Las Positas College
3000 Campus Hill Drive
Livermore, CA 94551-7650
(925) 424-1000
(925) 443-0742 (Fax)

Course Outline for CS 20

ADV PROG W/DATA STRUCTURES/C++

Effective: Fall 2003

I. CATALOG DESCRIPTION:
CS 20 — ADV PROG W/DATA STRUCTURES/C++ — 4.00 units

Design and implementation of larger programs in C++ using software engineering principles. Emphasis on definition and use of data structures. Includes specification of abstract data types, recursion, dynamic memory allocation, stacks, linked lists, queues, binary trees, random access files, and use of hash codes.

3.00 Units Lecture 1.00 Units Lab

Prerequisite
-
or

CS 2 - Computing Fundamentals II
with a minimum grade of C

Grading Methods:
Letter or P/NP

Discipline:

	MIN
Lecture Hours:	54.00
Lab Hours:	54.00
Total Hours:	108.00

II. NUMBER OF TIMES COURSE MAY BE TAKEN FOR CREDIT: 1

III. PREREQUISITE AND/OR ADVISORY SKILLS:

Before entering the course a student should be able to:

A. CS2

IV. MEASURABLE OBJECTIVES:

Upon completion of this course, the student should be able to:

A. define and understand basic operations on abstract data types (ADTs);
B. use dynamic memory allocation to create dynamic data structures;
C. implement programs using linked lists, stacks, queues and binary trees, including implementations with the Standard Template Library;
D. design and implement larger programming projects (greater than 500 lines);
E. understand the concept of time efficiency for algorithms using Big O notation;
F. be able to use advanced concepts of object oriented programming;
G. understand and use direct access methods, including hash codes;
H. be able to design and implement a search program using hashing functions;
I. understand and be able to use template functions and template classes.

V. CONTENT:

A. Data Abstraction
   1. Concept of an ADT
   2. Typical operations in an ADT
   3. ADT implementation issues
B. Big O notation for determining time efficiency of algorithms
   1. Definition
   2. Simple examples: Time order sequential search and binary search
C. Dynamic memory allocation for data structures
D. Linear structures
   1. ADT:List
   2. Linked lists; singly and doubly linked
   3. Space/time comparison of use of array vs. linked list implementation

4. Linked list applications
5. ADT:Queue
6. Implementation of queues by linked list and circular array
7. ADT:Stack
8. Implementation of stacks by array and by linked list
9. Big-O time order of traversal, insertion and deletion for above data structures;
10. Applications such as evaluation of postfix (RPN) and infix expressions

E. Implementation of algorithms that use ADTs (including implementations using the C++)
1. Linked list
2. Stack
3. Queue
4. String

F. Recursion
1. Definition
2. Coding recursive functions
3. Base and general case
4. Use of the program stack and understand stack activation record and its role in implementing recursive calls
5. Recursive data structures

G. Search algorithms
1. Direct access through use of hash codes
2. Iterative binary search
3. Recursive binary search
4. Using a binary search tree
5. Timing of search algorithms

H. Trees
1. General definition, types, and terminology of trees
2. ADT:Binary Tree
3. Implementation of Binary Trees
4. Applications of Binary Trees
5. Complete Binary Trees: building a heap
6. Time order of various Binary Tree operations

I. Sorting Algorithms: implementation and comparison
1. Elementary iterative sorting algorithms
2. Recursive sort such as QuickSort, MergeSort, or HeapSort
3. Timing of sort algorithms selected in course

J. Larger projects
1. Class design diagrams
2. Industry documentation standards
3. Use of inheritance for code reuse
4. Top-down and bottom-up design
5. Testing issues and techniques

K. Hashing Algorithms
1. Concept of searching by direct access: Hash Tables
2. Construction of an elementary hashcode function
3. Resolving collisions by linear probing and chaining
4. Application of hash codes to storage of records in a direct access file or object

VI. METHODS OF INSTRUCTION:

A. If taught online: 1. Reading assignments 2. Programming assignments 3. Online slide shows 4. Participation in online office hours 5. Participation in threaded discussions

B. If taught in class: 1. Reading assignments 2. Lecture 3. Discussion 4. Classroom demonstrations 5. Computer laboratory assignments

VII. TYPICAL ASSIGNMENTS:

A. Programming 1. Write, debug and test a C++ program that uses a stack to evaluate postfix expressions. 2. Write, debug and test a C++ program that compares the runtime efficiency of insertion sort and quicksort algorithms B. Reading 1. Study Chapter 8, “Priority Queues,” in the textbook. 2. Study an assigned website on the Standard Template Library.

VIII. EVALUATION:

A. **Methods**

B. **Frequency**

IX. TYPICAL TEXTS:

1. Budd, Timothy ¬*Data Structures in C++: Using the Standard Template Library (STL).* 1st ed., Addison Wesley, 1998.
2. Topp, William Ford and William *Data Structures with C++ Using STL.* 2nd ed., Prentice Hall, 2001.
3. Collins, William J. *Data Structures and the Standard Template Library.* 1st ed., McGraw-Hill, 2003.

X. OTHER MATERIALS REQUIRED OF STUDENTS:

A. Students taking the online version of this course must have access to a Pentium class computer with a CD drive and an Internet connection.