Mercer County Community College

Division of Business and Technology

IST 107

FUNDAMENTALS PROGRAMMING SKILLS USING C++

COURSE DESCRIPTION

Learn to program the computer to solve problems and to gain an appreciation for the role that computers and programs play in society today. Topics include programming environment, variables, methods, decisions, repetition, exceptions, arrays, and object-oriented programming. Laboratory exercises are aimed at helping the student to understand the lecture and reading materials and then at using the material to solve new problems.

Text(s): Reference Division Booklist

Prerequisites: ENG035 and MAT030

Co-requisites: None

Credits: 3 Lecture Hours: 1 Lab Hours: 4

Food and Drink are strictly prohibited in classrooms as per Health and Safety Laws. Students may not bring in chemicals or cleaning fluids of any kind without the appropriate MSD sheets

Course Coordinator: Winston H. Maddox Latest Review: Fall 2003
I. **OVERALL COURSE OBJECTIVES**

Upon completion of this course, students will be able to:

- Use C# as a tool to solve problems on the computer.
- Demonstrate the ability to construct procedural and object-oriented programming in C# using the correct syntax.
- Demonstrate through example programs, classroom discussions, and brief writing assignments, several of the factors that have led to the pervasive use of computers in our society.
- Students will demonstrate critical thinking skills by choosing the most appropriate constructs to complete assignments in C#.

II. **SPECIFIC MODULE OBJECTIVES**

A. Microsoft Visual Studio Environment and Introduction to Programming (Students will):

1. Recognize *that war and the fear of war generated lots of useful tools including computers*.
2. Observe and describe the compilation process and the *ability of computers to store both programs and data (von Neumann)*.
3. Students will demonstrate the ability to use the programming environment by studying a simple console application and then creating a C# console application to perform a similar but different simple process.
4. Students will demonstrate the ability to use the programming environment by studying a simple forms application and then creating a C# windows forms application to perform a similar but different simple process.

B. Variable Types and Arithmetic (Students will):

1. Students will demonstrate an understanding of the difference between variable names and keywords by preparing a memo that compares them.
2. Students will demonstrate the ability to use arithmetic and assignment operators by writing a simple program to perform simple calculations.
3. Students will demonstrate the ability to control the order of operations with parentheses by entering an improperly coded arithmetic expression, reviewing the resulting output and then correcting the program.
4. Compare and contrast the *accuracy of computers and humans*.

C. Methods (Students will):

1. Students will demonstrate that they can program a Method by moving code from Main() to an appropriately named Method().
2. Students will demonstrate that they can pass data to a Method() by studying a Method() that sums five numbers and then creating a Method() that receives ten numbers and averages them.
3. Student will demonstrate that they can return data from a Method() by modifying the Method() that averages numbers to return the standard deviation.
4. Students will demonstrate their understanding of the role that the speed of computers versus humans has had on the impact that computers have had in society by preparing a memo to discuss this factor.
D. **Branching or Decisions (Students will):**

1. Demonstrate the ability to code a simple if expression.
2. Students will Compare and contrast computer (or philosophical) logic with the use of AND and OR in the English language in a written memo.
3. After reviewing the development of code to determine a student’s letter grade based on their course average, students will demonstrate an understanding of if-else-if statements by creating methods that compute the U.S. Federal Income Tax for Single and Married taxpayers.
4. Students will demonstrate an understanding of why the “if-else” construct is more efficient than two separate if statements in a written memo.
5. Students will demonstrate an understanding of why the “if-else-if” construct is more efficient than a bunch of if statements in a written memo.
6. After reviewing the development of code to use the switch statement to select pass-fail or letter grading, students will demonstrate an understanding of switch statements by creating programs that compute the U.S. Federal Income Taxes for two individuals or for a married couple, based on their filing status.
7. Students will explain the differences between the way computers and people make decisions and discuss the advantages and disadvantages of both approaches in a written memo.

E. **Loops (Emphasize that a major value of computers is the ability to do simple things over and over, without error and without complaint.) Students will:**

1. After reviewing programs to sum the integers up to a specified number using for, while, and do loops, student will create programs to compute the factorial of the number, using all three looping constructs.
2. Students will demonstrate the ability to design, code, and test a simple console and a simple Windows Forms application by coding a console application that prompts the user for a number (N) and then uses a for-loop to generate the N numbers of the Fibonacci sequence and by coding a Windows Forms application that uses a simple textbox to accept N and a while-loop to generate the first N numbers in the Fibonacci sequence.
3. Students will explain the purpose of each of the three looping constructs in a written memo; since the rational for the various loops is poorly covered in the chosen text, students will be expected to do research in other texts or on the internet.
4. Students will **Describe why looping is important to the success of computers** in a written memo.

F. **Handling Exceptions (try, catch, and throw; Students will):**

1. Given programs that detect and handle a bad month, student will add code to detect and handle a bad day or a bad season.

---

1 This module has more than its share of written assignments. The instructor is encouraged to stagger the due dates.
G. Creating and using Classes and Objects (Students will):

1. After studying a “Circle” class with a constructor and a public Area(), students will demonstrate an understanding of methods by adding a public method called Circumference().
2. After studying and extending the “Circle” class, with a constructor and several public methods, student will demonstrate a basic understanding of Classes and Objects by programming and testing a new class “Sphere” with a constructor to load the radius, and methods to determine the great circle circumference, the great circle area, the surface area, and the volume.
3. After studying several examples of classes with constructors, student demonstrate the ability to develop and use Classes and Objects by programming a new class “RectangularPrism” with a constructor to load height, width, and depth, and methods for SurfaceArea() and Volume(). These methods should be used in a Window-Forms application to compute these values.

H. Values and References (Students will):

1. Students will be able to describe the differences between variables and classes, and values and references in a written memo.
2. Students will be able to describe “boxing” and “unboxing” in writing.

I. Enumeration and Structures (Students will):

1. After studying an example of the enumeration of Seasons and the use of an enumeration in Classes, students will develop their skills using step-by-step directions to enumerate Months.
2. After studying the Seasons and Months enumeration examples, students demonstrate an understanding of Enumeration by creating code that enumerates The-Days-Of-The-Week.
3. After studying an example of using structures to store and use time (hours, minutes, and seconds) variables, students will develop their skills using step by step directions to incorporate an American-Date method and date structure in a program.
4. Students will demonstrate an understanding of structures by creating an extension to the date structure to add and use a European-Date method.
5. Students will demonstrate the ability to create and apply structures by creating a new program that uses structures to store and retrieve names, addresses, and telephone numbers.

J. Arrays and Collections (Students will):

1. After studying a program that uses an array to store data and to compute the average of the data, student will add and use a method to compute the standard deviation.
2. Students will demonstrate the ability to apply the material that they have learned so far by completing a major programming project. Specifically, student will develop a complete console application to store student-names, individual test, homework, laboratory, project, and participation grades, with the weights for each of these grade categories. The program will compute the average in each category, the course average, and a recommended letter grade. Data should be stored in an object and use methods that are defined for the object.
3. Student will be able to describe the differences and similarities between collections and arrays, and queues and stacks in written memo.
4. After working through the card shuffling and dealing example, student will demonstrate the ability to use arrays and collections by creating and using methods to sort each hand and compute the total points based on “Goren’s point count” system.
K. Overloading and Parameter Arrays (Students will):

1. Students will create a Class with overloaded Methods to compute the square root of integers, longs, floats, doubles, and decimals.
2. Students will use command line parameters to create a new version of Hello World that also says hello to every name that is listed on the command line.

III. FINAL PROJECT

The final project will consist of developing two classes, Quadratic and Cubic and the code to load the coefficients of the cubic equation into the Cubic class. The Cubic class will have a method to extract a real root and to extract the coefficients of the reduced Quadratic equation. The extracted coefficients will be used to initialize the Quadratic class. The Quadratic class will have a method to identify the type of roots, real or imaginary, and to either extract the two real roots or the real and imaginary parts of complex roots. More details regarding this assignment, to exploit as many features of C# as possible, and to help students understand the arithmetic involved will be developed and passed out during the 10th week of the course.

Completion of this project will:

1. Demonstrate the student’s ability to solve problems independently using C#.
2. Help the student recognize that iteration permits computers to solve problems that cannot be completed by humans. Students will be expected to explain how iteration is used to extract a real root from the cubic equation.
3. Provide an opportunity for the student to present his project to his classmates using a code-walk through.

Students will be required to enter, compile, and test programs outside of class. There are suitably configured computers in the library. The primary text for this course includes a C# programming environment and a C# compiler.

Students will be expected to prepare for each class by reading assignments from the textbooks, by completing short-answer written exercise to demonstrate a factual understanding of the material, and by entering the code for the laboratory exercise in an appropriate text file.

The course will consist of one hour of lecture and four hours of laboratory each week. Whenever possible, these contact hours should be spread over at least two class periods each week; however, evening sessions will be offered. The hour of lecture will review the reading material and help students with any questions that they have. The lecture will be followed by about two hours of structured laboratory, where laboratory exercises will be completed step-by-step in unison. The student will document the results of his studies by adding detailed comments to the program. The remainder of the laboratory, about two hours, and the programming homework assignment will require the student to write a program that uses the same constructs to solve a similar but different problem.

During the course, each student will be expected to complete two significant programming projects. After the fifth module, about week 6 of the course, each student will be required to develop a console and a windows application to compute the first N number in the Fibonacci sequence. After the tenth module, about week 12 of the course, each student will be required to use object-oriented programming techniques to completely factor a cubic equation. Students are encouraged to propose programming projects that would be of more interest than the two that have been mentioned. Instructors are encouraged to accept alternate proposals if the proposed work is equivalent.


**Both Texts Are Required**

Ten memos (two pages or less each) are required during this course to demonstrate the students understanding of various concepts. Each will be graded and the lowest, non-zero, grade will be dropped.

When students are required to describe something in writing, they will be required to prepare a memo to the instructor explaining or answering the given question. Memos are expected to be typed and to follow standard business form and are not expected to exceed two pages in length. Students name must appear on both pages of a two-page memo. Multiple page memos should be stapled. Memos will be graded for content (65%), format or style (20%), and timely submission (15%).

Students will be expected to complete each of the step-by-step exercises in the first 11 chapters of Sharp and Jagger, or equivalent step-by-step assignments in an alternate text, in the laboratory portion of the course. Each assignment is to be thoroughly commented to demonstrate the student’s understanding of the code and turned in to the instructor. Each set of laboratories will be graded and the lowest, non-zero, grade will be dropped.

Students are expected to demonstrate an understanding of each of the 11 modules by either creating a program or by significantly modifying code that was provided in the Sharp & Jagger text. This work will be completed in laboratory and at-home. As in the case of laboratories, each assignment is to be thoroughly commented to demonstrate the student’s understanding of the code and turned in to the instructor. Each assignment will be graded and the lowest, non-zero, grade will be dropped.

There are two significant programming projects. All programming assignments, laboratories and homework will be graded based on the following:

- Submitted On-time: 20%
- Performs the Required Functions: 30%
- Uses All Previously Covered Material, Including the Current Topic: 10%
- Uses a Consistent and Correct Indentation Style: 10%
- Uses Sound Program Structuring Methods: 10%
- Contains Comments that Adequately Describe the Code: 10%
- Uses Constructs that are Typically Used by C# Programmers: 10%

The course grade will be based on the following weights:

- Final Examination: 25%
- Two major programming projects: 25%
- Laboratory Assignments: 20%
- Programming / Homework Assignments: 15%
- Writing Assignments: 10%
- Attendance and Participation: 5%
Because of the amount of homework that the instructor needs to grade, because students must stay caught up to master a subject like C# programming and because a reasonable number of low grades are being dropped, LATE HOMEWORK, LABORATORY EXERCISES, and PROGRAMMING ASSIGNMENTS WILL NOT BE ACCEPTED. Students are reminded of the school’s academic integrity policy. In this class, if a student submits work that has been copied from an outside source, without proper citations, or if two or more students submit work that is essentially identical, the grade for the assignment will be ZERO. Grades of ZERO will not be dropped. Students who receive two or three ZEROS, depending on the assignment, will not be able to pass the course.

The course percentage will be determined by multiplying the student’s average in each category by the specified weight and totaling these products. The letter grade will then be determined as follows:

<table>
<thead>
<tr>
<th>Percentage Range</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>90 ≤ Percentage ≤ 100</td>
<td>A</td>
</tr>
<tr>
<td>80 ≤ Percentage &lt; 90</td>
<td>B</td>
</tr>
<tr>
<td>70 ≤ Percentage &lt; 80</td>
<td>C</td>
</tr>
<tr>
<td>60 ≤ Percentage &lt; 70</td>
<td>D</td>
</tr>
<tr>
<td>0 ≤ Percentage &lt; 60</td>
<td>F</td>
</tr>
</tbody>
</table>

Students are reminded that they must be aware of the deadline for the drop-add period. The professors who teach this course are not obligated to withdraw students who stop coming to class. WITHDRAWING FROM ANY COURSE IS THE STUDENTS RESPONSIBILITY.