AXIA COLLEGE℠ of University of Phoenix℠ ONLINE

COURSE Syllabus

IT 210

Fundamentals of Programming with Algorithms and Logic

Version 3 09/01/08

Program Council

The Academic Program Councils for each college oversee the design and development of all University of Phoenix curricula. Council members include full-time and practitioner faculty members who have extensive experience in this discipline. Teams of full-time and practitioner faculty content experts are assembled under the direction of these Councils to create specific courses within the academic program.

Copyright

Course Syllabus

***Course Title*:** IT 210—Fundamentals of Programming with Algorithms and Logic

Required Texts

Venit, S. (2004). *Extended prelude to programming: Concepts and design* (2nd ed.). Boston: Scott/Jones.

Axia College's *Writing Style Handbook*, available online at https://axiaecampus.phoenix.edu/Writing_Style_Handbook_AxiaUOP.pdf

Electronic Resources

Please Note: All required text and materials are found on the **Materials** tab of the student web page. The student web page can be accessed through the Axia College of University of Phoenix Student and Faculty Web site at https://axiaecampus.phoenix.edu/

Course Overview

COURSE DESCRIPTION

This course provides students with a basic understanding of programming practices. Concepts covered include flowcharting, pseudocode methodologies, and an understanding of programming practices. Students will learn how these concepts, when properly applied, improve program design.

TOPICS AND OBJECTIVES

Introduction to Software Development

- Describe the importance of using a structured, modular approach when creating program requirements, design, and code.
- Identify how a computer processes and stores data.

Software Application Development

- Identify the purposes and definitions of software development concepts.
- Use pseudocode to design a program to solve a computational problem.
- Identify the application-level requirements of a conversion project.

Structured Programming, Part I

- Demonstrate the sequential processing control structure.
- Demonstrate the selection processing control structure.

Structured Programming, Part II

- Demonstrate the iteration control structure.
- Design complex program algorithms using the three basic control structures.

Complex Data Structures

- Explain the need for composite complex data structures.
- Generate the program design and pseudocode for a simple array.

Verification and Validation

- Recognize how requirements and desk review design are used to verify algorithms.
- Create test data to validate that algorithms handle user input data correctly.

File and Database Processing

- Determine when a sequential file is more useful than a database.
- Differentiate between a flat file and a relational database.
- Design a suitable program to solve given programming problems using the top-down modular approach and pseudocode.

Object-Oriented Design and Programming

- Identify both the top-level objects and the GUI interfaces of an electronic product.
- Describe object-oriented, event-driven programming.

- Describe a simple, object-oriented program.
- Recognize the difference between object-oriented and structured program design.

Requirement Specifications, Design Specifications, and Processing Models

- Create requirement specifications, design specifications, and processing models—including input/out tables, data flow models, and procedural processing.

Point Values for Course Assignments

Week One: Introduction to Software Development	
Discussion Questions	10
Participation	10
CheckPoint: Input Data and Output Process	15
Week Two: Software Application Development	
CheckPoint: Software Development Activities/Purposes	10
CheckPoint: Chapter 2 Programming Problem	30
Assignment: Application-Level Requirements	90
Week Three: Structured Programming, Part I	
Discussion Questions	10
Participation	10
CheckPoint: Sequential and Selection Processing Control Structure	30
Week Four: Structured Programming, Part II	
CheckPoint: Iteration Control Structure	30
Assignment: Currency Conversion Design	90
Week Five: Complex Data Structures	
Discussion Questions	10
Participation	10
CheckPoint: Simple Array Process	30
Week Six: Verification and Validation	
CheckPoint: Algorithm Verification	30
Assignment: Currency Conversion Test Procedure	90
Week Seven: File and Database Processing	
Discussion Questions	10
Participation	10
Exercise: Peer Reviews of Currency Conversion Test Procedure	20
CheckPoint: Chapter 5 Programming Problems	30
Week Eight: Object-Oriented Design and Programming	
CheckPoint: Interfaces and Communications Messages	30
CheckPoint: Object-Oriented Data and Processes	30
Assignment: Object-Oriented Design	90
Week Nine: Requirement Specifications, Design Specifications, and Processing Models	
Capstone Discussion Question	15
Participation	10
Final Project: Currency Conversion	250
Point Totals	**1,000**

Policies and Procedures

All students in this course are required to abide by the policies and procedures described in the **Policies** section of the Student Web Page. To view these policies, students must be logged in to the Student Web Page. The same information can be accessed via the **Policies** link located in the **Materials** section of the **Classroom** tab on the Student Web Page.

Week One

Introduction to Software Development

- Describe the importance of using a structured, modular approach when creating program requirements, design, and code.
- Identify how a computer processes and stores data.

ASSIGNMENTS

1. **Read** objectives and welcome.
    - **Read** instructor's bio, and post your own bio.
    - **Due Date:** Day 1 [post to the **Chat Room** forum]

2. **Read** Appendix A regarding the final project requirements.

3. **Read** Ch. 2, 3, & 7 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.).

4. **Discussion Question 1**
    - **Due Date:** Day 2 [post to the **Main** forum]
    - **Post** your response to the following: Why do you think the requirements analysis process is so difficult? Describe two things you can do to overcome these difficulties.

5. **Discussion Question 2**
    - **Due Date:** Day 4 [post to the **Main** forum]
    - **Post** your response to the following: When building a house, a structured, modular approach is better than a haphazard approach. Explain how a structured approach relates to developing programs and why using an organized approach is important.

6. **CheckPoint:** Input Data and Output Process
    - *Resources*: Appendix B and Appendix C
    - **Due Date:** Day 5 [post to the **Individual** forum]
    - **Review** the example in Appendix B.

- **Read** the following scenario:

  You want to build a program that will keep track of your CD and DVD collection at home.

- **Use** the table in Appendix C to complete this CheckPoint:

  - **Identify** at least three **processes** (capabilities) that are needed in order to keep track of your collection.
  - **Identify** the input data required for each of the processes.
  - **Identify** a logical name for each data output item and type of data output (real number, integer, text).

- **Post** the table as an attachment.

Week Two

Software Application Development

- Identify the purposes and definitions of software development concepts.
- Use pseudocode to design a program to solve a computational problem.
- Identify the application-level requirements of a conversion project.

ASSIGNMENTS

1. **CheckPoint:** Software Development Activities and Purposes

   - *Resource*: Appendix D
   - **Due Date:** Day 3 [**Individual**] forum
   - **Match** the software development activity or concept with the description or purpose of the activity using the table in Appendix D.
   - **Post** the table as an attachment.

2. **CheckPoint:** Chapter 2 Programming Problem

   - *Resources*: Appendix E; and pp. 33, 36, and 59 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.)
   - **Due Date:** Day 5 [**Individual**] forum
   - **Review** the example in Appendix E as well as the additional examples on pp. 33 and 36.
   - **Complete** Ch. 2, Programming Problem 2, on p. 59.

3. **Assignment:** Application-Level Requirements

   - *Resources*: Appendix B and Appendix F
   - **Due Date:** Day 7 [**Individual**] forum
   - **Review** the example in Appendix B.
   - **Complete** the following assignment using Appendix F:

     - **List** the application-level requirements for the Currency Conversion project.
     - **Use** a structured programming approach to generate an Input-Process-Output chart for the application.
     - **Generate** the hierarchy chart for the application.

   - **Post** the table as an attachment.

Week Three

Structured Programming, Part I

- Demonstrate the sequential processing control structure.
- Demonstrate the selection processing control structure.

ASSIGNMENTS

1. **Read** Ch. 4 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.).

2. **Discussion Question 1**

   - **Due Date:** Day 2 [**Main**] forum
   - **Post** your response to the following: Review the definition of *control structure* on p. 45 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.). Then, think about the pseudocode algorithm you would write for a simple task (making a peanut butter sandwich, for example) as well as three simple control structures that could be used to create this algorithm. What do you think is the most difficult part of creating the algorithm? What can you do to make this process easier?

3. **Discussion Question 2**

   - **Due Date:** Day 4 [**Main**] forum
   - **Post** your response to the following: How do you use the three **basic control structures**—sequential, repetition, and selection—in your everyday problem solving? Do you think there are any other control structures that would make your problem-solving skills more efficient? If so, describe them.

4. **CheckPoint:** Sequential and Selection Processing Control Structure

   - *Resource*: Appendix G
   - **Due Date:** Day 5 [**Individual**] forum
   - **Read** the following scenario:

     You are an accountant setting up a payroll system for a small firm. Each line of the table in Appendix G indicates an employee's salary range and corresponding base tax amount and tax percentage. Given a salary amount, the tax is calculated by adding the base tax for that salary range and the product of percentage of excess and the amount of salary over the minimum salary for that range.

   - **Design** a program that solves this problem.
   - **Generate** a set of input test values.

- **Perform** a design walkthrough to verify your design.

Week Four

Structured Programming, Part II

- Demonstrate the iteration control structure.
- Design complex program algorithms using the three basic control structures.

ASSIGNMENTS

1. **CheckPoint:** Iteration Control Structure

    - *Resource*: Ch. 4 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.)
    - **Due Date:** Day 5 [**Individual**] forum
    - **Design** a program that models the worm's behavior in the following scenario:

      A worm is moving toward an apple. Each time it moves, the worm cuts the distance between itself and the apple by its own body length until the worm is close enough to enter the apple. The worm can enter the apple when it is within a body length of the apple.

2. **Assignment:** Currency Conversion Design

    - *Resources*: Appendix H, Appendix I, and pp. 117 and 121-122 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.)
    - **Due Date:** Day 7 [**Individual**] forum
    - **Complete** the hierarchy chart in Appendix H and the flowcharts in Appendix I, based on the Currency Conversion requirements and Input-Process-Output table you generated in Week Two.
    - **Develop** the pseudocode for the program design.
    - **Post** the assignment as an attachment.

Week Five

Complex Data Structures

- Explain the need for complex data structures.
- Generate the program design and pseudocode for a simple array.

ASSIGNMENTS

1. **Read** Ch. 5 & 6 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.).

2. **Discussion Question 1**

    - **Due Date:** Day 2 [**Main**] forum
    - **Post** your response to the following: Identify at least two data structures that are used to organize a typical file cabinet. Why do you feel it is necessary to emulate these types of data structures in a computer program? For what kind of work project would you want to use this type of program?

3. **Discussion Question 2**

    - **Due Date:** Day 4 [**Main**] forum
    - **Post** your response to the following: Describe a programming project or situation in the workplace that would lend itself to array usage.

4. **CheckPoint:** Simple Array Process

    - *Resources*: Pp. 172-174 and 198 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.)
    - **Due Date:** Day 5 [**Individual**] forum
    - **Complete** Ch. 6, exercise 3, on p. 198. You are required to generate only the pseudocode, as described in the Week Two CheckPoint. No charting is required, but you may have to incorporate the bubble sort algorithm on pp. 172-174 to determine the number of salaries above and below the mean.

Week Six

Verification and Validation

- Recognize how requirements and desk review design are used to verify algorithms.
- Create test data to validate that algorithms handle user input data correctly.

ASSIGNMENTS

1. **CheckPoint:** Algorithm Verification

    - *Resource*: Appendix J
    - **Due Date:** Day 5 [**Individual**] forum
    - **Review** Appendix J.
    - **Answer** the following questions about the information in Appendix J:

        - What will be printed if the input is 0?
        - What will be printed if the input is 100?
        - What will be printed if the input is 51?
        - What will be printed if the user enters “Wingding”?
        - Is this design robust? If so, explain why. If not, explain what you can do to make it robust.
        - How many levels of nesting are there in this design?
        - Give a set of values that will test the normal operation of this program segment. Defend your choices.
        - Give a set of test values that will cause each of the branches to be executed.
        - Give a set of test values that test the abnormal operation of this program segment.

2. **Assignment:** Currency Conversion Test Procedure

    - **Due Date:** Day 7 [**Individual**] forum
    - **Generate** a set of test inputs and expected results for the Currency Conversion program.
    - **Post** the test procedure as an attachment.

Week Seven

File and Database Processing

- Determine when a sequential file is more useful than a database.
- Differentiate between a flat file and a relational database.
- Design a suitable program to solve given programming problems using the top-down modular approach and pseudocode.

ASSIGNMENTS

1. **Read** Ch. 8 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.).

2. **Discussion Question 1**

   - **Due Date:** Day 2 [**Main**] forum
   - **Post** your response to the following: Under what circumstances would you use a sequential file over a database? Describe these circumstances. When would a database be more beneficial than a sequential file? Is it possible for the two types of permanent storage to be used interchangeably? Explain your answers.

3. **Discussion Question 2**

   - **Due Date:** Day 4 [**Main**] forum
   - **Post** your response to the following: What are some of the key differences between a flat file and relational database? Which of the two storage methods do you think is most useful in a real-world application?

4. **Exercise:** Peer Reviews of Currency Conversion Test Procedure

   - *Resource*: Appendix K
   - **Due Date:** Day 3 [**Individual**] forum
   - **Perform** peer reviews of two classmates' Currency Conversion Test Procedures, which your instructor will place in your **Individual** forum on Day 1.
   - **Complete** the Appendix K form for each of the peer reviews.
   - **Post** the completed Appendix K forms in your **Individual** forum as an attachment.

5. **CheckPoint:** Chapter 5 Programming Problems

   - *Resource*: P. 158 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.)
   - **Due Date:** Day 5 [**Individual**] forum

- **Complete** Programming Problems 1 and 2.
- **Provide** the analysis and pseudocode only (no diagrams are required).

Week Eight

Object-Oriented Design and Programming

- Identify both the top-level objects and the GUI interfaces of an electronics product.
- Describe object-oriented, event-driven programming.
- Describe a simple object-oriented program.
- Recognize the difference between object-oriented and structured program design.

ASSIGNMENTS

1. **CheckPoint:** Interfaces and Communication Messages

   Understanding object-oriented methodologies is often difficult. You already understand that object-oriented analysis and design emulates the way human beings tend to think and conceptualize problems in the everyday world. With a little practice, object-oriented programming will become second nature to you.

   As an example, consider a typical house in which there are several bedrooms, a kitchen, and a laundry room—each with a distinct function. You sleep in the bedroom, you wash clothes in the laundry room, and you cook in the kitchen. Each room encapsulates all the items needed to complete the necessary tasks.

   You do not have an oven in the laundry room or a washing machine in the kitchen. However, when you do the laundry, you do not just add clothes to the washer and wait in the laundry room; once the machine has started, you may go into the kitchen and start cooking dinner. But how do you know when to go back to check the laundry? When the washer buzzer sounds, a message is sent to alert you to go back into the laundry room to put in a new load. While you are folding clothes in the laundry room, the oven timer may ring to inform you that the meat loaf is done.

   What you have is a set of well-defined components: Each provides a single service to communicate with the other components using simple messages when something needs to be done. If you consider a kitchen, you see it is also composed of several, smaller components, including the oven, refrigerator, and microwave. **Top-level objects** are composed of smaller components that do the actual work. This perspective is a very natural way of looking at our world, and one with which we are all familiar. We do the same thing in **object-oriented programming**:

   - Identify components that perform a distinct service
   - Encapsulate all the items in the component necessary to get the job done
   - Identify the messages that need to be provided to the other components

   Although the details can be quite complex, these details are the basic principles of object-oriented programming.

- *Resource*: Ch. 8 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.)
- **Due Date:** Day 3 [**Individual**] forum
- **Consider** the microwave oven in your kitchen, using the object-oriented thinking described above.
- **Create** a table with the following four column headings: Top-Level Objects, Communicates With, Incoming Messages, and Outgoing Messages.
    - **Identity** the top-level objects of the microwave.
    - **Explain** some of the graphical user interfaces (GUIs) and communications messages that occur during the operation of a microwave.
- **Describe** some of the advantages of having a *componentized* system. For example, what happens if the microwave breaks?
- **Post** your completed CheckPoint as an attachment.

2. **CheckPoint:** Object-Oriented Data and Processes

- *Resource*: Ch. 8 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.)
- **Due Date:** Day 5 [**Individual**] forum
- **Identify** a task you perform regularly, such as cooking, mowing the lawn, or driving a car.
- **Write** a short, structured design (pseudocode only) that accomplishes this task.
- **Think** about this task in an object-oriented way, and identify the objects involved in the task.
- **Identify** how you can encapsulate the data and processes you identified into an object-oriented design.
- **Describe** the architectural differences between the object-oriented and structured designs. Which of the designs makes more sense to you? Why?

3. **Assignment:** Object-Oriented Design

- *Resource*: Pp. 47-49, p. 251, and pp. 256-261 in *Extended Prelude to Programming: Concepts and Design* (2nd ed.)
- **Due Date:** Day 7 [**Individual**] forum
- **Generate** an object-oriented design for a system that keeps tracks of your CD and DVD collection.
- **Identify** each of the classes, associated data, and operations for the classes.
- **Generate** the pseudocode for each of the classes as demonstrated on p. 251.
- **Draw** a GUI that will create the objects and provide access to each object's processing methods. **Note:** Use the drawing tool in Microsoft® Word or in any other applicable drawing tool to complete this part of the assignment.

Week Nine

Requirement Specifications, Design Specifications, and Processing Models

- Create requirement specifications, design specifications, and processing models (including input/out tables, data flow models, and procedural processing).

ASSIGNMENTS

1. **Capstone Discussion Question**
    - **Due Date:** Day 3 [**Main**] forum
    - **Post** your response to the following: Drawing upon your knowledge of software development, which process—requirements, design, coding, or testing—do you think has more impact on the overall success and quality of development? Explain your answer.

2. **Final Project:** Currency Conversion
    - *Resource*: Appendix A
    - **Due Date:** Day 7 [**Individual**] forum
    - **Consolidate** all the sections of the Currency Conversion development documentation: menu selection, requirements, design, and testing.
    - **Incorporate** any changes recommended by the instructor.
    - **Post** the assignment as an attachment.