C and Software Tools

Course Syllabus

CSC 230, Section 1
Spring, 2016

Meeting:
MW 11:45 - 1:00 pm
EB1 1007

Instructor:
David Sturgill, EB2 2294

Office Hours:
Mo 1:00 - 2:00 pm, EB2 2294
Tu 10:00 - 11:00 am, EB2 2294
We 1:00 - 2:00 pm, EB2 2294
Th 10:00 - 11:00 am, EB2 2294

Section 2 Instructor:
Sarah Heckman, EB2 2297

Teaching Assistants:
Shijie (Jerry) Li (sli41@ncsu.edu)
Arnab Saha (asaha@ncsu.edu)
Yiqiao (Joe) Xu (yxu35@ncsu.edu)

Office Hours at:
https://courses.ncsu.edu/csc230/common/OfficeHours.html

Text:

Course Description

In this course, students will develop skills in several important areas. First, we’ll get some experience working in C, a language that lets us think like procedural rather than an object-oriented developers. Also, being a fairly low-level language, C lets us see and control more of what’s going on in the hardware. This can help us think about using the hardware more effectively, whether we’re actually programming in C or in a higher-level language. While we learn C, we’ll also learn about tools and techniques that help us build, manage, debug and analyze software projects.

The computer science department captures the essential content of all undergraduate courses in the form of a list of course objectives. This helps to make sure that the most important material gets emphasized even if different instructors lead the course from semester to semester.

Course Objectives

By the end of the course, students should be able to do the following.
• Write small to medium C programs having several separately-compiled modules.

• Explain what happens to a program during preprocessing, lexical analysis, parsing, code generation, code optimization, linking, and execution, and identify errors that occur during each phase. In particular, they will be able to describe the differences in this process between C and Java.

• Correctly identify error messages and warnings from the preprocessor, compiler, and linker, and avoid them.

• Find and eliminate runtime errors using a combination of logic, language understanding, trace printout, and gdb or a similar command-line debugger.

• Interpret and explain data types, conversions between data types, and the possibility of overflow and underflow.

• Explain, inspect, and implement programs using structures such as enumerated types, unions, and constants and arithmetic, logical, relational, assignment, and bitwise operators.

• Trace and reason about variables and their scope in a single function, across multiple functions, and across multiple modules.

• Allocate and deallocate memory in C programs while avoiding memory leaks and dangling pointers. In particular, they will be able to implement dynamic arrays and singly-linked lists using allocated memory.

• Use the C preprocessor to control tracing of programs, compilation for different systems, and write simple macros.

• Write, debug, and modify programs using library utilities, including, but not limited to assert, the math library, the string library, random number generation, variable number of parameters, standard I/O, and file I/O.

• Use simple command-line tools to design, document, debug, and maintain their programs.

• Use an automatic packaging tool, such as make or ant, to distribute and maintain software that has multiple compilation units.

• Use a version control tools, such as subversion (svn) or Git, to track changes and do parallel development of software.

• Distinguish key elements of the syntax (what’s legal), semantics (what does it do), and pragmatics (how is it used) of a programming language.

**Grading**

Your final grade in this course will reflect your score on two preliminary exams, a comprehensive final exam, small, programming exercises, larger homework assignments and online quizzes. These will be combined with the following weights:

<table>
<thead>
<tr>
<th></th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online Quizzes</td>
<td>8</td>
</tr>
<tr>
<td>Programming Exercises</td>
<td>8</td>
</tr>
<tr>
<td>Homework Assignments</td>
<td>40</td>
</tr>
<tr>
<td>First In-Class Exam</td>
<td>12</td>
</tr>
<tr>
<td>Second In-Class Exam</td>
<td>12</td>
</tr>
<tr>
<td>Final Exam</td>
<td>20</td>
</tr>
</tbody>
</table>

Grades on quizzes, homework assignments and exams will be reported in the Moodle gradebook. Exercise grades will be reported through wolfware classic. Moodle will automatically compute a semester average,
but it won’t be perfect since it will be missing the exercise grade and Moodle doesn’t know our policy for dropping quiz grades. If you need to check your exact average, you’ll want to compute it yourself.

After your final average is computed, your letter grade is determined based on the following table. If your average is at least the value on the left, you are guaranteed a grade that’s at least as good as the one on the right. For example, if you end up with an average of 86, you will get at least a B. You may even get a B+ if, for example, your performance shows a trend of improvement through the semester or you’ve taken advantage of opportunities for extra credit.

<table>
<thead>
<tr>
<th>Minimum Score</th>
<th>Letter Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>97</td>
<td>A+</td>
</tr>
<tr>
<td>93</td>
<td>A</td>
</tr>
<tr>
<td>90</td>
<td>A-</td>
</tr>
<tr>
<td>87</td>
<td>B+</td>
</tr>
<tr>
<td>83</td>
<td>B</td>
</tr>
<tr>
<td>80</td>
<td>B-</td>
</tr>
<tr>
<td>77</td>
<td>C+</td>
</tr>
<tr>
<td>73</td>
<td>C</td>
</tr>
<tr>
<td>70</td>
<td>C-</td>
</tr>
<tr>
<td>67</td>
<td>D+</td>
</tr>
<tr>
<td>63</td>
<td>D</td>
</tr>
<tr>
<td>60</td>
<td>D-</td>
</tr>
</tbody>
</table>

Students auditing this class are required to take all the exams and to earn an exam average of 60 or higher. Students enrolled for credit only must earn a course average of C- or better across all components of the course (exams, projects and exercises) in order to receive a grade of S.

### Evaluation

#### Exams

We have three exams in this course, two preliminary exams and a final exam. The windows for taking the first exam is February 15 - 17 and the second is March 28 - 30. The window for the final exam is April 27 - 29. Material in this course builds from basic language elements to larger ideas and constructs. This affects the exams. The second exam is intended focus on material covered since the first exam, but a good understanding of all previous material is necessary to do well on this exam. The final exam is intended to be comprehensive.

All exams are closed book, but students are permitted to bring one 3 × 5 note card to each exam. You may use both sides of your card, but your card must be hand written by you, with your name on it. Note cards will be collected after each exam.

#### Quizzes and Exercises

In this class, you get to earn some points toward your final grade through a number of online quizzes. Quizzes will be taken online through the course moodle page.

Exercises will give you a chance to write or fill in missing parts of short programs. They will generally be due the on Sunday evening, after they are assigned. This will include an exercise deadline of April 24, during the last week of class.

---

1Why all these peculiar rules about the note card? Well, if you’re going to have a note card during the exam, I want preparing that card to be one more chance to learn the material. If you’ve done a good job preparing your own note card, you may not even need to look at it during the exam.
Hopefully, the quizzes will give you a good reason to keep up with the readings and the lectures and to pay attention. The exercises should give you an opportunity to practice the programming language concepts as we learn them, instead of just trying to apply concepts when you’re working on a homework.

I understand that students sometimes have to miss class, or forget about an exercise. You’ll get to drop your three lowest quiz grades and your three lowest exercise grades. You’ll also get to drop 5 percent of the remaining grade from each of these categories. This is our mechanism for handling any missed quizzes and exercises; it will let you miss a few and do poorly on a few more without any penalty to your final grade.

What does it mean to drop 5 percent of your quiz grade? After dropping the three lowest quiz grades, I drop another 5 percent by adding up all the points available on the quizzes that remain. I reduce this by 5 percent and call it your adjusted maximum, $m$. Then, I add up all the points you earned on quizzes, clamp that to $[0, m]$ range and divide by $m$ (so, your quiz average won’t exceed 100 percent, even if your total exceeds the adjusted maximum). Your exercise average is computed the same way.

Homework Assignments

In addition to the frequent, small programming exercises, you’ll get to complete six homework assignments. These will give you a chance to use what you’re learning to solve larger, more interesting problems. As these homeworks are assigned, they will be posted to the course homepage. Electronic submission of homeworks will normally be due at 11:00 on their due date, with the last project due April 20.

If you have a conflict on the due date, just plan to complete and submit your homework early. Since homework submission is electronic, you can even submit your work while you’re away if you have to travel. If a documented emergency (e.g., hospitalization) prevents an assignment from being submitted, that assignment will simply be dropped from the student’s homework average.

Homework assignments may vary in points. For example, an easy homework may be worth 70 points and a more difficult one may be worth 120. To compute your homework average, just add up all the points you earned and divide by the total number of points available. So, homeworks that are worth more points will carry more weight in the average.

Late Submission

If you miss the submission deadline for an assignment, you can submit it late for a 20 percent penalty. Late homework submissions will normally be accepted for 24 hours after the regular submission deadline.

If you make a homework submission before the due date, then, after the due date, you realize that you’ve made a mistake, you can still make a late submission if it’s within 24 hours of the deadline. Of course, you’ll need to decide whether it’s better to lose points for a mistake on an on-time submission or to lose 20 percent for a perfect but late submission. In general, you’ll earn more points with a correct-but-late submission than you will with a broken-but-on-time submission.

Grade Appeals

On exams and programming projects, the instructor establishes guidelines for grading, and grading responsibility is shared with Teaching Assistants and graders. Instructors supervise the teaching assistants and spot check some of their work. If you believe an error has been made on a homework assignment, write up a short description of your case and send it to the TA responsible for that part of the project (email is fine). The teaching assistant will consider your appeal and adjust your grade if necessary. If you are not satisfied with the decision made by the TA, then take the appeal to the instructor, who will make the final decision.

For exams, regrade requests should go directly to the instructor.

Students have one week from when project or exam is returned to begin an appeal. These limits obviously can’t apply to any assignment or exam graded within a week of the end of the term. For those assignments, appeals must be made before 11:00 am on May 4.
Reading Assignments

Regular reading assignments are included in the schedule of topics on the course homepage. The reading will support much of the class discussion, and keeping up with the readings can help you to do well on exercises and homeworks.

Programming Guidelines

You will get a chance to do some C programming on exercises and homework assignments. Style requirements for the exercises are not strict; unless it’s specifically part of the problem you’re solving, you can generally format your code and use any variable names you’d like.

Style requirements for the programming projects are strict, very strict. The course homepage includes a document describing the style guidelines, along with some tips about common problems students have. Why are the requirements so strict? In the workplace, it’s common to have a set of style guidelines that all developers are expected to follow. In this class, we try to help students get used to this idea.

Programming assignments are expected to compile and execute on what we’re calling our common platform. This is an installation of Linux that’s available to all students. The course homepage includes a description of this platform, along with some help with developing and testing your code on this platform.

Class Membership and Participation

Class Communication

We will use Piazza for out-of-class discussions. A link is provided on the course homepage. Students are welcome and encouraged to post questions and are requested to help answering other students’ questions. There are two requirements for all posts to the message board.

- The post must be courteous to and respectful of students, staff, and the University community.
- Students must consider the academic integrity expectations and the difference between answering a question and providing a solution. It’s OK to help out by explaining a problem you had, posting sample input and output, or pointing out a tricky special case. It’s not OK to post your code when you’re asking for help, and it’s not OK to post a working solution (even in part) when you’re trying to help someone else out.

In addition to the course homepage in Moodle and the Piazza message board, there is a mail alias for the course. This will used to provide time-sensitive and important information, such as a change in the due date for a homework assignment. You will want to make sure Moodle is using an email address that you check regularly.

Dissemination of Information

The course website in Moodle is the primary distribution medium for this course. Assignments, lecture slides, many examples, study guides and other materials will be available from this site.

Academic Integrity and Attendance

In short, you are to do your own work. Academic integrity can sometimes be a problem, so this document gives it some extra attention to help people stay out of trouble.

You may only work on an assignment with another student if the assignment explicitly states that this is permitted. In completing your assignments, you can talk to the instructor or the teaching assistants to get specific help. You can use materials posted to the course homepage on Moodle and examples from your
book. You can talk to anyone about some general techniques for solving a problem, but you should never share copies of source code or other parts of your work with another student.

Students are expected to maintain high standards of academic integrity and honesty. Normally, a case of cheating will result in a grade of zero for an assignment. A major offense, including any violation on a test, could result in failure of the course.

Suspected violations of academic integrity will be reported to the Office of Student Conduct. This benefits the student in that it provides an opportunity for independent review of the evidence. It also benefits the university by maintaining a centralized record of violations.

Examples of Cheating (this list is NOT exhaustive):

• to give any student access to any of your work which you have completed for individual class assignments.

• It is cheating AND plagiarism to use another person’s work and claim it as your own. You are expected to complete all assignments on your own, unless otherwise specified in the assignment.

• to interfere with another student’s use of computing resources or to circumvent system security.

• to email, ftp, post on the Internet, bulletin boards, message boards, etc. your work for others to obtain. Do NOT use sites that allow you to anonymously post code. Those sites are searchable, and others may find your code (like the teaching staff).

• to ask or pay another person or persons to complete an assignment for you.

• It is cheating AND plagiarism to decompile any compiled code and use the decompiled source code as your own. You may also break the law by decompiling code.

• It is cheating AND plagiarism to use code that you find online.

• to give another student access to your account (NC State account or others that you use for university work) or to give them your account password.

• for you and another student to work collaboratively on an assignment, unless otherwise specified by the assignment.

• to circumvent the intention of the assignment and/or the automated grading system (e.g., by hardcoding test case solutions).

Examples of NOT Cheating (this list is NOT exhaustive):

• Using the code from the class website (with citations in the comments).

• Using code from other programs YOU wrote.

• Help from TAs or instructor (with citations in the comments).

• Using code from the textbook or textbook website (with citations in the comments).

Example Citations

/* (In file or function level comments)
 * I received help from the TA, Martha Washington on DATE during her office hours.
 * We discussed X.
 */

/*
* The code for this method is based on Exercise Y that I completed on date Z.
*/

Protecting Yourself

- Do not leave papers lying around your workstation.
- Do not dispose of important papers in the lab recycling bins and trash cans until after the assignment is graded.
- Do not give out your password.
- Do not leave your workstation unattended or forget to log yourself out.
- Do not leave your laptop unattended.
- Do not give other students access to any of your workspace or email them any code.
- Do not give other students access to your course materials or your personal computer.
- Do not email, ftp, or post your code on the Internet, message boards, etc.
- Keep all copies of final an intermediate work until after the assignment is graded.
- Keep all graded assignments until after you receive the final grade for the course.
- Do not discuss implementation details of the assignment with your peers.

Semester Schedule

The following table gives the planned schedule of lecture topics, readings, exams and other important events for the semester. We’ll try to stick to this schedule, especially for the planned exams. As a student in the online class, you have some flexibility in when you view the lectures. This will give you some guidelines for when you should view them if you want to be prepared for the exams and assignments. If weather requires the on-campus sections us to miss a lecture or two, or if we slip the deadline on a homework assignments, some adjustment may be required.

<table>
<thead>
<tr>
<th>Date</th>
<th>Events</th>
</tr>
</thead>
</table>
| 1/6/2016   | Lecture 01, Getting Started  
Reading: King, pages 1 - 8 (Chapter 1, Introducing C)  
Reading: King, pages 9 - 32 (Chapter 2, C Fundamentals) |
| 1/11/2016  | Lecture 02, IO, Compilation and Parsing  
Reading: King, pages 37 - 48 (Chapter 3, Formatted Input/Output) |
| 1/13/2016  | Lecture 03, Revision Control and Build Automation  
**Homework 1 Due** |
| 1/20/2016  | Lecture 04, Variables, Expressions and Types  
Reading: King, pages 53 - 68 (Chapter 4, Expressions)  
Reading: King, pages 73 - 93 (Chapter 5, Selection Statements) |
| 1/25/2016  | Lecture 05, Scanf and Type Conversion  
Reading: King, pages 99 - 120 (Chapter 6, Loops)  
Reading: King, pages 125 - 148 (Start of Chapter 7, Basic Types) |
| 1/27/2016  | Lecture 06, Program Structure  
Reading: King, pages 219 - 237 (Chapter 10, Program Organization)  
Reading: King, pages 457 - 479 (Chapter 18, Declarations) |
<table>
<thead>
<tr>
<th>Date</th>
<th>Lecture</th>
<th>Reading 1</th>
<th>Reading 2</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/1/2016</td>
<td>Lecture 07, Arrays</td>
<td>Reading: King, pages 161 - 177 (Chapter 8, Arrays)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/3/2016</td>
<td>Lecture 08, Pointers (part 1)</td>
<td>Reading: King, pages 241 - 255 (Chapter 11, Pointers)</td>
<td></td>
<td>Homework 2 Due</td>
</tr>
<tr>
<td>2/8/2016</td>
<td>Lecture 09, File I/O and Functions</td>
<td>Reading: King, pages 183 - 214 (Chapter 9, Functions)</td>
<td>Reading: King, pages 539 - 545 and 551 - 564 (Parts of Chapter 22, Input/Output)</td>
<td></td>
</tr>
<tr>
<td>2/10/2016</td>
<td>Lecture 10, Pointers (part 2)</td>
<td>Reading: King, pages 257 - 273 (Chapter 12, Pointers and Arrays)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/15/2016 - 2/17/2016</td>
<td>Exam 1 (Lectures 01 - 09)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/17/2016</td>
<td>Lecture 11, Strings</td>
<td>Reading: King, pages 277 - 308 (Chapter 13, Strings)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/22/2016</td>
<td>Lecture 12, Pointers (part 3)</td>
<td>Reading: King, pages 438 - 443 (Part of Chapter 17, Advanced Uses of Pointers)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/24/2016</td>
<td>Lecture 13, Dynamic Memory Allocation</td>
<td>Reading: King, pages 413 - 443 (Chapter 17, Advanced Uses of Pointers)</td>
<td></td>
<td>Homework 3 Due</td>
</tr>
<tr>
<td>2/29/2016</td>
<td>Lecture 14, Debugging</td>
<td>Reading: King, pages 627 - 631 (Part of Chapter 24, Error Handling)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3/2/2016</td>
<td>Lecture 15, Structs</td>
<td>Reading: King, pages 377 - 407 (Chapter 16, Structures, Unions and Enumerations)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3/14/2016</td>
<td>Lecture 16, Data Structures</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3/16/2016</td>
<td>Lecture 17, Bitwise Operations</td>
<td>Reading: King, pages 509 - 515 (Start of Chapter 20, Low-Level Programming)</td>
<td></td>
<td>Homework 4 Due</td>
</tr>
<tr>
<td>3/30/2016</td>
<td>Lecture 20, More Data Structures and Object Orientation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/4/2016</td>
<td>Lecture 21, Security</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/6/2016</td>
<td>Lecture 22, The Rest of C</td>
<td>Reading: King, pages 589 - 623 (Chapter 23, Library Support for Numbers and Character Data)</td>
<td>Reading: King, pages 682 - 692 (Part of Chapter 26, Miscellaneous Library Functions)</td>
<td>Homework 5 Due</td>
</tr>
<tr>
<td>4/11/2016</td>
<td>Lecture 23, Performance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/13/2016</td>
<td>Lecture 24, C++ (part 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/18/2016</td>
<td>Lecture 25, C++ (part 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/20/2016</td>
<td>Lecture 26, C++ (part 3)</td>
<td></td>
<td></td>
<td>Homework 6 Due</td>
</tr>
<tr>
<td>4/25/2016</td>
<td>Lecture 27, C++ (part 4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/27/2016 - 4/29/2016</td>
<td>Final Exam</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Course Materials Acknowledgment

This course features materials provided by other instructors at NC State. Thanks to Dr. Sarah Heckman, Barry Peddycord and Dr. Douglas Reeves for generously providing their materials and assistance and helping to make this course possible.