CSCI 0013 - PROGRAMMING CONCEPTS AND METHODOLOGY II

Catalog Description

Prerequisite: Completion of CSCI 12 with grade of "C" or better
Hours: 72 (54 lecture, 18 laboratory)
Description: Application of software engineering techniques to the design and development of large programs; data abstraction and structures and associated algorithms. (C-ID COMP 132) (CSU, UC)

Course Student Learning Outcomes

- CSLO #1: Implement linear lists with arrays and linked objects.
- CSLO #2: Write programs that use the stack and queue data structures, based on provided specifications.
- CSLO #3: Produce intermediate-sized programs using Object-Oriented design principles, with a combination of written and provided code.
- CSLO #4: Identify effective techniques to test, debug, and document larger programs.
- CSLO #5: Write a program that uses recursive algorithms operating on a binary tree.

Effective Term

Fall 2020

Course Type

Credit - Degree-applicable

Contact Hours

72

Outside of Class Hours

90

Total Student Learning Hours

162

Course Objectives

Lecture Objectives:
1. Discuss the representation and use of primitive data types and built-in data structures.
2. Describe how the data structures are allocated and used in memory.
3. Describe common applications for each data structure.
4. Compare alternative implementations of data structures with respect to performance.
5. Compare and contrast the costs and benefits of dynamic and static data structure implementations.
6. Choose the appropriate data structure for modeling a given problem.
7. Describe the concept of recursion and give examples of its use.
8. Identify the base case and the general case of a recursively defined problem.
9. Compare iterative and recursive solutions for elementary problems such as factorial.
10. Describe the divide-and-conquer approach.
11. Describe how recursion can be implemented using a stack.
12. Discuss problems for which backtracking is an appropriate solution.
13. Determine when a recursive solution is appropriate for a problem.
14. Explain the value of declaration models, especially with respect to programming-in the-large.
15. Identify and describe the properties of a variable such as its associated address, value, scope, persistence, and size.
16. Discuss type incompatibility.
17. Defend the importance of types and type-checking in providing abstraction and safety.
18. Evaluate trade-offs in lifetime management (reference counting vs. garbage collection).
19. Explain how abstraction mechanisms support the creation of reusable software components.
20. Defend the importance of abstractions, especially with respect to programming-in-the-large.
21. Describe how the computer system uses activation records to manage program modules and their data.
22. Justify the philosophy of object-oriented design and the concepts of encapsulation, abstraction, inheritance, and polymorphism.
23. Describe how the class mechanism supports encapsulation and information hiding.
24. Compare and contrast the notions of overloading and overriding methods in an object-oriented language.
25. Explain the relationship between the static structure of the class and the dynamic structure of the instances of the class.
26. Describe how iterators access the elements of a container.
27. Discuss the properties of good software design.
28. Compare and contrast object-oriented analysis and design with structured analysis and design.
Laboratory Objectives:
1. Implement the user-defined data structures in a high-level language.
2. Write programs that use each of the following data structures: arrays, records, strings, linked lists, stacks, queues, and hash tables.
3. Implement, test, and debug simple recursive functions and procedures.
4. Demonstrate different forms of binding, visibility, scoping, and lifetime management.
5. Demonstrate the difference between call-by-value and call-by-reference parameter passing.
6. Design, implement, test, and debug simple programs in an object-oriented programming language.
7. Design, implement, and test the implementation of "is-a" relationships among objects using a class hierarchy and inheritance.

General Education Information

- Approved College Associate Degree GE Applicability
- CSU GE Applicability (Recommended-requires CSU approval)
- Cal-GETC Applicability (Recommended - Requires External Approval)
- IGETC Applicability (Recommended-requires CSU/UC approval)

Articulation Information

- CSU Transferable
- UC Transferable

Methods of Evaluation

- Objective Examinations
  - Example: Multiple choice and short answer questions such as: 1. Which of the following is not a valid Java identifier? a) pressureA

b) ppO2 c) 4counter d) all of the above are valid Java identifiers Answer: c Identifiers must begin with an upper or lowercase letter.

- Problem Solving Examinations
  - Example: 1. Write a static method that computes the arithmetic mean of a list of numbers. Your methods should accept the list of numbers as a parameter. Rubric Grading. Answer: public static double mean(double[] x) { double average=0; //sum the values in the list for(int i=0; i average=average+x[i]; return average/ x.length; } 2. Laboratory assignments covering computer science principles and applications such as: 1. Implement the infix to postfix conversion program discussed in class. Your solution should make use of a reference-based stack. Your program should also check the validity of the input stream. Rubric Grading.
- Projects
  - Example: Create a program that simulates the flight of a model rocket. Your program should allow the user to specify the mass of the rocket, total impulse of the motor, and burn time of the motor. Your simulation should use the formulas discussed in class. For the simulation output, create a chart that shows the altitude of the rocket each 10th of a second.

Repeatable

No

Methods of Instruction

- Laboratory
- Lecture/Discussion
- Distance Learning

Lab:

1. Prior to the lab, ask the class to read the data structures chapter and pay close attention to the design and implementation of the queue. Instructor initiate the discuss for the following: How would you use the queue to simulate waiting in line for a ride at an amusement park? Students will be asked to create a Java program for this simulation. (Laboratory Objective 2)

Lecture:

1. Assume students have read the chapter on data structures. Instructor gives a 20 to 30 minute lecture discussing the array implementation of the stack. Make use of the whiteboard and a handout about the stack during the lecture. Discuss postfix notation and how a stack is used to process a postfix expression. Ask the students to develop pseudocode for processing a postfix expression using a stack. Ask three students to put their solution on whiteboard. Discuss each solution with class. Discuss optimal pseudocode using presentation software. (Lecture Objective 6)

Distance Learning

1. Through Distance Learning, the instructor will present a video lecture on inheritance. After the student views the lecture, they will then be asked to write the GeometricObject class, Circle class and rectangle class using the inheritance. (Laboratory Objective 7)

Typical Out of Class Assignments
Reading Assignments

1. Read and study the chapter on fundamental data structures. Focus your reading on the stack and queue. Be prepared to compare and contrast these two structures and their various implementations in class. 2.Read and study the handout on overloading and overriding. Compare and contrast the notions of overloading and overriding methods in an object-oriented language and be prepared to discuss in class.

Writing, Problem Solving or Performance

Typical early lab assignment: Goals: Cutting a string into pieces.Counting words. Description Write two separate Java programs for cutting a string (sentence) into individual tokens (words). Part 1 Write a Java program that uses the StringTokenizer class (in java.util) to break a string into tokens. Your program should print each word from the string on a separate line. In addition, your program should print the number of tokens found in the string. You may assume the tokens in the string are separated by one or more spaces and the string is terminated with a period. This first program may be written entirely in a main method. Please use the string under test data to test your program. Part 2 Your second program should also slice a string into tokens, but without the aid of the StringTokenizer class. Using the charAt( ) method, and the string length constant, to tokenize the test data string. Add each token from the string into an ArrayList object (see the Java API under java.util). Your program can simply add each token to the ArrayList as the tokens are discovered.

Other (Term projects, research papers, portfolios, etc.)
Required Materials

- Introduction to Java Programming
  - Author: Y. Daniel Liang
  - Publisher: Pearson Education
  - Publication Date: 2014
  - Text Edition: 10th
  - Classic Textbook?: No
  - OER Link:
  - OER:
- Starting out with Java
  - Author: Tony Gaddis
  - Publisher: Pearson
  - Publication Date: 2017
  - Text Edition: 6th
  - Classic Textbook?: No
  - OER Link:
  - OER:

Other materials and-or supplies required of students that contribute to the cost of the course.