EASTERN ARIZONA COLLEGE
C Programming I
Course Design
2019-2020

Course Information

Division	Business
Course Number	CMP 130
Title	C Programming I
Credits	3
Developed by	Lydia Mata
Lecture/Lab Ratio	2 Lecture/2 Lab

Transfer Status

ASU	NAU	UA
CSE 100, Computer/ Stats (CS)	CS 122; Science & Applied Science [SAS] --and-- CS 122L; Lab Science [LAB]	ECE 175

Activity Course	No
CIP Code	11.0100
Assessment Mode	Pre/Post Test (50 Questions/50 Points)
Semester Taught	Upon Request
GE Category	AAS degree only
Separate Lab	No
Awareness Course	No
Intensive Writing Course	No
Diversity and Inclusion Course	No

Prerequisites

None

Educational Value

This purpose of this course is to learn programming principles and techniques using the C language. This course would also be helpful in other programming courses such as Java Programming.

Description

A comprehensive introduction to the C language; preparation and writing of programs using C, using current programming techniques. A prior programming language is helpful. For two-year Computer majors or students transferring in a Computer Information/Management Information Systems degree. Identical to EGR 130.

Supplies

USB flash drive to save programs.

Competencies and Performance Standards

1. Identify the element of a C Program and the types of data that can be processed by C.

Learning objectives

What you will learn as you master the competency:

a. Evaluate C Programs using directives and identifiers.
b. Explain the importance of variables and data types.
c. Use executable statements to code algorithms.
d. Recognize the general form of a C Program.
e. Use arithmetic expressions to solve programming problems.
f. Specify the format of numbers in program output.
g. Differentiate between interactive mode and batch mode.
h. Locate errors in a C Program.

Performance Standards

Competence will be demonstrated:

- by successful completion of required program
- by successful completion of final exam

Criteria-Performance will be satisfactory when:

- learner evaluates a C Program using directives and identifiers
- learner explains the importance of variables and data types
- learner uses executable statements to code algorithms
- learner recognizes the general form of a C Program
- learner can use arithmetic expressions to solve programming problems
- learner specifies the format of numbers in program output
- learner can differentiate between interactive mode and batch mode
- learner can locate errors in a C Program

2. Use the top-down design techniques to simplify algorithms.

Learning objectives

What you will learn as you master the competency:

a. Build a program from existing information.
b. Use library functions to solve mathematical computations.
c. Use structure charts to track relationships among subproblems.
d. Define functions that have no arguments.
e. Write functions containing input arguments.

Performance Standards

Competence will be demonstrated:

- by successful completion of required program
- by successful completion of final exam

Criteria-Performance will be satisfactory when:

- learner can build a program from existing information
- learner uses library functions to solve mathematical computations

- o learner uses structure charts to track relationships among subproblems
- o learner can define functions that have no arguments
- o learner can write functions containing input arguments

3. Use selection structures containing the If and Switch statements.

Learning objectives

What you will learn as you master the competency:

a. Control the flow of execution in a program or function using control structures.
b. Establish criteria for a group of statements using conditions.
c. Write IF statements to execute given values.
d. Write IF statements with compound statements.
e. Write a program using decision steps.
f. Modify a program containing function subprograms.
g. Use nested IF statements to code decisions with multiple alternatives.
h. Use the Switch statement to execute simple expressions.

Performance Standards

Competence will be demonstrated:

- o by successful completion of required program
- o by successful completion of final exam

Criteria-Performance will be satisfactory when:

- o learner can control the flow of execution in a program or function using control structures
- o learner can establish the criteria for a group of statements using conditions
- o learner writes IF statements to execute given values
- o learner writes IF statements with compound statements
- o learner writes a program using decision steps
- o learner modifies a program containing function subprograms
- o learner uses nested IF statements to code decisions with multiple alternatives
- o learner uses the Switch statement to execute simple expressions

4. Write and use nested loops in programs.

Learning objectives

What you will learn as you master the competency:

a. Include a repetition structure in a program.
b. Code a pretest loop using the While statement.
c. Compute a sum or product in a loop.
d. Code a loop using the For statement.
e. Code conditional loops.
f. Formulate a valid loop structure.
g. Include nested loops in programs.
h. Code a loop using the do-while statement.
i. Use the top-down design to solve programming problems.

j. Recognize ways to debug and test programs.

k. Recognize common programming errors.

Performance Standards

Competence will be demonstrated:

- by successful completion of required program
- by successful completion of final exam

Criteria-Performance will be satisfactory when:

- learner includes a repetition structure in a program
- learner codes a pretest loop using the While statement
- learner can compute a sum or product in a loop
- learner codes conditional loops
- learner formulates a valid loop structure
- learner includes a nested loop in programs
- learner uses the top-down design to solve programming problems
- learner can recognize ways to debug and test programs
- learner recognizes common programming errors

5. Define modular programming.

Learning objectives

What you will learn as you master the competency:

a. Use output parameters to return multiple results from a function.

b. Create multiple calls to a function using input/output parameters.

c. Define and locate scope of names in programming.

d. Write formal output parameters as actual functions.

e. Manipulate numeric data to create programs with multiple functions.

f. Debug and test a program system.

g. Review common programming errors.

Performance Standards

Competence will be demonstrated:

- by successful completion of required program
- by successful completion of final exam

Criteria-Performance will be satisfactory when:

- learner uses output parameters to return multiple results from a function
- learner can create multiple calls to a function using input/output parameters
- learner can define and locate scope of names in programming
- learner can write formal output parameters as actual functions
- learner manipulates numeric data to create programs with multiple functions
- learner debugs and tests a program system
- learner reviews common programming errors

6. Define simple data types.

Learning objectives

What you will learn as you master the competency:

a. Differentiate the various numeric and data types.
b. Recognize the representation of character values.
c. Associate numeric codes in categories using enumerated type.
d. Illustrate methods for iteratively approximating a root of an equation.

Performance Standards

Competence will be demonstrated:

- by successful completion of required program
- by successful completion of final exam

Criteria-Performance will be satisfactory when:

- learner can differentiate among the various numeric and data types
- learner recognizes the representation of character values
- learner can associate numeric codes in categories using enumerated type
- learner can illustrate methods for iteratively approximating a root of an equation

7. Use arrays to store data.

Learning objectives

What you will learn as you master the competency:

a. Declare and reference arrays.
b. Recognize the distinction between an array subscript value and an array element value.
c. Use for loops for sequential access.
d. Use array elements as function arguments.
e. Write functions that have arrays as arguments.
f. Perform array searches and sorts.
g. Use multi-dimensional arrays to represent multi-dimensional objects.
h. Select array elements for processing.
i. Define subscript range errors.

Performance Standards

Competence will be demonstrated:

- by successful completion of required program
- by successful completion of final exam

Criteria-Performance will be satisfactory when:

- learner can declare and reference arrays
- learner recognizes the distinction between an array subscript value and an array element value
- learner uses for loops for sequential access
- learner uses array elements as function arguments
- learner can write functions that have arrays as arguments
- learner can perform array searches and sorts
- learner uses multidimensional arrays to represent multidimensional objects

- o learner selects array elements for processing
- o learner can define subscript range errors

8. Manipulate string data.

Learning objectives

What you will learn as you master the competency:

a. Review string basics.
b. Create string library functions.
c. Concatenate strings.
d. Compare a portion of a string variable's contents to another string.
e. Modify lists using array of pointers.
f. Analyze and convert characters.
g. Convert characters placed in a string.
h. Create an editor to process text.
i. Review common string programming errors.

Performance Standards

Competence will be demonstrated:

- o by successful completion of required program
- o by successful completion of final exam

Criteria-Performance will be satisfactory when:

- o learner reviews string basics
- o learner creates string library functions
- o learner can concatenate strings
- o learner compares a portion of a string variable's contents to another string
- o learner can modify lists using array of pointers
- o learner can analyze and convert characters
- o learner converts characters placed in a string
- o learner can create an editor to process text
- o learner reviews the common string programming errors

9. Use recursion as an alteration to iteration.

Learning objectives

What you will learn as you master the competency:

a. Define recursion.
b. Trace a recursive function.
c. Code recursive mathematical functions.
d. Implement recursive functions with arrays and string parameters.
e. Apply recursive functions to solve problems.
f. Compare iterative and recursive functions.
g. Solve recursive functioning errors.

Performance Standards

Competence will be demonstrated:

- by successful completion of required program
- by successful completion of final exam

Criteria-Performance will be satisfactory when:

- learner can define recursion
- learner can trace a recursive function
- learner implements recursive functions with arrays and string parameters
- learner can apply recursive functions to solve problems
- learner compares iterative and recursive functions
- learner solves recursive functioning errors

10. Define data types that represent structured collections of data pertaining to particular objects.

Learning objectives

What you will learn as you master the competency:

a. Define the components of a structured data object.
b. Use structure data types as input and output parameters.
c. Apply functions to return structured result values.
d. Manipulate structure types to problem solve.
e. Organize data using parallel arrays and arrays of structure.
f. Use union types to interpret a data object in a variety of ways.
g. Review common programming errors seen when using manipulated structure types.

Performance Standards

Competence will be demonstrated:

- by successful completion of required program
- by successful completion of final exam

Criteria-Performance will be satisfactory when:

- learner can define the components of a structured data object
- learner uses structure data types as input and output parameters
- learner can apply functions to return structured result values
- learner manipulates structure types to problem solve
- learner organizes data using parallel arrays and arrays of structure
- learner uses union types to interpret a data object in a variety of ways
- learner reviews common programming errors seen when using manipulated structure types

11. Compare the advantages and disadvantages of text and binary files.

Learning objectives

What you will learn as you master the competency:

a. Review input/output files.
b. Use binary files to store information.

c. Write a program that searches a database.

d. Review file programming errors.

Performance Standards

Competence will be demonstrated:

- by successful completion of required program
- by successful completion of final exam

Criteria-Performance will be satisfactory when:

- learner reviews input/output files
- learner uses binary files to store information
- learner can write a program that searches a database
- learner reviews file programming errors

12. Program large software systems.

Learning objectives

What you will learn as you master the competency:

a. Use abstraction to manage complexity.

b. Create a header file.

c. Create a library implementation file.

d. Review the five storage classes of C.

e. Modify functions for inclusion in a library.

f. Create conditional compilations.

g. Code arguments to function main.

h. Define macros with parameters.

i. Review common programming errors.

Performance Standards

Competence will be demonstrated:

- by successful completion of required program
- by successful completion of final exam

Criteria-Performance will be satisfactory when:

- learner uses abstraction to manage complexity
- learner creates a header file
- learner creates a library implementation file
- learner reviews the five storage classes of C
- learner modifies functions for inclusion in a library
- learner creates conditional compilations
- learner can code arguments to function main
- learner defines macros with parameters
- learner reviews common programming errors

13. Define dynamic data structures.

Learning objectives

What you will learn as you master the competency:

a. Review pointers and their uses.
b. Apply dynamic memory allocation.
c. Define linked lists.
d. Implement linked list operators using pointer variables.
e. Represent a stack data structure with a linked list.
f. Represent a queue with a linked list.
g. Maintain ordered lists.
h. Create a binary search tree.
i. Review common programming errors.

Performance Standards

Competence will be demonstrated:

- by successful completion of required program
- by successful completion of final exam

Criteria-Performance will be satisfactory when:

- learner reviews pointers and their uses
- learner applies the dynamic memory allocation
- learner defines linked lists
- learner can implement linked list operators using pointer variables
- learner represents a stack data structure with a linked list
- learner represents a queue with a linked list
- learner maintains an ordered list
- learner creates a binary search tree
- learner reviews common programming errors

14. Multiprocess using processes and threads.

Learning objectives

What you will learn as you master the competency:

a. Apply multitasking to operating systems.
b. Create processes.
c. Define interprocess communications and pipes.
d. Create a thread.
e. Use threads to accomplish more than one task at a time.
f. Review common programming errors with multithread programs.

Performance Standards

Competence will be demonstrated:

- by successful completion of required program
- by successful completion of final exam

Criteria-Performance will be satisfactory when:

- o learner applies multitasking to operating systems
- o learner creates processes
- o learner defines interprocess communications and pipes
- o learner creates a thread
- o learner uses threads to accomplish more than one task at a time
- o learner reviews common programming errors with multithreaded programs

Types of Instruction

Classroom Presentation
On-campus Laboratory

Grading Information

Grading Rationale

Post Test	10%
Quizzes	30%
Programming Assignments	60%

Grading Scale

A	90-100%
B	80-89%
C	70-79%
D	60-69%
F	0-59%