Course Acronym:	CSCI
Course Number:	30
Descriptive Title:	Advanced Programming in C++
Division:	Mathematical Sciences
Department:	Computer Science
Course Disciplines:	Computer Science
Catalog Description:	This course re-examines earlier C++ topics covered in Computer Science 2 in greater detail and with increased rigor. The course emphasizes the design of advanced Object-Oriented data structures. Topics include data abstraction, abstract classes, single and multiple inheritances, virtual and friend functions, operator overloading, generic data types, the Standard-Template-Library, pointers and dynamic memory management, algorithm efficiency, limits of computation, aggregation-composition modeling, and top-down design. The course concludes with a significant creative project.
Prerequisite:	Computer Science 2 with a minimum grade of C or equivalent
Co-requisite:	
Recommended Preparation:	
Enrollment Limitation:	
Hours Lecture (per week):	3
Hours Laboratory (per week):	3
Outside Study Hours:	6
Total Course Hours:	108
Course Units:	4
Grading Method:	Letter Grade only
Credit Status:	Credit, degree applicable
Transfer CSU:	Yes
Effective Date:	Prior to July 1992
Transfer UC:	Yes
Effective Date:	
General Education: ECC	
Term:	
Other:	
CSU GE:	

Term:	
Other:	
IGETC:	
Term:	
Other:	
Student Learning Outcomes:	**SLO #1 Document Programming Solutions** Students will design, code, compile, test, and document programming solutions to problems requiring the development of C++ classes (by inheritance, by composition; templates), requiring C++ operator overloading, requiring effective use of the Standard Template Library, requiring effective use of pointers and dynamic memory allocation. The students will be able to tell the asymptotic runtime complexity of a given solution using Big-O notation. **SLO #2 Tracing and Verifying** Students, when given a code segment involving the use of a class, will be able to trace the construction of class objects, trace the destruction of class objects, verify whether memory leaks have occurred, trace object assignment operations, verify when copy constructors are invoked and when overloading of copy constructors is required. **SLO #3 Identifying and Eliminating Errors** Students, when given C++ code with errors, will be able to identify what those errors are and will be able to modify the C++ code to eliminate those errors. **SLO #4 Explaining the Concept of C++** Students will be able to explain the concept of C++ class templates and how they relate to the concept of generics, the concept of virtual functions and polymorphism, the concept of multiple inheritances and virtual base classes, the concept of container types and the circumstances where specific containers should or should not be used.
Course Objectives:	1. Implement data abstraction and inheritance. 2. Implement function and operator overloading. 3. Use the Standard Template Library. 4. Create and use friend and virtual functions. 5. Create new classes by inheritance and composition. 6. Use multiple inheritances. 7. Implement virtual base classes. 8. Describe how C++ implements virtual functions. 9. Use exception handling and concurrency for complex programs 10. Use Big-O notation to express the complexity of an algorithm.
Major Topics:	**I. Pointers in C++ (6 hours, lecture)** A. Declare pointers 1. Pointer Types 2. Manipulation pointers B. Using pointers 1. Accessing variables 2. Accessing arrays

3. Const and static types
4. Passing to functions
5. Dynamic Memory Allocation

C. Smart Pointers
1. Unique
2. Shared
3. Weak

II. Classes (6 hours, lecture)

A. Inheritance
1. Base/parent class
2. Derived classes

B. Modes of visibility
C. Virtual functions
D. Friend functions
E. Move Semantics
1. R-value references
2. Move operations

III. Overloading (5 hours, lecture)

A. Function
B. Operator

IV. Templates (5 hours, lecture)

A. Template Class
B. Abstract data types
C. Perfect Forwarding
1. Forwarding Reference
2. Reference Collapsing

V. The Standard Template Library (6 hours, lecture)

A. Containers
1. Deque
2. List
3. Vector
4. Map

B. Algorithms
1. Initialization
2. Sorting
3. Searching
4. Transforming

C. Iterators

VI. Advance Inheritance (3 hours, lecture)

A. Multiple inheritances
B. Virtual base classes

VII Exceptions (6 hours, lecture)

A. Exception Handling
    1. Throwing an Exception
    2. Catching
    3. Function try Blocks
    4. STL Exception Class Hierarchies
B. Exception Safety
    1. No-throw guarantee
    2. Basic guarantee
    3. Strong guarantee

VIII. Object-Oriented Programming (6 hours, lecture)

A. Designing classes
B. Hierarchies

IX. Concurrency (6 hours, lecture)

A. Threads
    1. Race Conditions
    2. Deadlocks
    3. Mutexes
B. Tasks
C. Parallel algorithms
D. Amdahl's Law

X. Analysis of Algorithms (5 hours, lecture)

A. Empirical calculation of runtime performance based on benchmarking.
B. Theoretical computation of asymptotic performance.
C. Recursive algorithms
D. Dynamic programming
E. Study of classical Linear, LogLinear, Quadratic, and Exponential solutions.

XI. Pointers in C++ (6 hours, lab)

A. Declare pointers
    1. Pointer Types
    2. Manipulation pointers
B. Using pointers
    1. Accessing variables
    2. Accessing arrays
    3. Const and static types
    4. Passing to functions
    5. Dynamic Memory Allocation
C. Smart Pointers
    1. Unique
    2. Shared

3. Weak

XII. Classes (6 hours, lab)

A. Inheritance
    1. Base/parent class
    2. Derived classes
B. Modes of visibility
C. Virtual functions
D. Friend functions
E. Move Semantics
    1. R-value references
    2. Move operations

XIII. Overloading (5 hours, lab)

A. Function
B. Operator

XIV. Templates (5 hours, lab)

A. Template Class
B. Abstract data types
C. Perfect Forwarding
    1. Forwarding Reference
    2. Reference Collapsing

XV. The Standard Template Library (6 hours, lab)

A. Containers
    1. Deque
    2. List
    3. Vector
    4. Map
B. Algorithms
    1. Initialization
    2. Sorting
    3. Searching
    4. Transforming
C. Iterators

XVI. Advanced Inheritance (3 hours, lab)

A. Multiple inheritances
B. Virtual base classes

XVII Exceptions (6 hours, lecture)

A. Exception Handling
    1. Throwing an Exception
    2. Catching
    3. Function try Blocks

	4. STL Exception Class Hierarchies B. Exception Safety 1. No-throw guarantee 2. Basic guarantee 3. Strong guarantee **XVIII. Object-Oriented Programming (6 hours, lab)** A. Designing classes B. Hierarchies **XIX. Concurrency (6 hours, lecture)** A. Threads 1. Race Conditions 2. Deadlocks 3. Mutexes B. Tasks C. Parallel algorithms D. Amdahl's Law **XX. Analysis of Algorithms (5 hours, lecture)** A. Empirical calculation of runtime performance based on benchmarking. B. Theoretical computation of asymptotic performance. C. Recursive algorithms D. Dynamic programming E. Study of classical Linear, LogLinear, Quadratic, and Exponential solutions.
Total Lecture Hours:	54
Total Laboratory Hours:	54
Total Hours:	108
Primary Method of Evaluation:	2) Problem solving demonstrations (computational or non-computational)
Typical Assignment Using Primary Method of Evaluation:	Create a class, MyVector, to model vectors. The private data of the class should include a pointer to a double (storage for the elements of the vector) and an integer (the dimension of the vector). Remember that for classes with dynamic memory allocation, you will need to implement The Big Three. You are to overload the I/O operators, operators + and -, and overload the operator * to model the vector dot product. You are to write a test program to make sure your MyVector class does what it is supposed to do.
Critical Thinking Assignment 1:	Develop a class, MyMatrix, to model matrices. The initial implementation of the class will utilize a double pointer to a double data type to store the elements of the matrix, and integer values to hold the number of rows and columns. Since you will have dynamic memory allocation, The Big Three must be implemented. You will overload the I/O operators, operators +, -, and * (matrix multiplication). You will implement a public member function Determinant to return the determinant of the matrix if it is a square matrix. You are to use function recursion in the definition of the Determinant function.

	Finally, you are to write a test program to verify that your MyMatrix class is functioning in a manner consistent with a matrix object, and with the good programming practices you have been learning.
Critical Thinking Assignment 2:	Modeling the work we have been studying in the Vehicle inheritance class hierarchy, you are to create a Person inheritance class hierarchy. The classes Student, Voter, and Faculty will inherit directly from Person. The class GradStudent will inherit from Student. The class StudentVoter will inherit from both the Student class and the Voter class. You will be making use of at least one virtual base class, virtual functions, virtual inheritance, and multiple inheritance. Again, you are to write test code to verify that the classes of your hierarchy are working properly. In particular, you will verify that pointers to the base class Person may be used to properly create, dynamically, objects of any class in your hierarchy.
Other Evaluation Methods:	Laboratory Reports, Other Exams, Quizzes
Instructional Methods:	Lab, Lecture
If other:	
Work Outside of Class:	Problem solving activity, Required reading, Study
If Other:	
Up-To-Date Representative Textbooks:	Stanley B. Lippman, Josee Lajoie, Barbara E. Moo. C++ Primer. 5th ed. Addison Wesley, 2013. (Discipline Standard)
Alternative Textbooks:	
Required Supplementary Readings:	
Other Required Materials:	
Requisite:	Prerequisite
Category:	sequential
Requisite course(s): List both prerequisites and corequisites in this box.	Computer Science-2
Requisite and Matching skill(s):Bold the requisite skill. List the corresponding course objective under each skill(s).	**Write C++ code using programmer-defined classes.** CSCI 2 -Explain and implement basic data structure techniques: pointers, classes, recursion, searching, sorting, templates and dynamic memory allocation. **Write C++ code using function pointers.** CSCI 2 -Explain and implement basic data structure techniques: pointers, classes, recursion, searching, sorting, templates and dynamic memory allocation. **Write C++ code involving dynamic memory allocation and de-allocation.** CSCI 2 -Explain and implement basic data structure techniques: pointers, classes, recursion, searching, sorting, templates and dynamic memory allocation.
Requisite Skill:	

Requisite Skill and Matching Skill(s): Bold the requisite skill(s). If applicable	
Requisite course:	
Requisite and Matching skill(s):Bold the requisite skill. List the corresponding course objective under each skill(s).	
Requisite Skill:	
Requisite Skill and Matching skill(s): Bold the requisite skill. List the corresponding course objective under each skill(s). If applicable	
Enrollment Limitations and Category:	
Enrollment Limitations Impact:	
Course Created by:	Joseph Hyman
Date:	10/05/1990
Original Board Approval Date:	12/09/1991
Last Reviewed and/or Revised by:	Solomon Russell
Date:	11/13/2022
Last Board Approval Date:	07/17/2023 effective FALL 2024