

PALOMAR COLLEGE
COURSE OUTLINE OF RECORD
FOR DEGREE CREDIT COURSE

(FORM VERSION 5/95)

(DATE REVISED 3/11/2000)

TRANSFER COURSE A.A. DEGREE APPLICABLE COURSE

COURSE NUMBER AND TITLE: CSIS 235 C for Programmers

UNIT VALUE: 4

MINIMUM NUMBER OF SEMESTER HOURS: 80 Hours

BASIC SKILL REQUIREMENTS: Appropriate language and computational skills.

ENTRANCE REQUIREMENTS:

Prerequisite: None
Corequisite: None
Recommended Preparation: None

SCOPE OF COURSE:

Intended for students with high-level programming language experience. An introduction to the C programming language, emphasizing top down design and principles of structured programming. Includes hands-on laboratory experience, reinforcing the lecture material. Language syntax is covered, together with variations in standard control structures, data structures, pointers, function declarations, and file input/output. The use of header files and processor directives will be covered. Development and maintenance function libraries will be included. *Not recommended for students who have completed CSIS 220 in C.*

SPECIFIC COURSE OBJECTIVES: The successful student will be able to:

1. Correctly utilize the Integrated Development Environment for the C compiler.
2. Develop appropriate problem-solving methodologies for designing algorithmic solutions to problems.
3. Create a program using the syntax and semantics of the C programming language.
4. Utilize top-down design and stepwise refinement techniques in the development of C programs.
5. Utilize appropriate control structures and data types.
6. Utilize appropriate user-defined data structures and abstract data types.
7. Utilize design principles of hierarchy, modularity, and procedural abstraction.
8. Determine an algorithms space and time complexity.

CONTENT IN TERMS OF SPECIFIC BODY OF KNOWLEDGE:

- I. **Introduction**
 - A. Course requirements
 - B. Policies and method of evaluation
 - C. C design goals
 - D. Number systems (decimal, binary, hexadecimal)
 - E. DOS operating system
 - F. Components of a computer system
- II. **Integrated Development Environment**
 - A. Editor
 - B. Compiler
 - C. Debugger
 - D. Help Facility
 - E. Linker
- III. **Beginning C**
 - A. C program structure
 - B. Declarations and data types
 - C. Header files
 - D. Function prototypes
 - E. Printf() and output formats
 - F. Mixing data types
 - G. Constants
- IV. **Operators**
 - A. Arithmetic
 - B. Logical
 - C. Increment and decrement
 - D. Assignment
 - E. Conditional
 - F. Comma
 - G. Precedence

- V. Statements and Control Flow**
 - A. Statements and blocks
 - B. if
 - C. for
 - D. while
 - E. do
 - F. switch
 - G. break
 - H. continue

- VI. Input / Output**
 - A. getchar()
 - B. putchar()
 - C. scanf()

- VII. Functions**
 - A. Return statement and values
 - B. Pass by value

- VIII. Arrays and Strings**
 - A. Declarations, arguments and subscripts
 - B. gets() and puts()
 - C. String functions and libraries
 - D. Quicksort and binary search
 - E. Two-dimensional arrays

- IX. File Manipulation**

- X. Storage Class**
 - A. auto
 - B. extern
 - C. static
 - D. register

- XI. Pointers**
 - A. Address of operator
 - B. Indirection operator
 - C. Pointer arithmetic
 - D. Simulate call by reference
 - E. Pointers and arrays
 - F. Pointers and functions
 - G. Pointers and strings
 - H. Command line arguments
 - i. Variable length arguments

- XII. C Preprocessor**
 - A. Macros
 - B. File inclusion
 - C. Constants
 - D. Conditional compilation

- XIII. Bitwise Logical Operators**
- XIV. Structures**
 - A. Definitions and syntax
 - B. Pointers to structures
- XV. Enumerated Types**
- XVI. Typedef**
- XVII. Dynamic Allocation**
 - A. malloc() and free()
 - B. Linear linked lists and stacks
- XVIII. Textmode Programming**
 - A. conio.h header file
 - B. memory
 - C. Segmented architecture
 - D. Accessing video memory
 - E. Window mechanics
- XIX. Separate Compilation**
- XX. Problem Solving Methodologies**
 - A. Stepwise refinement
 - B. Top-down design
 - C. Recursion
 - D. Divide and conquer
 - E. Backtracking
- XXI. Abstract Data Types**

REQUIRED READING:

Lafore, Robert. C Programming Using Turbo C++ . Indianapolis: Sams Publishing, 1998.

SUGGESTED READING: None.

REQUIRED WRITING:

Problem solving exercises are more appropriate. Students are required to complete five or six computer programming labs. Each programming lab will consist of a hands-on exercise applying theory principles learned in class. Programs must be well documented (at least one paragraph) in terms of their overall design goals. Additionally, each subprogram must be documented (two or three sentences) as to its purpose and overall performance.

OUTSIDE ASSIGNMENTS:

Students are expected to spend a minimum of three hours per unit per week in class and on outside assignments, prorated for short term classes.

There are written homework exercises within each section of each chapter which are assigned. In addition, numerous programming assignments are assigned.

INSTRUCTIONAL METHODOLOGY:

Check the following that apply:

- lecture
- laboratory
- lecture/laboratory combination
- directed study

This course may be offered as a distance education course and meets Title 5 regulations 55370, 55372, 55374, 553765 and 55378. Yes No

If yes, check all that apply:

- telecourse
- mediated instruction
- computer assisted instruction

GRADING POLICY AND STANDARDS (include methods for determining whether the stated objectives have been met by students):

Grades for courses are based upon final examinations, mid-term examinations, other tests, assignments, projects, and participation. Faculty will inform students of their grading policy at the beginning of each semester.

IS COURSE REPEATABLE FOR REASON(S) OTHER THAN DEFICIENT GRADE?

YES__ NO Number of times course may be taken for credit 1.

If yes, identify specific provision of Division 2 section(s) 55761-55763 and 58161 which qualifies course as repeatable:

CONTACT PERSON: Richard Stegman x 2769

SIGNATURES ON FILE