

CS331 – Data Structures and Algorithms

Last Updated - 02/06/02

Course Manager - Dr. Gruia Calinescu, Assistant Professor

3 credit hours; required for CS & CPE; 100 min. lecture & 100 min. lab each week

Current Catalog Description - Implementation and application of the essential data structures used in computer science. Analysis of basic sorting and searching algorithms and their relationship to these data structures. Particular emphasis is given to the use of object-oriented design and data abstraction in the creation and application of data structures. Prerequisite: CS 106 or CS 200. (2-2-3)

Textbook

- Dale and Teague, *C++ Plus Data Structures*, Jones and Bartlett Publishers, 2nd Edition, ©2001, ISBN 0763714704
- Roberge, *Data Structures in C++: A Laboratory Course*, Jones and Bartlett Publishers, ©1997, ISBN-0763714232

References - other textbooks or materials

- Deitel and Deitel, *C++ How To Program*, Prentice-Hall, Inc., 3rd Edition, ©2001, ISBN-01308957171 (optional)
- Lippman, *C++ Primer*, Addison Wesley, 2nd Edition, @1991 ISBN-0201548488 (optional)

Course Goals - Students should be able to:

- Explain, implement and apply the following fundamental data structures:
 - lists (unordered & ordered), stacks, queues
- Analyze time and space complexity of algorithms using asymptotic upper bounds (the big-O notation)
- Explain and use pointers, dynamic memory allocation, and linked structures for the above listed data structures
- Outline basic object-oriented design concepts:
 - composition, inheritance, and polymorphism
- Write and test recursive procedures with linked structures, and explain the run-time stack concept
- Explain, implement and apply the following hierarchical data structures:
 - expression trees, binary search trees, and heaps
- Analyze sorting and searching algorithms, and explain their relationship to data structures
- Choose and implement appropriate data structures to solve an application problem

Prerequisites by Topic

- Experience object-oriented programming in C++
- Pre-calculus

Major Topics Covered in Course

1. Review of Primitive types, Arrays, Records, Strings and string processing	3 hours
2. Abstract Data Types (ADTs), implementing ADTs using classes	3 hours
3. List ADT, array implementation	4 hours
4. The concept and properties of algorithms: correctness and performance	1 hour
5. Sorted List ADT, binary search, and performance evaluation	1 hour
6. Constructors, destructor, overloading operators, templates	1 hour
7. Pointers and references	1 hour
8. Dynamic memory allocation	1 hour
9. Stack ADT, array and linked list implementations	4 hours
10. Queue ADT, array and linked list implementations	4 hours
11. Unsorted List ADT, singly linked list implementations	4 hours
12. Sorted, circular, and doubly linked lists	4 hours

13. Copying structures	1 hours
14. Composition, inheritance, and polymorphism	4 hours
15. Testing, empirical measurements of performance	2 hours
16. Programming with recursion, runtime stack	4 hours
17. Divide-and-conquer example: quicksort	1 hour
18. Trees, expression tree ADT, Binary search trees	8 hours
20. Heaps, priority queue ADT, heapsort	3 hours
Project(s) discussion, Midterm(s) and discussion, Project(s) evaluation	6 hours
Final Exam	-
	60 hours

Laboratory projects (specify number of weeks on each)

- 12 labs (1-2 labs each week, each lab contains multiple programming assignments, some with shells, and analysis work)
 - String ADT, Array Implementation of the List ADT, Stack ADT, Queue ADT, Singly Linked List Implementation of the List ADT, Doubly Linked List Implementation of the List ADT, Ordered List ADT, Recursion with Linked Lists, Expression Tree ADT, Binary Search Tree ADT, Heap ADT (time permitting), Performance Evaluation (or equivalent as part of the project)
- 1 object-oriented programming project (12 weeks, requiring at least two interacting classes and appropriate choice of data structure(s), design, implementation, testing, and evaluation)

Estimate CSAB Category Content in Credit Hours

	CORE	ADVANCED		CORE	ADVANCED
Data Structures	1.5	.5	Computer Organization and Architecture	0	
Algorithms	.33		Concepts of Programming Languages	.33	
Software Design	.33				

Oral and Written Communications - Every student is required to submit at least 1 written reports (not including exams, tests, quizzes, or commented programs) of typically 3-5 pages and to make 0 oral presentations of typically 0 minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

- Object-oriented design document required for object-oriented programming project

Social and Ethical Issues - Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

- Legitimate Code Re-Use, 1 hour, object-oriented programming project

Theoretical Foundations - Please list the types of theoretical material covered, and estimate the time devoted to such coverage in contact (lecture and lab) hours.

- Data Structures, 14 hours
- Searching, 7 hours
- Sorting, 3 hours
- Recursion, 3 hours
- Big-O analysis of space and time requirements, 5 hours.
- Memory management, 3 hours
- Object Oriented (data abstraction, subclassing, inheritance, polymorphism), 7 hours

Problem Analysis - Please describe the problem analysis experiences common to all course sections.

- Most labs include follow-up analysis on applying the concepts learned.

- Searching & Sorting topics include discussion of big-O analysis

Solution Design - Please describe the design experiences common to all course sections.

- 1 object-oriented programming project (12 weeks, requiring at least two interacting classes and appropriate choice of data structure(s), design, implementation, testing, and evaluation)

Other Course Information

- Additional Suggested Course Assignments
 - Every lab contains a homework-like assignment, the postlab
 - 1 or 2 midterm exams (100 minutes in total, around 80% programming)
 - 1 final exam (120 minutes, around 50% programming and around 50% design or theory)
- Planned Course Enhancements
 - Change the postlab assignments to include more theoretical content. (Fall 2002)
 - Move object-oriented design concepts of composition, inheritance, and polymorphism to CS200 (Spring 2003)
 - Change “Prerequisite: CS 106 or CS 200.” to “Prerequisite: Grade of ‘C’ or better in CS 106 or CS 200.”
 - Change programming language to Java. (depends on CS200 change to Java in Fall 2003)