GROSSMONT COLLEGE

OFFICIAL COURSE OUTLINE

Curriculum Committee Approval: 04/20/2021

GCCCD Governing Board Approval: 05/18/2021

COMPUTER SCIENCE INFORMATION SYSTEMS 119 – INTRODUCTION TO COMPUTER PROGRAMMING

1. Course Number Course Title Semester Units

CSIS 119 Introduction to Computer Programming 3

Semester Hours

3 hours lecture 48-54 total hours 96-108 outside-of-class hours 144-162 total hours

2. Course Prerequisites

None

Corequisite

None

Recommended Preparation

A “C” grade or higher or “Pass” in CSIS 110 and CSIS 112 or equivalent.

3. Catalog Description

An introductory course in computer programming as a foundation for more advanced programming, computer science, computer networking, or software engineering courses. Emphasis is on the development of problem solving skills as it introduces students to computer programming principles and best practices using modular and Object Oriented programming concepts. Attention is given to development of effective software engineering practices emphasizing such principles as analysis and design decomposition, encapsulation, procedural abstraction, testing, and software reuse. Students learn and apply standard programming constructs, problem-solving strategies, the concept of an algorithm, fundamental data structures, and the machine representation of data.

4. Course Objectives

The student will:

a. Demonstrate knowledge of software engineering methodologies and practices.

b. Develop skills in problem analysis and solution design.

c. Translate problem analyses to modular and object oriented software system designs.

d. Develop skills in the application of structured programming logic.

e. Apply the principles of a software development process.

f. Demonstrate introductory skills in an Object Oriented language.

g. Develop and test simple software systems.

5. Instructional Facilities

Standard Classroom

6. Special Materials Required of Student

None

7. Course Content

a. History of computing

b. Basic computability

c. Machine level representation of data

d. Assembly level machine organization

e. Software tools and environments

f. Software requirements and specifications

g. Software design (object oriented and procedural)

h. Software validation

i. Overview of programming languages

j. Object oriented programming (data hiding, encapsulation, inheritance, polymorphism)

k. Abstraction mechanisms

l. Fundamental programming constructs

m. Algorithms and problem solving

n. Fundamental data structures and computing algorithms

o. Declarations and types

p. Fundamental techniques in flowcharting

8. Method of Instruction

a. Demonstration in a traditional classroom or via electronic means.

b. Hands-on practice in either a dedicated or a virtual lab environment

c. Critical thinking exercises.

d. Topical discussion of current computer programming trends and issues

9. Methods of Evaluating Student Performance

a. Completion of exercises.

b. Successful analysis and design of solutions to projects.

c. Objective examinations and quizzes including a final examination.

d. Written quizzes and exams that measure students’ ability to describe computer programming principles, as well as the ability to analyze a scenario and choose the among the alternative paths.

e. Scenario-based lab activities that measure students’ ability to design and create computer programs.

10. Outside Class Assignments

a. Textbook reading assignments.

b. Problem solving exercises.

c. Complete Study Guides provided covering major topics.

d. Troubleshoot/analyze imposed programming scenarios, investigate potential alternatives, and implement action to achieve a determined result.

e. Complete and pass section quizzes and course final exam.

f. Read and analyze instructor assigned case studies; post analysis and comments to the class discussion board.

g. Respond to other students’ analysis and comments on the class discussion board.

11. Representative Texts

a. Representative Text(s):

Farrell, Joyce. *Programming Logic and Design (Comprehensive),* NinthEdition, Cengage Learning, 2018.

b. Supplementary texts and workbooks:

None

Addendum: Student Learning Outcomes

Upon completion of this course, our students will be able to do the following:

* 1. Apply problem analysis best practices to create a solution design.
  2. Evaluate existing problem solution pseudocode or flowchart for correctness and revise it if incorrect.
  3. Name, explain, create, compare and contrast the three best practice structures for computer programming (sequence, selection, iteration).