COURSE DESCRIPTION:

Prerequisites: **Take One Set:**
- Set 1: DMA-010, DMA-020, DMA-030, and DMA-040
- Set 2: MAT-060* and MAT-070
- Set 3: MAT-060* and MAT-080
- Set 4: MAT-060* and MAT-090
- Set 5: MAT-095
- Set 6: MAT-120
- Set 7: MAT-121
- Set 8: MAT-161
- Set 9: MAT-171
- Set 10: MAT-175

Corequisites: None

This course introduces computer programming and problem solving in a structured program logic environment. Topics include language syntax, data types, program organization, problem-solving methods, algorithm design, and logic control structures. Upon completion, students should be able to manage files with operating system commands, use top-down algorithm design, and implement algorithmic solutions in a programming language. *This course has been approved to satisfy the Comprehensive Articulation Agreement general education core requirement in natural science/mathematics (Quantitative Option).* Course Hours per Week: Class, 2. Lab, 3. Semester Hours Credit, 3.

LEARNING OUTCOMES:

Upon successful completion of this course, the student will be able to:

a. Describe the major components in problem solving for a computer program.
b. Apply top-down concepts in algorithm design.
c. Create flowcharts to illustrate program algorithm or process.
d. Analyze and write pseudocode to illustrate compact and informal high-level descriptions of computer programming algorithms.
e. Explain the concept of data storage and named memory locations.
f. Apply decision and repetition structures in program design.
g. Write and incorporate methods and functions to demonstrate program competence.
h. Define variables and arrays used in program methodology.
i. Implement input and output to access and process files.
j. Describe and apply object-oriented programming methodology.
k. Apply recursion techniques to problem solving.
OUTLINE OF INSTRUCTION:

I.  Introduction to Computers and Programming
    a)  Hardware and Software
    b)  Programs and Data

II. Input, Processing, and Output
    a)  Designing a Program
    b)  Input, Output, and Variables
    c)  Variable Assignment and Calculations
    d)  Variable Declarations and Data Types
    e)  Named Constants
    f)  Hand Tracing a Program
    g)  Documenting a Program

III. Modules
    a)  Introduction to Modules
    b)  Defining and Calling a Module
    c)  Passing Arguments to Modules
    d)  Local Variables, Global Variables and Global Constants

IV. Decision Structures and Boolean Logic
    a)  Introduction to Decision Structures
    b)  Dual Alternative Decision Structures
    c)  Comparing Strings
    d)  Nested Decision Structures
    e)  The Case Structure
    f)  Logical Operators
    g)  Boolean Variables

V.  Repetition Structures
    a)  Introduction to Repetition Structures
    b)  Condition-Controlled Loops: While, Do-While, and Do-Until
    c)  Count-Controlled Loops and the For Statement
    d)  Calculating a Running Total
    e)  Sentinels
    f)  Nested Loops

VI. Functions
    a)  Introduction to Functions
    b)  Writing Your Own Functions
    c)  More Library Functions

VII. Input Validation
    a)  Garbage In, Garbage Out
    b)  The Input Validation Loop
    c)  Defensive Programming
VIII. Arrays
   a) Array Basics
   b) Sequentially Searching an Array
   c) Processing the Contents of an Array
   d) Parallel Arrays
   e) Two-Dimensional Arrays
   f) Arrays of Three or More Dimensions

IX. Sorting and Searching Arrays
   a) The Bubble Sort Algorithm
   b) The Selection Sort Algorithm
   c) The Insertion Sort Algorithm
   d) The Binary Search Algorithm

X. Files
   a) Introduction to File Input and Output
   b) Using Loops to Process Files
   c) Using Files and Arrays
   d) Processing Records
   e) Control Break Logic

XI. Menu-Driven Programs
   a) Introduction to Menu-Driven Programs
   b) Modularizing a Menu-Driven Program
   c) Using a Loop to Repeat the Menu
   d) Multiple-Level Menus

XII. Recursion
   a) Introduction to Recursion
   b) Problem Solving with Recursion
   c) Examples of Recursive Algorithms

XIII. Object-Oriented Programming
   a) Procedural and Object-Oriented Programming
   b) Classes
   c) Using the Unified Modeling Language to Design Classes
   d) Finding the Classes and Their Responsibilities in a Problem
   e) Inheritance
   f) Polymorphism

REQUIRED TEXTBOOKS AND MATERIALS:

Text to be assigned by the instructor each semester