UNIVERSITY OF NEBRASKA AT OMAHA
COURSE SYLLABUS/DESCRIPTION

Department and Course Number	CSCI 2240
Course Title	Introduction to C Programming
Course Coordinator	Patrick Cavanaugh
Total Credits	3
Date of Last Revision	March 5, 2014

1.0 Course Description

1.1 Overview of content and purpose of the course (catalog description).
Programming in C in a UNIX® operating system environment; algorithm and program development and file manipulation using C; UNIX-like utility development.

1.2 For whom course is intended.
This course is designed primarily for individuals already possessing programming facility in another high-level language who want to learn how to program in C in the UNIX environment.

1.3 Prerequisites of the course (courses).
CSCI 1620 or equivalent experience with programming in a high-level language

1.4 Prerequisites of the course (topics)
- 1.4.1 Facility with a high-level programming language like Pascal, Modula, Java, or C++.
- 1.4.2 Ability to design and implement solutions to modest problems using assignment and flow control, procedures/subroutines/functions, scalars, arrays, records/structures, and simple input/output.

1.5 Unusual circumstances of the course.
None

2.0 Objectives

2.1 Be able to construct syntactically correct C programs using all language features.
2.2 Be able to recognize and correct syntax errors in C programs. This includes correct declarations and prototypes in both "K&R" C and ANSI C.
2.3 Be able to use primitive and complex data types and utilize/implement type conversion. This includes structures, typedefs, unions, enumerations and bit fields.
2.4 Be able to effectively use the operators in C.
2.5 Be able to explain the scope, lifetime, and initialization possibilities for all variable types in C. This includes the use of static to alter scope.
2.6 Be able to describe and the basic low-level UNIX input/output facilities, including at least the open, close, read, write, unlink, and lseek system calls.
2.7 Be able to explain and use the malloc and free functions to create and destroy simple dynamic data structures like variable-sized arrays and linked lists.
2.8 Be comfortable with standard Unix command line tools, make, man, nm, etc., as well as concepts of pipes and redirection in the shell and use of the preprocessor and linker.

3.0 Content and Organization

		Contact hours
3.1	Introduction	1.0
	3.1.1 History and characteristics of C	
	3.1.2 Creating, compiling, and running C programs on UNIX	
	3.1.3 Steps in C compilation	
	3.1.3.1 The preprocessor	
	3.1.3.2 The compiler	
	3.1.3.3 The assembler (optional)	
	3.1.3.4 The linker (linkage editor)	
	3.1.3.5 Typical compiler options	
	3.1.3.6 Locating information on-line	
3.2	Basics of the C language	3.0
	3.2.1 Fundamental data types	
	3.2.2 Variables and declarations	
	3.2.3 Numeric constants	
	3.2.4 Character and string constants	
	3.2.5 Characters as integers	
	3.2.6 Variable initialization	
	3.2.7 Arithmetic operators	
	3.2.8 Comparison operators	
	3.2.9 Logical operators	
	3.2.10 Bit-level operators	
	3.2.11 Order of precedence	
	3.2.12 Simple numeric input/output using scanf and printf	
3.3	Conditional execution	1.5
	3.3.1 The if statement	
	3.3.2 The ? operator	
	3.3.3 The switch statement	
3.4	Looping and iteration	2.0
	3.4.1 The for statement	
	3.4.2 The while statement	
	3.4.3 The do-while statement	
	3.4.4 The break and continue statements	
3.5	Arrays and strings	2.5
	3.5.1 Arrays with one dimension	
	3.5.2 Arrays with more than one dimension	
	3.5.3 Strings	
	3.5.4 Simple character input/output using getchar and putchar	
	3.5.5 Array and string initialization	
3.6	Functions	4.0
	3.6.1 Function definitions and prototypes	
	3.6.2 Parameters and arguments	
	3.6.3 Simulating call by reference for scalar parameters	
	3.6.4 Function return types, and void	
	3.6.5 The return statement	

3.7	Storage classes and scope	1.5
	3.7.1 The auto storage class	
	3.7.2 The static storage class	
	3.7.3 The register storage class	
	3.7.4 Local and global scope for variables and functions	
	3.7.5 External references	
	3.7.6 Recursion and storage allocation; the run-time stack	
	3.7.7 Typical errors	
3.8	Additional data types	4.0
	3.8.1 Structures	
	3.8.1.1 Declarations	
	3.8.1.2 Variable declaration	
	3.8.2 Structure initialization	
	3.8.3 Field alignment variations and portability	
	3.8.4 Unions	
	3.8.5 Enumerated types	
	3.8.6 Bit fields	
	3.8.7 Implementation of bit fields and portability	
	3.8.8 Arrays of structures	
	3.8.9 The typedef statement	
3.9	Pointers	5.0
	3.9.1 The address of a data item	
	3.9.2 Dereferencing a data item	
	3.9.3 Call by reference, revisited	
	3.9.4 Array names, array referencing and pointers	
	3.9.5 The name of a function and its address	
	3.9.6 The NULL pointer value	
	3.9.7 Initialization of pointer variables	
	3.9.8 Pointer comparison	
	3.9.9 Pointers to pointers	
3.10	Command line processing	3.0
	3.10.1 Command line parameters: argc and argc	
	3.10.2 UNIX expectations	
	3.10.3 Shell processing of command lines	
	3.10.4 I/O redirection	
	3.10.5 Pipes and pipelines	
3.11	The C preprocessor	2.0
	3.11.1 The include statement	
	3.11.2 Header files	
	3.11.3 Simple use of the define statement	
	3.11.4 Conditional compilation	
	3.11.4.1 The if statement	
	3.11.4.2 The endif statement	
	3.11.4.3 The else statement	
	3.11.4.4 The elseif statement	
	3.11.5 Macros (advanced use of define)	

3.14.2.1 opendir
3.14.2.2 closedir
3.14.2.3 readdir
3.14.3 Obtaining information about files
3.14.3.1 stat
3.14.3.2 fstat

3.15 String handling functions (string.h) 2.0
3.15.1 strcpy, strcat, strcmp, strlen
3.15.2 strchr, strstr, strtok, strspn, strpbrk, etc.
3.15.3 memset, memcmp, memcpy, memmove, memchr, etc.

3.16 Character classification functions (ctype.h) 0.5

3.17 Simple mathematical functions (math.h) 1.0
3.17.1 sqrt
3.17.2 exp, pow
3.17.3 fabs, abs
3.17.4 trigonometric functions
3.17.5 log, log10

3.18 Other standard library functions 1.0
3.18.1 malloc (review)
3.18.2 exit
3.18.3 qsort
3.18.4 rand/srand and random/srandom
3.18.5 atoi/atol/atof and pitfalls

4.0 Teaching Methodology

4.1 Methods to be used
This course is presented primarily through lectures, with on-line demonstrations.

4.2 Student role in the course
The student in this course will study the C programming language in depth, demonstrate understanding of the language by designing, writing and testing numerous programs, and take several quizzes and examinations.

4.3 Contact hours
Three hours per week

5.0 Evaluation

5.1 Type of student projects that will be the basis for evaluating student performance, specifying distinction between undergraduate and graduate, if applicable. For laboratory projects, specify the number of weeks spent on each project).

Students will write a significant number of C programs for the class, each of which is to be compiled and executed in a UNIX environment. All programs will use the C preprocessor, and an ANSI C-compiler (gcc is traditional); commands will be executed using a standard UNIX shell (e.g. sh, csh, ksh). Earlier programs should be completed in a week or less, while some programs later in the semester may require several weeks for completion. For some larger programs, the instructor may provide partially-complete solutions which are to be embellished or modified by the student. Each student will work independently.

5.2 Basis for determining the final grade

The major portion (typically 80 percent) of the student's grade will be determined by their success on the programming assignments. Programming assignments will be evaluated for correctness, readability, use of required features, and structure.

A few (three or four) short quizzes and a final examination will be given, and will be the basis for the remaining portion of the student's grade.

5.3 Grading scale and criteria

Points	Grade
97 – 100%	A+
93 – 96%	A
90 – 92%	A–
87 – 89%	B+
83 – 86%	B
80 – 82%	B–
77 – 79%	C+
73 – 76%	C
70 – 72%	C–
67 – 69%	D+
63 – 66%	D
60 – 62%	D–
0 – 59%	F

6.0 Resource Material

6.1 Textbooks and/or other required readings used in course

Kernighan and Ritchie, *The C Programming Language*, 2nd edition, Prentice Hall, 1988 (or a more recent text); although this is an "old" book, it is still one of the best available, especially for students who already know how to program in another high-level language (which is a prerequisite for the course).

6.2 Other suggested reading materials, if any

UNIX "man" and "info" pages for various commands, UNIX system calls, and library functions (in particular, gcc and standard C library functions); much of this material is covered in the Kernighan and Ritchie text, but students should become familiar with these resources for obtaining additional information.

6.3 Other sources of information

C is a widely-used programming language. There are numerous sites on the World Wide Web providing information on the language. A search for the literal phrase "C Programming" yielded over 300,000 hits.

6.4 Current bibliography of resource for student's information

Ammerall, *Programs and Data Structures in C*, John Wiley, 1987

Arnold and Peyton, *A C User's Guide to ANSI C*, Addison-Wesley, 1992

Deitel and Deitel, *C How to Program*, 3rd edition, Prentice Hall, 2001

Feuer, *The C Puzzle Book*, Prentice Hall, 1985

Harbison and Steele, *C: A Reference Manual*, 5th edition, Prentice Hall, 2002

Kelly and Pohl, *A Book on C: Programming in C*, 4th edition, Addison-Wesley, 1997

Kernighan and Ritchie, *The C Programming Language*, 2nd edition, Prentice Hall, 1988

Linden and Linden, *Expert C Programming*, Prentice Hall, 1994

Loudon, *Mastering Algorithms with C*, O'Reilly, 1999

Plauger, *The Standard C Library*, Prentice Hall, 1991

Press et al, *Numerical Recipes in C: The Art of Scientific Programming*, 2nd edition, Cambridge University Press, 1993

Reek, *Pointers on C*, Addison-Wesley, 1997

Stevens, *Advanced Programming in the UNIX Environment*, Addison-Wesley, 1992

Summit and Lafferty, *C Programming FAQs: Frequently Asked Questions*, Addison-Wesley, 1995

Tondo et al, *The C Answer Book*, 2nd edition, Prentice Hall, 1988

Vine, *C Programming for the Absolute Beginner*, Premier Press, 2002

7.0 Computer Science Accreditation Board (CSAB) Category Content (class time in hours)

CSAB Category	*Core*	*Advanced*
Data structures	5	
Computer organization and architecture		
Algorithms and software design	28	4
Concepts of programming languages	5	

8.0 Oral and Written Communications

Every student is required to submit at least __0___ written reports (not including exams, tests, quizzes, or commented programs) to typically _____ pages and to make ___0__ oral presentations of typically _____ minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

9.0 Social and Ethical Issues

No coverage

10.0 Theoretical content

Although the course does not cover any theoretical topics in depth, it does provide some theoretical of the following topics.

		Contact Hours
10.1	Data representation (in memory and on disk)	0.5
10.2	Explicit and implicit dynamic memory management	1.0

10.3 Recursion 1.0
10.4 Linked lists 1.0

11.0 Problem analysis

Students are expected to bring with them some of the analysis skills necessary for this course. Additional skills are obtained by working through the numerous assignments.

12.0 Solution design

The focal point of the course is the design and implementation of solutions to problems, primarily in the context of the C programming language and the UNIX operating system.

CHANGE HISTORY

Date	***Change***	***By whom***	***Comments***
6/22/2003	Cleanup, ABET-specific material	Wileman	
12/2/2008	Cleanup, ABET-specific material	Clark	
3/5/2014	Course Number Change, Coordinator	Cavanaugh	