CS G175: C++ PROGRAMMING 1

Item	Value
Curriculum Committee Approval Date	11/05/2024
Top Code	070710 - Computer Programming
Units	3 Total Units
Hours	90 Total Hours (Lecture Hours 36; Lab Hours 54)
Total Outside of Class Hours	0
Course Credit Status	Credit: Degree Applicable (D)
Material Fee	No
Basic Skills	Not Basic Skills (N)
Repeatable	No
Open Entry/Open Exit	No
Grading Policy	Standard Letter (S), • Pass/No Pass (B)

Course Description

Formerly: C++ Programming. This course introduces the fundamentals of software development using the C++ programming language. Software development process will include: designing, writing source code, compiling, linking, executing, and debugging. Data types, arithmetic and logical expressions, debugging, looping, branching, modularization, static and dynamic memory allocation, classes and objects will be presented in lectures and practiced through lab projects. Console applications will be designed and implemented. ADVISORY: CS G102 and course taught at the level of intermediate algebra or appropriate math placement. Transfer Credit: CSU; UC. C-ID: COMP 122. **C-ID:** COMP 122.

Course Level Student Learning Outcome(s)

1. Course Outcomes
2. Create a program that uses object oriented programming constructs.
3. Apply the techniques of structured (functional) decomposition to break a program into smaller pieces.
4. Construct error handling routines.

Course Objectives

- 1. Summarize the evolution of programming languages.
- 2. Design software solutions for business and scientific problems.
- 3. Formulate and document the problem solution.
- 4. Translate mathematical formulas/expressions, and algorithms in the C++ language.
- 5. Resolve coding and logic errors using sophisticated debugging tools.
- 6. Apply optimization techniques.
- 7. Develop a large software solution into modules using structured decomposition.
- 8. Analyze static and dynamic memory allocations.
- 9. Modify input, output devices and files.
- 10. Develop applications using Object-Oriented Programming (OOP) paradigms.

Lecture Content

Survey of Programming Languages Binary instructions and low level languages Compiled vs. translated languages Scripting languages Programming languages evolution Programming concepts Compiling and linking Data types Variables and constants Arithmetic and logical expressions Conversion of business and scientific formulas Branching and looping Arrays, vectors, and strings Input/Output (I/O) formatting Modularization using structured decomposition Functions Parameter passing Local and global variables Static variables Default arguments Overloading functions Functions and menu-driven programming Stubs and drivers Memory management Dynamic memory allocation and deallocation Garbage collection Pointers Pointers to arrays and strings Memory watch for debugging Searching and sorting Linear vs. binary search Sorting algorithms Algorithm complexity analysis Recursion Recursively defined problems Recursive vs. iterative implementations Files and streams Sequential-access for ASCII files Binary files Random-access files Command-line arguments Object-oriented programming (OOP) paradigm Abstract data types Classes and objects Constructors, accessors, and mutators Static members Friends of classes Operator overload UML diagrams Aggregation and composition CRC (Class Responsibility and Collaboration) cards Inheritance Errors and exceptions Input errors Exception handling Programming paradigms Console programming Event-driven programming Software development process Development methodologies Requirements and specifications Designing solution: charts and UML diagrams Coding and unit testing Documenting requirements, specifications, solution options, and user guides Testing and debugging

Lab Content

Introduction to computers and C++ programming C++ basics More flow of control Procedural abstraction and functions that return a value Functions for all subtasks I/O streams as an introduction to objects and classes Arrays Strings and vectors Pointers and dynamic arrays Defining classes Friends, overloaded operators, and arrays in classes Separate compilation and namespaces Pointers and linked lists Recursion Inheritance Exception handling Templates Standard Template Library (STL) and C++11

Method(s) of Instruction

- Lecture (02)
- DE Live Online Lecture (02S)
- DE Online Lecture (02X)
- Lab (04)
- DE Live Online Lab (04S)
- DE Online Lab (04X)

Reading Assignments

Textbook and Websites

Writing Assignments

Students will be required to complete software development projects presented to them in the form of business automation problems requiring solution implementation. Students will be required to write documentation for their projects.

Out-of-class Assignments

An optional library research paper will promote further study and research in current Windows Programming or other related topics selected by the student and approved by the instructor.

Demonstration of Critical Thinking

Students will analyze requirements and select the appropriate programming structures for an efficient solution implementation. Testing and debugging will require students to perform data tracing and error isolation during program execution.

Required Writing, Problem Solving, Skills Demonstration

Students will be required to complete software development projects presented to them in the form of business automation problems requiring solution implementation. Students will be required to write documentation for their projects.

Eligible Disciplines

Computer science: Master's degree in computer science or computer engineering OR bachelor's degree in either of the above AND master's degree in mathematics, cybernetics, business administration, accounting or engineering OR bachelor's degree in engineering AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in mathematics AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in any of the above AND a master's degree in information science, computer information systems, or information systems OR the equivalent. Note: Courses in the use of computer programs for application to a particular discipline may be classified, for the minimum qualification purposes, under the discipline of the application. Master's degree required.

Textbooks Resources

1. Required Tony Gaddis, Judy Walters, Godfrey Muganda. Starting Out with C++: Early Objects, 10th ed. Pearson, 2020 Rationale: .

Other Resources

1. Instructor prepared materials.