**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 2020**

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 file input/output; binary file input/output; arrays; vectors; abstract data types including user-defined data structures and enumerated data types.

3.00 Units Lecture 1.00 Units Lab

**Strongly Recommended**
MATH 107 - Pre-Algebra
with a minimum grade of C

CS 7 - Introduction to Computer Programming Concepts
with a minimum grade of C

**Grading Methods:**
Letter or P/NP

**Discipline:**
- Computer Science

| | MIN |
|---|---|
| **Lecture Hours:** | 54.00 |
| **Expected Outside of Class Hours:** | 108.00 |
| **Lab Hours:** | 54.00 |
| **Total Hours:** | 216.00 |

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
   1. Perform accurate computations with whole numbers, fractions and decimals, signed and unsigned, without using a calculator
   2. Simplify and evaluate variable expressions
   3. Use the English and metric units of length, area, volume, mass, temperature and time
   4. Calculate mean, median and mode from a set of data
   5. Apply the concepts learned to specific real-life applications, such as, simple interest, business and finance, restaurants, bank statements, etc.

B. CS7
   1. Design simple algorithms to solve a variety programming problems.
   2. Design and implement programs of short to medium length, using standard elements of programming languages such as variables, input/output, control structures, functions/methods and arrays.
   3. Explain what an algorithm is and its importance in computer programming.
   4. Design and implement specific program steps and components to achieve desired program behavior.
   5. Design and organize elements of a program using a structured representation such as pseudocode and/or flowcharts.

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 and enumerated data types 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 characteristics of common programming paradigms.
Q. Design, implement, test, and debug programs using basic computation, simple file input/output, standard conditional and iterative structures, and the definition of functions.
R. Develop a complex C++ project comprised of source and header files with multiple compilation steps

## V. CONTENT:

A. Tools and Procedures
   1. Creation, editing and building programs
   2. Compiling and linking C++ on multiple platforms, including working in a Unix/Linux environment with standard tools and multiple file C++ projects

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 strategies 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
   6. Functions using header files and multiple source files

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. Two-Dimensional Arrays in C++
   1. Declaring and accessing two-dimensional arrays
   2. Two-dimensional arrays as function parameters
   3. Typical two-dimensional array uses (e.g., matrix calculations)

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

I. Vectors in C++
   1. Vectors vs. arrays
   2. Declaring and using Vectors
   3. Storing structure variables within Vectors
   4. Vectors as both function parameters and returns

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

K. C/C++ Enumerated data types
   1. Declaring new enumerated data types (EDT)
   2. Accessing constants within the EDT
   3. Using enumerated data types within a C++ structure

L. Binary File Input/Output
   1. Sequential binary file processing
   2. Random binary file processing

## VI. METHODS OF INSTRUCTION:

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

## VII. TYPICAL ASSIGNMENTS:

A. Create a C++ program that inputs, displays and finds the maximum, minimum and mean values 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.
E. Create a multiple file C++ project which includes header and source files containing constants, function prototypes and function definitions

## VIII. EVALUATION:

**Methods/Frequency**

A. Exams/Tests
   Mid-term and Final Exams. Minimum of two per semester
B. Quizzes
   Quizzes for each chapter of the text as completed during class.
C. Projects
   Programming Projects testing the students retention of lecture and lab materials. Project(s) will be given at the conclusion of each chapter during class.
D. Class Participation
   Participate in class discussions during class, each class taught.
E. Lab Activities
   Working out classroom activities ensuring lecture content retention, each class taught

## IX. TYPICAL TEXTS:

1. Gaddis, Tony. *Starting Out with C++: From Control Structures through Objects* . 9th ed., Addison-Wesley, 2018.
2. Stroustrup, Bjarne. *The C++ Programming Language*. 4th ed., Addison-Wesley, 2013.
3. Stroustrup, Bjarne. *Programming: Principles and Practice Using C++*. 2nd ed., Addison-Wesley, 2014.
4. Savitch, Walter. *Absolute C++.* 6th ed., Pearson, 2017.

## 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 of a cloud-storage service.