This course will cover the fundamentals of software development using the C++ Language. The discussions of software development process will include: designing, writing source code, compiling, linking, executing, and debugging. Data types, arithmetic and logical expressions, debugging, looping, branching, modularization, static and dynamic memory allocation, classes and objects will be discussed in lectures and practiced through lab projects. Both console and GUI-based (Graphical User Interface) applications will be designed and created. ADVISORY: Computer Science G102, Math G030

PREREQUISITES:

CS G102: Computer Software Development, Introduction
MATH G030: Intermediate Algebra
or
MATH G040: Accelerated Elementary and Intermediate Algebra

ASSIGNED DISCIPLINES:
Computer science

MATERIAL FEE: Yes [ ] No [X] Amount: $0.00

CREDIT STATUS: Noncredit [ ] Credit - Degree Applicable [X] Credit - Not Degree Applicable [ ]

GRADING POLICY: Pass/No Pass [X] Standard Letter [X] Not Graded [ ] Satisfactory Progress [ ]

OPEN ENTRY/OPEN EXIT: Yes [ ] No [X]

TRANSFER STATUS: CSU Transferable[ ] UC/CSU Transferable[X] Not Transferable[ ]

BASIC SKILLS STATUS: Yes [ ] No [X] LEVELS BELOW TRANSFER: Not Applicable

CALIFORNIA CLASSIFICATION CODES: Y - Not Applicable

NON CREDIT COURSE CATEGORY: Y - Not applicable, Credit Course

OCCUPATIONAL (SAM) CODE: C

REPEATABLE ACCORDING TO STATE GUIDELINES: No [X] Yes [ ] NUMBER REPEATS:

REQUIRED FOR DEGREE OR CERTIFICATE: No [ ] Yes [X]
Associate of Arts: Liberal Arts: Emphasis in Business and Technology(Associate in Arts)
Computer Science(Associate in Science for Transfer)
Mathematics(Associate in Science for Transfer)
Software Development(Certificate of Achievement)
Software Development(Associate in Arts)
Video Game Development(Certificate of Achievement)
**Video Game Development (Associate in Arts)**

**GE AND TRANSFER REQUIREMENTS MET:**

CSU Transfer Course
- A. Transfers to CSU
- Transfer Model Curriculum AS-T

UC Transfer Course
- A. Transfers to UC

**COURSE LEVEL STUDENT LEARNING OUTCOME(S) Supported by this course:**

1. Given a set of requirements for a small business or scientific problem, prepare the software development specification.
2. Design the software components and draw flow-charts for the complex code sections.
3. Design, implement, test, and debug a 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.
4. Apply the techniques of structured (functional) decomposition to break a program into smaller pieces.
5. Describe and utilize the mechanics of parameter passing.
6. Design a simple user interface to satisfy the user interactions.
7. Code all the necessary expressions, branches, loops, functions, classes.
8. Add the appropriate error handling routines.

**COURSE OBJECTIVES:**

1. Assess, analyze, and design software solutions for moderately complex business and scientific problems
2. Properly document the solution
3. Write the software code, mathematical formulas/expressions, and algorithms in the C++ language
4. Eliminate coding and logic errors using sophisticated debugging tools
5. Understand and apply basic optimization techniques
6. Break a large software solution into manageable modules using structured decomposition
7. Manage static and dynamic memory allocations
8. Interact with input, output devices and files
9. Implement applications using Object Oriented Programming (OOP) paradigms through the use of classes and objects
10. Summarize the evolution of programming languages illustrating how this history has led to the paradigms available today

**COURSE CONTENT:**

**LECTURE CONTENT:**

- A. Software Development Process
  1. Collecting Requirements
  2. Preparing Specifications
  3. Designing Solution Options Including Structure Charts and Flow Charts
  4. Coding
  5. Documenting Requirements, Specifications, Solution Options, and User Guides
  6. Compiling & Linking
7. Executing and Debugging

B. Programming Concepts
   1. Programming paradigms and language history
   2. Data Types
   3. Variables and Constants
   4. Arithmetic and Logical Expressions
   5. Conversion of Business and Scientific Formulas
   6. Branching and Looping
   7. Modularization using Structured Decomposition
   8. Functions
   9. Parameter Passing
   10. Dynamic Memory Allocation
   11. Pointers
   12. Arrays and Strings
   13. Files and Streams
   14. Object Oriented Programming (OOP) paradigm
   15. Classes and Objects
   16. Inheritance
   17. Errors and Exceptions
   18. Console Programming
   19. Event-Driven and GUI Programming

LABORATORY CONTENT:

1) Given a set of requirements for a small business or scientific problem, prepare the software development specification
2) Design the software components and draw flow-charts for the complex code sections
3) Create a project with the correct file structure
4) Break the programs into appropriate classes
5) Design a simple user interface to satisfy the user interactions
6) Code all the necessary expressions, branches, loops, functions, classes
7) Add the appropriate error handling routines

METHODS OF INSTRUCTION:

A. Lecture:
B. Lab:
C. Online:
D. Independent Study:

INSTRUCTIONAL TECHNIQUES:

Lecture and/or discussion on specific topics with sample code demonstration using the projector and the white board.

COURSE ASSIGNMENTS:

Reading Assignments

Text & Websites

Out-of-class Assignments

An optional library research paper will promote further study and research in current Windows Programming or other related topics selected by the student and approved by the instructor.

Writing Assignments

Students will be required to complete software development projects presented to them in the form of business automation problems requiring solution implementation. Students will be required to write documentation for their projects.
METHODS OF STUDENT EVALUATION:
Midterm Exam
Final Exam
Short Quizzes
Written Assignments
Essay Examinations
Objective Examinations
Projects (ind/group)
Problem Solving Exercises
Oral Presentations
Skills Demonstration

Demonstration of Critical Thinking:
Students will be demonstrating their laboratory projects. Optional research papers and classroom presentations will further demonstrate their ability in critical thinking and problem solving.

Required Writing, Problem Solving, Skills Demonstration:
Students will be required to complete software development projects presented to them in the form of business automation problems requiring solution implementation. Students will be required to write documentation for their projects.

TEXTS, READINGS, AND RESOURCES:

TextBooks:
1. Tony Gaddis, Judy Walters, Godfrey Muganda. Starting Out with C++: Early Objects, 8th ed. Addison-Wesley, 2013

Other:
1. A syllabus and multiple reference material related to the latest software development technologies will be distributed by the instructor.

LIBRARY:
Adequate library resources include: Print Materials

Comments:

Attachments:
Attached Files