Course Information Form (CIF)

Course Code: CSC 125 (IAI CS 912)
Course Title: Computer Science II (C++)

Department: Business/Computer Science and Technologies
Effective Date: Summer 2026
PCS Code: 1.1 - Baccalaureate/Transfer
CIP Code: 11.0202
Repeatability: 0

Credit Hours

Catalog Notation: 2-2-3
Credit Hour Distribution:

Lecture: 2	Lab: 2	Clinical: 0	**Total: 3**

General Course Information

Catalog Description

Advanced topics in computer science, C++ object-oriented programming, fundamental data structures, and development of a larger-scale program.

General Course Objectives

Students will be able to develop large scale programs using the C++ language, incorporating advanced computer science concepts and data structures in C++ language programs.

Minimum Placement Levels

English	Reading	Math
None	None	None

Prerequisites

Credit in CSC 123 with a grade of C or higher, or an equivalent C or C++ Computer Science I (IAI CS 911) course

Methods of Evaluation

Quizzes (8 or more), Programming Projects (8 or more), a Midterm Exam, a Comprehensive Final, and Lab Programming Assignments (16 or more). Instructors should use additional methods as they see fit.

Instructional Materials and Additional Supplies

Problem Solving with C++ (w/cd), by Savitch, Addison-Wesley. ISBN 0-321-53134-5

Course Content

General Learning Outcomes (GLOs)

- Reasoning and Inquiry: Students will demonstrate the ability to solve problems using deductive reasoning and logic, quantitative reasoning, or the scientific method.
- Technology: Students will demonstrate the ability to evaluate, select, and appropriately use current and emerging tools.

Course Segments and Student Learning Outcomes

Course Segment	Learning Outcomes	Lecture Hours	Lab Hours	Clinical Hours
Unix and Software Development	1. Identify major components of the UNIX operating system. 2. Demonstrate the navigation of the UNIX file system with basic UNIX commands. 3. Identify and apply software development steps (edit, preprocess, compile, link). 4. Demonstrate multiple file compilations. 5. Demonstrate the access of a remote computer using ssh via command line and Graphical Interface. 6. Demonstrate the access of a remote computer using scp/sftp via a Graphical Interface. 7. Demonstrate the editing of a file stored on a remote system. 8. Describe the difference between header and source files. 9. Plan and develop programs that consist of multiple source files.	3	3	0
Object-Oriented Design Review/Overview	1. Describe object-oriented principles. 2. Describe reference variables, static variables, and methods. 3. Plan and develop C++ classes as data models with functions in object-oriented design. 4. Demonstrate object-oriented principles in a C++ programming environment. 5. Plan and develop simple C++ classes. 6. Explain the concept of separating implementation from interface with C++ classes. 7. Explain the concept of information hiding; implement accessors and mutators. 8. Explain and write class methods with emphasis on constructors/destructors. 9. Explain and use reference variables, the keyword this, static variables, and methods. 10. Distinguish between low and high cohesion of an object. 11. Distinguish between low and high coupling of two objects. 12. Explain and use both instance variables and methods, and class variables and methods.	4	4	0
Pointers, Structures, Functions, and Memory Concepts	1. Plan and develop code with C++ pointers and pointers to objects as function arguments. 2. Demonstrate the fundamental memory concepts of stack and heap. 3. Plan and develop code with C++ references and references to objects as function arguments. 4. Plan and develop code with C++ iterators to access containers of objects and containers of pointers to objects. 5. Plan and develop code using new and delete with containers of pointers to objects.	1	1	0

Course Segment	Learning Outcomes	Lecture Hours	Lab Hours	Clinical Hours
Operator Overloading	1. Plan and develop code that applies function overloading and/or operator overloading where language applicable. 2. Plan and develop code with overloaded methods for object assignment and copy constructors. 3. Plan and develop code with overloaded methods for logical comparison. 4. Identify which operators are required to be members, which are required to be non-members, and which can be either in C++. 5. Plan and develop code with overloaded methods for ++ prefix operators and ++ postfix operators. 6. Identify the difference between shallow (physical) and deep (logical) copies.	2	2	0
Linked Lists	1. Identify fundamental concepts of data structures. 2. Plan and develop code using both arrays and linked structures to implement linked lists, including basic operations such as insertion, deletion, and traversal. 3. Plan and develop code using both arrays or equivalent structures and linked structures to implement stacks and queues, including basic operations such as insertion and deletion. 4. Apply recursive programming techniques to linked lists; identify the base case in recursion and correctly implement it. 5. Construct code with merge sort and linear search algorithms on linked lists, and identify basic complexity issues. 6. Plan and develop code with methods for object assignment, copy constructors, and deep/shallow copy for a linked list. 7. Plan and develop code with examples of stack and queue C++ classes using the methods of a linked list.	6	6	0
Templates	1. Apply and use concepts of templates in containers. 2. Construct a linked list implementation of a class template.	1	1	0
Inheritance	1. Identify and explain concepts and application of inheritance. 2. Plan and develop code with simple class hierarchies. 3. Apply concepts of dynamic binding, polymorphism, virtual functions, and pure virtual functions. 4. Apply concepts of concrete/abstract classes. 5. Identify relations between classes as has-a, knows-a, and is-a. 6. Plan and develop code with stack and queues as private, derived classes. 7. Plan and develop code with derived classes as part of a large scale program.	4	4	0
Binary Search Trees	1. Identify fundamental concepts of data structures and containers. 2. Plan and develop code with arrays or equivalent structures and linked structures to implement binary trees, including basic operations such as insertion, deletion, and traversal. 3. Apply recursive programming techniques to trees; identify the base case in recursion and correctly implement it. 4. Plan and develop code with various traversal and search algorithms to trees and analyze the complexity of each.	3	3	0

Course Segment	Learning Outcomes	Lecture Hours	Lab Hours	Clinical Hours
STL Containers and Algorithms	1. Identify fundamental STL data structures (stacks, queues, vectors, sets, maps). 2. Identify fundamental concepts of the STL such as containers (data structures), iterators, and algorithms. 3. Plan and develop code using a C++ STL priority_queue container (data structure) with separate key and data. 4. Demonstrate how a heap (data structure) implements a priority queue. 5. Demonstrate how a heap (data structure) implements a Binary Tree. 6. Plan and develop code using a C++ STL map container (data structure) with separate key and data. 7. Differentiate C++ STL map containers (data structure) and C++ STL set containers (data structure). 8. Analyze the running time of various implementations of STL containers.	4	4	0
Exceptions	1. Construct code with simple exception handling in C++.	1	1	0
Other C++ Topics	1. Identify and use the constructs added to C++ in C++11. 2. Identify and use the constructs added to C++ in C++14.	1	1	0

Total Contact Hours

Lecture Hours	Lab Hours	Clinical Hours
30	30	0