

COURSE OUTLINE

Computer Science/Information Systems 212 Advanced Data Structures

I. Catalog Statement

CS/IS 212 is designed to provide a thorough coverage of data structures with data abstraction applied to a broad spectrum of practical applications. Students who take this course master the principles of programming as a tool for problem solving. Students solve practical problems in a computer-equipped laboratory using an object oriented programming language, such as JAVA. Some specific topics covered include hash tables, trees, persistent structures, indexed files, and databases.

Total Lecture Units: 3.0

Total Laboratory Units: 0.0

Total Course Units: 3.0

Total Lecture Hours: 48.0

Total Laboratory Hours: 0.0

Total Laboratory Hours To Be Arranged: 0.0

Total Faculty Contact Hours: 48.0

Prerequisite: CS/IS 211 or equivalent

II. Course Entry Expectations

Prior to enrolling in the course, the student should be able to:

- create computer programs using data structures such as arrays, records, strings, linked lists, stacks, queues, and hash tables;
- create simple recursive functions and procedures;
- explain how abstraction mechanisms aid in creating reusable software components;
- create simple programs in an object-oriented programming language;
- compare and contrast object-oriented analysis and design with structured analysis and design.

III. Course Exit Standards

Upon successful completion of the required coursework, the student will be able to:

- create computer programs solving more complex OOP problems;
- explain more complex abstract data types such as trees, graphs, hash tables, and heaps;
- explain queues, dequeues, and priority queues;
- write programs utilizing trees, binary trees, full binary trees, and complete binary trees.

IV. Course Content

Total Faculty Contact Hours = 48.0

- A. Review of Basic Algorithms (**7 hours**)
 1. Recursive solutions
 2. Array Searching
 3. File searching
- B. Linked Lists (**3 hours**)
 1. List implementations that use arrays
 2. List implementations that link data
 3. Inheritance and lists
 4. Sorted lists
- C. Stacks (**5 hours**)
 1. The Abstract Data Type (ADT) stack
 2. Simple application of the ADT stack
 3. Applications utilizing Postfix and Infix expressions
 4. The relationship between stacks and recursion
- D. Queues (**5 hours**)
 1. Queues
 2. Deques
 3. Priority queues
- E. Class Relationships (**5 hours**)
 1. Inheritance revisited
 2. Dynamic binding and abstract classes
 3. Applications
 4. Advantages of an objects-oriented approach
- F. Trees (**5 hours**)
 1. Terminology
 2. The ADT binary tree
 3. The ADT binary search tree
 4. General trees
- G. Advanced Implementation of Tables (**5 hours**)
 1. Balanced search trees
 2. Hashing
 3. Data with multiple organizations

- H. Graphs (**5 hours**)
 - 1. Terminology
 - 2. Graphs as ADT
 - 3. Graph traversals
 - 4. Applications of graphs
- I. External Methods (**5 hours**)
 - 1. External storage
 - 2. Sorting data in an external file
 - 3. External tables
- J. Advanced Topics in Data Structures (**3 hours**)

V. Methods of Instruction

The following methods of instruction may be used in the course:

- classroom lecture and demonstrations;
- individual assistance in the classroom.

VI. Out of Class Assignments

The following out of class assignments may be used in the course:

- homework exercises;
- programming problems(programming trees, binary trees, full binary trees, and complete binary trees).

VII. Methods of Evaluation

The following methods of evaluation may be used in the course:

- midterm examinations and quizzes;
- final examination.

VIII. Textbook(s)

Carrano, Frank and Timothy Henry. *Data Abstraction and Problem Solving with C++:
Walls and Mirrors*. 6th ed. New York: Addison Welsey. 2012. Print.
12th Grade Textbook Reading Level. ISBN: 978-0132923729.

IX. Student Learning Outcomes

Upon successful completion of the required coursework, the student will be able to:

- develop computer programs using more complex OOP problems;
- analyze and explain complex abstract data types such as such as trees and graphs;
- write programs utilizing trees, binary trees, full binary trees, and complete binary trees.