Public Course Outline

Modify Credit Course: CS 151 - Advanced C++ Programming

Units Lecture 2.00	**Units Lab** 1.00	**Units Total** 0.00 - 3.00
Lecture Weekly Contact Hours 2.00	**Lab Weekly Contact Hours** 3.00	**Total Weekly Contact Hours** 5.00
Lecture Weekly Out of Class Hours 4.00	**Lab Weekly Outside of Class Hours** 0.00	**Total Weekly Outside of Class Hours** 4.00
Total Contact Hours 80.00 - 90.00	**Total Outside of Class Hours** 64.00 - 72.00	**Total Course Hours** 144.00 - 162.00

Typically Offered: Fall, Spring - F,SP

COURSE DESCRIPTION

In this advanced programming course, students design and implement increasingly complex C++ programs that build upon skills acquired in C++ Programming (CS 150). Students also develop appropriate and efficient methods to test their programs. Topics include polymorphism, inheritance, class libraries, the standard template library, pointers, advanced file input/output operations, recursion, virtual functions, exception handling, dynamic memory management, bitwise operators, and data structures, such as linked lists, stacks, queues, and binary trees.

ENROLLMENT RESTRICTIONS

Prerequisite
CS 150

OUTLINE OF COURSE LECTURE CONTENT

The course lecture will address the following topics:
I. Advanced object-oriented programming (OOP) design and analysis
A. Encapsulation, inheritance, polymorphism
B. Abstract classes and generics or parameterized types
C. Templates and Standard Template Library (STL)
D. Pointers(C/C++), virtual functions, and dynamic memory management, including malloc and free functions.

II. Software engineering design techniques and strategies
A. Memory management, garbage collecting
B. Visibility, scope, lifetime, and binding
C. Event and error handling.

III. Advanced language elements
A. Copy constructors
B. Advanced File I/O techniques
1. Text and binary files
2. Random access files.
C. Bitwise operators
1. Shift
2. Two's complement
3. IEEE floating-point representation
4. Bit access functions
5. Rounding and overflow.

IV. Advanced data structures
A. Vectors, linked lists
B. Iterators (tools to walk through a data structure).

OUTLINE OF COURSE LAB CONTENT

The course lab will address the following topics:
Individual and group projects in the lab are designed to be hands-on activities that support, compliment, and extend the material and theory presented in the lectures.

PERFORMANCE OBJECTIVES

Upon successful completion of this course, students will be able to do the following:
1). Demonstrate and apply advanced concepts of the C++ object-oriented programming language.
2). Employ inheritance and polymorphism in programs.
3). Utilize pointers, reference variables, and dynamic memory allocation.
4). Apply class libraries and the standard template library (STL).
5). Design robust constructor, destructor, and virtual functions.
6). Employ advanced file-handling methods.
7). Create programs using advanced interface design techniques.
8). Design effective exception handling in programs.
9). Construct event-driven programs.
10). Assess the correct data structure for a specific program.
11). Compare and develop efficient and appropriate testing methods.
12). Assess important trends in programming standards.
13). Apply Interprocess communication skills as time permits.

READING ASSIGNMENTS

Reading assignments will be consistent with, but not limited by, the following types and examples:
1). Read about object-oriented programming using the C++ language in the class textbook as well as using and understanding specialized terms, concepts, and theories.
2). Reinforce textbook topics by researching the material on the Web and other texts for further explanation.
3). Read instructor-distributed handouts on specialized topics relevant to successful programming in industry using C++ and extending this knowledge to class projects.

WRITING ASSIGNMENTS

Writing assignments will be consistent with, but not limited by, the following types and examples:
1). Create and write a requirements specification that defines what the program will accomplish and how the user will interact with it for each project done as homework and in class.
2). Write an essay on a current topic assigned in class, such as "Cloud Computing, and what it means and provides for computer scientists."
3). Create an essay and bullet points on a current topic of interest in the computer science field, such as "Algorithmic Bias," and debate the topic with classmates.

OUTSIDE-OF-CLASS ASSIGNMENTS

Outside-of-class assignments will be consistent with, but not limited by, the following types and examples:
1). Complete weekly programming assignments including a requirements specification.
2). Collaborate and complete group programming projects.
3). Prepare for individual presentations on specific topics discussed in the text.

STUDENT LEARNING OUTCOMES

1. The student will be able to analyze designs incorporating more complex control structures and object-oriented design techniques which will result in appropriate solutions to specified problems.
2. The student will be able to implement efficient, effective solutions to more complex programming problems. These solutions will incorporate the principles of inheritance and polymorphism, fault-tolerance through exception and error handling, utilize data files and implement more advanced data structures and associated algorithms.
3. The student will be able to convey more advanced concepts of the object-oriented paradigm, describe more complex syntactic and program control structures, write more sophisticated programs as well as explain program designs verbally and in writing.
4. The student will be able to work in a team, agree upon an organizational model, establish roles, and delegate tasks to develop object-oriented software designs and complete programming projects on time in a cooperative and collective manner.

METHODS OF INSTRUCTION

Instructional methodologies will be consistent with, but not limited by, the following types or examples:
1). Group projects in class allow students to practice critical thinking skills in solving a problem as they evaluate their peers' code.
2). Group projects in class also give students the opportunity to see how other students approach and solve a problem, which enhances their programming skills.
3). Peer learning on practice quizzes and tests helps students learn the material.
4). Instructor lecture and instructor-led discussion.

METHODS OF EVALUATION

Evaluation methodologies will be consistent with, but not limited by, the following types or examples:
1). Individual programming projects will assess the student's ability to understand the material presented in the text and to properly apply it to writing a program.
2). Individual and group programming projects will be evaluated on the following:
a). Use of requirement specifications, proper syntax, and the application of object-oriented concepts and techniques that result in appropriate solutions to a specified problem
b). Ability to integrate student's understanding of programming with the application of industry-standard coding practices.
3). Group programming projects will be evaluated using a grading rubric to assess the students' ability to contribute both verbally and with written code to a programming project within the group context.
4). Class presentations of coding assignments will be evaluated to assess the students' ability to understand and accurately communicate their knowledge to others of modern and object-oriented programming techniques.
5). Performance-based exams will be assessed to evaluate how well the student understands, uses, and retains object-oriented programming principles and techniques.

REQUIRED TEXTBOOKS

Examples of typical textbooks for this course include the following:

1. **Author** Gaddis, Tony, Judy Walters, and Godfrey Muganda
   Title Starting Out with C++ Early Objects
   Edition 10th ed.
   Publisher Pearson
   Year 2020
   ISBN 978-0135235003
   This is the most current, in-print edition. No

2. **Author** Savitch, Walter
   Title Problem Solving with C++
   Edition 10th ed.
   Publisher Pearson
   Year 2018
   ISBN 978-0134448282
   This is the most current, in-print edition. No

3. **Author** Stroustrup, Bjarne
   Title The C++ Programming Language
   Edition 4th ed.
   Publisher Addison-Wesley
   Year 2014
   ISBN 978-0321563842
   This is the most current, in-print edition. No

COURSE REPEATABILITY

Total Completions Allowed: 1

Rationale for multiple enrollments:

Courses Related in Content (CRC) in Physical Education, Visual Arts, and Performing Arts:

DISTANCE ED (FORM A)

Type of Approval: 100% Online or Hybrid

You may indicate here which component(s) of the course should never be conducted online (e.g. proctored exams, labs, in-person orientation, etc.):

ARTICULATION

Transfer Status: Acceptable for Credit: CSU, UC -

CSU/IGETC GE Area(s): 103 - CSU, UC