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

Course Outline for CS 33

ADVANCED C++ PROGRAMMING

Effective: Fall

I. CATALOG DESCRIPTION:
CS 33 — ADVANCED C++ PROGRAMMING — 4.00 units

This is an advanced course in C++ programming. Advanced topics will be covered which will build on the skills acquired in earlier courses. Topics include advanced OOP, class libraries, STL, templates, Input and Output, graphics, files, multimedia, database, prototyping, interface design.

3.00 Units Lecture 1.00 Units Lab

Prerequisite
CS 30 - C++ Programming
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. CS30
1. GENERIC: These outcomes are being developed throughout the entire programming sequence. Upon completion of the course, to an intermediate level, students should be able to: Programming Skills
2. Present the elements and features of the development environment
3. Explain and use the design process
4. Define and use functions and storage classes
5. Define and explain trends in programming standards
6. Write, compile, test and debug programs
7. Present the characteristics of object-oriented programming
8. Define and use data types and variables
9. Define and use multi-dimensional arrays
10. Define and use constructor and destructor functions
11. Define and use function overloading
12. Define and use operator overloading
13. Define and use inheritance mechanisms in OOP
14. Define and use user interfaces
15. Define and use file I/O
16. Define and develop class modules
17. Develop and use event-driven programs
18. Systems Analysis
19. Develop high-level systems and functional specifications
20. Define general scope of work to meet requirements and constraints
21. Systems Design
22. Specify major subsystems and interfaces
23. Develop detail design specifications
24. Select design methodology and tools
25. Identify maintenance requirements
26. Technical Documentation
27. Write in a concise and precise form appropriate for technical documentation
28. Explain and use the processes and techniques of technical documentation

29. Record system specifications accurately and completely
30. Testing and Debugging
31. Select debugging and testing methodology, and develop comprehensive and systematic test plan
32. Develop testing procedures
33. Conduct tests in the most efficient way
34. Test programs, and document errors and solutions
35. User Interface Design
36. Define the requirements for the user interface
37. Detail the development process and methods best suited for the project
38. Develop user interface (UI) to meet user requirements
39. Test UIs
40. Problem Solving
41. Recognize a wide range of problems, and assess their impact on the system
42. Use a wide range of troubleshooting methods and tools to isolate problems
43. Select the appropriate approach to identify causes of the problem based on the given situation
44. SPECIFIC: These outcomes are detailed specifically for this course. Upon completion of the course students should be able to: Give an overview of the evolution and present state of programming languages
45. Describe and employ basic principles of software engineering.
46. Define and use abstract data types in program applications.
47. Define and employ overloading of functions and operators.
48. Write functions implementing iteration.
49. Define and illustrate encapsulation, inheritance and polymorphism in C++.
50. Write programs that use file I/O techniques.
51. Write programs as console applications and windows applications.

B. CS2

IV. MEASURABLE OBJECTIVES:

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

A. GENERIC: These outcomes are being developed throughout the entire programming sequence. Upon completion of the course, to an advanced level, students should be able to: Programming Skills
1. Explain and use the design process
2. Define and use functions and storage classes
3. Define and explain trends in programming standards
4. Write, compile, test and debug programs
5. Present the characteristics of object-oriented programming
6. Define and use pointers
7. Define and use constructor and destructor functions
8. Define and use function overloading
9. Define and use operator overloading
10. Define and use inheritance mechanisms in OOP
11. Define and use user interfaces
12. Define and use file I/O
13. Define and develop class modules
14. Develop and use event-driven programs
15. Embed one language in another
16. Use pointers to data, objects and functions
17. Use dynamic memory allocation

B. Database Design
1. Explain database design concepts and the role of database components
2. Create and customize forms and reports
3. Explain the use of databases and information in the business environment
4. Develop database business applications

C. Systems Design
1. Specify major subsystems and interfaces
2. Develop detail design specifications
3. Select design methodology and tools
4. Identify physical requirements for systems implementation

D. Technical Documentation
1. Write in a concise and precise form appropriate for technical documentation
2. Explain and use the processes and techniques of technical documentation

E. Testing and Debugging
1. Select debugging and testing methodology, and develop comprehensive and systematic test plan
2. Design testing programs to uncover hardware compatibility problems during the development phase of the project
3. Develop testing procedures
4. Conduct tests in the most efficient way
5. Test programs, and document errors and solutions

F. User Interface Design
1. Define the requirements for the user interface
2. Develop and test prototypes
3. Construct user interfaces for flexibility and adaptability

G. Problem Solving
1. Recognize a wide range of problems, and assess their impact on the system
2. Use a wide range of troubleshooting methods and tools to isolate problems

H. Task Management
1. Break down projects and activities into a series of tasks

I. SPECIFIC: These outcomes are detailed specifically for this course. Upon completion of the course students should be able to: Write programs that involve advanced OOP.

J. Write programs using Class libraries and STL.

K. Write programs using class and function templates.

L. Write programs using advanced I/O.

M. Write programs using graphics.

N. Write programs using multimedia.

O. Write programs using a database.

P. Write programs using advanced interface design.

Q. Write programs that use exception handling.

V. CONTENT:

A. Advanced OOP

B. Polymorphism
C. Virtual Functions
D. Class libraries
E. STL
F. Templates
G. Advanced Input and Output
H. Graphics
I. Multimedia
J. Database
K. Prototyping
L. Advanced interface design
M. Exception handling

VI. METHODS OF INSTRUCTION:

A. **Lecture** -
B. **Demonstration** -
C. **Projects** - Optional: Programming project completed in teams
D. **Lab** - Lab Programming Assignments
E. **Discussion** -

VII. TYPICAL ASSIGNMENTS:

A. Write a program to create a database of customers: 1. The customer record contains the following data a. Name (20 char) b. Address (50 char) c. Phone (20 char) d. Number of Employees e. Average salary of employees 2. The application should be capable of creating, modifying, and deleting all customer records. B. Write a team programming project to create a WEB application that will act as a web browser and as a database which contains users and their web sites visited: 1. The browser application behaves just as normal browsers behave but that each user is added to the list of users and the web sites visited are tracked within the database. 2. The system administrator to the Web application can view the list of web sites and disable selected sites. 3. When a user starts the application and attempts to visit a restricted site they are denied access.

VIII. EVALUATION:

A. **Methods**

B. **Frequency**

1. Frequency of evaluation
   a. Recommend 2 or 3 exams plus final examination
   b. Recommend programming assignment to cover each topic within course content. Contents can be combined.
2. Types of Exam Questions
   a. Change the following function into a template:

```
1. int sum(int a, int b, int c)
2. {
3.     return a + b + c;
4. }
```

   b. Write a constructor which has default parameters for the following class:

```
1. class Client
2. {
3.         char name[20];
4.         int    age;
5.         bool married;
6.       private:
7.         Client( . . . );
8.  //  write this constructor to use default values of:
9.  //          name = "", age = 0, married = false;
10. };
```

IX. TYPICAL TEXTS:

1. Hubbard, John *Programming with C++*., McGraw-Hill (Schaum's Outline), 1996.
2. Prata, Stephen *C++ Primer Plus.* 3rd ed., Sams Publishing (The Waite Group), 1998.
3. Schildt, Herbert *C++: The Complete Reference.* 3rd ed., Osborne McGraw-Hill, 1998.
4. Deitel & Deitel *C++ How to Program.* 2nd ed., Prentice-Hall, 1998.

X. OTHER MATERIALS REQUIRED OF STUDENTS:

A. Current version of Microsoft Visual C++
B. Current version of Borland C++ BUILDER
C. Current version of Linux C++ compilers.