

COMPUTER SCIENCE 261

COMPUTER SCIENCE II

I. Introduction

A. Catalog Description

This course is a continuation of the topics introduced in CSC1161. It provides an introduction to the study of fundamental data structures and their associated algorithms. Students learn how to choose appropriate data structures and algorithms for particular problems. They learn about lists, stacks, queues, trees, sorting, searching, abstract data types, and object-oriented programming using an object-oriented programming language. *Prerequisites: CSCI 161 together with MATH 170 or 180; or permission of the instructor.* Satisfies the Mathematical Approaches core requirement.

B. Learning Objectives

1. Learn about data structures and their associated algorithms.
2. Be introduced to proofs of program correctness.
3. Continue the development of good programming style, including object-oriented methods.
4. Be introduced to more advanced methods of program design (including data abstraction, information hiding, and object-oriented design), debugging, and testing.
5. Learn the fundamentals of algorithmic analysis.
6. Work with other students on programming projects.

This course satisfies the Mathematical Approaches category of the university's core curriculum by developing an appreciation of the power of Computer Science and formal methods to provide a way of understanding a problem unambiguously, describing its relation to other problems, and specifying clearly an approach to its solution. A student in this course will develop a variety of mathematical skills, an understanding of formal reasoning, and a facility with applications. Specifically, this course will provide the student with the ability to analyze a problem, to design a systematic way of addressing that problem (an algorithm), and to implement that algorithm in a computer programming language.

C. Prerequisites

1. Computer Science 161 together with Math 170, or 180. A grade of C- is required in the prerequisite courses.

II. Required Topics

A. Program Design

1. Top-down and bottom-up implementation; object-oriented design; testing, profiling, and debugging techniques.

II. Required Topics (cont.)

B. Simple Data Structures

1. Data abstraction. Basic implementation techniques and algorithms associated with stacks, queues, linked lists, and binary trees.

C. Arrays, Searching and Sorting

1. Unsorted lists: Insertion, deletion, and linear search
2. Sorted lists: Insertion, deletion, linear search, and binary search
3. Sorting: At least one $O(n^2)$ algorithm, and one $O(n \log n)$ algorithm

D. Linked Lists

1. Pointers
2. Traversal, insertion, deletion and search operations
3. Doubly-linked list representation and operations

E. Queues and Stacks

1. Array and linked list implementations
2. Insertion and deletion operations

F. Trees

1. Pointer representations of a binary tree
2. Binary search trees
3. Recursive binary tree traversals: inorder, preorder, and postorder
4. Heaps

G. Analysis of Algorithms

1. Fundamentals of algorithm analysis, asymptotic behavior and space vs. time tradeoffs.

H. Program Verification

1. Loop invariants, mathematical induction, preconditions and postconditions.

I. Recursion

1. Recursive data structures and algorithms. When to use recursion and when not to.

J. Introduction to Hash Tables

1. Direct Address Tables
2. Basic hash functions
3. Collision-resolution policies: linear probing and separate chaining.

III. Optional Topics

- A. Huffman coding
- B. Object serialization
- C. Binary file I/O
- D. GUI and event-driven programming

IV. Bibliography

- | | |
|------------|---|
| D. Gries, | <u>The Science of Programming</u> |
| E. Knuth, | <u>The Art of Computer Programming, Vols. I, II, III</u> |
| L. Nyhoff | <u>C++ An Introduction of Data Structures</u> |
| C. Shaffer | <u>A Practical Introduction to Data Structures and Algorithm Analysis</u> |
| R. Lafore | <u>Data Structures and Algorithms in Java</u> |
| N. Wirth, | <u>Algorithms + Data Structures = Programs</u> |
| (video) | <u>Sorting Out Sorting</u> |