GAVILAN COLLEGE
5055 Santa Teresa Blvd
Gilroy, CA 95023

Course Outline

COURSE: CSIS 5 DIVISION: 50 ALSO LISTED AS:

TERM EFFECTIVE: Spring 2016 CURRICULUM APPROVAL DATE: 04/27/2015

SHORT TITLE: C++ SCIENTIFIC PROG

LONG TITLE: C++ Scientific Programming

Units	Number of Weeks	Type	Contact Hours/Week	Total Contact Hours
3	18	Lecture:	2	36
		Lab:	3	54
		Other:	0	0
		Total:	5	90

COURSE DESCRIPTION:

An introduction to computer problem solving and programming using the C++ language for science and engineering majors. Students will write programs for a variety of scientific and mathematical applications. This course has the option of a letter grade or pass/no pass. PREREQUISITE: Mathematics 1A ADVISORY: Completion of CSIS 10.

PREREQUISITES:

Completion of MATH 1A, as UG, with a grade of C or better.

COREQUISITES:

CREDIT STATUS: D - Credit - Degree Applicable

GRADING MODES

L - Standard Letter Grade
P - Pass/No Pass

REPEATABILITY: N - Course may not be repeated

SCHEDULE TYPES:

02 - Lecture and/or discussion
03 - Lecture/Laboratory
04 - Laboratory/Studio/Activity
05 - Hybrid
72 - Dist. Ed Internet Delayed

STUDENT LEARNING OUTCOMES:

1. Analyze and explain the behavior of programs involving the fundamental C++ programming constructs.

Measure: exams, discussion, programming problems

PLO:1,2,3,4 ILO: 7,2,3,1

GE-LO:

Year assessed or anticipated year of assessment: 2015

2. Modify and expand short programs that use standard conditional and iterative control structures and functions to solve numerical and scientific problems.

Measure: exams, homework, programming problems

PLO: 4,3,2,1 ILO: 7,2,3

GE-LO:

Year assessed or anticipated year of assessment: 2015

3. Design, implement, test, and debug a numerically intensive program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions.

Measure: case studies, homework, programming problems

PLO: 4,3,2,1 ILO: 7,2,3

GE-LO:

Year assessed or anticipated year of assessment: 2015

CONTENT, STUDENT PERFORMANCE OBJECTIVES, OUT-OF-CLASS ASSIGNMENTS

Curriculum Approval Date: 04/27/2015

(6 Hours) Lectures: Introduction to Computers and C++ Programming I - Computer Systems, Programming and Problem-Solving, Introduction to C++, and Testing and Debugging. C++ Basics - Variables and Assignments, Input and Output, Data Types and Expressions, Simple Flow of Control, and Program Style.

Student Performance Objectives: List the main components of a computer. Define: a machine-language program, a high-level language program, a source program, and an object program. Explain the role of a compiler. Describe an operating system and its purpose. Explain linking. Define an algorithm. Discuss the main phases of the program design process. Name the main kinds of program errors and state what kinds of errors are discovered by the compiler. Discuss the following and explain how a C++ program utilizes them: variables, identifiers, assignments, input and output, and data types and expressions. Explain ways to add flow of control to your programs.

Out of Class Assignments: Read Introduction to Computers and C++ Programming I chapter. Read chapter on C++ Basics. HWand Lab: Complete programming projects such as: entering the C++ program shown in the display and then modifying or changing the program as directed and writing a program that uses several types of variables.

(3 Hours) Lectures: More Flow of Control - Using Boolean Expressions, Multiway Branches, More About C++ Loop Statements, and Designing Loops.

Student Performance Objectives: Describe the purpose of a Boolean expression in C++ programming. Name two kinds of statements in C++ that alter the order in which actions are performed and give some examples. Define: a switch statement, a break statement, a block, and a loop statement. Explain the commonly used methods for terminating an 'input loop'.

Out of Class Assignments: Read chapter on More Flow of Control. HW and Lab: Complete scientific programming problems utilizing the concepts covered in class.

(6 Hours) Lectures: Procedural Abstraction and Functions That Return a Value - Top-Down Design, Predefined Functions, Programmer-Defined Functions, Procedural Abstraction, Scope and Local Variables, and Overloading Function Names. Functions for All Subtasks - 'void' Functions, Call-By-Reference Parameters, Using Procedural Abstraction, Testing and Debugging Functions, and General Debugging Techniques.

Student Performance Objectives: Describe the top-down design method. Demonstrate some of the C++ predefined functions used for mathematical calculations. Explain and demonstrate several of your own functions. Define the term procedural abstraction. Discuss what is meant by overloading the function name.

Define a 'void' function in C++ programming. Explain the 'call-by-reference' mechanism. Define a driver program and a stub and demonstrate how they are used for debugging. Choosing the appropriate input values for debugging a program.

Out of Class Assignments: Read chapter on procedural abstraction and functions that return a value. Read functions for all subtasks chapter. HW and Lab: Complete mathematically intensive and scientific programming problems utilizing the concepts covered in class.

(6 Hours) Lectures: I/O Streams as an Introduction to Objects and Classes - Steams and Basic File I/O, Tools for Stream I/O, and Character I/O. Arrays - Introduction to Arrays, Arrays in Functions, Programming with Arrays, and Multidimensional Arrays.

Student Performance Objectives: Define the C++ construct known as a 'stream'. Explain and demonstrate how to write programs using I/O. Name at least three member functions associated with an 'iostream' object and give examples of the usage of each. Define and utilize a 'manipulator'. Explain and demonstrate how an array is used, including sorting and searching.

Out of Class Assignments: Read chapter on I/O Streams as an Introduction to Objects and Classes and chapter on Arrays. HW and Lab: Complete mathematically intensive programming problems utilizing concepts covered in class.

(6 Hours) Lectures: Strings and Vectors - An Array Type for Strings, The Standard 'string' Class, and Vectors. Pointers and Dynamic Arrays.

Student Performance Objectives: Discuss and demonstrate the use of strings and vectors as they relate to arrays. Define and utilize a 'dynamic array'. Explain the concept of a pointer in C++.

Out of Class Assignments: Read chapters on Strings and Vectors and Pointers and Dynamic Arrays. HW and Lab: Complete programming problems specific to the concepts covered in class.

(6 Hours) Lectures: Defining Classes - Structures, Classes, Abstract Data Types, and Introduction to Inheritance. Friends, Overloaded Operators, and Arrays in Classes - Friend Functions, Overloading Operators, Arrays and Classes, and Classes and Dynamic Arrays.

Student Performance Objectives: Define and explain the purpose of a 'structure'. Describe what makes for a good class definition and provide some techniques that will help define your classes based on modern programming practices. Define a 'class' and demonstrate its use with a member function. State what ADT means. Describe how inheritance support code reuse and make code easier to maintain. Discuss techniques for defining operations on objects as nonmember functions. Define a 'friend function' and demonstrate its use. Explain the difference between a (binary) operator and a function. Discuss and demonstrate the ways that you can combine arrays, structures, and classes to form intricately structured types such as arrays of structures, arrays of classes, and classes with arrays as member variables.

Out of Class Assignments: Read chapter on Defining Classes and read chapter on friends, overloaded operators, and arrays in classes. HW and Lab: Complete scientific programming problems specific to the material covered in class.

(6 Hours) Lectures: Separate Compilation and Namespaces. Pointers and Linked Lists - Nodes and Linked Lists and Stacks and Queues.

Student Performance Objectives: Discuss how a C++ program can be distributed across a number of files so that when some parts of the program change, only those parts need to be recompiled. Discuss and demonstrate the use of namespaces. Define 'nodes' and 'linked list' and demonstrate their applications. Discuss the data structures known as a 'stack' and a 'queue'.

Out of Class Assignments: Read chapters on Separate Compilation and Namespaces and Pointers and Linked Lists. HW and Lab: Complete scientific programming problems specific to the concepts covered in class.

(6 Hours) Lectures: Recursion - Recursive Functions for Tasks, Recursive Functions for Values, and Thinking Recursively. Inheritance - Inheritance Basics, Inheritance Details, and Polymorphism.

Student Performance Objectives: Define, write, and use recursive functions. Explain the C++ process of 'inheritance'. Discuss what 'polymorphism' refers to and apply its function. Explain why you cannot assign a base class object to a derived class variable.

Out of Class Assignments: Read chapter on Recursion and chapter on Inheritance. HW and Lab: Complete mathematical programming problems specific to the material covered in class.

(7 Hours) Lectures: Exception Handling - Exception-Handling Basics and Programming Techniques for Exception Handling. Templates - Templates for Algorithm Abstraction and Templates for Data Abstraction. Standard Template Library - Iterators, Containers, and Generic Algorithms.

Student Performance Objectives: Discuss the purpose of 'exception-handling'. Describe what happens when an exception is never caught. Discuss C++ templates and demonstrate their application. Name the main kinds of 'iterators'. Explain the major difference between a 'vector' and a 'list'.

Out of Class Assignments: Read chapters on Exception Handling, Templates, and Standard Template Library. HW and Lab: Complete programming problems specific to the concepts covered in class.

(2 Hours) Final.

METHODS OF INSTRUCTION:

Lecture, computer demonstration.

METHODS OF EVALUATION:

The types of writing assignments required:

Written homework

The problem-solving assignments required:

Homework problems

Quizzes

Exams

The types of skill demonstrations required:

Class performance

Performance exams

The types of objective examinations used in the course:

Multiple choice

True/false

Matching items

Completion

Other category:

None

The basis for assigning students grades in the course:

Writing assignments: 5% - 20%

Problem-solving demonstrations: 30% - 50%

Skill demonstrations: 15% - 50%

Objective examinations: 10% - 40%

Other methods of evaluation: 0% - 0%

REPRESENTATIVE TEXTBOOKS:

Required:

Savitch, Walter. Problem Solving with C++. Addison-Wesley, 2014. Or other appropriate college level text.

Reading level of text, Grade: 12Verified by: MS Word

Other textbooks or materials to be purchased by the student:

Guide to Scientific Computing in C++ by Joe Pitt-Francis, Jonathan Whiteley , Springer, 2012.

ARTICULATION and CERTIFICATE INFORMATION

Associate Degree:

CSU GE:

IGETC:

CSU TRANSFER:

Transferable CSU, effective 200630

UC TRANSFER:

SUPPLEMENTAL DATA:
Basic Skills: N
Classification: Y
Noncredit Category: Y
Cooperative Education:
Program Status: 1 Program Applicable
Special Class Status: N
CAN: CSCI4
CAN Sequence: XXXXXXXX
CSU Crosswalk Course Department: CSIS
CSU Crosswalk Course Number: 5
Prior to College Level: Y
Non Credit Enhanced Funding: N
Funding Agency Code: Y
In-Service: N
Occupational Course: C
Maximum Hours:
Minimum Hours:
Course Control Number: CCC000076696
Sports/Physical Education Course: N
Taxonomy of Program: 070710