*Lecture Contact Hours: 48-54; Outside-of-Class Hours: 96-108;*
*Laboratory Contact Hours: 48-54; Outside-of-Class Hours: 0;*
*Total Student Learning Hours: 192-216*

**CUYAMACA COLLEGE**
COURSE OUTLINE OF RECORD

**Computer Science 181 – Introduction to C++ Programming**

3 hours lecture, 3 units
3 hours laboratory, 1 unit
Total units: 4 units

**Catalog Description**
Introduction to computer programming using a C family language. Students with no previous programming experience in C++ will learn computer organization and operation, binary representation of information, how to plan and create well-structured programs, write programs using sequence, selection and repetition structures, and create and manipulate sequential access files, structs, classes, pointers and arrays.

**Prerequisite**
None

**Recommended Preparation**
"C" grade or higher or "Pass" in CS 119 or equivalent, and intermediate algebra

**Entrance Skills**
Without the following skills, competencies and/or knowledge, students entering this course will be highly unlikely to succeed:
1) Demonstrate knowledge of software engineering methodologies and practices.
2) Identify common tools, functions, classes and utilities available in modern programming languages.
3) Define and utilize structured programming principles.
4) Identify logical "objects" and their "attributes" and "behaviors."
5) Design "behaviors" that follow all structured programming guidelines.
6) Utilize the three basic logic control structures of sequence, decision and repetition.
7) Conceptualize abstraction, polymorphism, inheritance and encapsulation.
8) Manipulate arrays.
9) Contrast procedural vs. object-oriented languages.
10) Trace the flow of execution and values of all variables through a complex object-oriented program.

**Course Content**
1) Introduction to Computers, the Internet, and the Web
2) Short history of C-based languages
3) Introduction to C Programming
4) Typical C Program Development Environment
5) Structured Program Development in C
    a. Algorithms
    b. Pseudocode
    c. If
    d. While
    e. Operators
6) Program Control
7) Functions

   a. Modules
   b. Libraries
   c. Definitions
   d. Headers
   e. Random Number Generator
   f. Recursion
8) Arrays
   a. Arrays to Functions
   b. Sorting
   c. Searching
9) Pointers
   a. Operators
   b. Passing Arguments to Functions by Reference
   c. Expressions and Arithmetic
   d. Arrays
10) Characters and Strings
   a. Character-Handling Library
   b. Input/Output
   c. Search
   d. Memory
11) Formatted Input/Output
   a. printf
   b. Integers, floating-point numbers, strings, characters
   c. scanf
12) Structures, Unions, Bit Manipulations and Enumerations
13) File Processing
14) Data types and structures
15) Preprocessor
16) Classes and objects
17) Multidimensional arrays
18) Inheritance, polymorphism and operator overloading

**Course Objectives**

Students will be able to:

1) Storage class characteristics for variables, objects and methods and justify the reason(s) for the selection
2) Solve a variety of programming problems using sound structured, top-down, object-oriented design principles
3) Use operator overloading and inheritance to minimize reinvention of similar objects
4) Access standard C++ library modules
5) Explain and apply pointers and memory allocation techniques
6) Explain the difference between (and implications of) reference vs. primitive data types
7) Design, create and utilize complex classes and their associated objects
8) Input and output data to and from standard devices and files
9) Visualize array subscripting as memory offset calculations and explain some of the implications. Represent this visualization in drawings
10) Utilize fundamental program control structures (sequence, decision, and repetition)
11) Explain the role of pre-processed code and effectively utilize it
12) Code efficient sort and search algorithms
13) Identify the scope of a C++ variable in a program
14) Demonstrate in a program the use of a method that returns an array
15) Trace program flow and predict outcomes

**Method of Evaluation**
A grading system will be established by the instructor and implemented uniformly. Grades will be based on demonstrated proficiency in subject matter determined by multiple measurements for evaluation, one of which must be essay exams, skills demonstration or, where appropriate, the symbol system.
1) Quizzes and exams that measure students' ability to use programming terminology and explain technical concepts related to program development and implementation in C++.
2) Practical exams requiring students to trace programming code to predict outcome. For example, a 10-20 line section of code is provided and students must be able to identify how all of the values in all variables are altered by that code and predict the final output as specified in the code.
3) Projects requiring students to write complete application programs demonstrating effective use of object-oriented principles, methodology and concepts.

**Special Materials Required of Student**
Flash drive

**Minimum Instructional Facilities**
Computer lab with networked computers and software to create C++ programming environment

**Method of Instruction**
1) Lecture and demonstration
2) Hands-on practice
3) Lab problems

**Out-of-Class Assignments**
1) **Reading:** review lecture notes, course materials, digital references, and more.
2) **Writing:** design, code, and debug multiple C++ programs and objects that demonstrate class objectives; post analysis comments on the class discussion board; and more.
3) **Other:** analyze instructor-assigned pre-coded C++ programs, trace programming code to predict outcome; and more.

**Texts and References**
1) Required (representative examples):
   a. Gaddis, Tony. Starting Out With C++: From Control Structures through Objects. 10th edition. Pearson, 2021.
   b. Paul J. Deitel and Harvey Deitel. C++ How to Program, plus MyProgramming Lab with Pearson eText 9th Edition, Pearson, 2025.
2) Supplemental: None

**Exit Skills**
Students having successfully completed this course exit with the following skills, competencies and/or knowledge:
1) Write well-designed C++ code to solve a variety of scientific problems as standalone programs.
2) Select appropriate storage class characteristics for variables as well as methods.
3) Solve a variety of programming problems using sound structured, top-down, object-oriented design principles.
4) Effectively use inheritance to minimize reinvention of similar objects.
5) Effectively access standard C++ library modules.
6) Recognize the difference between (and implications of) reference vs. primitive data types.
7) Design, create and utilize complex classes and objects.
8) Input and output data to and from standard devices and files.
9) Visualize array subscripting as memory offset calculations and recognize some of the implications.
10) Understand and effectively utilize fundamental program control structures.
11) Understand the relative efficiency of the various sort and search algorithms.

12) Untangle poorly designed code.

**Student Learning Outcomes**

Upon successful completion of this course, students will be able to:

1) Decompose problems and design program solutions using flowcharts, pseudocode, models, or other tools.
2) Properly code applications using the fundamental coding structures: sequence, selection, and loops.
3) Test and debug applications using debugging tools such as trace execution.