Las Positas CurricUNET

LAS POSITAS COLLEGE

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

Course Outline for CS 2

COMPUTING FUNDAMENTALS II

Effective: Fall 2010

I. CATALOG DESCRIPTION:
CS 2 — COMPUTING FUNDAMENTALS II — 4.00 units

Object-oriented programming methods applied to intermediate-level problems using C++. Pointers and dynamic allocation; classes; encapsulation; inheritance and polymorphism; object and function overloading; recursive algorithms; introduction to searching and sorting; introduction to abstract data types.

3.00 Units Lecture 1.00 Units Lab

Strongly Recommended
CS 1 - Computing Fundamentals I

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 this course, it is strongly recommended that the student should be able to:

A. CS1

IV. MEASURABLE OBJECTIVES:

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

A. explain and apply basic principles of software engineering;
B. define and use C++ abstract data types in program applications;
C. define and implement C++ dynamic data structures (stacks, queues, etc.);
D. define and use overloaded functions and operators in C++, including friend functions;
E. create and use C++ function templates;
F. define and manipulate pointers;
G. use C++ new and delete mechanisms for dynamic memory allocation;
H. design and use C++ classes, including inheritance within a class hierarchy;
I. define and use virtual member functions to implement polymorphism;
    1. design and implement simple recursive functions;
    2. explain and use simple search and sort algorithms;

V. CONTENT:
A. Program Design and development
    1. Software life cycle
        a. Needs analysis
        b. Algorithm design
        c. Testing
    2. Bottom-up vs Top-down design
    3. Use of drivers and stubs in program development
B. Structured Programming Elements
    1. Review of built-in C++ data types
    2. Review of control structures
        a. Iteration: for, while, do-while
        b. Selection: if, if-else, switch
        c. Other: break, continue
    3. Pointers
        a. Definition and initialization

        - b. Pointer manipulation and dereferencing
        - c. Array name as a pointer
        - d. Dynamic allocation: new and delete operators
    - 4. Operators
        - a. Dereference and address-of operators
        - b. Scope-resolution operator
        - c. Ternary conditional operator (“ : ? *)
    - 5. Functions
        - a. Review: prototypes, parameter passing, calling
        - b. Function templates
        - c. Overloaded functions
        - d. Default arguments
    - 6. Arrays
        - a. Accessing with standard and pointer notation
        - b. Arrays of pointers
        - c. Nested structures
- C. Object Oriented Programming Concepts and C++ Syntax
    - 1. Basic concepts and vocabulary
        - a. Class
        - b. Object
        - c. Method or member function
        - d. Field or member variable
        - e. Encapsulation
        - f. Instantiation
    - 2. Class versus Structs
    - 3. Constructors and Destructors
    - 4. Member function design
        - a. Syntax and use
        - b. Const-ness
    - 5. Access Levels and Scope
        - a. This pointer
        - b. Private vs protected vs public
    - 6. Operator Overloading’
    - 7. Class Templates
    - 8. Friend Functions
    - 9. Polymorphism
        - a. Virtual functions
- D. Introduction to Abstract Data types
    - 1. List ADT
        - a. Basic operations
        - b. Static array implementation
        - c. Dynamic array implantation
        - d. Linked list implementation
    - 2. Stack ADT
        - a. Basic operations
        - b. Array vs linked-list implementation
- E. Introduction to Searching and Sorting
    - 1. Linear vs binary search
    - 2. Selection vs insertion sort
- F. Introduction to Recursion
    - 1. Recursive definitions
    - 2. Simple recursive functions
    - 3. Recursion vs interation

VI. METHODS OF INSTRUCTION:

- A. **Lecture** -
- B. **Discussion** -
- C. Lab Programming Assignments
- D. Interactive activities
- E. **Demonstration** -

VII. TYPICAL ASSIGNMENTS:

- A. Write a program that creates three objects of class Rectangle, with length and width as class data members. Write class member functions that return the area and perimeter of the object.
- B. Write a program that demonstrates C++ class inheritance. Define a class Shape and sub-classes Rectangle, Triangle, and Circle that inherit from Shape. Declare appropriate data members and member functions for each class and sub-class. Write a driver to test the classes by instantiating objects of each and calling their member functions.

VIII. EVALUATION:

A. Methods

- 1. Other:
    - a. Midterms and/or quizzes
    - b. Final exam
    - c. Written homework assignments
    - d. Assigned programming tasks (including labs)

B. Frequency

- 1. At least two in-class midterm examinations, or one in-class midterm examination and several quizzes
- 2. One in-class comprehensive final examination
- 3. Several programming assignments of substantial size and complexity, incorporating all topics in the course.

IX. TYPICAL TEXTS:

1. Deitel & Deitel *C++: How to Program.* 6th ed., Prentice Hall, 2007.
2. Savitch, W. *Absolute C++.* . 4th ed., Addison Wesley, 2009.

X. OTHER MATERIALS REQUIRED OF STUDENTS:

- A. Portable storage device (e.g., USB drive) is strongly recommended.