Mathematics/Science Department
Kirkwood Community College

Course Syllabus

Data Structures

CSC-153

Bob Driggs
Dean

Cate Sheller
Instructor

2010
Data Structures (CSC-153)

Course Description

This course continues the study of program design and construction begun in CSC-142 (Computer Science). The course emphasizes topics in data structures and practice in their specification, design, implementation and use. Topics include container classes, arrays, lists, stacks, queues, trees, graphs, algorithm analysis, object-oriented programming, data abstraction, and searching and sorting techniques.

Prerequisite: CSC-142 (Computer Science)

Credit Hours: 4

Resources

- Books:
  

  Main, Michael. *Data Structures and Other Objects Using Java* 3rd edition, 2006, Pearson/Addison-Wesley


- Web Sites:
  
  Java API

- Software: Java compiler/IDE (e.g. BlueJ, jGrasp)
- Hardware: Personal computers

General Course Objectives

- To develop skills in the implementation and use of data structures and data abstraction.
- To develop skill in algorithm analysis and verification
- To design and implement complex programs that are both functional and readable
To develop advanced skills in the design and implementation of computer programs utilizing an object-oriented language

Detailed Objectives

Unit 1: Foundations
At the conclusion of this unit, students will be able to:

- Effectively document methods using pre and post-condition statements
- Implement basic container classes including bags, lists, and sets
- Write client applications that utilize basic container classes
- Perform time analysis of program methods
- Use Big-O notation to express results of algorithm analysis
- Write programs that demonstrate information hiding
- Define linked list classes and member functions
- Implement doubly-linked lists and circular linked lists
- Use linked lists to implement basic container classes
- Write application programs that rely on linked list operations
- Compare implementations of container classes that use static arrays, dynamic arrays, and linked lists, and explain the relative efficiencies, advantages and disadvantages of each
- Recognize problems for which a recursive solution in most appropriate
- Trace recursive operations using run-time stack illustrations
- Use inductive reasoning to prove the effectiveness of recursive operations
- Explain how to prevent infinite recursion in terms of variant expressions and threshold values
- Hand-simulate and implement serial and binary search algorithms
- Hand-simulate and implement hash table algorithms, using open-address, chained, and double hashing
- Hand-simulate and implement sorting algorithms, including selectionsort, insertionsort, quicksort, and mergesort
- Give examples of situations in which one sort or another is more or less efficient

Unit 2: Basic Data Structures and Techniques
At the conclusion of this unit, students will be able to:

- Recognize situations in which inheritance can simplify implementation of groups of related classes
- Implement super and sub classes
- Implement derived classes and applications given an abstract superclass
- Implement and utilize stacks
- Use stacks to perform evaluation of arithmetic expressions in prefix, infix, and postfix forms
Implement and utilize queues and priority queues
Use queues in programs that simulate client/server scenarios
Explain tree-based concepts and algorithms using standard terminology
Design and implement classes and functions for binary tree nodes and generalized tree nodes
Hand trace binary search tree searches, insertions, removals and traversals
Implement binary search tree algorithms

Unit 3: Advanced Tree Structures and Searching and Sorting Algorithms
At the conclusion of this unit, students will be able to:

- Determine whether or not a given tree satisfies the rules of a binary search tree, heap or B-tree
- Perform insertion and removal of an item in a heap, with reheapification up or down, as necessary
- Hand trace B-tree searches, insertions and removals
- Use a B-tree to implement a Set class
- Use a heap to implement a priority queue class
- Explain graph-based algorithms using standard terminology
- Design and implement graph classes
- Hand-trace and implement depth-first and breadth-first graph traversal algorithms
- Design and implement simple path algorithms

Instructional Methods

- Lecture/discussion/demonstration

  Class meets four times a week for one hour at a time. The primary method of instructional delivery during these sessions will be lecture/discussion, with demonstration of key concepts followed by practice, as described below.

- Closed labs: approximately one per week
- Homework assignments

  Students will gain mastery of the concepts and techniques taught in class through a series of homework assignments in which they will write original programs given problem specifications

Evaluation

- Exams and Quizzes: 60%
- Homework Problems, Labs, and Programs: 40%