Concepts in the classroom, programming in the lab. *The Papers of the Twenty-Fifth SIGCSE Technical Symposium on Computer Science Education*, 26, 164–168.
- Gregorc, A. F. (1984). *Gregorc style delineator: Development, technical and administration manual*. Columbia, ST: Gregorc Associates, Inc.
- Goldin, G. (1990). Epistemology, constructivism, and discovery learning in mathematics. *Journal for Research in Mathematics, Monograph 4*, 31–47.
- Haase, R. F., & Ellis, M. V. (1987). Multivariate analysis of variance. *Journal of Counseling Psychology*, 34(4), 404–413.
- Hartel, P. H., & Hertzberger, L. O. (1995). Paradigms and laboratories in the core computer science curriculum: An overview. *ACM SIGCSE Bulletin*, 27(4), 13–20.

- Heshusius, L. (1995). Holism and special education: There is no substitute for real life purposes and processes. In T. M. Skrtic (Ed.), *Disability and democracy: Reconstructing (special) education for postmodernity* (pp. 166–189). New York: Teachers College, Columbia University.
- Hagan D., & Markham S. (2000). Does it help to have some programming experience before beginning a computing degree program? *Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education*. 25–28.
- Houston, D. M. (1993). An exploration and analysis of the relationship among learning styles, teaching styles, gender and performance (Doctoral dissertation, Kansas State University, 1993). *Dissertation Abstracts International*, 54(8A), 2867.
- Indiana University Center for Postsecondary Research (2007). *National survey of student engagement*. Retrieved August 21, 2007 from <http://nsse.iub.edu/index.cfm>
- Kayes, D. C (2005). Internal validity and reliability of Kolb's learning style inventory version 3 (1999). *Journal of Business and Psychology*, 20(2), 249.
- Knight, J. (2003). *Crossing boundaries: What constructivist instruction can teach us about intensive-explicit instruction, and what intensive-explicit*

- instruction can teach us about constructivist instruction*. Retrieved October 4, 2005 from <http://www.ku-crl.org/htmlfiles/SE2005/constructivist.rtf>
- Kolb, D. A. (1976). *Learning style inventory: Self-scoring test and interpretation booklet*. Boston: McBer and Company.
- Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. Upper Saddle River, NJ: Prentice-Hall.
- Kolb, D. A. (1999). *Learning style inventory, version 3*. Boston: Hay/McBer.
- Kolb, D. A. (2005). *Learning style inventory, version 3.1*. Boston: Hay/McBer.
- Kolb, D., & Kolb S. (2005) *The Kolb learning style inventory—Version 3.1 2005 technical specifications*. Retrieved January 3, 2006 from http://www.hayresourcesdirect.haygroup.com/Learning_Self-Development/Assessments_surveys/Learning_Style_Inventory/Downloads/lsi%20tech%20manual.pdf
- Konvalina, J., Stephens, L., & Wileman, S. (1983). Identifying factors influencing computer science aptitude and achievement. *AEDS Journal*, 16(2),106–112.
- Kumar, A. N. (2003). The effects of closed labs in computer science I: An assessment. *Journal of Computing Science in Colleges*, 18(5), 40–48.
- Kumar, V., Winne, P., Hadwin, A., Nesbit, J., Jamieson-Noel, D., Calvert, T., et al. (2005). Effects of self-regulated learning in programming. *Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies*, 383–387.

- Lewin, K., Lippitt, R., & White, R. (1939). Pattern of aggressive behavior in experimentally created "social climates." *Journal of Social Psychology, 10*, 271–299.
- Loyd, B. H., & Loyd, D. E. (1985). The reliability and validity of an instrument for the assessment of computer attitudes. *Educational and Psychological Measurement, 45*, 903–908.
- Mather, M., & Adams, D. (2007). *The crossover in male-female enrollment rates*. Population Reference Bureau. Retrieved September 19, 2007 from <http://www.prb.org/Articles/2007/CrossoverinFemaleMaleCollegeEnrollmentRates.aspx>
- McFarland, R. D. (2004). Development of a CS0 course at Western New Mexico University. *Journal of Computing Sciences in Colleges, 20*(1), 308–313.
- Meyers, I. B. (1962, 1975) *The Meyers-Briggs type indicator: Manual*. Palo Alto, CA: Consulting Psychological Press.
- Mitchell, W. (2001). Another look at CS0. *Journal of Computing Science in Colleges, 17*(1), 194–205.
- National Science Foundation. (2005). Division of Science Resources Statistics. *Table C-14: Bachelor's degrees, by race/ethnicity, citizenship, sex, and field: 2004*. Retrieved September 30, 2007 from <http://www.nsf.gov/statistics/wmpd/underdeg.htm#bachelor>

- Newby, M., & Fisher, D. L. (1997). Development and use of the Computer Laboratory Environment Inventory. *Proceedings, 14th Annual Conference of the Australian Society for Computers in Tertiary Education*. 430–435.
- Nussbaum, J., & Novick, N. (1982). Alternative frameworks, conceptual conflict, and accommodation: Toward a principled teaching strategy. *Instructional Science*, 11, 183–200.
- Ormrod, J. E. (2004). *Human learning* (4th ed.). Upper Saddle River, NJ: Prentice-Hall.
- Perkins, D. (1999). The many faces of constructivism. *Educational Leadership*, 57(3), 6–11.
- Phillips, D. C. (2000). An opinionated account of the constructivist landscape. In D. C. Phillips (Ed.), *Constructivism in education* (pp. 1–16). Chicago: National Society for the Study of Education, University of Chicago Press.
- Piaget, J. (1964). *The early growth of logic in the child*. New York: Harper & Row.
- Piburn, M. D. (1989). Reliability and validity of the propositional logic test. *Educational and Psychological Measurement*, 49, 667–672.
- Pillay N., & Jugoo, V. R. (2005) An investigation into student characteristics affecting novice programming performance. *ACM SIGCSE Bulletin*, 37(4), 107–110.

- Pines, A. L., & West, L. H. T. (1986). Conceptual understanding and science learning: An interpretation of research with a sources-of-knowledge framework. *Science Education*, 70(5), 583–604.
- Pintrich, P. R., & De Groot, E. V. (1990). Motivational and self-regulated learning components of classroom academic performance. *Journal of Educational Psychology*, 82(1), 33–40.
- Pintrich, P. R., Marx, R. W., & Boyle, R. A. (1993). Beyond cold conceptual change: The role of motivational beliefs and classroom contextual factors in the process of conceptual change. *Review of Educational Research*, 63(2), 167–199.
- Pintrich, P. R., Smith, D. A., Garcia, T., & McKeachie, W. J. (1991). *A manual for the use of the motivated strategies for learning questionnaire (MSLQ)*. Ann Arbor, Michigan: National Center for Research to Improve Postsecondary Teaching and Learning, University of Michigan.
- Poplin, M. S. (1988a). Holistic/constructivist principles of the teaching/learning process: Implications for the field of learning disabilities. *Journal of Learning Disabilities*, 21(7), 401–416.
- Poplin, M. S. (1988b). The reductionist fallacy in learning disabilities: Replicating the past by reducing the present. *Journal of Learning Disabilities*, 21(7), 380–400.

- Priebe, R. (1997). The effects of cooperative learning in a second-semester university computer science course. (ERIC Document Reproduction Service No. ED406189)
- Ramaingam, V., & Wiedenbeck, S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analysis of novice programmer self-efficacy. *Journal of Educational Computing Research, 19*(4), 367–381.
- Raths, J. (1973). The emperor's clothes phenomenon in science education. *Journal of Research in Science Teaching, 10*(3), 201–211.
- Sabin, R. E., & Sabin, E. P. (1994). Collaborative learning in an introductory computer science course. *ACM SIGCSE Bulletin, 26*(1), 304–328.
- Segal, L. (2001). *The dream of reality: Heinz von Foerster's constructivism* (2nd ed.). New York: Springer.
- Shaver, J. H. (1983). The verification of independent variables in teaching methods research. *Educational Researcher, 12*(8), 3–9.
- Smith, P. (1996). A process education approach to teaching computer science. *Proceedings of the 1996 Annual ASCUE Conference*. Retrieved February 15, 2006 from <http://www.saintmarys.edu/psmith/psmith96.html>
- Stevens, J. P. (2002). *Applied multivariate statistics for the social sciences* (4th ed.). Mahwah, NJ: Lawrence Erlbaum Associates.

- Taylor, H. G., & Mounfield L. (1991). An analysis of success factors in college computer science: High school methodology is a key. *Journal of Research on Computing in Education*, 24(2), 240–246.
- Thomas, R., & Upah, S. (1998). Give programming instruction a chance. *Journal of Research on Computing in Education*, 29(1), 96–109.
- Thweatt, M. (1994). CS1 closed lab vs. open lab experiment. *The Papers of the Twenty-Fifth SIGCSE Technical Symposium on Computer Science Education*, 26, 80–82.
- University of North Alabama. (2007). *UNA 2005–2006 institutional data book*. Retrieved June 10, 2007 from <http://www2.una.edu/research/DataBook%202006.pdf>
- Veres, J. G., Sims, R. R., & Locklear, T. S. (1991). Improving the reliability of Kolb's revised LSI. *Educational and Psychological Measurement*, 51, 143–150.
- Vygotsky, L. S. (1987). *The collected works of L. S. Vygotsky* (R. W. Rieber & A. S. Carton, Eds.). New York: Plenum Press.
- Wierstra, R. F. A., & DeJong, F. P. C. M. (2002). A scaling theoretical evaluation of Kolb's Learning Style Inventory-2. In M. Vacke & D. Gombeir (Eds.) *Learning styles: Reliability and validity* (pp. 431–440). Proceedings of the 7th annual ELSIN Conference, 26-28 June, Ghent: University of Ghent.

- Wilson, B. (2002). A study of factors promoting success in computer science including gender differences. *Computer Science Education*, 12(1-2), 141-164.
- Walker, G. N. (2004). Experimentation in the computer programming lab. *ACM SIGCSE Bulletin*, 36(4), 69-72.
- Wu, C., Lin, J. M., & Hsu, I. Y. (1997). Closed laboratories using SimLIST and SimRECUR. *Computers & Education*, 28(1), 55-64.
- Yager, R. E. (1991). The constructivist learning model. *The Science Teacher*, 51(6), 52-58.
- Zak, D. (2005). *An introduction to programming C++* (4th ed.). Belmont, CA: Thomson Course Technology.
- Zweben, S. (2007). 2005-2006 Taublee Survey: Record Ph.D. production continues; Undergraduate enrollments turning the corner. *Computing Research News*, 19(3), 7-22.

APPENDIX A
INSTRUMENTS

Sample of KLSI - 3.1

LEARNING-STYLE INVENTORY

The Learning-Style Inventory describes the way you learn and how you deal with ideas and day-to-day situations in your life. Below are 12 sentences with a choice of endings. Rank the endings for each sentence according to how well you think each one fits with how you would go about learning something. Try to recall some recent situations where you had to learn something new, perhaps in your job or at school. Then, using the spaces provided, rank a "4" for the sentence ending that describes how you learn best, down to a "1" for the sentence ending that seems least like the way you learn. Be sure to rank all the endings for each sentence unit. Please do not make ties.

Example of completed sentence set:

1. When I learn: 2 I am happy. 1 I am fast. 3 I am logical. 4 I am careful.

Remember: 4 = most like you 3 = second most like you 2 = third most like you 1 = least like you

	A		B		C		D	
1. When I learn:	—	I like to deal with my feelings.	—	I like to think about ideas.	—	I like to be doing things.	—	I like to watch and listen.
2. I learn best when:	—	I listen and watch carefully.	—	I rely on logical thinking.	—	I trust my hunches and feelings.	—	I work hard to get things done.
3. When I am learning:	—	I tend to reason things out.	—	I am responsible about things.	—	I am quiet and reserved.	—	I have strong feelings and reactions.
4. I learn by:	—	feeling.	—	doing.	—	watching.	—	thinking.
5. When I learn:	—	I analyze.	SAMPLE				—	I like to try things out.
6. When I am learning:	—	I am practical.	SAMPLE				—	I am a logical person.
7. I learn best from:	—	observing.	SAMPLE				—	a chance to try out and practice.
8. When I learn:	—	I like to see results from my work.	—	I like ideas and theories.	—	I take my time before acting.	—	I feel personally involved in things.
9. I learn best when:	—	I rely on my observations.	—	I rely on my feelings.	—	I can try things out for myself.	—	I rely on my ideas.
10. When I am learning:	—	I am a reserved person.	—	I am an accepting person.	—	I am a responsible person.	—	I am a rational person.
11. When I learn:	—	I get involved.	—	I like to observe.	—	I evaluate things.	—	I like to be active.
12. I learn best when:	—	I analyze ideas.	—	I am receptive and open-minded.	—	I am careful.	—	I am practical.

MCB200C

© 1993 David A. Kolb, Experience-Based Learning Systems, Inc. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means without permission in writing from the Hay Group 116 Huntington Ave., Boston, MA 02116. Telephone 1 800 729 8074 / 1 617 425 4500.

Attitude towards Computers and Computer Courses Scale

Below are 28 statements about computers and computer science that represent a variety of opinions. This is not a test and there are no right or wrong answers. Please read each statement carefully and then circle the number that most closely corresponds to the way you personally feel about the given statement. The number you circle should correspond to the following scale.

	Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree
	1	2	3	4	5
1. I do not think I will ever use what I learned in this class.	1	2	3	4	5
2. I feel comfortable when a conversation turns to computers.	1	2	3	4	5
3. Studying about computers is a waste of time.	1	2	3	4	5
4. It is fun to find out how computer systems work.	1	2	3	4	5
5. This class is providing me with skills I expect to use in the future.	1	2	3	4	5
6. I feel at ease when I am around computers.	1	2	3	4	5
7. My future career will require a knowledge of computers.	1	2	3	4	5
8. I enjoy using a computer.	1	2	3	4	5
9. This class is increasing my technical skills.	1	2	3	4	5

	Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree
	1	2	3	4	5
10. Working with a computer makes me very nervous.	1	2	3	4	5
11. I cannot imagine getting a job that does not involve using computers.	1	2	3	4	5
12. I think working with computers would be enjoyable and stimulating.	1	2	3	4	5
13. I am gaining few useful skills from this class.	1	2	3	4	5
14. I get a sinking feeling when I think about trying to use a computer.	1	2	3	4	5
15. Computers are an important factor in the success of a business.	1	2	3	4	5
16. The challenge of solving problems using a computer does not appeal to me.	1	2	3	4	5
17. The skills gained in this class are too specific to be generally useful in the future.	1	2	3	4	5
18. Computers make me feel uncomfortable.	1	2	3	4	5
19. The use of computers will increase in the future.	1	2	3	4	5
20. I would like to work with computers.	1	2	3	4	5

	Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree
	1	2	3	4	5
21. This class is helping develop my problem-solving skills.	1	2	3	4	5
22. Computers make me feel uneasy and confused.	1	2	3	4	5
23. All college students need a course about using computers.	1	2	3	4	5
24. I enjoy learning on a computer.	1	2	3	4	5
25. As a result of this class I feel confident about tackling unfamiliar problems involving computers.	1	2	3	4	5
26. I feel aggressive and hostile towards computers.	1	2	3	4	5
27. Knowledge of the use of computers will help me get a job.	1	2	3	4	5
28. Learning about computers is boring.	1	2	3	4	5

Test 1

Choose the best answer or ending for each question or statement.

1. Comments that should be ignored by the compiler are denoted using:
 - a. Two forward slashes (//).
 - b. A /* at the beginning of the comment and a */ at the end of the comment
 - c. All of the above
 - d. **None of the above**
2. Which of the following is not a syntax error?
 - a. `std::cout << 'Hello world! ';`
 - b. `std::cout << "Hello
world! ";`
 - c. `std::cout << "Hello world! ";`
 - d. `std::cout << Hello world!;`
3. Which of the following statements would display the phrase C++ is fun?
 - a. `std::cout >> "C++ is fun ";`
 - b. `std::cout >> 'C++ is fun';`
 - c. `std::cout << "C++ is fun";`
 - d. `std::cout << C++ is fun;`
4. Which of the following is not a valid C++ identifier?
 - a. my Value
 - b. _AAA1
 - c. length
 - d. m_x

5. Which is the output of the following statements?

```
std::cout << "Hello\n";  
std::cout << "World";
```

- a. Hello World
- b. HelloWorld
- c. Hello
World
- d. Hello

World

6. Which of the following code segments prints a single line containing hello there with the words separated by a single space?
 - a. `std::cout << "hello " << endl;`
`std::cout << " there";`
 - b. `std::cout << "hello" , " there";`
 - c. `std::cout << "hello \n";`
`std::cout << "there";`
 - d. `std::cout << "hello";`
`std::cout << " there";`

7. Which of the following is a valid variable declaration statement?
 - a. `int total;`
 - b. `#include <iostream>`
 - c. `int main()`
 - d. `// first string entered by user`

8. A(n) _____ enables a program to read data from the user (keyboard).
 - a. `cout` statement
 - b. `cin` statement
 - c. return statement
 - d. variable declaration.

9. How would you write an expression that gives the remainder of dividing integer1 by integer2
 - a. `integer1 / integer2`
 - b. `integer2 / integer1`
 - c. `integer1 % integer2`
 - d. `integer2 % integer1`

10. The value of the C++ expression `11 + 22 % 4` is:
 - a. 13
 - b. 1
 - c. 8
 - d. 16

11. Which of the following is a valid input statement?
 - a. `cin >> studentAge;`
 - b. `cin << studentAge;`
 - c. `studentAge >> cin;`
 - d. `studentAge << cin;`

12. Which of the following statements about the C++ main function is true?
 - a. Every program must have a function named main.
 - b. Program execution begins with the first executable statement in the main function
 - c. The word `int` in the function heading means that the main function returns an integer value (to the operating system).
 - d. All of the statements are true.

13. Which C++ statement is used to return a value from the main indicating that the program has terminated successfully?
 - a. `int main()`
 - b. `return 0;`
 - c. `"return 0";`
 - d. None of the above

14. Which of the following statements will compute the product of a and b and assign the result to x?
 - a. `x = a * b;`
 - b. `a * b = x;`
 - c. `cout << a * b;`
 - d. all of the above

15. A location in the computer's memory that may contain different values at various times throughout the execution of a program is called a
- bit
 - integer
 - constant
 - variable
16. Most calculations are performed by a(an) _____ statement.
- declaration
 - output
 - assignment
 - input
17. _____ are used to document a program and improve its readability
- Declarations
 - Comments
 - Output statements
 - All of the above
18. The body of the main function (and every other function) begins with _____ and ends with _____.
- ()
 - begin end**
 - { }
 - []
19. C++ statements end with
- endl**
 - ;
 - .
 - Nothing
20. To use a function the program must invoke it with a(n) _____
- Message
 - Call
 - Variable
 - Declaration
21. The keyword _____ is used in a function header to indicate that a function does not return a value.
- int**
 - return**
 - void**
 - none of the above
22. If we assume 30 is stored in **result**, which of the following statements will print The product is 30
- cout << The product is << result;**
 - cout << "The product is, result;**
 - cout << "The product is "<< result;**
 - cout << " The product is " << " result";**

23. The _____ program combines the output of the compiler with various library functions to produce an executable image.
- Loader
 - Editor
 - Linker
 - Pre-processor

24. Statement A: All variables must be declared before they are used.
Statement B: All variables must be given a type when they are declared.

Which of the two above statements are true?

- All of the statements are true
- None of the statements are true
- Statement A is true but Statement B is false
- Statement A is false but Statement B is true

25. *Statement X*: The modulus operator % must have integer operands
Statement Y: C++ considers the variable **total** to be the same as
TOTAL

Which of the above statements are true?

- All of the statements are true
- None of the statements are true
- Statement X* is true but *Statement Y* is false
- Statement X* is false but *Statement Y* is true

26. *Statement S*: Declarations must be within the first few lines of the main function.
Statement R: The statement `cout << "\n";` causes the cursor to position at the beginning of the next line on the screen

Which of the above statements are true?

- All of the statements are true
- None of the statements are true
- Statement S* is true but *Statement R* is false
- Statement S* is false but *Statement R* is true

```
# include <iostream>
using namespace std;
```

```
int GetData()
{
    int y;
    cin >> y;
    return y;
}
```

```
void PrintOut(int y)
{
    cout << y;
}
```

```
int main()
{
    int y;
    return 0;
}
```

27. In the main function in the program above what would be a valid call for the user-defined function **GetData**?
- a. **GetData();**
 - b. **int GetData;**
 - c. **y = GetData;**
 - d. **y = GetData();**
28. In the main function in the program above what would be a valid call for the user-defined function **PrintOut**?
- a. **PrintOut(y);**
 - b. **void PrintOut(int y);**
 - c. **y = PrintOut;**
 - d. **y = PrintOut(y);**

Test 2

Choose the best answer or ending for each question or statement.

- To execute multiple statements when an **if** statement's condition is **true**, enclose those statements in a pair of:
 - Parentheses, ().
 - Square Brackets, [].
 - Braces, { }.
 - Angle brackets, <>.
- Which of the following is true of a pseudocode program?
 - They are executed by the computer.
 - They help the programmer "think out" a program.
 - They include declarations and all types of statements.
 - All of the above are false.
- In C++, the condition $(4 > y > 1)$:
 - Evaluates correctly and should not be replaced.
 - Does not evaluate correctly and should be replaced by $(4 > y \ \&\& \ y > 1)$.
 - Evaluates correctly but could be replaced by $(4 > y \ \&\& \ y > 1)$.
 - Does not evaluate correctly and should be replaced by $(4 > y \ || \ y > 1)$.
- Consider the following code, assuming that **x** is an **int** with an initial value of **12**

```
if( x = 6 )  
    cout << x;
```

What is the output?

- 6.
 - 12.
 - Nothing.
 - A syntax error is produced.
- When a programmer wishes to read into a string variable several words separated by spaces they should use
 - `cin << sentence;`
 - `cin.getline(sentence);`
 - `getline(cin, sentence);`
 - all of the above

For questions 6 – 8 use the following class definition

```
class doodles
{
    public:

    doodles()
    {
        x = 0;
        name = "XXX";
    }

    doodles(int y, string name1)
    {
        x = y;
        name = name1;
    }

    int getnumber()
    {
        return x;
    }

    void set( int z, string name2)
    {
        x = z;
        name = name2;
    }

    private:
    int x, string name;
};
```

6. A call to the initializing constructor for doodles is
 - a. doodles mydoodle;
 - b. doodles mydoodle();
 - c. doodles mydoodle(10, "cat");
 - d. a and c are both correct.
7. A call to the "get" doodles class function is
 - a. getnumber();
 - b. cout << getnumber();
 - c. int y = getnumber();
 - d. b and c are both correct.
8. A call to the "set" doodles class function is
 - a. set()
 - b. set(int x, string name);
 - c. set(3, "Bob");
 - d. cout << set(x, name);

9. After execution of the following code, what will be the value of angle if the input value is 10?

```
cin >> angle;
if (angle > 5)
    angle = angle + 5;
else if (angle > 2)
    angle = angle + 10;
```

- a. 10
- b. 15
- c. 20
- d. 25

10. After execution of the following code, what will be the value of angle if the input value is 10?

```
cin >> angle;
if (angle > 5)
    angle = angle + 5;
if (angle > 2)
    angle = angle + 10;
```

- a. 10
- b. 15
- c. 20
- d. 25

11. An object declaration may contain:

- a. Parentheses.
- b. The name of the object.
- c. The name of the class.
- d. All of the above.

12. Calling a member function of an object from outside the class requires

- a. The name of the object followed by the dot separator.
- b. The name of the function followed by open and close parenthesis.
- c. The class name followed by the dot separator.
- d. None of the above

13. In the UML, the top compartment of the rectangle modeling a class contains:

- a. The class's name.
- b. The class's attributes.
- c. The class's behaviors.
- d. All of the above.

14. What is the name of the values the class function call passes to the class function for the parameters? (what are 3 and "Bob" in the class function call mydoodles.set(3, "Bob");)

- a. Arguments.
- b. References.
- c. Objects.
- d. Values.

15. Assuming that text is a variable of type string, what will be the contents of text after the statement `cin >> text;` is executed if the user types `Hello World!` and then presses **Enter**?
- H
 - Hello
 - Hello World
 - Hello World!
16. Attributes of a class are also known as:
- Constructors.
 - Local variables.
 - Data members.
 - Classes.
17. A default constructor has how many parameters?
- 0.
 - 1.
 - 2.
 - Variable.
18. A constructor can specify the return type:
- int.**
 - string.**
 - void.**
 - A constructor cannot specify a return type.**
19. The compiler will implicitly (automatically) create a default constructor if:
- The class does not contain any data members.
 - The programmer specifically requests that the compiler do so.
 - The class does not define any constructors.
 - The class already defines a default constructor.**
20. Statement 1: Class data members are declared in the body of member functions.
Statement 2: Class member functions are usually declared in the private section of the class body.
- Both statements are true.
 - Both statements are false.
 - Statement 1 is true but Statement 2 is false.
 - Statement 1 is false but Statement 2 is true.
21. _____ can access and change the values stored in class data members.
- The main
 - Class member functions
 - Other class data members
 - Client functions
22. The purpose of a "Set" function is
- To initialize or change the values stored in class data members
 - To provide the values stored in class data members
 - To create an object
 - To create and initialize an object.
23. The purpose of a "Get" function is

- a. To initialize or change the values stored in class data members
 - b. To provide the values stored in class data members
 - c. To create an object
 - d. To create and initialize an object.
24. The purpose of an initializing constructor is
- a. To initialize or change the values stored in class data members
 - b. To provide the values stored in class data members
 - c. To create an object
 - d. To create and initialize an object.
25. If a class has an explicitly defined default constructor, written by the class programmer as part of the class, what values are stored in the class data members when an object is declared?
- a. Whatever value was in the memory location last.
 - b. Default values provided by the compiler
 - c. Default values provided by the constructor.
 - d. Values provided by the main or client function.
26. If a class has an initializing constructor, what values are stored in the class data members when an object is declared that calls the initializing constructor.
- a. Whatever value was in the memory location last.
 - b. Default values provided by the compiler
 - c. Default values provided by the constructor.
 - d. Values provided by the main or client function.
27. Every class definition begins with the keyword _____ followed by the class _____.
- a. public :
 - b. class name
 - c. void name
 - d. public name
28. The three types of control structures are:
- a. Sequence, functions, classes
 - b. Functions, classes, main
 - c. Sequence, selection, repetition
 - d. If, while, for
29. The _____ selection statement is used to execute one action when the statement is true or a different action when the statement is false.
- a. if
 - b. if
else
 - c. if
else if
else if ...
else
 - d. all of the above

30. In order to test for multiple cases (more than 2) the _____ selection statement is used.
- a. if
 - b. if
else
 - c. if
else if
else if ...
else
 - d. all of the above

Test 3

Choose the best answer or ending for each question or statement.

1. What is wrong with the following while loop?

```
sum = 200;
while ( sum <= 1000 )
    sum = sum - 30;
```

- a. The parentheses should be braces.
- b. Braces are required around `sum = sum - 30;`.
- c. There should be a semicolon after `while (sum <= 1000)`.
- d. `sum = sum - 30` should be `sum = sum + 30` or else the loop will be infinite.

2. How many times will the following loop print hello?

```
i = 1;
while ( i <= 10 )
    cout << "hello";
```

- a. 0.
 - b. 9.
 - c. 10.
 - d. unknown: the loop is an infinite loop.
3. In an interactive program, indefinite repetition (where you do not know how many times the loop should repeat) should be controlled by a:
- a. Counter.
 - b. Sentinel value.
 - c. Absence of a condition.
 - d. Non-constant condition.
4. To handle situations where a loop must reinitialize a variable at the beginning of each pass through the loop, such reinitialization could be performed by:
- a. An assignment statement as the first statement in the loop body.
 - b. A declaration as the first statement in the loop body.
 - c. An assignment statement after the loop body.
 - d. A declaration after the loop body.
5. If `x` initially contains the value 3, which of the following sets `x` to 7?
- a. `x ++ 4;`
 - b. `x += 4;`
 - c. `x =+ 4;`
 - d. `x + 4 = x;`

6. Which of the following will not increment c by 1?
- `c + 1;`
 - `c++;`
 - `++c;`
 - `c += 1;`
7. Which of the following does counter-controlled repetition require?
- An initial value for the counter.
 - A condition that tests for the final value in the counter.
 - An increment or decrement by which the counter variable is modified each time through the loop.
 - Counter-controlled repetition requires all of the above.
8. If a variable is declared in the initialization expression of a for structure, then:
- It is automatically reinitialized to zero once the loop is finished.
 - The scope of the variable is restricted to that particular for loop.
 - It retains its final value after the loop is finished.
 - It can not be used in any structures that are nested in that for structure.
9. Which of the following **for** headers is not valid?
- `for (int i = 0; i < 10; i++).`
 - `int i = 0;`
`for (i=0; i < 10; i++).`
 - `for (int i = 0; j = 5; i++).`
 - `for (int i = 0; i < 10; ++i).`
10. If a do...while structure is used:
- An infinite loop will not take place.
 - Counter-controlled repetition is not possible.
 - The body of the loop will execute at least once.
 - An off-by-one error will not occur.
11. What will the following program segment do?
- ```
int counter = 1;
do {
 cout << counter << " ";
 ++counter;
} while (counter <= 10);
```
- Print the numbers 1 through 11.
  - Print the numbers 1 through 10.
  - Print the numbers 1 through 9.
  - Cause a syntax error.

12. A switch statement should be used:
  - a. As a single-selection structure.
  - b. As a double-selection structure.
  - c. As a multiple-selection structure.
  - d. To replace all if and if...else statements.
  
13. In a switch structure:
  - a. a semicolon is required after the end brace.
  - b. Multiple actions for a single case do not need to be enclosed in braces.
  - c. A default case is required.
  - d. A break is required after the default case.
  
14. Which of the following is correct when labeling cases in a switch structure?
  - a. case 1
  - b. Case 1
  - c. case 1
  - d. Case 1
  
15. switch can be used to test:
  - a. int constants.
  - b. float constants.
  - c. string constants.
  - d. all types of constants.
  
16. When using (`infile >> x`) as the condition for a while loop
  - a. You should perform a priming read of x from infile before the loop heading.
  - b. You should include a read of x from infile within the loop body.
  - c. No other read of x from infile should occur.
  - d. Both a and b are correct.
  
17. When using a sentinel-controlled while loop to read a file
  - a. You should perform a priming read of x from the file before the loop heading.
  - b. You should include a read of x from the file within the loop body.
  - c. No read of x from the file should occur.
  - d. Both a and b are correct.
  
18. When using a counter controlled while loop to read a file
  - a. You should perform a priming read from the file before the loop heading.
  - b. You should include a read from the file within the loop body.
  - c. No read from the file should occur.
  - d. Both a and b are correct.
  
19. You should check for file not found when using
  - a. a do-while loop
  - b. a counter-controlled while loop
  - c. a sentinel-controlled while loop
  - d. all of the above

20. In order to find the average of an unknown number of values you must have
- a counter variable
  - an accumulator variable
  - an averaging variable
  - both a and b.
21. What type of loop would you need to read a file of social security numbers until a specific number is found.
- a counter-controlled loop
  - an end of file-controlled loop
  - a sentinel-controlled loop
  - a **for** loop
22. Which of the conditions below would be true when count is greater than 10 or found is true (but false otherwise).
- (count > 10 || found)
  - (count > 10 || found = true)
  - (count > 10 && found)
  - (count > 10 && found = false)
23. *Statement 1*: The body of the while loop always executes at least once.  
*Statement 2*: The body of the do – while loop always executes at least once.
- Both statements are true
  - Both statements are false
  - Statement 1* is true but *Statement 2* is false
  - Statement 1* is false but *Statement 2* is true
24. *Statement 1*: A semicolon should not be placed at the end of the condition in a while loop.  
*Statement 2*: A semicolon is required at the end of the condition in a do – while loop.
- Both statements are true
  - Both statements are false
  - Statement 1* is true but *Statement 2* is false
  - Statement 1* is false but *Statement 2* is true
25. *Statement 1*: The variable used in a switch expression may be of type int or float.  
*Statement 2*: The default case is required in a switch statement.
- Both statements are true
  - Both statements are false
  - Statement 1* is true but *Statement 2* is false
  - Statement 1* is false but *Statement 2* is true
26. *Statement 1*: An expression containing the || operator is true if either or both of its operands are true.  
*Statement 2*: An expression containing the ! operator is true if its operand is true.
- Both statements are true
  - Both statements are false
  - Statement 1* is true but *Statement 2* is false
  - Statement 1* is false but *Statement 2* is true

27. What is printed by the following code?

```
int j = 0;
cout << j;
for (j = 1; j < 10; ++j);
 cout << j << " ";
```

- a. 0
- b. 0 10
- c. 0 1 2 3 4 5 6 7 8 9
- d. 0 1 2 3 4 5 6 7 8 9 10

28. What is printed by the following code?

```
int j = 2;
switch(j)
{
 case 1: cout << "How ";
 case 2: cout << "Are ";
 case 3: cout << "You ";
 default : cout << "nothing?";
}
```

- a. Are You
- b. How Are You
- c. Are
- d. Are You nothing?

29. for (j = 0; j < 10; \_\_\_\_\_)

In order to perform the loop 10 times the for loop above should have what in the blank?

- a. ++j
- b. j++
- c. j+=1
- d. any of the above

30. If an input file does not exist what happens when a program tries to open the file?

- a. nothing
- b. the program terminates with an error message
- c. the identifier associated with the file is set to false or 0.
- d. the file is created and a marker is placed at the beginning of the file

31. If a file exists but is empty what happens when a program tries to read from the file?

- a. nothing
- b. the program terminates with an error message
- c. the identifier associated with the file is set to false or 0.
- d. a marker is placed at the beginning of the file

32. If an output file does not exist what happens when a program tries to open the file?

- a. nothing
- b. the program terminates with an error message
- c. the identifier associated with the file is set to false or 0.
- d. the file is created and a marker is placed at the beginning of the file

33. If an output file exists what happens when a program tries to open the file?
  - a. nothing
  - b. the program terminates with an error message
  - c. the identifier associated with the file is set to false or 0.
  - d. the file is erased and a marker is placed at the beginning of the file
  
34. When simulating a count-controlled loop using a while statement, the counter variable should be
  - a. declared before the loop
  - b. initialized before the loop
  - c. updated inside the loop body
  - d. all of the above
  
35. When using a for loop with the heading **for (count = 0; count < 10; ++count)** to simulate a count-controlled loop in general the counter variable should **not** be
  - a. declared before the loop
  - b. initialized before the loop
  - c. updated inside the loop body
  - d. all of the above
  
36. In a switch statement placing a break statement at the end of the block of statements for each case
  - a. is a syntax error.
  - b. causes a branch to the end of the case statement.
  - c. causes a branch to the default case.
  - d. is not necessary.
  
37. Which of the looping statements in C++ are pretest loops?
  - a. while
  - b. for
  - c. do – while
  - d. both a and b
  
38. Which of the looping statements in C++ are posttest loops?
  - a. while
  - b. for
  - c. do – while
  - d. both a and b
  
39. What five things should you do to use files for input or output in a program
  - a. include <ifstream>, include <ofstream>, open the file, use the file identifier in I/O statements, close the file
  - b. include <fstream>, declare the file identifier, open the file, use the file identifier in I/O statements, close the file
  - c. include <ifstream>, declare the file identifier, open the file, use the file identifier in I/O statements, close the file.
  - d. include <ofstream>, declare the file identifier, open the file, use the file identifier in I/O statements, close the file.



40. Which of the following is false?
- a. break and continue statements alter the flow of control.
  - b. continue statements skip the remaining statements in current iteration of the body of the loop in which they are embedded.
  - c. break statements exit from the loop in which they are embedded.
  - d. continue and break statements may be embedded within all C++ structures.
41. A counter-controlled loop can be simulated using
- a. A while loop
  - b. A for loop
  - c. A do-while loop
  - d. All of the above
42. A while loop is usually used to write a
- a. Sentinel-controlled loop
  - b. End of file controlled loop
  - c. Interactive loop
  - d. All of the above
43. *Statement 1*: In counter controlled loops, a counter is used to repeat the loop body a specified number of times  
*Statement 2*: In an event controlled loop, some condition within the loop changes and this causes the loop to terminate.
- a. Both statements are true
  - b. Both statements are false
  - c. *Statement 1* is true but *Statement 2* is false
  - d. *Statement 1* is false but *Statement 2* is true

## Test 4

Choose the best answer or ending for each question or statement.

1. In C++ new types can be created using
  - a. functions
  - b. classes
  - c. **typedef**
  - d. both b and c
  
2. The function prototype  
**double mySqrt( int x );**
  - a. Declares a function called **mySqrt** which takes an integer as an argument and returns a double.
  - b. Defines a function called **double** which calculates square roots.
  - c. Defines a function called **mySqrt** which takes an argument of type **x** and returns a **double**.
  - d. Declares a function called **mySqrt** which takes a **double** as an argument and returns an integer.
  
3. The argument list of a function call must match the parameter list of the called function in all of the following details, except:
  - a. The number of arguments/parameters in the list.
  - b. The types of arguments/parameters in the list.
  - c. The names of arguments/parameters in the list.
  - d. The argument list and parameter list must match in all of the above details.
  
4. A function prototype does not have to:
  - a. Include parameter names.
  - b. Terminate with a semicolon.
  - c. Agree with the function definition.
  - d. Match with all calls to the function.
  
5. The only identifiers that can be reused everywhere in a program without any ambiguity are:
  - a. Global variables.
  - b. static local variables.
  - c. Those in the parameter list of a function prototype.
  - d. Those in the parameter list of a function definition.
  
6. An activation record will be popped off the function call stack whenever:
  - a. A function returns control to its caller.
  - b. A function calls another function.
  - c. A function calls itself.
  - d. A function declares a local variable.
  
7. When an argument is passed-by-value, changes in the calling function \_\_\_\_\_ affect the original variable's value; when an argument is passed call-by-reference, changes \_\_\_\_\_ affect the original variable's value.
  - a. Do not, do.
  - b. Do not, do not.
  - c. Do, do.
  - d. Do, do not.

8. A reference parameter:
  - a. Is an alias for its corresponding argument.
  - b. Is declared by following the parameter's type in the function prototype by an ampersand (&).
  - c. Cannot be modified.
  - d. Both (a) and (b).
  
9. Call-by-reference can achieve the security of call-by-value when:
  - a. The value being passed is small.
  - b. A large argument is passed in order to improve performance.
  - c. The **const** qualifier is used.
  - d. Both a and b.
  
10. An array is not:
  - a. A consecutive group of memory locations.
  - b. Subscripted by integers.
  - c. Declared using braces, [].
  - d. Made up of different data types.
  
11. Which of the following is not true?
  - a. The first element of an array is the zeroth.
  - b. The last element of an array has position number one less than the array size.
  - c. The position number contained within square brackets is called a subscript.
  - d. A subscript cannot be an expression.
  
12. Which statement would be used to declare a 10-element integer array c?
  - a. `array c = int[ 10 ];`
  - b. `c = int[ 10 ];`
  - c. `int array c[ 10 ];`
  - d. `int c[ 10 ];`
  
13. Which of the following is not a correct way to initialize an array?
  - a. `int n[ 5 ] = { 0, 7, 0, 3, 8, 2 };`
  - b. `int n[] = { 0, 7, 0, 3, 8, 2 };`
  - c. `int n[ 5 ] = { 7 };`
  - d. `int n[ 5 ] = { 9, 1, 9 };`
  
14. Constant variables:
  - a. Can be assigned values in executable statements.
  - b. Do not have to be initialized when they are declared.
  - c. Can be used to specify array sizes.
  - d. b and c
  
15. Referencing elements outside the array bounds:
  - a. Can result in changes to the value of an unrelated variable.
  - b. Is impossible because C++ checks to make sure it does not happen.
  - c. Is a syntax error.
  - d. Enlarges the size of the array.

16. Which of the following is false about a function to which an array is being passed?
  - a. It knows the size of the array that is being passed.
  - b. It is being passed the address of the first element in the array.
  - c. It is able to modify the values stored in the array.
  - d. The array name is used as an argument in the function call.
17. To prevent modification of array values passed to a function:
  - a. The array must be declared static in the function.
  - b. The array parameter can be preceded by the **const** qualifier.
  - c. A copy of the array must be made inside the function.
  - d. The array must be passed by reference.
18. Passing by value is used if a parameter's data flow is
  - a. one-way, into the function.
  - b. one-way, out of the function.
  - c. two-way, into and out of the function
  - d. either b or c
19. Passing by reference is used if a parameter's data flow is
  - a. one-way, into the function.
  - b. one-way, out of the function.
  - c. two-way, into and out of the function
  - d. either b or c
20. In C++, a function prototype is
  - a. a declaration but not a definition.
  - b. a definition but not a declaration.
  - c. both a declaration and a definition.
  - d. neither a declaration nor a definition.
21. Given the function prototype

```
void Fix(int&, float);
```

which of the following is an appropriate function call? (**someInt** is of type **int**, and **someFloat** is of type **float**.)

- a. **Fix(24, 6.85);**
  - b. **someFloat = Fix(someInt, 6.85);**
  - c. **Fix(someInt, someFloat);**
  - d. b and c
22. The \_\_\_\_\_ of an identifier is the portion of a program in which the identifier can be used.
    - a. storage class
    - b. lifetime
    - c. scope
    - d. b and c

23. The \_\_\_\_ of an identifier is the time during the execution of a program when an identifier actually has memory associated with it.
- storage class
  - lifetime
  - scope
  - b and c
24. Which of the following statements can be used to declare a constant **size** for an array of strings **List**?
- `string List[size];`
  - `string List[size = 10];`
  - `const size = 10;`
  - `const int size = 10;`
25. What is the name of the second element in the array **List**?
- `List[0];`
  - `List[1];`
  - `List[2];`
  - `List[3];`
26. Which statement below assigns the value 0 to **List** array element 4?
- `List[3] = 0;`
  - `List[4] = 0;`
  - `List[5] = 0;`
  - none of the above
27. Which statements below are needed to print the float variable **SomeFloat** rounded to 2 decimal places.
- `cout << setprecision(2);`
  - `cout << fixed;`
  - `cout << showpoint;`
  - a and b
28. Which file must you include in order to use `setprecision`?
- `cmath`
  - `string`
  - `iomanip`
  - `cstdlib`
29. Which of the following is a valid function prototype for a function **FindArea** which has two **float** value parameters **side1** and **side2** and returns a **float**?
- `float FindArea(float, float);`
  - `float FindArea(float side1, float side2);`
  - `void FindArea(float side1, float side2);`
  - a and b

30. Which of the following is a valid function call for the function **FindArea** (assume all variables are **float**)?
- Findarea(x, y);**
  - Area = Findarea(float x, float y);**
  - Area = Findarea(5, 7);**
  - b and c
31. A function **SomeFunc** has two parameters, **alpha** and **beta**, of type **int**. The data flow for **alpha** is one-way, into the function. The data flow for **beta** is two-way, into and out of the function. What is the most appropriate function heading for **SomeFunc**?
- void SomeFunc( int& alpha, int& beta )**
  - void SomeFunc( int alpha, int& beta )**
  - void SomeFunc( int& alpha, int beta )**
  - void SomeFunc( int alpha, int beta )**
32. Which of the following statements about value parameters is true?
- The caller's argument is never modified by execution of the called function.
  - The parameter is never modified by execution of the called function.
  - The caller's argument must be a variable.
  - b and c above

33. Given the function definition

```
void Twist(int a, int& b)
{
 int c;

 c = a + 2;
 a = a * 3;
 b = c + a;
}
```

what is the output of the following code fragment that invokes **Twist**? (All variables are of type **int**.)

```
r = 1;
s = 2;
t = 3;
Twist(t, s);
cout << r << ' ' << s << ' ' << t << endl;
```

- 1 14 3
  - 1 10 3
  - 5 14 3
  - 1 14 9
34. If a variable **alpha** is accessible only within function **F**, then **alpha** is either
- a global, a local variable within **F**, or a parameter of **F**.
  - a local variable to the main or a parameter of **F**.
  - a global variable or an argument to **F**.
  - a local variable within **F** or an argument to **F**.

35. A programmer wishes to write a function to completely fill a one-dimensional array of float **List** with length **size**. Which of the following is the correct function header:
- `void (float List[], int size);`
  - `void(float List);`
  - `void(float List[size]);`
  - `void(float List[]);`
36. Which of the following is a valid declaration for a 2-dimensional array **Table**.
- `float Table [10];`
  - `float Table[2] [10];`
  - `typedef float TableType[2] [10];  
TableType Table;`
  - both b and c.
37. \_\_\_\_\_ in a function passes the value of an expression back to the main.
- A value parameter
  - A reference parameter
  - The return statement
  - b and c above
38. Which of the following is a valid prototype for a void function **GetData** which has 2 **float** reference parameters **side1** and **side2**.
- `void GetData(float& side1, float& side2);`
  - `void GetData(&float, &float);`
  - `void GetData(&float side1, &float side2);`
  - b and c
39. *Statement 1*: If there are fewer initializers in an initializer list than the number of elements in the array, the remaining elements are initialized to the last number.  
*Statement 2*: Arguments to a function call are always passed by value;
- Both statements are true.
  - Both statements are false.
  - Statement 1* is true but *Statement 2* is false.
  - Statement 1* is false but *Statement 2* is true.
40. The number used to refer to a particular element of an array is called its
- subscript
  - index
  - type
  - both a and b

41. Suppose the first few lines of a function are as follows:

```
void Calc(float beta)
{
 alpha = 3.8 * beta;
```

Then the variable alpha must be

- a. a local variable.
- b. a global variable.
- c. a parameter.
- d. an argument.



## **APPENDIX B**

### **LAB AND PROGRAMMING ASSIGNMENTS**

## Lab Assignments

### Laboratory Assignment 1

#### Purpose:

- To learn to use C++.net to edit, compile, and run C++ programs.
- To see the advantages of white space in C++ programs.

**Reading Assignment:** Sections 1.8, 1.9, 1.12, 1.16 and handout "Using C++.net"

#### Lab Programming Assignment:

1. Type in program 1 provided with this assignment.
2. Build your project and run the program.
3. Open the program 2 file located on Blackboard and copy it over your program 1.
4. Clean the solution and build your project again
5. Run program 2.

```
// Program 1 for Laboratory 1
// Printing a message to the screen
#include <iostream>
using namespace std;

int main()
{
 cout << "Welcome to the world of C++!\n";
 return 0;
}
```

### Post Lab Assignment 1

1. What was the output of program1?
2. What was the output of program 2?
3. What was the difference in the code in program 2 and the code you typed in?
4. Observations and Discoveries
5. Problems with the Lab

## Laboratory Assignment 2

### Purpose:

- To understand the parts of a basic C++ program.
- To use the cout statement.

**Reading Assignment:** Sections 2.1, 2.2, 2.3

### Lab Programming Assignment:

1. Type in the provided template of a basic C++ program.
2. Modify the template to print out on separate lines a message that includes the name of this assignment, your name, the name of this course, your instructor's name, and the date your post lab is due.
3. If you have time, surround your message with a box of stars.

```
//Laboratory 2
// Outputs a message to the screen

//insert include statement here
int main()
{
 //Insert output lines here
 return 0;
}
```

## Post Lab Assignment 2

1. What was the output of the program?
2. Observations and Discoveries
3. Problems with the Lab

### Laboratory Assignment 3

**Purpose:**

- To practice the use of integer variables in programs.
- To practice using the assignment operator.
- To understand integer operations of divide and modulus
- To practice printing out the contents of integer variables.

**Reading Assignment:** Sections 2.4 and 2.5

**Lab Programming Assignment:**

- Type in the program on the next page or copy the lab 3 program from blackboard
- Modify the program
  1. In one statement declare an integer variable **num2** and assign it an initial value of 10
  2. Run your program. Your compiler may complain about printing a number that has not been assigned a value but make it do it.
  3. In an assignment statement assign the value 3 to num1.
  4. In an assignment statement have the program calculate num1 times num2 and store the result in product. Remember to use \* for multiply.
  5. In another assignment statement have the program calculate num2 divided by num1 and store the result in quotient.
  6. In a fourth assignment statement have the program calculate num2 modulus num1 (hint use the % operator). Store the result in remainder.
  7. Clean your solution and then build your project and run the program again.

```
//Laboratory 3
/*This program practices the use of the assignment operation, declaring variables, and integer
operations.*/
#include <iostream>

int main()
{
 int num1;

 //1. declare and initialize num2 here

 int product = 0, quotient = 0, remainder = 0;

 //3-6 Insert assignment statements here

 std :: cout << "The first number is " << num1 << std :: endl;
 std :: cout << "The second number is " << num2 << std :: endl;
 std :: cout << "The second number times the first is " << product << std :: endl;
 std :: cout << "The second divided by the first is " << quotient << std :: endl;
 std :: cout << "The second modulus the first is " << remainder << std :: endl;
 return 0;
}
```

### Post Lab Assignment 3

1. What was the output of this lab's program on the first execution?
2. What was the output of this lab's program on the second execution?
3. Make a list of the variables used in this program.
4. Make a list of the literals used in this program.
5. Observations and Discoveries
6. Problems with the Lab



## Laboratory Assignment 4

### Purpose:

- To understand and practice the use of the insertion operator (>>) and cin
- To use a constant in a program
- To understand the difference between constants and variables.
- To understand the precedence of operators and to change the order of evaluation using parenthesis.
- To understand the importance of verifying correctness of output.

**Reading Assignment:** Section 2.6

### Lab Programming Assignment:

1. Type in the program on the next page.
2. Declare a constant whose identifier is **three** and contains the value 3.
3. Modify the program to prompt the user for three numbers and store the user input into the variables already declared.
4. Run the program. Did the program correctly find the average of the three numbers?
5. Insert parenthesis so that the program finds the average of the three numbers correctly.

```
//Laboratory 4
// Your identifying comments
/*This program prompts the user for 3 numbers and finds the average of the three numbers */
#include <iostream>
using namespace std;

//2. Declare the constant three here

int main()
{
 int num1, num2, num3, average, result;

 /*3. Prompt the user for 3 numbers here and store them in num1, num2,
 and num3.*/

 average = num1 + num2 + num3/three; /*5. Change this statement to work
 correctly */
 cout << "The average of " << num1 << ", " << num2 << ", and " << num3 << " is " <<
 average << endl;
 return 0;
}
```

#### Post Lab Assignment 4

1. What was the output of this lab's program without parenthesis?
2. What was the output of the program after you inserted the parenthesis?
3. Make a list of all the variables used in this program.
4. Where are variables usually declared?
5. Make a list of the constants used in this program.
6. Where are constants usually declared?
7. Observations and Discoveries
8. Problems with the Lab

## Laboratory Assignment 5

### Purpose:

- To practice using relational operators.
- To practice the use of the if statement
- To practice the algorithm for finding the minimum of a list of numbers.
- To recognize the need for testing of all possible paths of execution.

### Reading Assignment: 2.7, 4.5

### Lab Programming Assignment:

1. Modify the program given to find the minimum of two numbers entered by the user.
  - a. Write a C++ statement that stores the first number in minimum
  - b. Write a C++ statement that stores the second number in minimum when the second number is smaller than the first
2. Run the program at least three times: Once where the first number is the smallest, once where the first number is largest, once where the numbers are equal.

```
//Laboratory 5
// Identifying comments
/*This program gets two numbers from the user, finds the smaller number then prints out the
original numbers and the smaller appropriately labeled.*/
#include <iostream>
using namespace std;

int main()
{
 int num2, minimum;
 cout << "Please enter a number" << endl;
 /*Replace this comment with an input statement that stores the user entry
into minimum*/
 cout << "Please enter a second number" << endl;
 cin >> num2;

 /* Replace this comment with an if statement that stores the second
number in minimum when it is smaller than the first number*/

 cout << "The smallest number is " << minimum << endl;
 return 0;
}
```

### **Post Lab Assignment 5**

1. What was the output of this lab's program on the first execution?
2. What was the output of this lab's program on the second execution?
3. What was the output of this lab's program on the third execution?
4. Observations and Discoveries
5. Problems with the Lab

## Laboratory Assignment 6

### Purpose:

- To declare a variable of type double
- To be able to use the C++ standard library functions from `cmath`.
- To be able to write and call a value returning function with one parameter.
- To understand the difference between arguments and parameters.

**Reading Assignment:** Section 6.1, 6.2, 6.3

### Lab Programming Assignment:

Modify the program provided with this laboratory.

1. Include the C++ standard library `cmath`
2. Declare `x` to be of type double and initialize `x` to 10.0
3. Call the `sqrt` function to find the square root of 2.0
4. Define a function `findCube` which returns a double and has one double parameter `c`.
5. Call your function `findCube` with the argument `x`.
6. Build the project and run the program.

```
//Laboratory 6
//This program using the sqrt function and a user-defined function cube.

#include <iostream>
//1.insert include statement here
using namespace std;

//4. insert function definition for findCube here

int main()
{
 //2.Declare and initialize x here
 cout << "The square root of "<< x << " is " << sqrt(x) << endl;
 cout << "The square root of "<<2.0<<" is "<< /* 3. insert call here */
 << endl;
 cout << "The cube of " << x << " is " << /*5. insert call here*/ << endl;
 cout << "The cube of " << 2.0 << " is " << findCube(2.0) << endl;
 return 0;
}
```

### Post Lab 6

1. What is the output of this program?
2. What are the parameters of findCube?
3. What are the parameters of sqrt?
4. What literals and or variables did you use as arguments in this program?
5. Observations and Discoveries.
6. Problems with the lab.

## Laboratory Assignment 7

### Purpose:

- To be able to define and call a void function with a parameter.

**Reading Assignment:** Section 6.4

### Lab Programming Assignment:

Modify the program from Laboratory Assignment 6 to use a function to print the results.

1. Insert the function heading for `printResults` above the main. `printResults` is a void function with 1 float parameter (`x`).
2. Move the output (`cout`) statements from the main to the body of `printResults`
3. Call your `printResults` function in the main.
4. Run your program. What is the output?

```
//Laboratory 7
//This program uses a user-defined void function with several parameters

#include <iostream>
#include <cmath>
using namespace std;

float findCube(float c)
{
 return pow(c,3);
}
//1. insert the heading for printResults here
{
 //2. insert the body for printResults here
}
int main()
{
 float x = 10.0;

 // 2. move all the cout statements to the body of printResults
 cout << "The square root of "<< x << " is " << sqrt(x) << endl;
 cout << "The square root of " << 2.0 << " is " << sqrt(2.0) << endl;
 cout << "The cube of " << x << " is " << findCube(x) << endl;
 cout << "The cube of " << 2.0 << " is " << findCube(2.0) << endl;

 //3. call the printResults function here.
 return 0;
}
```

### Post Lab Assignment 7

1. What was the output of this lab's program on the first execution
2. What are the benefits of using functions?
3. What are the disadvantages of using functions.
4. Observations and Discoveries
5. Problems with the Lab



## Laboratory Assignment 8

### Purpose:

- To understand the syntax and purpose of a class.
- To practice writing and calling a member function without parameters.

**Reading Assignment:** Sections 3.1, 3.2, 3.3, 3.4 to end of page 70.

**Lab Programming Assignment:** Copy or type the program for this lab into the C++ editor. The purpose of this program is to test a class you are to write.

1. Type a class definition for a class called `WebAccount` just above your main function. Include only a public area at this time.
2. In the public area of your `WebAccount` class, write a member function called `displayWelcomeMessage`. The function should print out a message welcoming the new client to Accounts on the Web.
3. Place a call to the member function in the main using the object `VacationAccount`. Run the program.

```
// Laboratory 8
//Your identifying comments
// This program creates a class and uses it's function member.
```

```
#include <iostream>
using namespace std;
```

```
//1. Write your class definition here
```

```
int main()
{
 WebAccount VacationAccount ;

 //2. Call your class member function here
 return 0;
}
```

### **Post Lab Assignment 8**

1. What was the output of this lab's program?
2. Did you use any objects in this program? If so what were they?
3. Observations and Discoveries
4. Problems with the Lab

## Laboratory Assignment 9

### Purpose:

- To understand the difference between int and string variables.
- To practice the use of string variables
- To understand the difference between a string variable and a literal.
- To understand when to use the cin and the getline command.

Reading Assignment: page 43, 44, 72-74

**Lab Programming Assignment:** Using the basic C++ program template, write a program which

1. prompts the user for their first name, middle initial, last name, (on one line) then their street address on a second line, their city on a third line, their state on a fourth line and their zip on a fifth line. Store the full name, street address, city, state and zip in five separate string variables. Use getline for the full name, street address, city and state. Use the standard input stream operator (cin >>) for the zip. Be careful. Remember that while the standard stream operator skips over whitespace, getline does not and while getline consumes the end of line character, the standard stream operator does not.
2. You will need to include the string file.
3. Echo print the input with appropriate labels as shown below.

Name: Bob A. Smith

Address: 115 Pine St.

City: New York

State : New York

Zip: 35630

```

//Laboratory 9
//This program prompts the user for name and address and echo prints the output.
#include <iostream>
//2. Replace this comment with the include statement
using namespace std;
int main()
{
 string name, street, city, state, zip;
 cout << "Enter your full name and press the enter key.";

 //1.replace this comment with your getline statement

 cout << "Enter your street address and press the enter key.";

 //1.replace this comment with your getline statement

 cout << "Enter your city and press the enter key.";

 //1.replace this comment with your getline statement

 cout << "Enter your zip and press the enter key.";

 //1.replace this comment with your cin statement

 //3.replace this comment with your output statements
 return 0;
}

```

### Post Lab Assignment 9

1. What was the output of this lab's program?
2. Make a list of the string variables used in this program.
3. Make a list of the literals used in this program
4. Observations and Discoveries
5. Problems with the Lab

## Laboratory Assignment 10

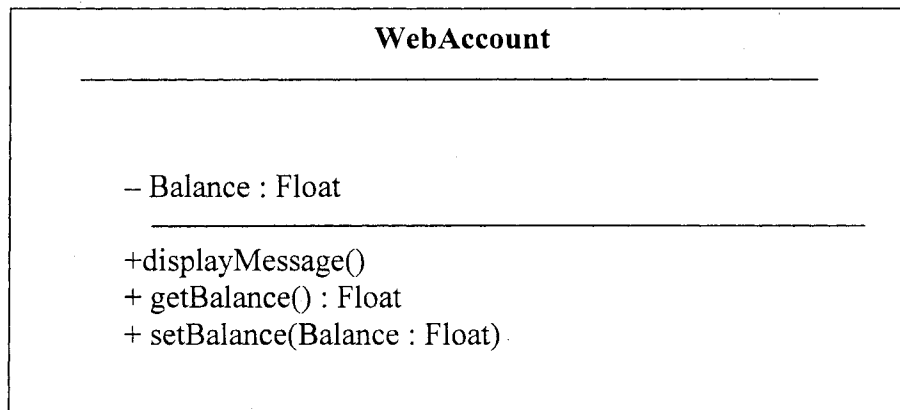
### Purpose:

- To understand Class data members
- To understand how to access a class's data member.
- To understand why we write set and get class member functions

### Reading Assignment: Section 3.6

**Lab Programming Assignment:** Modify your program from lab assignment 8.

1. Add a class data member called Balance (of type Float) that is private to the class.
2. Add a Get class member function getBalance to your class. getBalance returns a float value which is the current balance
3. As a part of an output statement write a call to getBalance for your class object VacationAccount.
4. Compile and Run your program.
5. Add a set class member function setBalance to your class. setBalance has one float parameter (amount). setBalance stores amount into the class data member, Balance.
6. Add a call to your setBalance member function using 0.00 as the argument.
7. Compile and run your program again.



```

// Laboratory 10
//Your identifying comments
/* This program adds a data member and a set and a get function to the WebAccount class*/
#include <iostream>
#include <string>
using namespace std;
class WebAccount
{
public:

 void displayWelcomeMessage()
 {
 cout<<"Welcome to Accounts on the Web" <<endl;
 }

 //2.. Replace this comment with the class member function getBalance

 // 5. Replace this comment with the class member function setBalance

private:

 //1. Replace this comment with the new data member

};
int main()
{
 WebAccount VacationAccount;
 float Balance = 60.50;
 VacationAccount.displayWelcomeMessage();

 //6. Replace this comment with a call to setBalance.

 /*3. replace this comment with a cout statement that includes a call to
 getBalance.*

 return 0;
}

```

### Post Lab Assignment 10

1. What was the output of this lab's program on the first run?
2. Why didn't the class data member function *Balance* have 60.50 in it?
3. What was the output of this lab's program on the 2<sup>nd</sup> run?
4. Observations and Discoveries
5. Problems with the Lab



## Laboratory Assignment 11

### Purpose:

- To practice declaring and initializing a class object.
- To practice the syntax of constructors.
- To understand the need for constructors

### Reading Assignment: 6.1 – 6.3

### Lab Programming Assignment: Modify your program from laboratory assignment 10.

1. Modify your program to include a default constructor which initializes the Balance to 0
2. Run the program
3. Write an initializing constructor that initializes the amount in Balance to an amount provided in the declaration.
4. Change the declaration of VacationAccount (in the main) to include an argument of 100.50
5. Run the program again.

```

//Laboratory 11
//Your identifying comments
// This program adds data members and a constructor to a class.

#include <iostream>
using namespace std;

class WebAccount
{
 public:
 //1. replace this comment with your default constructor
 //3. Replace this comment with an initializing constructor
 void displayWelcomeMessage()
 {
 cout << "Welcome to Accounts on the Web!" << endl;
 }

 float getBalance()
 {
 return Balance;
 }
 private:
 float Balance;
};

int main()
{
 WebAccount VacationAccount ; //4.change this declaration.

 VacationAccount.displayWelcomeMessage();
 cout << "Your Balance is " << VacationAccount.getBalance() << endl;
 return 0;
}

```

### **Post Lab Assignment 11**

1. What was the output of this lab's program on the first run?
2. What was the output on this lab's program on the second run?
3. Did which constructor you used make a difference?
4. Observations and Discoveries
5. Problems with the Lab

## Laboratory Assignment 12

### Purpose:

- To practice writing pseudocode.

### Reading Assignment: 4.1- 4.3

### Lab Programming Assignment:

1. Write the pseudocode for the algorithm to get two numbers from a user, find the smaller of two numbers and print the smaller number.
2. Exchange your pseudocode with someone else.
3. If you have time write the program from the other person's pseudocode.

### Post Lab Assignment 12

1. Show your pseudocode below.
2. Was the other student's pseudocode easy to understand?
3. Was it exactly like yours?
4. Observations and Discoveries
5. Problems with the Lab

## Laboratory Assignment 13

### Purpose:

- To practice using the if-then-else statement.
- To practice writing class member functions from UML specification diagrams.
- To practice calling class functions

### Reading Assignment: 4.5

Lab Programming Assignment: Modify the class WebAccount program from Laboratory Assignment 10.

1. Write the two class functions as specified in the UML diagram below.

| WebAccount                  |
|-----------------------------|
| +Deposit(Amount:float)      |
| + Withdraw( Amount : float) |

Deposit should add Amount to the value in Balance.

Withdraw should subtract Amount from balance if Balance is big enough. Otherwise it should print an appropriate message.

2. Call Deposit on the class object with the literal 100 as its argument.
3. Using the getBalance class function, print out the Balance of the class object appropriately labeled.
4. In the main call your Withdraw class function on the class object that has already been declared as members of the WebAccount class using 10 as the withdrawal amount.
5. Using the get Balance class function, print out the Balance of the class object appropriately labeled.
6. In the main call your Withdraw class function on the class object with 200 as the argument.
7. Using the get Balance class function, print out the Balance of the class object appropriately labeled.
8. Run your program

```

// Lab programming assignment 13

//Your identifying comments
// This program adds datamembers to a class
#include <iostream>
using namespace std;
class WebAccount
{
 public:
 WebAccount()
 {
 Balance = 0;
 }
 void displayWelcomeMessage()
 {
 cout<<"Welcome to Accounts on the Web " << endl;
 }

 float getBalance()
 {
 return Balance;
 }
//insert Deposit function member here
// insert Withdraw function member here
 private:
 float Balance;
};
int main()
{
 WebAccount VacationAccount;
 VacationAccount.displayWelcomeMessage();
//2. Replace this comment with deposit class function call
//3. Replace this comment with an output statement that prints the Balance.
//4. Replace this comment with the withdraw call for 10 dollars.
//5. Replace this comment with an output statement that prints the Balance.
//6. Replace this comment with the withdraw call for 200 dollars
//7. Replace this comment with an output statement that prints the Balance.

 return 0;
}

```

### **Post Lab Assignment 13**

1. What was the output of this lab's program?
2. Observations and Discoveries
3. Problems with the Lab



## Laboratory Assignment 14

### Purpose:

- To practice using the if-else if-else if- else statements.
- To practice using a menu
- To practice calling class functions

### Reading Assignment: Section 4.6, 4.7

Lab Programming Assignment: Modify the class WebAccount program from Laboratory Assignment 13.

1. Change your main to print a menu which allows the user to select which of 3 class member functions they would like to perform.
  - 1 Deposit
  - 2 Withdraw
  - 3 Print the Balance
2. Get the choice from the user
3. use an if ... else if...else if...else statement to perform the users choice.
  - Deposit
    - a. Inside this if you must prompt the user for an amount and call Deposit
  - Withdraw
    - a. Inside this if you must prompt the user for an amount and call Withdraw
  - Print the Balance
    - a. Inside this if you must call printBalance
  - The final else
    - a. Inside this if you should print a message that the selection was not valid
4. Run your program at least 4 times
  - Once where the user selects Deposit
  - Once where the user selects Withdraw
  - Once where the user selects Print the Balance.
  - Once where the user inputs a number not in the menu

```

// Lab programming assignment 14
//Your identifying comments
// This program adds a menu to the main
#include <iostream>
using namespace std;
class WebAccount
{
 public:
 WebAccount()
 {
 Balance = 0;
 }
 void displayWelcomeMessage()
 {
 cout<<"Welcome to Accounts on the Web " << endl;
 }

 float getBalance()
 {
 return Balance;
 }
 void Deposit(float amount)
 {
 Balance = Balance + amount;
 }
 void Withdraw(float amount)
 {
 if (amount <=Balance)
 Balance = Balance - amount;
 else
 cout << "Insufficient Funds." << endl;
 }
 private:
 float Balance;
};
int main()
{
 WebAccount VacationAccount;
 VacationAccount.displayWelcomeMessage();
 //1. Replace this comment with the menu
 //2. Replace this comment with a cin statement to get the user's choice
 //3. Replace this comment with if..else if..else if ...else statements

 return 0;
}

```

### **Post Lab Assignment 14**

1. What was the output of this lab's program on the first execution?
2. What was the output of this lab's program on the second execution?
3. What was the output of this lab's program on the third execution?
4. Observations and Discoveries
5. Problems with the Lab

## Laboratory Assignment 15

### Purpose:

- To practice creating a counted while loop
- To use a counter variable

### Reading Assignment: Section 4.8

### Lab Programming Assignment:

1. Modify your main from Laboratory Assignment 14 to allow the user to Deposit several checks at one time.
  - a. Declare a variable to use as the loop counter and one to use as the number of checks.
  - b. Inside the deposit option
    - i. Prompt the user for the number of checks they wish to deposit
    - ii. get the number
    - iii. Construct the heading for the while loop
    - iv. Construct the body of the while loop which will consist of
      1. an opening brace
      2. The prompt for the amount of the check and the statement to get the amount
      3. an update of the counter
      4. a closing brace
2. Run the program with the selection 1 for deposit.
3. Input 10 checks of varying amounts

```

// Lab programming assignment 15

//Your identifying comments
// This program adds a while loop to the deposit option in the main
#include <iostream>
using namespace std;
class WebAccount
{
 public:
 WebAccount()
 {
 Balance = 0;}
 void displayWelcomeMessage()
 {
 cout<<"Welcome to Accounts on the Web " << endl;}
 float getBalance()
 {
 return Balance;}
 void Deposit(float amount)
 {
 Balance = Balance + amount;}
 void Withdraw(float amount)
 {
 if (amount <=Balance)
 Balance = Balance - amount;
 else
 cout << "Insufficient Funds" << endl;
 }
 private:
 float Balance;
};
int main()
{
 WebAccount VacationAccount;
 int choice;
 //Declare loop counter and number of checks variable
 float amount;
 cout << " 1. Deposit \n 2. Withdraw\n 3. Print the Balance\n " << endl;
 cout << endl << "Enter your choice: 1, 2, or 3. " <<endl;
 cin >> choice;
 if (choice == 1)
 {
 //insert prompt for user here
 //get number of checks to deposit here
 // insert while statement here
 // begin while loop body here
 cout << "How much is the check you wish to deposit? ";
 cin >> amount;
 VacationAccount.Deposit(amount);
 //update counter here
 //end while loop body here
 }
 else if (choice ==2)
 {
 cout << "How much is the withdrawal? ";
 cin >> amount;
 VacationAccount.Withdraw(amount);
 }
}

```

```
}
else if (choice == 3)
 cout << "Your Balance is " << VacationAccount.getBalance() << endl;
else
 cout << "Your made an invalid choice. " << endl;
return 0;
}
```

### Post Lab Assignment 15

1. What were the amounts of the 10 checks you entered?
2. What was the output of this lab's program ?
3. Observations and Discoveries
4. Problems with the Lab

## Laboratory Assignment 16

### Purpose:

- To practice creating a sentinel controlled while loop
- To practice formatting real number output

### Reading Assignment: Section 4.9

### Lab Programming Assignment:

4. Modify your main from Laboratory Assignment 14 to allow the user to perform several operations
  - a. Inside the main option
    - i. Before the while loop, modify the menu prompt to allow the user to select 4: Quit
    - ii. Construct the heading for the while loop with the condition that the loop continue so long as the user entry does not equal 4.
    - iii. The body of the while loop which will consist of
      1. an opening brace
      2. The if – else if – else statement as previously programmed
      3. Another prompt menu and a cin statement to get the choice from the user.
      4. a closing brace
5. Modify your output statement that prints the Balance to include formatting that forces the Balance to print with 2 decimal places.
6. Run your program.
7. As the user:
  - a. Select deposit
  - b. Select print the balance
  - c. Select withdraw
  - d. Select print the balance



```

// Lab programming assignment 16
//Your identifying comments
// This program adds a sentinel controlled while loop to the main
#include <iostream>
// include iomanip here
using namespace std;
class WebAccount
{
 public:
 WebAccount()
 { Balance = 0;}
 void displayWelcomeMessage()
 { cout<<"Welcome to Accounts on the Web " << endl;}
 float getBalance()
 { return Balance;}
 void Deposit(float amount)
 { Balance = Balance + amount;}
 void Withdraw(float amount)
 {
 if (amount <=Balance)
 Balance = Balance - amount;
 else
 cout << "Insufficient Funds" << endl;
 }
 private:
 float Balance;
};
int main()
{
 WebAccount VacationAccount;
 int choice;
 float amount;
 cout<<" 1. Deposit \n";
 cout <<" 2. Withdraw\n";
 cout <<" 3. Print the Balance\n"<<endl;
 cout << endl << "Enter your choice: 1, 2, or 3. " <<endl;
 //Modify the above menu prompt to include the option 4: Quit
 cin >> choice;
 //insert while loop header here
 //begin body of while loop here
 if (choice == 1)
 {
 cout <<"How much is the check you wish to deposit? ";
 cin >> amount;
 VacationAccount.Deposit(amount);
 }
 else if (choice ==2)
 {
 cout << "How much is the withdrawal? ";
 cin >> amount;
 VacationAccount.Withdraw(amount);
 }
}

```

```
 }
 else if (choice == 3)
 cout << "Your Balance is " << VacationAccount.getBalance() << endl;
 /*change this statement to include setprecision, showpoint, and fixed
 stream manipulators */
 else
 cout << "You made an invalid choice. " << endl;
 //Get another choice from the user here
 //end while loop here
 return 0;
```

### **Post Lab Assignment 16**

1. What was the output of this lab's program?
2. Observations and Discoveries
3. Problems with the Lab

## Laboratory Assignment 17

### Purpose:

- To practice reading and writing to a file
- To practice creating an end of file controlled while loop

Reading Assignment: none

### Lab Programming Assignment:

1. Modify your WebAccount class to include a class function **PrintStatement()** that prints a monthly statement. The function should
  - i. Get the beginning balance from the file **datain.txt**
  - ii. in a **while** loop get the list of deposits/withdrawals, print( to an ouput file named **statement.txt**) the amount then add the amounts to the balance and print (to an output file) the new balance. (Be careful! This running balance for the **PrintStatement** member function is not the class data member **Balance**)
  - iii. print to an output file the ending balance
  - iv. output to a file as specified below

|                   |         |
|-------------------|---------|
| Beginning Balance | 250.25  |
| Amount            | Balance |
| -2.25             | 248.00  |
| 10.00             | 255.00  |
| Ending Balance    | 255.00  |

2. Create a text file **datain.txt** with at least 10 entries. The first should be the beginning balance. The rest of the entries should be a mixture of positive (deposits) and negative (withdrawls) amounts.
3. Construct a main that calls **PrintStatement** for the object **VacationAccount**
4. Run your program.

| <b>WebAccount</b>         |
|---------------------------|
| + <b>PrintStatement()</b> |

```

// Lab programming assignment 17
//Your identifying comments
/*This program adds a end of file controlled while loop to the class*/
#include <iostream>
// include fstream here
using namespace std;
class WebAccount
{
 public:
 WebAccount()
 {
 Balance = 0;}
 void displayWelcomeMessage()
 {
 cout<<"Welcome to Accounts on the Web " << endl;}
 float getBalance()
 {
 return Balance;}
 void Deposit(float amount)
 {
 Balance = Balance + amount;}
 void Withdraw(float amount)
 {
 if (amount <=Balance)
 Balance = Balance - amount;
 else
 cout << "Insufficient Funds" << endl;
 }
 void PrintStatement()
 {
 //declare and open files
 //declare local variables as needed
 /*get beginning balance from infile and print to
 outfile*/
 while (/* put read condition here to get amount*/)
 {

 //add to balance
 //print amount and balance to outfile
 }
 //Print ending balance to outfile
 }
 private:
 float Balance;
};
int main()
{
 WebAccount VacationAccount;
 //Call PrintStatement here
 return 0;
}

```

### Post Lab Assignment 17

1. What was the output to the file when you ran this lab's program ?
2. What are the advantages of reading from files over getting your data from the user?
3. Observations and Discoveries
4. Problems with the Lab

## Laboratory Assignment 18

### Purpose:

- To practice creating a For Loop

### Reading Assignment: 5.1 – 5.4

### Lab Programming Assignment:

8. Change your while loop from Laboratory Assignment 15 to a For Loop. Remember a while loop counter is updated inside the loop but a For loop's counter is updated in the For loop heading.
  - a. Have the loop counter start at 1 and continue so long as the loop counter is less than or equal to the user entered number. Run your program.
  - b. Have the loop counter start at 0 and continue so long as the loop counter is less than the number entered by the user. Run your program
  - c. Have the loop counter start at the number entered by the user and continue so long as the counter is greater than 0 (the counter should decrement by one each time through the loop). Run your program

```

// Lab programming assignment 18
//Your identifying comments
// This program adds a end of file controlled while loop to the class
#include <iostream>
using namespace std;
class WebAccount
{
 public:
 WebAccount()
 {
 Balance = 0;}
 void displayWelcomeMessage()
 {
 cout<<"Welcome to Accounts on the Web " << endl;}
 float getBalance()
 {
 return Balance;}
 void Deposit(float amount)
 {
 Balance = Balance + amount;}
 void Withdraw(float amount)
 {
 if (amount <=Balance)
 Balance = Balance - amount;
 else
 cout << "Insufficient Funds" << endl;
 }
 private:
 float Balance;
};
int main()
{
 WebAccount VacationAccount;
 int choice;
 int counter = 1;//remove this statement
 int numChecks;
 float amount;
 cout << " 1. Deposit \n 2. Withdraw\n 3. Print the Balance\n " << endl;
 cout << endl << "Enter your choice: 1, 2, or 3. " <<endl;
 cin >> choice;
 if (choice == 1)
 {
 cout << " How many checks do you wish to deposit\n";
 cin >> numChecks;
 while (counter <= numChecks) //change to a for statement
 {
 cout << "How much is the check you wish to deposit? ";
 cin >> amount;
 VacationAccount.Deposit(amount);
 ++counter; //remove this statement
 }
 }
 else if (choice ==2)
 {
 cout << "How much is the withdrawal? ";
 cin >> amount;
 VacationAccount.Withdraw(amount);
 }
 else if (choice ==3)
 cout << "Your Balance is " << VacationAccount.getBalance() << endl;
 else

```



```
 cout << "Your made an invalid choice. " << endl;
 return 0;
}
```

### Post Lab Assignment 18

1. How many times did the loop execute when you ran your program the first time?
2. The second time?
3. The third time?
4. Did it execute the same number of times for all three loops?
5. Do you prefer the For loop or the while loop?
6. Observations and Discoveries
7. Problems with the Lab

## Laboratory Assignment 19

### Purpose:

- To practice changing a while loop to a do while loop

### Reading Assignment: Section 5.5

### Lab Programming Assignment:

1. Modify your main from Laboratory Assignment 16 to include a do while loop in place of the while loop.
  - a. Inside the main option
    - i. remove the while statement and its accompanying open brace.
    - ii. Since a do while does not need a priming read, place the do reserved word above the first printout of the menu. Include on the next line the accompanying open brace.
    - iii. remove the second printout of the menu and the 2<sup>nd</sup> cin statement.
    - iv. end the do --- while loop after the last else.
    - v. It will be necessary to include an else if for condition 4. If the user chooses 4 the program should print Goodbye to the screen
2. Run your program. As the user: select quit
3. Run your program again. As the user:
  - a. Select deposit
  - b. Select print the balance
  - c. Select withdraw
  - d. Select print the balance
  - e. Select Quit

```

// Lab programming assignment 19
// This program adds a do while loop to the main
#include <iostream>
#include <iomanip>
using namespace std;
class WebAccount
{
 public:
 WebAccount()
 {
 Balance = 0;}
 void displayWelcomeMessage()
 {
 cout<<"Welcome to Accounts on the Web " << endl;}
 float getBalance()
 {
 return Balance;}
 void Deposit(float amount)
 {
 Balance = Balance + amount;}
 void Withdraw(float amount)
 {
 if (amount <=Balance)
 Balance = Balance - amount;
 else
 cout << "Insufficient Funds" << endl;
 }
 private:
 float Balance;
};
int main()
{
 WebAccount VacationAccount;
 int choice;
 float amount;
 //begin do loop here
 cout<< " 1. Deposit \n";
 cout <<" 2. Withdraw\n";
 cout <<" 3. Print the Balance\n";
 cout <<" 4. Quit "<< endl;
 cout << endl << "Enter your choice: 1, 2, 3, or 4. "<<endl;
 cin >> choice;
 while(choice != 4)//remove while loop
 {/remove
 if (choice == 1)
 {
 cout <<"How much is the check you wish to deposit? ";
 cin >> amount;
 VacationAccount.Deposit(amount);
 }
 else if (choice ==2)
 {
 cout << "How much is the withdrawal? ";
 cin >> amount;
 VacationAccount.Withdraw(amount);
 }
 else if (choice ==3)

```

```

 cout << "Your Balance is " << fixed<< setprecision(2) <<
 VacationAccount.getBalance() << endl;
//Add an else if for when choice is 4
 else
 cout << "Your made an invalid choice. " << endl;
//remove 2nd printout of menu and 2nd cin statement
 cout<<" 1. Deposit \n";
 cout <<" 2. Withdraw\n";
 cout <<" 3. Print the Balance\n"<<endl;
 cout << endl << "Enter your choice: 1, 2, 3, or 4. "
 <<endl;
 cin >> choice;
}//end do while loop here.
 return 0;
}

```

### Post Lab Assignment 19

1. What was the output of this lab's program on the first run?
2. What was the output on the second run?
3. Why was it necessary to have an else if for user choice of 4 in the do .. while but not in the while loop?
4. Observations and Discoveries
5. Problems with the Lab

## Laboratory Assignment 20

### Purpose:

- To practice using the char variable type
- To practice writing a switch statement

### Reading Assignment: Section 5.6

### Lab Programming Assignment:

1. Modify your main from Laboratory Assignment 19 to use a switch statement instead of a if .. else if
  - a. Inside the main option
    - i. remove the else if statements and replace them with a switch and case statements
    - ii. Add a break statement to the end of each case statement
2. Run your program.
3. As the user:
  - a. Select deposit
  - b. Select print the balance
  - c. Select withdraw
  - d. Select print the balance
  - e. Select Quit
4. Change the type of choice to char
5. Change the prompt to the user to read  
D: Deposit  
W: Withdraw  
P: Print your Balance  
Q: Quit  
Enter your Choice: D, W, P, or Q.
6. Change your case labels appropriately to recognize capitals or lower case responses.
7. Change the While statement at the end of the loop to continue so long as choice is not Q and choice is not q.
8. Run your program again with the same selections as before.

```

// Lab programming assignment 20
// This program adds a switch to the main and a char variable
#include <iostream>
#include <iomanip>
using namespace std;
class WebAccount
{ public:
 WebAccount()
 { Balance = 0;}
 void displayWelcomeMessage()
 { cout<<"Welcome to Accounts on the Web " << endl;}
 float getBalance()
 { return Balance;}
 void Deposit(float amount)
 { Balance = Balance + amount;}
 void Withdraw(float amount)
 { if (amount <=Balance)
 Balance = Balance - amount;
 else
 cout << "Insufficient Funds" << endl;
 }
private:
 float Balance;
};
int main()
{
 WebAccount VacationAccount;
 int choice;
 float amount;
 do
 {
 cout<<" 1. Deposit \n";
 cout <<" 2. Withdraw\n";
 cout <<" 3. Print the Balance\n"<<endl;

 cout << endl << "Enter your choice: 1, 2, 3, or 4. " <<endl;
 cin >> choice;
 //start switch statement here
 if (choice == 1)//remove if and replace with case
 {//remove the open brace, it is not needed.
 cout << "How much is the check you wish to deposit? ";
 cin >> amount;
 VacationAccount.Deposit(amount);
 //Add the break statement
 }//remove the close brace, it is not needed.
 else if (choice ==2) //remove else if and replace with case
 {
 //remove the open brace, it is not needed.
 cout << "How much is the withdrawal? ";
 cin >> amount;
 VacationAccount.Withdraw(amount);
 //Add the break statement
 }//remove the close brace, it is not needed
 }
}

```



```
 else if (choice ==3) //remove else if and replace with case
 cout << "Your Balance is " << fixed<< setprecision(2) <<
 VacationAccount.getBalance() << endl;
 //Add the break statement
 else if (choice == 4) //remove else if and replace with case
 cout << "Goodbye"<< endl; //Add the break statement
 else //remove else and replace with default case
 cout << "You made an invalid choice. " << endl;
 //end case
}while (choice != 4);
return 0;
}
```

### Post Lab Assignment 20

1. What was the output of this lab's program on the first run?
2. What was the output on the second run?
3. Observations and Discoveries
4. Problems with the Lab

## Laboratory Assignment 21

### Purpose:

- To be able to code nested loops

### Reading Assignment: none

### Lab Programming Assignment:

1. Modify the lab 21 program by adding a For loop nested inside a Do ... While loop.
2. The purpose of the for loop is to get 12 fahrenheit temperatures from a file, convert them to centigrade and print them on one line. The 12 temperatures were measured on one day at 2 am, 4, 6,....midnight.
3. The purpose of the do..while loop is to force the for loop to execute for several days.
4. Insert an if statement to test for file not found.
5. Run your modified program

```

//Laboratory Assignment 21
// Identifying comments
//This program uses a nested loop
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
class Temperature
{
 public:
 double getFahrenheit()
 {
 return Fahrenheit;
 }
 double getCentigrade()
 {
 return Centigrade;
 }
 void setFahrenheit(double F)
 {
 Fahrenheit = F;
 }
 void setCentigrade(double C)
 {
 Centigrade = C;
 }
 void ConvertFahrenheitToCentigrade()
 {
 Centigrade = (Fahrenheit - 32)*5/9.0;
 }
 private:
 double Fahrenheit, Centigrade;
};
int main()
{
 double F;
 int counter = 1;
 Temperature localtemp;
 ifstream infile;
 infile.open("temperaturedata.txt");
 // Insert an if statement to test for file not found
 cout << setw(32) << "Time" << endl;
 cout << "Day";
 for (int hour = 200; hour <2500; hour+=200)
 {
 cout<< setw(5)<< hour;
 }
 cout <<endl<< fixed << setprecision(0);
 // Begin do loop here

 cout << setw(3) <<counter;
 // Insert a for loop to get each days 12 temperatures
 ++counter;
 // Insert a statement to "throw the carriage" to the next line
 // End do ... while loop here
 return 0;
}

```

### **Post Lab Assignment 21**

1. What was the output of this lab's program?
2. How many times did the first for loop execute?
3. How many times did the outer do....while loop execute?
4. How many times did the inner for loop execute?
5. Observations and Discoveries
6. Problems with the Lab

## Laboratory Assignment 22

**Purpose:**

- To be able to write prototypes, and function implementations
- To practice using the rand, srand, and time functions

**Reading Assignment:** 6.5, 6.6, 6.7

### Lab Programming Assignment:

1. Add the code to Lab program 22 to seed the rand function with the time function and call the rand function twice and store the results in the two number variables.
2. Run your program.
3. Replace the two function definitions above the main with their function prototypes and move the implementation of both functions to below the main.
4. Run your program again.

```
//Laboratory 22
// Identifying comments
/*This program generates two random numbers, finds their average, then outputs the results*/
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <iomanip>
using namespace std;
const int count = 2;

float findaverage(int n1, int n2)
{
 float A;
 A = float(n1 + n2)/count;
 return A;
}

void printfunction(int N1, int N2, float a)
{
 cout << fixed << setprecision(1);
 cout << "Number 1\tNumber 2\tAverage" << endl;
 cout << setw(5) << N1 << setw(17) << N2 << setw(15) << a << endl;
}

int main()
{
 int num1 = 5, num2 = 10;
 float average;
 //seed the rand function with the time function
 //call the rand function twice, once for each number.
 average= findaverage(num1, num2);
 printfunction(num1, num2, average);

return 0;
}
```

### Post Lab Assignment 22

1. What was your output on the first run?
2. What was your output on the second run?
3. Observations and Discoveries
4. Problems with lab.

## Laboratory Assignment 23

### Purpose:

- To be able to differentiate between global and local variables and parameters.
- To be able to determine the scope of a variable.

**Reading Assignment:** Section 6.9, 6.10

### Lab Programming Assignment:

1. Type in the program below and execute it. What is the output?
2. Write cout statements in the main that use all the variables whose scope includes the main.
3. Write cout statements in the function that use all the variables whose scope includes the function
4. Execute the program again. What is the output?

```
//Laboratory 23
//This program tests the scope of variables
#include <iostream>
using namespace std;

int A= 1;
int a =0;

void function1(int);

int main()
{
 cout << "First A is " << A << endl;
 int A = 2;
 int b = 0;

 cout << " Then A is " << A << endl;
 for(int i = 0; i < 10; ++ i)
 ++b;
 function1(b);
 cout << " After the function A is " << A << endl;
 return 0;
}

void function1(int c)
{
 int A = 3;
 int d = 0;
 cout << " Inside the function A is" << A << endl;
}
```



### Post Lab Assignment 23

1. What was the output of this lab's program on the first execution.
2. On the second execution?
3. There are 3 declarations for a variable named A.
  - a. What is the value of the global variable A?
  - b. What is the value of the local to the main variable A?
  - c. What is the value of the local to the function variable A?
4. Make a list of all the global identifiers with file scope.
5. Make a list of all the local identifiers with block scope.
6. Make a list of all the variables used as function parameters.
7. Make a list of all the variables used as function arguments.
8. Observations and Discoveries
9. Problems with the Lab

## Laboratory Assignment 24

**Purpose:** To be able to diagram the call stack and activation records of a program

**Reading Assignment:** 6.11

**Lab Programming Assignment:**

1. Using the program for this lab, diagram the call stack of activation records that will occur as the program executes

```
//Laboratory 24
// Identifying comments
//This program generates two random numbers, finds their average, then outputs the results
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <iomanip>
using namespace std;
const double count = 2;

float findaverage(int n1, int n2);
void printfunction(int N1, int N2, float a);

int main()
{
 int num1, num2;
 float average;
 srand(unsigned(time(0)));
 num1 = rand();
 num2 = rand();
 average= findaverage(num1, num2);
 printfunction(num1, num2, average);
 return 0;
}

float findaverage(int n1, int n2)
{
 float A;
 A = (n1 + n2)/count;
 return A;
}

void printfunction(int N1, int N2, float a)
{
 cout << fixed << setprecision(1);
 cout << "Num 1\tNum 2\tAverage" << endl;
 cout << N1 << "\t" << N2 << "\t" << a << endl;
}
```

### Post Lab Assignment 24

1. Diagram the call stack below:
2. Observations and Discoveries
3. Problems with lab.

## Laboratory Assignment 25

**Purpose:** To be able to use call by reference and call by value and to know the difference.

**Reading Assignment:** Section 6.14

**Lab Programming Assignment:**

Modify the program below:

1. Write the prototype above the main and implementation below the main for the getData function whose call is in the main.
2. Make num1 and num2 value parameters just like we have done in the past.
3. Run your program.
4. Change the prototype and heading so that both num1 and num2 are passed by reference.
5. Run your program again.

```
//Laboratory Assignment 25
// Identifying comments
//This program uses a function to get two numbers and stresses the
//need for passed by reference.
#include <iostream>
using namespace std;

//insert prototype here

int main()
{
 int num1 = 0;
 int num2 = 0;
 getData(num1, num2);
 cout << "The numbers are " << num1 << " and " << num2 << endl;
 return 0;
}

//insert implementation here
```

### Post Lab Assignment 25

1. What was the output of this lab's program on the first execution
2. What was the output of this lab's program on the last execution?
3. How do you indicate that a parameter is to be passed by reference?
4. Observations and Discoveries
5. Problems with the Lab

## Laboratory Assignment 26

### Purpose:

- To be able to declare arrays
- To be able to initialize arrays inline.
- To use a FOR loop to initialize a large array.

Reading Assignment: Section 7.1, 7.2, 7.3

### Lab Programming Assignment:

Modify the program with this lab by

1. Declaring an array (call it A) of type double, of size 5.
2. Initializing that array to the values

|   |     |     |   |    |
|---|-----|-----|---|----|
| 0 | 7.1 | 2.3 | 5 | 10 |
|---|-----|-----|---|----|

3. Declaring another array (call it B) of integers of size MAX.
4. Using a FOR loop to initialize all the cells to -1.
5. Run your program. Depending on your compiler, you should get some kind or error on the first cout line.
6. Change the subscripts on A and run your program again.

```
//Laboratory Assignment 26
// Identifying comments
//This program declares, initializes and prints two arrays.
#include <iostream>
using namespace std;
const int MAX = 100;

int main()
{
 //declare and initialize array A here
 //declare array B here
 //insert a for loop to initialize B here

 cout << A[1] << " " << A[2] << " " << A[3] << " " << A[4] << " " << A[5] << endl;
 for(int i = 0; i < MAX; ++i)
 cout << B[i] << " ";
 cout << endl;
 return 0;
}
```

### Post Lab Assignment 26

1. What was the error on this lab's program on the first execution?
2. What was the output on the second execution?
3. Observations and Discoveries
4. Problems with the Lab

## Laboratory Assignment 27

**Purpose:** To be able to access and print the elements of an array.

**Reading Assignment:** Section 7.4

**Lab Programming Assignment:** Modify the program this lab's program to

1. Print the list
2. Run your program.

```
//Laboratory Assignment 27
// Identifying comments
//This program prints the elements of an array.
#include <iostream>
using namespace std;
const int MAX = 10; //Max is the maximum size of the array
class listOfNumbers
{
 public:
 listOfNumbers() //constructor
 {length = 0;
 }
 void initializeList()
 {
 cout << " How many numbers are in your list? ";
 cin >> length;
 if (length > 10)
 cout << "List is too long. " << endl;
 else
 for (int i = 0; i<length; ++i)
 {
 cout << "Enter number " << i+1 << ": ";
 cin >> Numbers[i];
 }
 }

 //insert your printList member function here

 private:
 int Numbers[MAX];
 int length;
};
int main()
{
 listOfNumbers MyList; //object which is a list of numbers
 MyList.initializeList();

 //call your printList class member function here

 return 0;
}
```



### **Post Lab Assignment 27**

1. What was the output of this lab's program on the first execution
2. Observations and Discoveries
3. Problems with the Lab

## Laboratory Assignment 28

### Purpose:

- To be able to search an array.
- To use parallel arrays

### Reading Assignment: Section 7.7

### Lab Programming Assignment:

Modify the program this lab's code to include a search routine inside the class AccountList.

1. The search member function should have an integer called key as it's only parameter and should return an integer.
2. Using a linear search, when the key is found the function should stop and return the index value.
3. If the key is not found the function should return a -1

Call your search member function from the main.

Include, in the main, an error message to be printed if the key is not found. The main should stop execution once the error message is printed.

```

// Lab programming assignment 27
/* This program searches an array for a key value and returns the associated value in a parallel
array*/
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
using namespace std;
int const MAX = 500;
class AccountList
{public:
 AccountList()
 { length = 0;}
 AccountList(ifstream& infile)
 { length = 0;
 while(infile >> idList[length])
 infile >> balanceList[length++];
 }
 int getID(int index)
 { return idList[index];}
 float getBalance(int index)
 { return balanceList[index];}

 //insert your search list class function here

private:
 int idList[MAX];
 float balanceList[MAX];
 int length;
};
class WebAccount
{ public:
 WebAccount()
 { balance = 0;}
 void displayWelcomeMessage()
 {cout<<"Welcome to Accounts on the Web "
 << endl;
 }
 float getBalance()
 { return balance;}
 void setBalance(float b)
 { balance = b;}
 void setID(int num)
 { ID = num;}
 void Deposit(float amount)
 { balance = balance + amount;}
 void Withdraw(float amount)
 { if (amount <=balance)
 balance = balance - amount;
 else
 cout << "Insufficient Funds" << endl;
 }
};

```

```

 }
private:
 float balance; int ID;
};
int main()
{
 ifstream infile;
 infile.open("sunriseAccounts.txt");
 AccountList SunriseBank(infile);
 WebAccount VacationAccount;
 int ID;
 int index;
 VacationAccount.displayWelcomeMessage();
 cout << "What is your account ID number? ";
 cin >> ID;
 //insert your call to search the list for ID here
 /*if the ID is not found insert your error message here to be printed and then end the
 program. */
 VacationAccount.setID(ID);
 VacationAccount.setBalance(SunriseBank.getBalance(index));
 cout << "Your current balance is " << fixed << setprecision(2) <<
 VacationAccount.getBalance() << endl;
 return 0;
}

```

### Post Lab Assignment 28

1. What was the output of this lab's program on the first execution
2. Usually a webpage will give you more than one chance to enter a valid ID. What changes do you think would need to be made to allow the user more than one chance to get his/her ID correct?
3. Observations and Discoveries
4. Problems with the Lab

## Laboratory Assignment 29

### Purpose:

- To be able to sort an array and maintain the associated parallel array

### Reading Assignment: Section 7.8

### Lab Programming Assignment:

Modify the program this lab's code to include an insertion sort routine inside the class AccountList which will sort the ID array in ascending order.

1. The sort member function will not return a value and has no parameters
2. As you sort the array of IDs and move the members of the ID array, you must also move the members of the Balance array.

Call your sort member function from the main.

```

// Lab programming assignment 29
// This program sorts two parallel arrays
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
using namespace std;
int const MAX = 500;
class AccountList
{public:
 AccountList()
 { length = 0;}
 AccountList(ifstream& infile)
 { length = 0;
 while(infile >> idList[length])
 infile >> balanceList[length++];
 infile.close();
 }
 int getID(int index)
 { return idList[index];}
 float getBalance(int index)
 { return balanceList[index];}
 void printToFile(ofstream& outfile)
 { for(int i = 0; i < length; ++i)
 outfile << idList[i] <<"\t"<<balanceList[i] << endl;
 outfile.close();
 }
 //insert your sort function here
private:
 int idList[MAX]; float balanceList[MAX]; int length;
};
int main()
{ ifstream infile;
 ofstream outfile;
 infile.open("sunriseAccounts.txt");
 outfile.open("sunriseAccountssorted.txt");
 AccountList SunriseBank(infile);
 // call your sort function here

 SunriseBank.printToFile(outfile);
 return 0;
}

```

### Post Lab Assignment 29

1. Inspect your output file. Is it sorted by ID?
2. Compare your output file to your input file. Do the balances still correspond to the correct IDs?
3. Observations and Discoveries
4. Problems with the Lab



## Laboratory Assignment 30

### Purpose:

- To be able perform operations on a two dimensional array.

**Reading Assignment:** Section 7.9, 7.10

### Lab Programming Assignment:

Modify the program this lab's program

1. Write a print member function to print the grades array.
2. Call your print member function for the Spring Grade book
3. Call your print member function for the Fall Grade book.

```

//Laboratory Assignment 30
//This program declares and prints two 2-dimensional arrays
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
const int maxStudents = 10;
const int maxTests = 3;
typedef int table[maxStudents][maxTests];
class Gradebook
{
 public:
 Gradebook() //default constructor
 { courseName = "XXX";
 numStudents = 0;
 numTests = 0;
 }
 Gradebook(string name, table arrayOfGrades, int num1,
 int num2)
 //initializing constructor
 { courseName = name;
 numStudents = num1;
 numTests = num2;
 for (int r = 0; r<numStudents; ++r)
 for(int c=0; c<numTests; ++c)
 grades[r][c] = arrayOfGrades[r][c];
 }
 void fillGrades(ifstream & infile)
 /*This function fills the grades Array from a file*/
 { for(int r= 0; r< numStudents; r++)
 for (int c=0; c< numTests; c++)
 infile >> grades[r][c];
 }
 int findMaximum()
 { int highgrade = 0;
 for(int r =0; r<numStudents; r++)
 for(int c=0; c<numTests; c++)
 if(highgrade < grades[r][c])
 highgrade = grades[r][c];
 return highgrade;
 }
}

```

```

double findTestAverage(int testNum)
/*This function finds and returns the average grade for
all students taking one test*/
{ double sum = 0;
 for (int r=0; r<numStudents; r++)
 sum = grades[r][testNum];
 return sum/numStudents;
}
void findStudentAverages(double Averages[])
/*This function finds the averages for all the students
and stores those averages in a one dimensional
array, Averages.*/
{ double sum=0;
 for(int r =0; r<numStudents; r++)
 {
 for(int c = 0; c<numTests; c++)
 sum += grades[r][c];
 Averages[r] = sum/numTests;
 }
}
//Insert member function to print the grades array here
private:
 string courseName; table grades;
 int numStudents, numTests;
};
int main()
{
 table gradesList = {{90,90,90},{80,80,80},{70,70,70},
{60,60,60},{50,50,50},{40,40,40},{30,30,30},{20,20,20}};
 string name = "CS 155";
 Gradebook fallGradebook;
 Gradebook springGradebook(name, gradesList, 8, 3);
 // insert your call to print the spring Gradebook here
 //insert your call to print the fall Gradebook here
 return 0;
}

```

### **Post Lab Assignment 30**

1. What was the output of this lab's program on the first execution
2. Observations and Discoveries
3. Problems with the Lab



## Programming Assignments

### Programming Assignment 1

Write a program that finds the sum, difference, and product of two integers of your choice. The two integers should be stored as constants. The sum, product, and difference should be stored as variables. The program should output to the screen, with appropriate labels, the original numbers, their sum, their difference and their product. Example output is shown below.

```
First number: 5
Second number: 30
Sum: 35
Difference: -25
Product: 150
```

Note: You will be graded not only for correct program and output but also for use of whitespace, good identifiers (variable names), and use of comments. In order to receive full credit, you must use the required elements (constants, variables, output to screen, etc). Remember your program must include as comments at the top: your name, the programming assignment number, the date the program is due, and a short sentence explaining what the program does. Additionally it is a good idea to include under these comments a variable dictionary.

## Programming Assignment 2

According to the Boy Scout Handbook, you can estimate the current temperature in Fahrenheit by counting the number of cricket chirps you hear in a 15 second interval. For example if you hear 30 cricket chirps in 15 seconds the approximate current temperature is 67 degrees Fahrenheit.

Write a program that uses 3 user defined functions (plus the main)

- One function should prompt the user for the number of cricket chirps they have heard in the last 15 seconds and store that number in memory
- One function should calculate the approximate Fahrenheit temperature by adding 37 to the user entered number
- One function should output the results as specified below.

OUTPUT:

The approximate temperature is 67 degrees Fahrenheit.

Note: Your main should consist only of variable declarations and the three function calls.

You will be graded on:

Correctness of program

- Your program must get the correct answer
- Your output must be as specified
- You must use the three user-defined functions as specified

Correct use of whitespace in your program

Correct use of meaningful identifiers (variable and function names).

Correct use of comments

### Programming Assignment 3

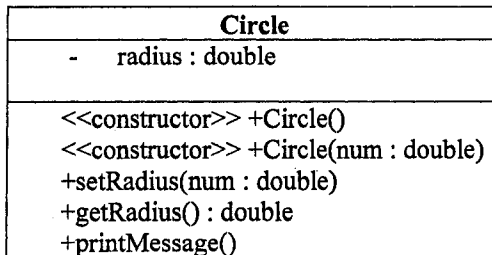
Write a class **Circle**. Your class should include a private data member, radius (of type double), a default constructor, an initializing constructor, a “get” member function, a “set” member function and a printMessage function which prints the radius of the circle appropriately labeled. Write a main to test your class. In the main you must at least

- Declare an object of type circle with no parameters.
- Prompt the user for a circle’s radius.
- Set the data member of your circle to the value obtained from the user.
- Print a message about your circle (using the member function)
- Declare another object of type circle with 1 argument (value of your choice).
- Get the radius of this new circle.
- Print the results of that Get, appropriately labeled

Output should be as specified below:

The radius of the circle is 2.33

The radius of the second circle is 4.55



Notes: Use the UML diagram above to help you in writing this program. You will be graded on :

- correctness of program (correct output)
- Output as specified
- Use of required elements
- Use of comments and whitespace
- Use of meaningful identifiers



### Programming Assignment 4

Write a program that prompts the user for an unknown number of gas purchases and miles traveled since the last purchase (use a sentinel controlled while loop). The program should cue the user to enter -1 when all gas purchases have been entered. The program should find the miles per gallon for each fillup individually and should also calculate and print the overall mpg. Output as specified below:

```
MPG this tankful: 22.076923
Total MPG; 22.076923
MPG this tankful: 20.000000
Total MPG: 21.173913
```

.  
.  
.

Use classes as described in the UML diagrams below.

| <b>FillupMPG</b>                                                                                                           |
|----------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>- gallons: Decimal</li> <li>- distance: Decimal</li> <li>- MPG : Decimal</li> </ul> |
| <pre>+ setGallons( g : Decimal) + setDistance(d: Decimal) + calculateMPG() + getMPG() : Decimal</pre>                      |

| <b>OverallMPG</b>                                                                                                                                                     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>- totalGallons: Decimal</li> <li>- totalDistance: Decimal</li> <li>- TotalMPG : Decimal</li> </ul>                             |
| <pre>«constructor» + OverallMPG () + accumulateTotalGallons( g : Decimal) + accumulateTotalDistance(d: Decimal) + calculateTotalMPG() + getTotalMPG() : Decimal</pre> |

Note: You will be graded on

1. your use of the required elements in this program.
2. Your output should be as specified
3. You should include required comments at the top of your program. You should document identifiers with comments. Be sure to document parts of the program if any parts are unclear.
4. You should use self-documenting identifier wherever possible.
5. Do not use unnecessary documentation such as `int x //declaring x`

### Programming Assignment 5

Write a program that gets 100 Fahrenheit temperatures from a file and finds the average and standard deviation of the temperatures. Since this is a counted loop, you must use a For Loop. You must also use the class Temperature that we developed in class (on the next page) in your program but you will not use all the class functions (This program does not use Centigrade at all). You should get the temperatures from the file in your main and then use a set function to store the values into the class data members. You will need to write one class function for your temperature class that accumulates the sum of the temperatures and one that accumulates the sum of the temperatures squared as they are read from the file. You will also need to add two data members to your class: totalTemperature and totalTemperatureSquared. You may calculate the average (correct to one decimal point) and the standard deviation (correct to 2 decimal points) in the main or in the class (your choice). Output as specified below.

Temperatures

90

56

78

.

.

99

47

Average Temperature = 58.6

Standard Deviation = 5.78

---

Note: Use the following formula to calculate the standard deviation

$$\text{standard deviation} = \sqrt{\frac{\sum(T^2) - 100 * (\text{average})^2}{99}}$$

You will be graded on:

1. Use of required elements in your program
2. Correctness of results
3. Output formatted as specified
4. Use of comments, whitespace, and meaningful identifiers

```

class Temperature
{ public:
 double getFahrenheit()
 {
 return Fahrenheit;
 }
 double getCentigrade()
 {
 return Centigrade;
 }
 void setFahrenheit(double F)
 //F is the Fahrenheit temperature from the client function
 {
 Fahrenheit = F;
 }
 void setCentigrade(double C)
 //C is the Centigrade temperature from the client function
 {
 Centigrade = C;
 }
 void ConvertFahrenheitToCentigrade()
 {
 Centigrade = (Fahrenheit - 32)*5/9.0;
 }
private:
 double Fahrenheit, Centigrade;
};

```

### Programming Assignment 6

Write a program that reads from a file named Program6data.txt into 2 parallel arrays the name and grade point average (GPA) of an unknown number of students (maximum of 100 students). The program must find the average GPA and then print out the names of all students whose GPA is greater than the average GPA. Sample input and output is given below.

#### Sample input file

```
Smith 2.75
Jones 3.65
Johnson 2.11
```

#### Sample Output

```
Jones
```

Note: You must use a class for your data and functions. Your main should consist of calls to your class functions.

You will be graded on:

1. Correctness and format of output
2. Use of specified elements
3. Use of comments to document the program, parts of the program, and identifiers.
4. Use of whitespace and meaningful identifiers

### Scoring Rubric For Programming Assignments

|                                                                                                       | Prog<br>1 - 3 | Prog<br>4 - 6 |
|-------------------------------------------------------------------------------------------------------|---------------|---------------|
| Program compiles and executes for test data *                                                         |               |               |
| Program gets correct answer(s) and output is formatted as specified                                   | 2             | 2             |
| Code makes good use of white space, indentation, other delineators                                    | 2             | 1             |
| Code includes necessary comments                                                                      | 3             | 2             |
| correct spelling and syntax                                                                           |               |               |
| required identification comments                                                                      |               |               |
| each variable/object commented or variable/object dictionary                                          |               |               |
| function and class identifiers commented                                                              |               |               |
| Pseudocode in comments or separate attachment and matches code                                        | 0             | 3             |
| Code makes use of meaningful identifiers                                                              | 2             | 1             |
| Code makes use of required C++ statements/elements                                                    | 5             | 5             |
| Variables/objects/constants typed correctly, declared in correct positions, Parameters used correctly | 1             | 1             |

\*Program must compile in order to be graded

**APPENDIX C**  
**CLASSROOM AND COURSE INFORMATION**

### **Catalog Description for CS 1 at Athens State University**

**Computer Science I (C++) 3 Semester Hours.** This is the first course for any new CS or CIS student at ASU who lacks programming experience or has never been exposed to C++. It provides an introduction to computers and programming with problem solving techniques. Arithmetic and relational operations as well as I/O for elementary data types are covered as basic language constructs for alternation and iteration. Students are introduced to the concepts and the rationale for structured programming, using functions. The course will take the student through the use of structured data types strings, arrays, text files and records (strucs). Programming assignments focus on the techniques of good programming style and how to design, code, debug, and document programs.

### **Catalog Description for CS 1 at University of North Alabama**

CS 155. (3) **Computer Science I.** An introduction to the theoretical foundations of computer science, the components of algorithms and the representation of these components using a high-level programming language. Special emphasis on software development and an introduction to object-oriented programming.

## Class Calendar

| class | Chapter/<br>Section | Lecture Topic                                                                                                            | Tests/<br>Surveys           | Lab<br># |
|-------|---------------------|--------------------------------------------------------------------------------------------------------------------------|-----------------------------|----------|
| 1     |                     | Welcome to Class. Who am I? Who are you? What is Computer Science?                                                       |                             |          |
| 2     |                     | Explanation of Study                                                                                                     | KLSI, ACCC,<br>Consent Form |          |
| 3     | 1                   | History of C and C++, C++ Standard Library, Visual C++, C++ development environment, using C++ editors and compilers.    |                             | 1        |
| 4     | 2.1–2.3             | Basic C++ program, standard stream output statements, syntax and logic errors, literals, comments                        |                             | 2        |
| 5     | 2.4, 2.5            | Reserved words, identifiers, variables, integer data type, assignment statement, constants                               |                             | 3        |
| 6     | 2.6                 | Standard stream input statements, operator precedence, using namespace std                                               |                             | 4        |
| 7     | 2.7, 4.4            | Relational operators, sequence, selection, the if statement,                                                             |                             | 5        |
| 8     |                     | review and catch up                                                                                                      |                             |          |
| 9     |                     | session test                                                                                                             |                             |          |
| 10    |                     | After test review                                                                                                        |                             |          |
| 11    | 6.1–6.3             | Math library functions, arguments, parameters, writing functions with a parameter that return a value                    |                             | 6        |
| 12    | 6.4, 6.12           | void functions with no parameters, functions with multiple parameters                                                    |                             | 7        |
| 13    | 3.1–3.4             | Introduction to objects and UML. Classes, objects, writing member functions without parameters, calling class functions. |                             | 8        |
| 14    | p.43, 44,<br>72–74  | char and string data types, get and getline, formatting with setw                                                        |                             | 9        |
| 15    | 3.6                 | Accessing private data members                                                                                           |                             | 10       |
| 16    |                     | Classes with data members, using constructors to initialize private data members                                         |                             | 11       |
| 17    | 4.1–4.3             | Algorithms, writing pseudocode                                                                                           |                             | 12       |
| 18    | 4.5                 | The if-then-else statement, UML specification diagrams, boolean data type                                                |                             | 13       |
| 19    | 4.6, 4.7            | The if-else if statement, float and double data type                                                                     |                             | 14       |
| 20    | 4.8                 | repetition control statements, the counter-controlled while loop, increment and decrement operators, counters            |                             | 15       |
| 21    |                     | review and catch up                                                                                                      | ACCC                        |          |
| 22    |                     | session test                                                                                                             |                             |          |
| 23    |                     | After Test Review                                                                                                        |                             |          |
| 24    | 4.9                 | The sentinel-controlled while loop, abbreviating assignment expressions                                                  |                             | 16       |
| 25    |                     | Using files, the end of file controlled while loop                                                                       |                             | 17       |
| 26    | 5.1–5.4             | The for loop, formatting real number output                                                                              |                             | 18       |



| class | Chapter/<br>Section | Lecture Topic                                                                 | Tests/<br>Surveys | Lab<br># |
|-------|---------------------|-------------------------------------------------------------------------------|-------------------|----------|
| 27    | 5.5                 | The do-while loop, using the eof marker                                       |                   | 19       |
| 28    | 5.6, 5.7            | Switch and case statements, the break and continue statements, char variables |                   | 20       |
| 29    | 5.8                 | Nested loops, logical operators                                               |                   | 21       |
| 30    |                     | catch up and review                                                           |                   |          |
| 31    |                     | session test                                                                  |                   |          |
| 32    |                     | After Test review                                                             |                   |          |
| 33    | 6.5–6.8             | Function prototypes and implementations, random and time functions            |                   | 22       |
| 34    | 6.9–6.10            | Scope, local and global variables                                             |                   | 23       |
| 35    | 6.11                | The function call stack and activation records                                |                   | 24       |
| 36    | 6.14                | Call by reference, call by value                                              |                   | 25       |
| 37    | 7.1–7.3             | Arrays, declaring and initializing                                            |                   | 26       |
| 38    | 7.4                 | Accessing and printings arrays                                                |                   | 27       |
| 39    | 7.7                 | Parallel arrays, Searching arrays                                             |                   | 28       |
| 40    | 7.8                 | Sorting arrays                                                                |                   | 29       |
| 41    | 7.9, 7.10           | 2-dimensional arrays, typedef statements, passing 2-dim arrays                |                   | 30       |
| 42    |                     | Catch up and Final Review                                                     | ACCC              |          |
|       |                     |                                                                               | Final Exam        |          |

365

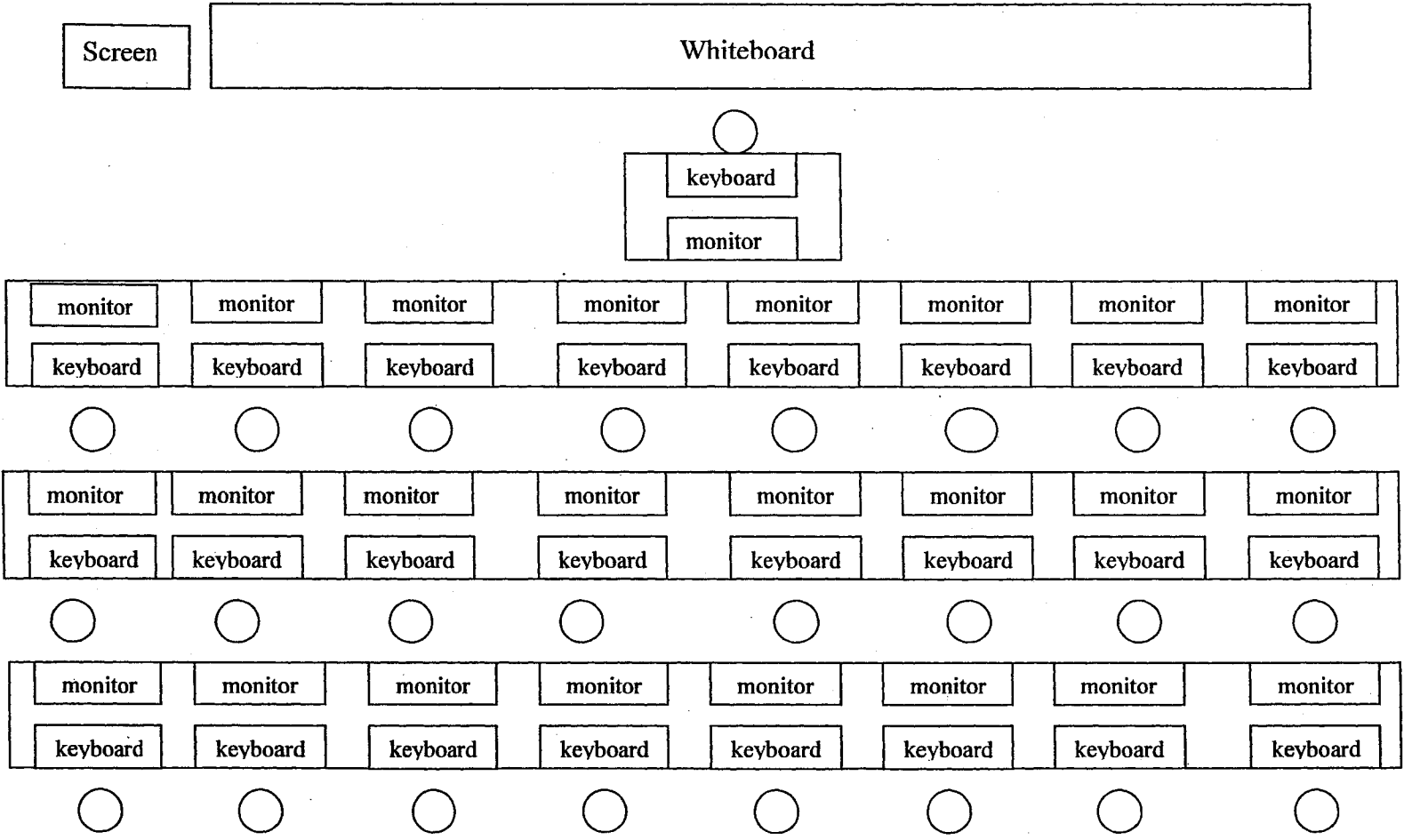


Figure A.2 Classroom at ASU

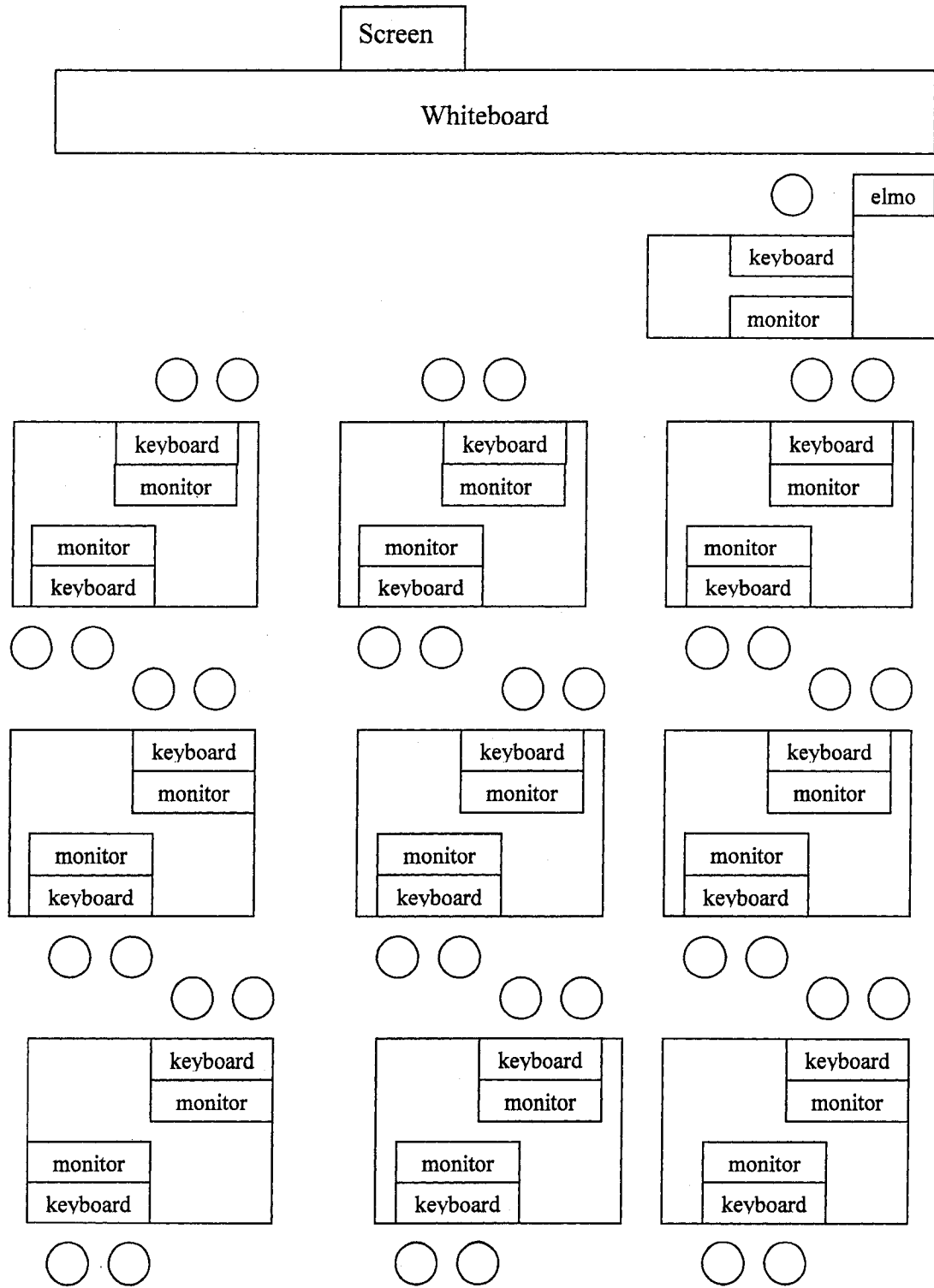


Figure A.1 Classroom at UNA

**APPENDIX D**  
**INSTITUTIONAL REVIEW BOARD**



*Florida Institute of Technology*

Institutional Review Board Office  
Dr. Lisa Steelman, Chair IRB  
School of Psychology  
(p) 674-8104  
[lsteelma@fit.edu](mailto:lsteelma@fit.edu)  
<http://www.fit.edu/research/committees/irb/index.html>

Researcher Information

- Your research has been approved by Florida Tech's IRB for **one year** from the date on your signature page. If your research runs beyond this date, please submit a Continuing Review Form.
- If **changes** to your research protocol are made you must submit a Revision Request Form.
- Should an **adverse event** that is serious and unexpected happen to a participant as a result of participating in your research study, you must submit an Adverse Events Reporting Form within 24 hours of the event.
- Your IRB identification number is: 05-2107

All forms may be found on the IRB website.

STUDENT APPLICATION  
RESEARCH INVOLVING HUMAN SUBJECTS  
EXPEDITED REVIEW

Name: Jean Henderson Date: 10/20/2005  
Major: Science Education Course: EDS 6999 Dissertation Research -Science Education  
Title of Project: The Effect of Classroom Structure on Student Achievement in and Attitudes toward Computer Science: A Comparison of Three Classroom Settings

Directions: Please provide the information requested below (please type). Use a continuation sheet if necessary, and provide additional supporting documentation. Please sign and date this form and then return it to your major advisor. You should consult the university's document "Principles, Policy, and Applicability for Research Involving Human Subjects" prior to completion of this form. Copies may be obtained from the Office of the Vice President for Research.

**1. List the objectives of the proposed project.**

The purpose of this study is to investigate the relationships among various student attributes and different classroom structures on student achievement in and attitudes toward computer science. Student attributes include gender, mathematics background, computer education, and learning styles. The classroom structures include a traditional class with a separate lab component, and two classes that integrate the lab component: a hands-on-closed setting and an instructor-led interactive setting. Students in an introduction to computer science course will be given specific laboratory assignments that complement the coursework and required to complete them in one of the three settings.

**2. Briefly describe the research project design/methodology.**

Two different true experimental research designs will be used in this study: randomized posttest-only control group for achievement and randomized pretest-posttest control group for attitude. Intact classes over two semesters will be used. In the first semester, Fall 2005, all students will be assigned to a control group. In the second semester, Spring 2006, students will be randomly assigned to one of two experimental groups. Pre-existing student attributes of math ability, previous programming experience, and gender will be measured and used as covariates to adjust for group differences and to establish group equivalency. Students in all classes will be surveyed for these student attributes at the beginning of each semester. All groups will be administered a pre-attitude assessment at the beginning of each semester and a post-attitude assessment at the end of each semester. Students will be administered a comprehensive teacher-constructed final exam to assess overall achievement in computer science at the end of the semester.

**3. Describe the characteristics of the subject population, including number, age, sex, etc.**

The subject population will consist of college students enrolled in a beginning computer science class at the University of North Alabama in Florence, Alabama. Most of these students will be between 18 and 25 years old although some may be returning older students. There will be a mixture of male and female participants. The majority of the students will be computer science or math majors. Approximately 60 students will participate in the study.

**4. Describe any potential risks to the subjects – physical, psychological, social, legal, etc. – and assess their likelihood and seriousness.**

I do not foresee any risks to the participants other than a possible breach of confidentiality. This is explained in the attached consent form.

**5. Describe the procedures you will use to maintain confidentiality from your research subjects and project data. What problems, if any, do you anticipate in this regard?**

Each participant will be assigned a pseudonym. I will be the only one who knows the participants' true identity. I will maintain the only copy of the raw data and will destroy original assessment forms at the conclusion of the study.

6. Describe your plan for informed consent (attach proposed form).

At the beginning of each semester students will be informed about the study and will be asked to read and sign the attached informed consent form.

7. Discuss what benefits will accrue to your subjects and the importance of the knowledge that will result from your study.

This study will provide further information about the types of classroom structure that may benefit students in beginning computer science courses and lead to lower levels of attrition in those courses. It will also provide information about the correlation between classroom structure and student attributes such as previous computer education, math background, gender, and learning styles. A product of this study will be a profile of the successful student in an introductory computer science course relative to the class structure.

8. Explain how your proposed study meets criteria for exemption from Institutional Review Board review.

This research is being conducted in an established educational setting involving normal educational practices. It is also using educational tests where the subjects will remain anonymous. Under Department of Health and Human Services Federal regulations [45 CFR 46.101(b)], these criteria exempt the research from review by the IRB.

I understand Florida Institute of Technology's policy concerning research involving human subjects and I agree:

1. to accept responsibility for the scientific and ethical conduct of this research study.
2. to obtain prior approval from the Institutional Review Board before amending or altering the research protocol or implementing changes in the approved consent form.
3. to immediately report to the IRB any serious adverse reactions and/or unanticipated effects on subjects which may occur as a result of this study.
4. to complete, on request by the IRB, a Continuation Review Form if the study exceeds its estimated duration.

Signature Jean Hendon

Date 9/12/2005

This is to certify that I have reviewed this research protocol and that I attest to the scientific merit of the study, the necessity for the use of human subjects in the study to the student's academic program, and the competency of the student to conduct the project.

Major Advisor Michael Gallo

Date 10/20/05

This is to certify that I have reviewed this research protocol and that I attest to the scientific merit of this study and the competency of the investigator(s) to conduct the study.

Academic Unit Head [Signature]

Date 10/20/05

IRB Approval [Signature]  
Name

Date 11/10/05

IRB # 05-2167 Exempt  
Title

## CONSENT FORM

### Jean Henderson

Florida Institute of Technology

Science and Mathematics Education Department

Research Title: *The Effect of Classroom Structure on Student Achievement in and Attitudes toward Computer Science: A Comparison of Three Classroom Settings*

### Invitation to Participate

You are invited to participate in a research study designed to examine the effect of different classroom structures on beginning computer science students' achievement in and attitudes toward computer science. It involves answering questions regarding gender, number of college math courses previously taken, number of computer programming courses previously taken, learning style, and attitudes toward computer science. Each questionnaire should take approximately 20–30 minutes or less to complete, depending on your responses. In addition, your instructor in Introduction to Computer Science, CS 155, will provide the researcher with a copy of your lab grades, your homework grades and your final course grade. It is imperative that you read and understand the answers to the following questions, which address the informed consent issues of this study, including those of confidentiality and anonymity. Note that your responses will be treated strictly confidential and will be accessible only by those in the research group. Your responses will remain completely anonymous and no identifying information will be collected from your responses. No reference will be made in oral or written reports that could connect you in any way to this study.

**1. What is the purpose of the study?** The purpose of this study is to gain a better understanding of the impact of different classroom structures on students' achievement in and attitudes toward computer science. Two different classroom structures will be examined: The traditional 3 days per week/50-minute classroom that includes a separate laboratory component independent of classroom instruction, and two different classrooms that integrate the laboratory component (25 minutes of instruction followed by 25 minutes of lab participation). Selected students enrolled in Introduction to Computer Science, CS 155, are being asked to participate.

**2. What will be involved in participating?** Participation in this study involves performing all of the activities that are normally associated with being a student in this college computer science courses. As part of the normal requirements of the course you will be expected to complete computer programming laboratory assignments. The type of lab in which you will complete these assignments will be a function of the classroom to which you have been assigned. In addition to these activities, you will be required to complete a survey that assesses your attitudes toward computer science. This survey will be administered during the 1<sup>st</sup>, 6<sup>th</sup>, 10<sup>th</sup> and 14<sup>th</sup> weeks of the semester. You will also be required to complete a learning styles inventory during the 3<sup>rd</sup> week to determine your learning style. These surveys are in hard copy form and will be administered during your regular class time. Student demographical data (such as student number, gender, college level math courses taken, and computer programming courses taken) will be collected anonymously and used as part of this study.

**3. Who will know what I say?** Your responses to all assessment items will be treated as strictly confidential and will be accessible only by those in the research group, including professors on the research committee at Florida Institute of Technology and the primary researcher. Your responses will remain completely anonymous and no identifying information will be collected from your answers. This means that no one will be able to identify you by name. Furthermore, no reference will be made in any oral or written reports that could connect you in any way to this study.

**4. What risks and benefits are associated with participation?** It is believed that there are no risks involved in participating in this study other than a possible breach of confidentiality, which is



addressed in the next section. As a participant, you might find it interesting and helpful to explore your attitudes toward computer science and your achievement in computer science. The results of the study will be made available in a report as part of the requirements of a doctoral dissertation. Finally, from a research perspective, the results of this study will also help further inform the teaching community on the nature of college students' learning success in this introductory computer science course, if the results of the study so indicate.

**5. What are the possible breaches to confidentiality and security?** Because one part of this study involves a questionnaire and assessment, there is a very small risk that completion of these may compromise your privacy. A description of these risks follows.

- a. To prevent multiple submissions from the same source, this study will record your unique identification number assigned to you by your college. This information will be stored in a computer file until the research is completed. Neither the researcher nor any member of the research team will have access to your personal information in a way that will identify you by name.
- b. Given the nature of data collection, all of the questionnaire responses received will be sent immediately to the primary researcher (Professor Jean Henderson), who will store the responses in a private folder accessible only by the primary researcher. Although highly unlikely, it is possible that the information could be the target of an unauthorized access. To help prevent this from happening, folders containing information will be placed in a secure location. Any computer files will be accessible only by the researchers involved in this study. Furthermore, in the unlikely event files are compromised, the data stored cannot be associated with you. All of the stored data will be deleted from the computer at the conclusion of the study.
- c. Although responding to a questionnaire does not involve e-mail contact, if you were to contact the researchers via e-mail for additional information about the study, there is a risk that others using your computer or sharing your e-mail account will be able to read your e-mail or the researcher's reply. As a result, you are encouraged to protect the privacy of your e-mail account. Contact your local ISP for information on how to do this.

**6. Are there any other possible threats to confidentiality?** Yes, there are potential legal threats, which are inherent in all research studies. For example, the court can subpoena research records but, in so doing, must protect the participant's rights to privacy. Certain types of information can also be requested under federal and state freedom-of-information or search-and-seizure laws. None of these actions, however, are anticipated for this study.

**7. What are my rights as a respondent?** You may ask any questions regarding the research, and they will be answered fully. Your participation in the study is voluntary and you may withdraw at any time.

**8. What will be published?** Following the completion of this study, results will be available initially in a summary report, and ultimately in a doctoral dissertation. It is also likely that a written report of results will be published in a scholarly journal or presented in written or oral form at national or international conferences.

**9. If I want more information, whom may I contact about the study?** For more information regarding this study you may contact the primary researcher via e-mail at [jfhenderson@una.edu](mailto:jfhenderson@una.edu). Additionally, you may contact the researcher's advisor, Dr. Michael Gallo at 321-674-7203 or via e-mail at [gallo@fit.edu](mailto:gallo@fit.edu). In addition you may contact the UNA Office of Research, Planning and Institutional Effectiveness at 765-4221. Finally, you may contact Florida Institute of Technology's Institutional Review Board for the Protection of Human Subjects who have approved this study. This board can be contacted through its staff office at 321-674-8120.

Please verify that each of these conditions is true of you and check the box below:

- I am 18 years of age or older. If I am younger than 18, I have enrolled in a UNA college course.
  - I have read and understand this informed consent statement.
  - I voluntarily agree to participate in the research.
- I would like to participate in this study and I agree to allow UNA to provide my gender, high school transcript(s), and college transcript(s) to the study's principal investigator.

If you do not wish to participate, check the box below:

I do not wish to participate

Student Number: \_\_\_\_\_ Date: \_\_\_\_\_

Name: (Printed)

\_\_\_\_\_

Signature:

\_\_\_\_\_

**Application Project Review by  
The Human Subjects Committee of the University of North Alabama  
(Institutional Review Board for the Protection of Human Subjects)**

(Please Type)

Project Director      Last Name Henderson      First Name Jean      M.I. F

Department/Organization Math/Computer Science

Phone -4762      Address Box 5051      E-Mail jfhenderson@una.edu

Title of Project  
The Effect of Classroom Structure on Student Achievement in and Attitudes Toward Computer Science: A Comparison of Three Classroom Settings

If Project Director is not a UNA faculty member, provide the name, department/organizational affiliation, phone number and mailing address of faculty/staff supervising the project.

Name \_\_\_\_\_

Dept./Org. \_\_\_\_\_

Phone \_\_\_\_\_      Address \_\_\_\_\_      E-Mail \_\_\_\_\_

Is the above research to be funded?       Yes       No  
If yes, by what agency? \_\_\_\_\_

Will this research be replicated using the methodology herein proposed?       Yes       No  
If yes, how many times will data be collected?      4  
Approximately how many years will be involved in the data collection process?      1.5

Based on the Federal and University guidelines for the use of human subjects in research, the proposed research should qualify for the following review (check one)

Exempt       Expedited       Full Review

(Request for exempt, expedited, or full review status is to be approved by the Human Subjects Committee of UNA prior to the initiation of data collection)

I certify that the above project will conform to Federal and University guidelines<sup>\*\*\*</sup> for the protection of human subjects.

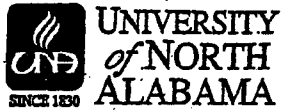
3/25/2006  
(Date)

Jean Henderson  
(Signature of Project Director)

<sup>\*\*\*</sup>See Federal Register Vol.56, No. 177, 28003-28032, 18 June 1991 (available at Office of Research, Bibb Graves 17) and UNA Policy on the Use of Human Subjects.

**SUBMISSION PROCEDURE:** Submit the original and 8 completed copies of this form and 9 copies of a project proposal/protocol to the Office of Research. Except for a full review, there is no deadline for submission; allow at least two weeks before the announced meeting of the Human Subjects Committee. Information on how to prepare a project proposal/protocol can be found in the University's Human Subjects Research Policy.

Revised 11/05/2003



Florence, Alabama 35632-0001

*Department of Sociology*  
[www2.una.edu/sociology](http://www2.una.edu/sociology)  
UNA Box 5010  
(256) 765-4200  
Fax (256) 765-4179

Project Director: Jean Henderson

Faculty Supervisor: not applicable

Title of Research Proposal:

Laboratories and Their Effects on Student Achievement In and Attitudes Toward Computer Science: A Comparison Between Three Different Laboratory Settings.

Date: August 12, 2005

IRB Action:

This proposal complies with University and Federal Regulations for the protection of human subjects (45 CFR46). Approval is effective for a period of 18 months from the date of this notification.

---

Craig T. Robertson, Ph.D.  
Chair, Human Subjects Committee

CTR/go

\*\*This form is designed to be filled out on the computer and not a typewriter.

**Application for Project Approval  
HUMAN SUBJECTS COMMITTEE (HSC)**

1. **Name of Investigator(s):** Jean Henderson
2. **Department Affiliation:** Computer Science
3. **Campus or home address:** 1127 Henson Dr. Florence, AL 35630
4. **Phone Number(s):** 256-760-1233
5. **Name of faculty member(s):** Jean Henderson
6. **Type of Investigator and Nature of Activity (check appropriate categories):**
  - Faculty or staff at Athens State University**
  - Project to be submitted for extramural funding**  
**Agency:** \_\_\_\_\_
  - Project to be submitted for intramural funding**  
**Source:** \_\_\_\_\_
  - Project Unfunded**
  - Other**
  - Student at Athens State University**
  - Undergraduate**  **OR**  **Special**
  - Class Project (number & title of class)** \_\_\_\_\_
  - Independent Study (name of faculty supervisor)** \_\_\_\_\_
  - Other (please explain)** \_\_\_\_\_

7. **Title of Investigation:** The Effect of Course Structure on Student Achievement in and Attitudes toward Computer Science: A Comparison of Two Laboratory Settings

**\*\*ALL STUDENT APPLICATIONS SUBMITTED TO THE HSC FOR REVIEW MUST BE SIGNED BY ALL INVESTIGATORS INCLUDING THE FACULTY MEMBER SUPERVISING THE RESEARCH ACTIVITY.**

8.  **Individuals other than faculty, staff, or students at Athens State University.**

**Please identify investigators and research group:** \_\_\_\_\_

9. **Certifications**

I am familiar with the research policies and procedures of Athens State University (3.11 in the Faculty Handbook) regarding human subjects in research.

I subscribe to the standards and will adhere to the policies and procedures.

I am familiar with the published guidelines for the ethical treatment of subjects associated with my particular field of study (e.g., as published by the American Psychological Association, American Sociological Asso., etc).

Signed: Jean Henderson  
First Investigator

Signed: Jean Henderson  
Faculty Supervisor

Date: 6/26/2006

Date: 6/26/2006

Approved: \_\_\_\_\_

**Chair, Human Subjects Committee**

Subject Human Subjects Application/Research Project  
From Tina Hicks <Tina.Hicks@athens.edu>  
Date Tuesday, July 11, 2006 10:36 am  
To jfhenderson@una.edu

Ms. Henderson,  
You have approval for your Human Subjects Project request.

Tina Hicks  
Executive Assistant to the Vice President of Academic Affairs  
Athens State University  
300 N. Beaty Street  
Athens, Alabama 35611  
256-233-8214  
Tina.Hicks@athens.edu

**APPENDIX E**

**RAW DATA**

Table E1

Raw Data: Observation 1:Participants 1-39

| Y1  | Y2  | X1 | X2 | M1 | M2 | M3 | L1 | L2 | L3 | S1   | S2  | S3    | X3 | ST1 | ST2 |
|-----|-----|----|----|----|----|----|----|----|----|------|-----|-------|----|-----|-----|
| 270 | 140 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 496 | 280 |
| 197 | 109 | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 0.5  | 0.5 | 0.25  | 0  | 363 | 216 |
| 183 | 124 | 1  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 346 | 220 |
| 258 | 122 | 1  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 0  | 447 | 245 |
| 157 | 102 | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 288 | 230 |
| 254 | 136 | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 1  | 0.5  | 0.5 | 0.25  | 0  | 427 | 261 |
| 200 | 133 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 328 | 264 |
| 275 | 127 | 1  | 1  | 0  | 1  | 0  | 0  | 0  | 1  | 0.5  | 0.5 | 0.25  | 0  | 550 | 237 |
| 261 | 89  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 500 | 193 |
| 210 | 110 | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 0.5  | 0.5 | 0.25  | 0  | 386 | 223 |
| 251 | 103 | 1  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 461 | 203 |
| 220 | 105 | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 413 | 214 |
| 125 | 104 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 218 | 206 |
| 275 | 131 | 1  | 1  | 0  | 1  | 0  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 0  | 512 | 265 |
| 188 | 105 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0.5  | 0.5 | 0.25  | 0  | 358 | 194 |
| 221 | 123 | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 0  | 417 | 245 |
| 256 | 131 | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 519 | 265 |
| 227 | 107 | 1  | 0  | 0  | 1  | 0  | 1  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 469 | 228 |
| 245 | 83  | 0  | 1  | 1  | 0  | 0  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 0  | 442 | 166 |
| 263 | 134 | 1  | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 492 | 268 |
| 271 | 135 | 1  | 1  | 0  | 1  | 0  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 0  | 520 | 259 |
| 275 | 116 | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0.5  | 0.5 | 0.25  | 0  | 533 | 233 |
| 268 | 104 | 1  | 0  | 0  | 1  | 0  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 0  | 543 | 213 |
| 264 | 135 | 1  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 0  | 525 | 266 |
| 195 | 124 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 352 | 250 |
| 174 | 100 | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 0  | 303 | 198 |
| 226 | 124 | 0  | 1  | 1  | 0  | 0  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 1  | 422 | 248 |
| 209 | 105 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | -0.5 | 0.5 | -0.25 | 1  | 388 | 217 |
| 181 | 113 | 1  | 1  | 1  | 0  | 0  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 1  | 360 | 228 |
| 222 | 125 | 1  | 1  | 0  | 0  | 1  | 1  | 0  | 0  | -0.5 | 0.5 | -0.25 | 1  | 404 | 255 |
| 229 | 121 | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | -0.5 | 0.5 | -0.25 | 1  | 396 | 236 |
| 198 | 124 | 1  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 1  | 375 | 262 |
| 228 | 139 | 0  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 1  | 412 | 279 |
| 246 | 123 | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | -0.5 | 0.5 | -0.25 | 1  | 420 | 253 |
| 200 | 107 | 1  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 1  | 380 | 214 |
| 224 | 103 | 1  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 1  | 403 | 206 |
| 268 | 124 | 0  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 1  | 532 | 251 |
| 151 | 116 | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | -0.5 | 0.5 | -0.25 | 1  | 321 | 215 |
| 210 | 96  | 1  | 0  | 0  | 1  | 0  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 1  | 391 | 194 |

Note: Y1 = Achievement, Y2 = Attitude, X1 = Gender, X2 = Previous Computer Programming Course, M1 = Math Background-1, M2 = Math Background-2, M3 = Math Background-3, L1 = Learning Style-1, L2 = Learning Style-2, L3 = Learning Style-3, S1 = School, S2 = Term, S3 = Interaction of School and Term, X3 = Treatment, ST1 = Subjects Total-1, ST2 = Subjects Total-2.



Table E2

Raw Data: Observation 1:Participants 40–72

| Y1  | Y2  | X1 | X2 | M1 | M2 | M3 | L1 | L2 | L3 | S1   | S2   | S3    | X3 | ST1 | ST2 |
|-----|-----|----|----|----|----|----|----|----|----|------|------|-------|----|-----|-----|
| 146 | 108 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | -0.5 | 0.5  | -0.25 | 1  | 306 | 216 |
| 215 | 127 | 0  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | -0.5 | 0.5  | -0.25 | 1  | 399 | 258 |
| 234 | 99  | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 0  | 381 | 183 |
| 219 | 132 | 1  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | -0.5 | -0.5 | 0.25  | 0  | 369 | 219 |
| 229 | 135 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 0  | 355 | 273 |
| 271 | 128 | 1  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 0  | 472 | 253 |
| 250 | 123 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 0  | 454 | 246 |
| 223 | 127 | 1  | 1  | 0  | 1  | 0  | 0  | 0  | 1  | -0.5 | -0.5 | 0.25  | 0  | 433 | 250 |
| 246 | 117 | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 0  | -0.5 | -0.5 | 0.25  | 0  | 453 | 238 |
| 228 | 94  | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 0  | 387 | 196 |
| 271 | 122 | 0  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | -0.5 | -0.5 | 0.25  | 0  | 490 | 244 |
| 261 | 94  | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 0  | 471 | 201 |
| 242 | 81  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 0  | 404 | 181 |
| 211 | 127 | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 353 | 253 |
| 271 | 134 | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 433 | 266 |
| 275 | 124 | 1  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | 0.5  | -0.5 | -0.25 | 1  | 546 | 249 |
| 264 | 140 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 483 | 280 |
| 208 | 84  | 1  | 1  | 1  | 0  | 0  | 1  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 289 | 136 |
| 218 | 100 | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 353 | 197 |
| 232 | 91  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 440 | 188 |
| 270 | 101 | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 0.5  | -0.5 | -0.25 | 1  | 488 | 195 |
| 262 | 101 | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 483 | 196 |
| 209 | 123 | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 406 | 252 |
| 216 | 98  | 0  | 1  | 1  | 0  | 0  | 1  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 406 | 198 |
| 218 | 108 | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 397 | 217 |
| 248 | 129 | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 1  | 0.5  | -0.5 | -0.25 | 1  | 436 | 249 |
| 249 | 140 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 444 | 280 |
| 252 | 112 | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0.5  | -0.5 | -0.25 | 1  | 407 | 229 |
| 230 | 104 | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0.5  | -0.5 | -0.25 | 1  | 309 | 176 |
| 227 | 116 | 1  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 342 | 231 |
| 150 | 122 | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 1  | 235 | 194 |
| 134 | 105 | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0.5  | -0.5 | -0.25 | 1  | 207 | 173 |

Note: Y1 = Achievement, Y2 = Attitude, X1 = Gender, X2 = Previous Computer Programming Course, M1 = Math Background-1, M2 = Math Background-2, M3 = Math Background-3, L1 = Learning Style-1, L2 = Learning Style-2, L3 = Learning Style-3, S1 = School, S2 = Term, S3 = Interaction of School and Term, X3 = Treatment, ST1 = Subjects Total-1, ST2 = Subjects Total-2.

Table E3

Raw Data: Observation 2:Participants 1-39

| Y1  | Y2  | X1 | X2 | M1 | M2 | M3 | L1 | L2 | L3 | S1   | S2  | S3    | X3 | ST1 | ST2 |
|-----|-----|----|----|----|----|----|----|----|----|------|-----|-------|----|-----|-----|
| 226 | 140 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 496 | 280 |
| 166 | 107 | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 0.5  | 0.5 | 0.25  | 1  | 363 | 216 |
| 163 | 96  | 1  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 346 | 220 |
| 189 | 123 | 1  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 1  | 447 | 245 |
| 131 | 128 | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 288 | 230 |
| 173 | 125 | 1  | 1  | 1  | 0  | 0  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 1  | 427 | 261 |
| 128 | 131 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 328 | 264 |
| 275 | 110 | 1  | 1  | 0  | 1  | 0  | 0  | 0  | 1  | 0.5  | 0.5 | 0.25  | 1  | 550 | 237 |
| 239 | 104 | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 500 | 193 |
| 176 | 113 | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 0.5  | 0.5 | 0.25  | 1  | 386 | 223 |
| 210 | 100 | 1  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 461 | 203 |
| 193 | 109 | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 413 | 214 |
| 93  | 102 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 218 | 206 |
| 237 | 134 | 1  | 1  | 0  | 1  | 0  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 1  | 512 | 265 |
| 170 | 89  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0.5  | 0.5 | 0.25  | 1  | 358 | 194 |
| 196 | 122 | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 1  | 417 | 245 |
| 263 | 134 | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 519 | 265 |
| 242 | 121 | 1  | 0  | 0  | 1  | 0  | 1  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 469 | 228 |
| 197 | 83  | 0  | 1  | 1  | 0  | 0  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 1  | 442 | 166 |
| 229 | 134 | 1  | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 492 | 268 |
| 249 | 124 | 1  | 1  | 0  | 1  | 0  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 1  | 520 | 259 |
| 258 | 117 | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0.5  | 0.5 | 0.25  | 1  | 533 | 233 |
| 275 | 109 | 1  | 0  | 0  | 1  | 0  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 1  | 543 | 213 |
| 261 | 131 | 1  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 0.5  | 0.5 | 0.25  | 1  | 525 | 266 |
| 157 | 126 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 352 | 250 |
| 129 | 98  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | 0.5 | 0.25  | 1  | 303 | 198 |
| 196 | 124 | 0  | 1  | 1  | 0  | 0  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 0  | 422 | 248 |
| 179 | 112 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | -0.5 | 0.5 | -0.25 | 0  | 388 | 217 |
| 179 | 115 | 1  | 1  | 1  | 0  | 0  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 0  | 360 | 228 |
| 182 | 130 | 1  | 1  | 0  | 0  | 1  | 1  | 0  | 0  | -0.5 | 0.5 | -0.25 | 0  | 404 | 255 |
| 167 | 115 | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | -0.5 | 0.5 | -0.25 | 0  | 396 | 236 |
| 177 | 138 | 1  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 0  | 375 | 262 |
| 184 | 140 | 0  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 0  | 412 | 279 |
| 174 | 130 | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | -0.5 | 0.5 | -0.25 | 0  | 420 | 253 |
| 180 | 107 | 1  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 0  | 380 | 214 |
| 179 | 103 | 1  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 0  | 403 | 206 |
| 264 | 127 | 0  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 0  | 532 | 251 |
| 170 | 99  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | -0.5 | 0.5 | -0.25 | 0  | 321 | 215 |
| 181 | 98  | 1  | 0  | 0  | 1  | 0  | 0  | 1  | 0  | -0.5 | 0.5 | -0.25 | 0  | 391 | 194 |

Note: Y1 = Achievement, Y2 = Attitude, X1 = Gender, X2 = Previous Computer Programming Course, M1 = Math Background-1, M2 = Math Background-2, M3 = Math Background-3, L1 = Learning Style-1, L2 = Learning Style-2, L3 = Learning Style-3, S1 = School, S2 = Term, S3 = Interaction of School and Term, X3 = Treatment, ST1 = Subjects Total-1, ST2 = Subjects Total-2.

Table E4

Raw Data: Observation 2:Participants 40-71

| Y1  | Y2  | X1 | X2 | M1 | M2 | M3 | L1 | L2 | L3 | S1   | S2   | S3    | X3 | ST1 | ST2 |
|-----|-----|----|----|----|----|----|----|----|----|------|------|-------|----|-----|-----|
| 160 | 108 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | -0.5 | 0.5  | -0.25 | 0  | 306 | 216 |
| 184 | 131 | 0  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | -0.5 | 0.5  | -0.25 | 0  | 399 | 258 |
| 147 | 84  | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 1  | 381 | 183 |
| 150 | 87  | 1  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | -0.5 | -0.5 | 0.25  | 1  | 369 | 219 |
| 126 | 138 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 1  | 355 | 273 |
| 201 | 125 | 1  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 1  | 472 | 253 |
| 204 | 123 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 1  | 454 | 246 |
| 210 | 123 | 1  | 1  | 0  | 1  | 0  | 0  | 0  | 1  | -0.5 | -0.5 | 0.25  | 1  | 433 | 250 |
| 207 | 121 | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 0  | -0.5 | -0.5 | 0.25  | 1  | 453 | 238 |
| 159 | 102 | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 1  | 387 | 196 |
| 219 | 122 | 0  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | -0.5 | -0.5 | 0.25  | 1  | 490 | 244 |
| 210 | 107 | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 1  | 471 | 201 |
| 162 | 100 | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | -0.5 | -0.5 | 0.25  | 1  | 404 | 181 |
| 142 | 126 | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 353 | 253 |
| 162 | 132 | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 433 | 266 |
| 271 | 125 | 1  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | 0.5  | -0.5 | -0.25 | 0  | 546 | 249 |
| 219 | 140 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 483 | 280 |
| 81  | 52  | 1  | 1  | 1  | 0  | 0  | 1  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 289 | 136 |
| 135 | 97  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 353 | 197 |
| 208 | 97  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 440 | 188 |
| 218 | 94  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 0.5  | -0.5 | -0.25 | 0  | 488 | 195 |
| 221 | 95  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 483 | 196 |
| 197 | 129 | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 406 | 252 |
| 190 | 100 | 0  | 1  | 1  | 0  | 0  | 1  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 406 | 198 |
| 179 | 109 | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 397 | 217 |
| 188 | 120 | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 1  | 0.5  | -0.5 | -0.25 | 0  | 436 | 249 |
| 195 | 140 | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 444 | 280 |
| 155 | 117 | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0.5  | -0.5 | -0.25 | 0  | 407 | 229 |
| 79  | 72  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0.5  | -0.5 | -0.25 | 0  | 309 | 176 |
| 115 | 115 | 1  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 342 | 231 |
| 85  | 72  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0.5  | -0.5 | -0.25 | 0  | 235 | 194 |
| 73  | 68  | 1  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0.5  | -0.5 | -0.25 | 0  | 207 | 173 |

Note: Y1 = Achievement, Y2 = Attitude, X1 = Gender, X2 = Previous Computer Programming Course, M1 = Math Background-1, M2 = Math Background-2, M3 = Math Background-3, L1 = Learning Style-1, L2 = Learning Style-2, L3 = Learning Style-3, S1 = School, S2 = Term, S3 = Interaction of School and Term, X3 = Treatment, ST1 = Subjects Total-1, ST2 = Subjects Total-2.