Las Positas College
3000 Campus Hill Drive
Livermore, CA 94551-7650
(925) 424-1000
(925) 443-0742 (Fax)

Course Outline for CS 1

COMPUTING FUNDAMENTALS I

Effective: Fall 2015

I. CATALOG DESCRIPTION:
CS 1 — COMPUTING FUNDAMENTALS I — 4.00 units

Introduction to programming and problem-solving using C++. Problem solving techniques and algorithms; program design, development, style, testing and debugging. C++ syntax covered includes: variables; data types; operators and expressions; control structures; library and user-defined functions; basic input/output; binary input/output; arrays; vectors, user-defined data structures.

3.00 Units Lecture 1.00 Units Lab

Strongly Recommended
MATH 107 - Pre-Algebra

Grading Methods:
Letter or P/NP

Discipline:

	MIN
Lecture Hours:	54.00
Lab Hours:	54.00
Total Hours:	108.00

II. NUMBER OF TIMES COURSE MAY BE TAKEN FOR CREDIT: 1

III. PREREQUISITE AND/OR ADVISORY SKILLS:

Before entering this course, it is strongly recommended that the student should be able to:

A. MATH107

IV. MEASURABLE OBJECTIVES:

Upon completion of this course, the student should be able to:

A. Design, create and compile C++ programs within multiple development environments and operating systems, including the use of command-line tools in Unix/Linux.
B. Interpret and apply C++ control structures for sequencing, selection and iteration.
C. Interpret and implement programmer-defined functions in C++.
D. Create and interpret expressions involving arithmetic and logical operators;
E. Interpret and apply arrays and simple programmer-defined data structures in C++.
F. Modify and expand short programs that use standard conditional and iterative control structures and functions.
G. Choose appropriate conditional and iteration constructs for a given programming task.
H. Apply the techniques of structured (functional) decomposition to break a program into smaller pieces.
I. Analyze and explain the behavior of simple programs.
J. Describe, interpret and apply the mechanics of parameter passing.
K. Discuss and apply the concept of algorithms in problem-solving processes.
L. Judge the correctness and quality of algorithms, identifying necessary properties of good algorithms.
M. Describe and apply effective debugging strategies.
N. Identify properties of variables and apply different forms of variable binding, visibility, scoping, and lifetime management.
O. Explain, interpret and apply elements of syntax related variable types, including type-checking, abstraction, type incompatibility and type safety.
P. Summarize the evolution of programming languages and distinguishing characterisitcs of common programming paradigms.
Q. Design, implement, test, and debug programs using basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions.

V. CONTENT:
A. Tools and Procedures
1. Creation, editing and building programs
2. Compiling and Linking in C++ on multiple platforms, including working in a Unix/Linux environment with standard tools
B. History and Background
1. History of programming languages

2. Brief survey of programming paradigms, including procedural and object-oriented languages.

C. Problem Solving and Algorithms
   1. Problem-solving strategies
   2. The concept and properties of algorithms
   3. Implementation strategeis for algorithms within problem-solving processes
   4. Debugging strategies

D. Fundamental Programming Concepts
   1. Basic syntax of C++
   2. Simple I/O in C++
      a. Elementary reading/writing of text files
   3. Variables, types, expressions and assignment
      a. Concept and rules for types and type-checking
      b. Literals and constants
      c. C++ arithmetic operators, including precedence rules
      d. C++ relational operators
      e. C++ logical operators
   4. Conditional and iterative control structures
      a. Nesting control structures

E. Functions and parameter passing
   1. Calling existing functions
   2. Function prototypes and definitions
   3. Parameter passing -- by-value and by-reference
   4. Void vs value-returning functions
   5. Structured decomposition

F. One-Dimensional Arrays in C++
   1. Declaring and accessing one-dimensional arrays
   2. Arrays as function parameters
   3. Searching within an array
   4. Typical array uses (e.g., summation)

G. C++ Strings
   1. Operators: concatenation, character access
   2. C++ string methods (e.g., find, substr)
   3. Comparison with C-style strings

H. C++ Structs
   1. Declaring new struct types
   2. Accessing struct components
   3. Structs as function parameters

I. Vectors in C++
   1. Vectors vs. Arrays
   2. Declaring and using Vectors
   3. Storing Structure variable into Vectors
   4. Vectors as both function parameters and returns

J. Binary File Input/Output
   1. Sequencial Binary File Processing
   2. Random Binary File Processing

VI. METHODS OF INSTRUCTION:

A. **Demonstration** -
B. **Classroom Activity** -
C. **Lecture** -
D. **Projects** -
E. **Lab** -

VII. TYPICAL ASSIGNMENTS:

A. Create a C++ program that inputs, displays and finds the maximum, minimum and mean of an array of decimal numbers.
B. Create a C++ program that reads a file of words, then finds and reports the number of words longer than a given length.
C. Create a C++ program that allows the user to manage inventory information for a store selling a specified kind of items, including inserting and removing items, editing item properties, and recording and querying sales data.
D. Create a C++ program that facilitates the playing of a simple two-player game, enforcing game rules, validating players' input, tracking player turns and scores, and determining whether end-game conditions are met.

VIII. EVALUATION:

A. **Methods**

   1. Other:
      a. Midterm examinations and/or quizzes
      b. Final exam
      c. Programming assignments
      d. Written homework assignments

B. **Frequency**

   1. There should be at least two midterm examinations, or one midterm examination and several quizzes throughout the course.
   2. One comprehensive final examination should be given at the end of the course.
   3. Programming assignments should cover each topic within the course content. Approximately one programming assignment per week is recommended.
   4. Written homework assignments are recommended to parallel reading assignments and to be given weekly. However, it is optional to include written assignments in the instructor's official evaluation of students; instead, the instructor may choose to arrange written assignments to allow self-evaluation by students.

IX. TYPICAL TEXTS:

1. Savitch, Walter. *Problem Solving with C++* . 9th ed., Addison-Wesley, 2014.
2. Gaddis, Tony. *Starting Out with C++: From Control Structures through Objects* . 8th ed., Addison-Wesley, 2014.

X. OTHER MATERIALS REQUIRED OF STUDENTS:

A. It is strongly recommended that each student have a portable storage device (e.g, USB drive) and/or access to an individual account on a cloud-storage service.