Northern Arizona University • College of Engineering, Forestry, and Natural Sciences
School of Informatics, Computing, and Cyber Systems

Course Information

Catalog Description:	Intermediate programming using C including program design, algorithm design and data structures.
Broad Topics:	Algorithmic Design, C syntax, makefiles, headers, defensive programming, pointers, structures, lists, sorting, trees.
Prerequisites:	CS 122 or CS 128 or CS 126 or equivalent and MAT 136.
Co-requisites:	None
Skill Level:	Intermediate
Credit Hours:	3
Class Number:	LEC 8315 Section 002
Meeting Times:	MWF 10:20-11:10, Engineering Building 69, Room 106
Final Exam Time:	Monday, May 8, 2017, 10:00am - 12:00pm
Required Texts:	• Etter, "Engineering Problem Solving With C", 4th Ed. Pearson, 2013. ISBN **9780136085317**.
Recommended Texts:	• Kernighan and Ritchie, "The C Programming Language", 2nd Ed. Prentice Hall, 1988. ISBN **9780131103627**. • Shaw, "Learn C the Hard Way", Pearson, 2002. ISBN **9780131103627**. • Harbison and Steele, "C: A Reference Manual", 5th Ed. Pearson, 2016. ISBN **9780321884923**.
Web Page:	BbLearn (bblearn.nau.edu)

Instructor Information

Instructor:	Patrick Kelley, BSCS
Office Hours:	Engineering Bldg. Rm 243 M, W, F 8:00 - 10:00 Tu,Th 8:00 - 12:00 Tu,Th 1:00 - 3:00
Email:	Patrick.Kelley (at) nau.edu or webmaster (at) flion.com
Phone:	699-7455 (email preferred)
NAU Address:	Box 15600 Flagstaff, AZ 86011

Course Description

Beginning with a review of basic program design and programming style, we will explore writing programs in C. The student will learn to write structured programs, use makefiles to build applications, and use headers to organize their code. C language features such as pointers and structures will be covered and the end of the semester will introduce some basic data structures such as linked lists and trees, along with some common tools such as sorting and searching.

Course Objectives

By the end of the semester you should be able to:

- Develop an algorithmic solution to a problem.
- Translate an algorithm into C code.
- Use the C compiler and GNU debugger.
- Create an organized project using makefiles and header files.
- Use data types, pointers, and structures correctly.
- Produce working C code to implement linked lists, stacks, and trees.
- Implement sorts and searches in C code.

Specific learning outcomes are detailed at the end of this syllabus.

Coursework

The coursework includes the following assignments and tests:

- Homework Assignments

  Homework assignments involve research using book and online resources to answer specific questions. They help to fully prepare you for and familiarize you with the current lecture topics. Points are awarded for correct answers.

- Programming Projects

  Programming projects are where you put the knowledge you've gained into practice, transitioning from the theoretical to the practical with hands-on experience. Points are awarded based on the completeness and quality of your work and the thoroughness of your project report.

- 1 Midterm Exam
- 1 Final Exam

  The midterm and the final are an incentive for you to ensure you fully understand the topics being covered - as well as demonstrating that fact to the instructor. Points are awarded for correct answers.

Your class grade is based on the standard scale of points earned: 90%=A, 80%=B, 70%=C, 60%=D, below 60%=F. **No grades are curved or dropped**, though there are opportunities for extra credit. Projects are individual effort.

Assignments are due according to the posted schedule in Bb Learn.

Extra Credit

There will be a number of extra credit possibilities that involve doing extra work on assignments. Besides that, you can also get +2 points on any project by turning it in at least two days early (ex. If it is due midnight Friday, then it must be in by midnight Wednesday to qualify for extra credit). **Note that your project must fully work to receive the extra credit**.

Late Policy

Project assignments are accepted up to a week late at a pro-rated 40% point penalty. Other assignments are not accepted after the due date.

If you miss a test or know you *will* miss an assignment or test, discuss the matter with me as soon as possible. Preferably beforehand so we can make other arrangements.

Attendance

Regular attendance is expected. Don't be late, and don't leave until class is dismissed. Roll isn't taken, but each lecture you miss seriously jeopardizes your overall comprehension of the material and your chances to do well. Not to mention that this class will be blended, meaning that some of your project work will be done together during the class period.

Final Exam Policies

- If you score less than 50% on the final exam, your final class grade (which includes the final exam grade) will be reduced by one letter grade. This means that doing very poor on the final can severely hurt your class grade. If you had 480/480 points ("A") before the final and made 50/120 points on the final, your final class average would be 530/600 (88%) and your final grade would be "C".
- The final exam is mandatory to receive a passing grade in the course. Even if you are carrying an "A" before the final, skipping the final will result in a failing grade.

Lectures and the Book

The lecture topics follow the same general outline as the books. However, the lecture complements the book rather than being a mirror of it. If you *only* read the book or *only* pay attention to the lecture you're likely to end up missing some key concepts. To get the most from the class, read each chapter before we discuss the corresponding topic in the lecture, then use the lecture as an opportunity to reconsider the key points of the material and ask questions on anything you're confused on.

Plagiarism and Cheating

Grades are a way to motivate students and to evaluate students' mastery of a subject and their ability to get work done. The grades you get are not themselves truly important, but instead are representative of your knowledge, capabilities, and work ethic, and ***those*** are the things that matter. If you plagiarize source code, fabricate results, make fraudulent claims, or attempt to cheat in any way, you are misrepresenting yourself, your level of understanding, your capabilities, and your ability to accomplish things. It is dishonest and unethical.

Anyone who plagiarizes, copies, fabricates, or cheats will at the *least* receive a zero on that assignment or test.

Consulting with others and using their advice on projects is fine. However, the programs you submit should be your own work that you thoroughly understand and are entirely responsible for. Don’t share code, just ideas.

Bb Learn

Most assignments and handouts will only be available on Bb Learn - they will not be handed out in class. Any clarifications, corrections, and announcements will be posted on Bb Learn.

I expect every student to follow the class progress in Bb Learn. Assignments and projects will be handed in via Bb Learn and your grades will be posted there. Read the assignment policies at http://www.cefns.nau.edu/~pek7/Common/assignment.pdf to see how I expect work to be submitted. Failure to follow the guidelines will affect your grade.

University Policies

There are a number of university policies that govern your education and safety that all students should be aware of. These are:

- Safe Working and Learning Environment
- Students With Disabilities
- Accommodation of Religious Observance And Practice
- Institutional Review Board (And Use Of Human Subjects)
- Academic Dishonesty
- Medical Insurance Coverage For Students
- Classroom Management
- Evacuation Policies

You will find a complete description of each policy here:

http://www.cefns.nau.edu/~pek7/Common/policies.pdf

EXTENDED COURSE DESCRIPTION

1. Explanation of Prerequisites
   CS 122, CS 128, or CS 126
   - An ability to program alone and with a team.
   - An ability to solve small to medium sized problems with structured programming.
   - An ability to employ the software design process and software design tools to solve problems with computation.

   MAT 136
   - An ability to use applied mathematics and logical thinking.

2. Core Topics

   - Algorithmic Design
   - Programming Basics
     - C program structure
     - Programming Style
     - Makefiles
     - Debugging
     - Converting an algorithm to code
   - C Syntax
     - Data types
       - Basic types
       - Arrays
       - Strings
     - Operations
     - Control Structures
       - Conditional
       - Loops
       - Functions
     - C Features
       - Pointers
       - Allocating and Deallocating memory
       - Defined types
       - Structures
   - Data Structures
     - Linked Lists
     - Queues
     - Stacks
     - Trees
     - Sorts
     - Searches

3. Tools

   Tools used in this class include:

   - Linux (XSession from the lab computers or Remote Desktop)
   - GCC (Note: NOT G++; we are not using C++ in this class)
   - GDB
   - EMACS

   Note: It is beneficial to learn to work in a simple environment without an IDE. Thus, use of an IDE is discouraged and there will be no support for them, or other tools, in the classroom environment.

4. Learning Outcomes

L1. An ability to design and implement algorithms.

By the end of this course students should be able to:

a. Develop and test an algorithm for simple problems.
b. Implement and algorithm in code.
c. Test an implementation for correctness.

This outcome supports BSCS ABET Student Outcomes A, E and K.
This outcome supports BSACS ABET Student Outcomes A, E and K.
This outcome supports BSEE ABET Student Outcomes A, C, E and K.

L2. An ability to generate reports reflecting progress on assigned projects.

By the end of this course students should be able to:

a. Write and organize a report in a clear and engaging manner.
b. Describe in writing the project outcomes.

This outcome supports BSCS ABET Student Outcome G.
This outcome supports BSACS ABET Student Outcome G.
This outcome supports BSEE ABET Student Outcome G.

L3. An ability to create organized ANSI C projects

By the end of this course students should be able to:

a. Use makefiles to handle dependancies.
b. Use header files to assist code organization.
c. Use basic C data types correctly and appropriately.
d. Use C control structures correctly and appropriately.
e. Create and call functions correctly.
f. Compile and link multiple code files to a single executable.

This outcome supports BSCS ABET Student Outcomes E and K.
This outcome supports BSACS ABET Student Outcomes E and K.
This outcome supports BSEE ABET Student Outcomes E and K.

L4. An ability to test and debug C code

By the end of this course students should be able to:

a. Create unit test code for their C projects.
b. Use GDB to debug C code.

This outcome supports BSCS ABET Student Outcomes E and K.
This outcome supports BSACS ABET Student Outcomes E and K.
This outcome supports BSEE ABET Student Outcomes E and K.

L5. An ability to use advanced C features.

By the end of this course students should be able to:

a. Use C pointer notation to create and dereference pointers.
b. Use data type casts correctly and appropriately.
c. Use C structures and unions.
d. Allocate and deallocate memory.

This outcome supports BSCS ABET Student Outcomes E and K.
This outcome supports BSACS ABET Student Outcomes E and K.
This outcome supports BSEE ABET Student Outcomes E and K.

L6. An ability to implement data structures in C code.

By the end of this course students should be able to:

a. Create nodal data structures such as linked lists, trees, or matricies.
b. Implement binary search and list/tree traversal operations.
c. Implement stacks and queues.
d. Implement various sorting algorithms.

This outcome supports BSCS ABET Student Outcomes A, E and K.
This outcome supports BSACS ABET Student Outcomes A, E and K.
This outcome supports BSEE ABET Student Outcomes A, C, E and K.