Saddleback College Course Outline

CS 1B - INTRODUCTION TO COMPUTER SCIENCE II

Catalog Id 192070.05

State Id CCC000589855

Division/School Science, Technology, Engineering and Math

Department Computer Science

Program CMPSCI - Computer Science

Subject Computer Science

History and Status

Course Status
Active (Fully Approved)

Course Originator Perez, Larry

- Tech Review Approval
  07/20/2022
- Division Approval
  07/20/2022
- Curriculum Committee Approval
  07/20/2022
- Board of Trustees
  08/29/2022
- State Approval
  06/28/2013
- Technical Change Date
  02/20/2018

Technical Change Comment
SAM fr D--5604-lab fee-73101-rmv CAN&IVC-12301-chg corq fr n; 6/28/13 rec apprl ltr chng fr 3 units to 5; 2/20/18 chngd cc # fr 546315 to 589855

Brief Description

Short Title INTRO TO COMP SCI II

Full Title INTRODUCTION TO COMPUTER SCIENCE II

Catalog Description

The second course in the Introduction to Computer Science series. Covers programming concepts including the properties of modularity and applies a contemporary high level programming language, currently C++, to the solutions of a wide variety of problems relating to science and business. Emphasis is on development, debugging, and testing of programs that use a wide variety of simple and composite data types. Includes functional and object-oriented programming.

Course Requisites

Prerequisite: CS 1A

Course Functions

Course Prior To Y - Not applicable

Course Classification Y - Credit Course

Degree Transfer Applicability

CB04: Credit Status D - Credit - Degree Applicable

Proposed Transfer Types

Acceptable to CSU, UC or Private

Explanation

General Education

UC Approval Date

CSU Approval Date

GE Approval Date

IGETC Approval Date

Local GE Approval Date

Comparable Irvine Course(s)

Comparable Transfer Courses

1. Programming Concepts

   Course ID
   CPSC 121

   Articulation College System
   CSU

   4-yr Institution
   CSU Fullerton

2. Programming in C/C++ as a Second Language

   Course ID
   I&C SCI 45C

   Articulation College System
   UC

4-yr Institution
UC Irvine

SC/IVC Code Yes

- NA - Not Applicable

  Rationale

  Approval Date

Cal-GETC No

CSU GE Yes

- TR - Transferable as an elective-does not fit GE pattern

  Approval Date

IGETC Yes

- NA - Not Applicable

  Approval Date

UC Transferable Course Y - UC Credit

General Education Status Y - Not Applicable

TOP Code 070600 - Computer Science (Transfer)

SAM Code E - Non-Occupational

CAN Code

C-ID

Course Options

Grading Option Letter Grade or Pass/No Pass

Open Entry No

This course is variable No

Repeatable No

Is this a cross listed course? No

Course Values

Method of Instruction Lecture/Lab Combination

Maximum Enrollment 35

Average Enrollment 30

Maximum WSCH
175.000

Average WSCH
150.000

Total Min Units/Hours Calculation

	Lecture	Lab	Learn Ctr	Total
Weekly Faculty Contact Hours	3.00	2.00	0.00	5.00
Total Contact Hours	49.80	33.20	0.00	83.00
Lecture Hour Equivalent	3.00	1.67	0.00	4.67
Full Time Equivalent Faculty	20.00	11.13	0.00	31.13
Units	3.00	0.50	0.00	3.50
Outside of Class Hours				99.60
Total Student Learning Hours				182.60

Schedule Description

Study of the concepts of structured and object-oriented programming using a high-level programming language, currently C++.

Course Content - Topics Covered

Lecture Topics

1. Further development of the following fundamental topics:
    1. Basic syntax and semantics of a higher-level language
    2. Variables, types, expressions, and assignment
    3. Simple Input and Output
    4. Conditional and iterative control structures
    5. Functions and parameter passing (value and reference)
    6. Structured decomposition
    7. Problem solving strategies
    8. The role of algorithms in the problem solving process
2. Implementation strategies for algorithms
3. Debugging strategies
4. The concept and properties of algorithms
5. History of programming languages
6. Brief survey of programmings paradigms
7. Procedural languages
8. Overview of type-checking
9. Declaration models (binding, visibility, scope and Lifetime)
10. Functional decomposition and algorithms (divide and conquer)
11. Emphasis on best practices in functional decomposition (low coupling/high cohesion)
12. Principles of I/O streams, Standard I/O devices
13. Functions (arithmetic and logical)
14. Algorithm development, program development and debugging
15. Design, implement, test, and debug intermediate level programs in C++
16. Analyze the difference between call-by-value and call-by-reference parameter passing
17. Best practices in software development including (but not limited to) style, proper construct use, documentation, testing, reusability, and maintainability
18. Arithmetic and logical operations
19. Scalar, structured and dynamic data types

20. Object-oriented paradigm and programming
21. The conception of types as a set of values together with a set of operations
22. Principles of I/O streams, Standard I/O devices
23. Virtual functions and polymorphism
24. Subprograms
25. Strings: processing
26. Parallel and Multidimensional arrays
27. Pointers
28. Stacks and queues
29. Searching and sorting algorithms
30. Other composite data types
31. Program design tools and programming environments
32. Pair Programming
33. Intro to Libraries
34. Standardized documentation and style
35. Runtime memory management (garbage collection)

Lab/Learning Center Content
Through lab experiences, students will practice skills related to the course content in the following areas:

1. Paired programming
2. Best practices in functional decomposition including separation of concerns
3. Algorithms and problem-solving strategies including representing solutions in flowchart and/or pseudocode form
4. Introduction to IDEs
5. The role of algorithms in the problem-solving process
6. Implementation strategies for algorithms
7. Testing and debugging strategies
8. Introduction to object oriented programming
9. Memory concepts
10. Single dimensional arrays
11. Control structures, logical operators, and basic I/O
12. Conditional and iterative control structures
13. Preprocessor directives and character strings
14. Functions, function calls, scope and lifetime of parameters and identifiers
15. Friend functions, friend classes, and "this" pointer
16. Functions and parameter passing
17. Arrays: passing array to functions, sorting and searching
18. Pointers and arrays
19. Dynamic memory with "new" and "delete"
20. Polymorphism
21. C++ I/O streams, stream manipulations
22. Exception handling
23. Standard template library

24. Structured decomposition
25. Linked lists

Course Content - Learning Objectives

Students participating in this class will:

1. General understanding of basic testing and debugging techniques. Describe strategies that are useful in debugging.
2. Break down a problem into the steps necessary for its solution. Be able to make general assertions about the efficiency of various algorithms.
3. Generate alternative solutions to a variety of problems. Write code using libraries.
4. Select solutions that conform to the principles of structured and object-oriented programming (OOP).
5. Represent solutions in flowchart and/or pseudocode form.
6. Create programs to execute the solutions to a variety of problems using C++.
7. Appraise the results of programs for accuracy and completion utilizing basic testing strategies.
8. Revise solutions and programs whose results are not accurate and/or complete.
9. Describe a problem and its solution in words as part of a program's internal documentation.
10. Understand and apply best practices in programming.
11. Create a problem solution with well constructed functions that conform to commonly understood best practices within the industry.
12. Understand the benefits of pair programming.
13. Apply the techniques of structured (functional) decomposition to break a program into smaller pieces. Choose appropriate conditional and iteration constructs for a given programming task.
14. Discuss the importance of algorithms in the problem-solving process. Identify the necessary properties of good algorithms.
15. Create algorithms for solving problems. Use pseudocode or a programming language to implement, test, and debug algorithms for solving simple problems.
16. Recognize and produce proper C++ syntax. Code algorithms into the C++ language.
17. Introduction to Abstract Data Types (ADTS): Use correct data type and data structures, including objects, linked lists, stacks, and queues. Utilize iteration, arrays, and pointers.
18. Demonstrate the paradigm of object oriented programming. Utilize encapsulation, overloading, inheritance and polymorphism.
19. Compare and contrast object-oriented analysis and design with structured analysis and design.

Course Content - Methods of Evaluation

Evaluation of the student will be based upon the following items:

Writing Assignments

- short answers

  Description

- laboratory reports

  Description

- Other (please describe)

  Description
  Students will demonstrate their understanding of the theories of structured and Object Oriented Programming (OOP) through short answer questions and computer program lab reports.

Problem Solving Demonstrations

- exams

  Description

- quizzes

  Description

- homework problems

  Description

- laboratory report(s)

  Description

- Other (please describe)

  Description
  Students will be evaluated on programming assignments and labs focusing on problem solving skills through algorithm development and analysis of code. b. Students will be evaluated on their performance on programming assignments requiring students to interpret, analyze, apply, and solve problems using the skills and concepts included in the topics covered in this course. c. Major project is evaluated using a rubric and is calculated as a part of the final grade.

Skill Demonstrations

- performance (exam)

  Description

- Other (please describe)

  Description
  Students will be evaluated on their ability to code effectively as demonstrated through assignments and labs.

Examinations

- multiple choice, true/false

  Description

- matching items

  Description

- completion

  Description

- Other (please describe)

  Description
  Students will be required to demonstrate details in their understanding of programming constructs and concepts through T/F, multiple choice questions and well as by write code or algorithms to solve specified problems as well as answer open response questions.

Other

Course Content - In and Out-of-Class Assignments

Typical Reading Assignments
Reading from a college-level text

Typical Writing Assignments
1. Write 10 to 12 programs to solve a variety of problems with scientific, mathematic, and general-purpose applications during the semester. 2. Write internal documentation in the programs above to explain the logic of the solution. 3. Write pseudocode that represents the solution to a variety of problems.

Typical Oral Assignments
Students will participate in weekly class discussions and 3 or 4 small-group discussions.

Typical Other Assignments
Using algorithm design tools that represent the solutions to 6 or 8 previous programming assignments, write a program as a major project that performs several functions. This project includes a complete set of external documentation comprising a description of the program development process, the test data and how it was selected, and a set of instructions for the program user.

Course Content - Other Requirements

Textbooks / Supplies
D.S. Malik , C++ Programming: Program Design Including Data Structures, 8 Ed. Cengage Learning . 2018

Material Fee 0.00

Requisite Validation

1. Prerequisite

   Comment

   Legacy Validation

CS 1A:

1. Describe the mechanics of parameter passing. Describe various data representations.
2. Describe the software development life-cycle. Explain the function of the statements that form the basis of a programming language, using examples in several different languages.
3. Describe the principles of structured programming and be able to describe, design, implement, and test structured programs using currently accepted methodology.
4. Explain what an algorithm is and its importance in computer programming. Represent the solution to a problem in flowchart and pseudocode form.
5. Describe various data representations.
6. Utilize text editors, compilers and IDEs.
7. Break down a problem into the steps necessary for its solution. Translate those solutions into code (created to solve the same problem) in C++.
8. Demonstrate number system conversion to and from binary, decimal and hexadecimal.
9. Discuss basic fundamental units of digital computers.
10. Describe computer organization and operating system features.
11. Analyze and design efficient algorithms for problem solving.
12. Utilize appropriate data types and structures.
13. Write, organize and assemble program documentation.
14. Create correct code, and debug simple errors in C++.
15. Define terms relating to computers and computer science.
16. Describe the historical development of computing machinery.
17. Create a worksheet using electronic spreadsheet software.

Subject

CS - Computer Science

Requisite Course

CS 1A - INTRODUCTION TO COMPUTER SCIENCE I (Active (Fully Approved))

Content Review Type