GROSSMONT COLLEGE

 COURSE OUTLINE OF RECORD

Curriculum Committee Approval: 04/20/2021

GCCCD Governing Board Approval: 05/18/2021

COMPUTER SCIENCE INFORMATION SYSTEMS 297 – INTERMEDIATE C++ PROGRAMMING

 1. Course Number Course Title Semester Units

 CSIS 297 Intermediate C++ Programming 4

Semester Hours

3 hours lecture: 48-54 hours 96-108 outside of class hours 3 hourslaboratory: 48-54 hours

192-216 total hours

 2. Prerequisites

A “C” grade or higher or “Pass” 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:

1. Demonstrate knowledge of object-oriented C++ programming.
2. Demonstrate working solutions to problems in C++ requiring recursion.
3. Create programs using polymorphism and virtual functions.
4. Develop debugging strategies.
5. Demonstrate efficient error handling methods.
6. Design and implement structured data constructs.
7. Demonstrate linked lists using abstract data types.
8. Prepare solutions to programming challenges using stacks and queues.
9. Search, sort, and update binary trees.

 5. Instructional Facilities

Standard Classroom

 6. Special Materials Required of Student

 None

7. Course Content

1. Object-Oriented programming techniques
	1. Structures, classes, and objects
	2. Public and private members
	3. Constructors and destructors
	4. Overloading
	5. Object arrays
	6. Polymorphism and inheritance
2. C++ programming standards and guidelines plus features of individual C++ implementations
	1. ANSI standards, Microsoft standards
	2. Templates and the standard template library
3. Visual studio GUI
4. The integrated Debugger
	1. Watch Windows, array debugging, data memory and stack dumps
	2. Tracing, stepping, breakpoints using multiple source file programs
5. Exceptions and exception strategies
6. Pointers to structures, enumeration, unions
7. Linked list operations
	1. Comparison of linked lists to arrays and vectors
	2. The process of building linked lists
	3. The process of transversing linked lists
	4. Examples of appending nodes, inserting nodes, deleting nodes
	5. Destructors and linked lists
	6. Linked list templates
8. 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
9. Binary trees
	1. How to build and use binary trees, best use
	2. Binary search techniques, and the concept of transversing the tree
	3. The concept and techniques to Insert and delete ‘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. Examinations and quizzes including a final examination.

 b. Skills demonstration: An example would be having a button trigger a handler method that then calls another UI method to change the button color.

 c. Projects and hands-on lab assignments: An example would be making an application that tracks contacts allowing entry and look up.

 d. Projects and scenario-based lab activities: To include multiple hands-on C++ activities applied from the textbook and other sources, such as chapter-by-chapter projects.

10. Outside Class Assignments

a. Team and individual projects, an example would include the writing of a program to solve a problem within a scientific or business situation or an interactive game utilizing the C++ programming language.

 b. Read and analyze instructor assigned case studies; post analysis and comments to the class discussion board.

 c. Participate in and respond to other students’ analysis and comments on the class discussion board.

d. Complete and pass section quizzes and course final exam.

11. Representative Texts

 a. Representative Text(s):

 Gaddis, Tony. *Starting Out with C++Early Objects.* 10 th edition. Addison-Wesley, 2020.

 b. Supplementary texts and workbooks:

 None

 Addendum: Student Learning Outcomes

 Upon completion of this course, our students will be able to do the following:

* 1. Prepare C++ programs optimizing speed, memory, and transportability
	2. Demonstrate a working knowledge of C++ objects and techniques, using data structures.