

GROSSMONT COLLEGE

Official Course Outline

COMPUTER SCIENCE INFORMATION SYSTEMS 297 – INTERMEDIATE C++ PROGRAMMING

<u>1. Course Number</u>	<u>Course Title</u>	<u>Semester Units</u>	<u>Hours</u>
CSIS 297	Intermediate C++ Programming	4	3 hours lecture 3 hours laboratory

2. Prerequisites

A “C” or “CR” grade or higher in CSIS 296 or equivalent.

Corequisite

None.

Recommended Preparation

None.

3. Catalog Description

This second course in C++ programming explores some of the more advanced concepts of the language including object oriented programming, error handling, and data structures.

4. Course Objectives

The student will:

- a. Demonstrate knowledge of object-oriented C++ programming.
- b. Use C++ programming standards and guidelines.
- c. Analyze working solutions to problems in C++ utilizing an operating systems interface.
- d. Develop debugging strategies.
- e. Demonstrate efficient error handling methods.
- f. Design and implement structured data constructs.
- g. Demonstrate linked lists using abstract data types.
- h. Prepare solutions to programming challenges using stacks and queues.
- i. Search, sort, and update binary trees.

5. Instructional Facilities

- a. Access to the internet.
- b. Classroom with one microcomputer workstation per student.

6. Special Materials Required of Student

Electronic storage media.

7. Course Content

- a. Object-Oriented programming techniques.
  - (1) Structures, classes, and objects.
  - (2) Public and private members.
  - (3) Constructors and destructors.

7. Course Content continued

- (4) Overloading.
- (5) Object arrays.
- (6) Polymorphism and inheritance.
- b. C++ programming standards and guidelines plus features of individual C++ implementations.
  - (1) ANSI standards, Microsoft standards.
  - (2) Templates and the standard template library.
- c. Visual studio GUI.
- d. The integrated Debugger.
  - (1) Watch Windows, array debugging, data memory and stack dumps.
  - (2) Tracing, stepping, breakpoints using multiple source file programs.
- e. Exceptions and exception strategies.
- f. Pointers to structures, enumeration, unions.
- g. Linked list operations.
  - (1) Comparison of linked lists to arrays and vectors.
  - (2) Building linked lists.
  - (3) Transversing linked lists.
  - (4) Appending nodes, inserting nodes, deleting nodes.
  - (5) Destructors and linked lists.
  - (6) Linked list templates.
- h. Creating and using stacks and queues.
  - (1) Dynamic and static stacks, stack operations.
  - (2) STL stack container.
  - (3) Dynamic queue ADTs, queue operations.
  - (4) Circular arrays and full/empty queues.
  - (5) STL deque and queue containers.
- i. Binary trees.
  - (1) Building and using binary trees, best use.
  - (2) Binary searches, transversing the tree.
  - (3) Inserting nodes, deleting nodes.
  - (4) Template considerations.

8. Method of Instruction

- a. Lectures.
- b. Hands-on demonstration.
- c. Lab exercises and projects.

9. Methods of Evaluating Student Performance

- a. Quizzes.
- b. Project and lab activities.
- c. Tests and essay questions.
- d. Objective exams including a written final examination.

10. Outside Class Assignments

- a. Read and study textbook.
- b. Prepare C++ programs and documentation.
- c. Hands-on lab assignments.

11. Texts

- a. Required Text(s):  
Deitel, Harvey M. and Paul J. Deitel. C++ How to Program. Paramus, NJ: Prentice Hall, 2007.
- b. Supplementary texts and workbooks:  
None.

Date approved by the Governing Board: April 17, 2007