CS 210 - Fundamentals of Programming I  
Spring 2015 - Syllabus

Instructor
Dr. Deborah Hwang  
KC-264, 488-2193, hwang@evansville.edu  
Home page: http://csserver.evansville.edu/~hwang

Office Hours: See instructor's home page.

Class Hours:
Section 01: TT, 2:00pm - 3:50pm, KC-267  
Section 02: MW, 2:00pm - 3:50pm, KC-267

Course Home Page
Announcements regarding handouts and assignments will be made in class. Assignments will be available only at the course home page (http://csserver.evansville.edu/~hwang/s15-courses/cs210.html). It is your responsibility to consult the course home page on a regular basis. Grades will be posted to Blackboard (http://bblearn.evansville.edu/).

Catalog Data
Emphasizes problem-solving techniques used in the analysis and design of software solutions, including structured top-down design, abstraction, good programming style, debugging, and testing. Programming constructs covered include control structures, functions, and basic and aggregate data types. Introduction to recursion and dynamic allocation.

Objectives
Learning problem solving techniques used in programming software solutions including structured design, good programming style, testing strategies, and debugging strategies. Note that you will be learning and using the C language as a vehicle towards these goals, but this course is not about learning C in its entirety. More specifically:

- Students will be able to write programs using selection, repetition, functions, and dynamic allocation in one major high level language.  
- Students will be able to write programs using the aggregate data types arrays, strings, and structs.  
- Students will be introduced to the fundamentals of functional decomposition and design.  
- Students will be introduced to testing and debugging strategies.  
- Students will be able to construct software solutions that use structured analysis and design with good programming style.  
- Students will be proficient using a Microsoft Windows IDE, such as Visual Studio or Code Blocks, to implement and debug programs.  
- Students will be introduced to contemporary professional issues

Prerequisites: None.
**Required Textbook**

**Daily Requirements**
You are expected to read the assigned chapter(s) from the textbook. There is not enough time during lecture to cover all the material in the depth necessary for mastery. You are responsible for all of the material in assigned chapters even if it is not explicitly covered in lecture, unless otherwise noted.

Most classes will have in-class exercises on material up to and including the reading assigned for the day. These exercises may be written and/or programming exercises, and may be individual or group exercises. Generally, these exercises are due by the end of the class period.

**Homework and Programming Assignments**
There will be weekly written homework exercises. Generally homework exercises will be assigned on Monday/Tuesday and due the following Monday/Tuesday. There will be programming assignments of varying lengths and difficulty. Generally, they will be assigned on Wednesday/Thursday and due the following Wednesday/Thursday with focus on a single topic. At least two larger assignments will be cumulative in nature and about two weeks in duration worth twice a much as a regular programming assignment.

**Exams and Evaluation**
There will be two in-class practical programming exams during the term, one in-class written midterm exam, and a comprehensive written final exam. The exams tentatively are scheduled as shown in the class schedule. The scheduled final exam for Section 01 is on . The scheduled final exam for Section 02 is on

Final grades will be based on the following weighted distribution:

<table>
<thead>
<tr>
<th>Weight</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20%</td>
<td>Two in-class practical exams (5% and 15%, respectively)</td>
</tr>
<tr>
<td>15%</td>
<td>Written midterm exam</td>
</tr>
<tr>
<td>20%</td>
<td>Comprehensive written final exam</td>
</tr>
<tr>
<td>10%</td>
<td>In-class exercises and homework (weighted as indicated)</td>
</tr>
<tr>
<td>35%</td>
<td>Programming assignments (weighted as indicated)</td>
</tr>
</tbody>
</table>

**Students must pass the second practical exam with a grade of 70% or better in order to earn a final grade of C- or better for the course, regardless of performance on other assignments.** Students who do not pass the second practical exam will be given an opportunity to take it again with a 10% penalty. Final grades are based on the final weighted percentage with adjustments depending on class distribution. Historically, the final grade A/B line falls around 88% ± 2% with subsequent grade levels every 10%.

**Grading of Programming Assignments and Programming Projects**
Programming assignments and programming projects grading will be divided as follows:

<table>
<thead>
<tr>
<th>Weight</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15%</td>
<td>Analysis and design - appropriate structure and modularity as specified by the assignment and indicated by appropriate commenting</td>
</tr>
<tr>
<td>70%</td>
<td>Implementation - correct results, appropriate error checking</td>
</tr>
<tr>
<td>15%</td>
<td>Coding format guidelines</td>
</tr>
</tbody>
</table>
The analysis is be included as comment blocks before each function. The design is indicated by in-lined comments. Programs are required to be formatted and commented as described in handout A C Programming Style Guideline. This style is substantially the same as the one in the textbook.

All programs will be submitted electronically for testing using the submission system at http://submission.evansville.edu. Programs may be submitted multiple times with the final submission being the graded submission. Ignore the score! The results of the submission system testing is only part of an assignment's grade. Also, the late penalties are computed differently than given below. (Dr. Roberts wrote the submission system, so it uses his late penalty scheme, which is different than that given below.)

Late Homework, Late Programming Assignments
Homework and programming assignments are due at the instructor's office and/or electronically submitted as appropriate by 4:30pm on the date specified unless otherwise noted. Any assignments arriving after 4:30pm are considered late. The following automatic late penalties will be applied:

- 10% if handed in by 4:30pm, one day late
- 20% if handed in by 4:30pm, two days late
- 30% if handed in by 4:30pm, three days late

Unexcused late work will not be accepted for credit after three days after the due date without prior arrangements. For the purpose of counting days, Friday 4:30pm to Monday 4:30pm is considered one day. (For example, an assignment due on Wednesday/Thursday may be submitted for late credit until Monday/Tuesday.) Please note that the purpose of the automatic late extension is to allow students leeway when needed. It is usually better to hand in something late and completed than on-time and incorrect. However, chronically handing in late submissions will lower your final grade.

Valid excuses for missing exams, missing classes, and handing assignments in late include illness, family emergencies, religious observances, official UE events such as varsity games and concerts, etc. They do not include (most) work conflicts, studying for other classes, leaving a day early or staying home an extra day over a weekend or holiday, etc. In general, an excused absence is one caused by circumstances beyond your control.

The instructor will rely on your integrity for getting work excused. If you have a valid excuse, send an email to the instructor with the details. However, the instructor reserves the right to request written documentation in cases of chronic absences. For religious observances and official UE events, you must inform the instructor that you will be absent before the absence occurs, otherwise it will be considered an unexcused absence.

Excused work must be made up within one calendar week from the original due date for full credit. Late excused work will not be accepted. Exceptions will be made for serious or prolonged illness, or other serious problems. Please note: It is your responsibility to take care of missed or late work.

Attendance Policy - Missed Classes and In-class Exercises
Due to the integrated lecture and lab format of the course, attendance is mandatory. Graded in-class exercises may not be made up, regardless of reason for an absence. However, excused absences will be noted and taken into consideration when assigning final grades. Students are responsible for all material covered in class. If you miss a class, find out what was covered from another student. You are responsible for checking the course home page for new assignments even if you miss class.
In addition, many classes will end with more than adequate time to complete any in-class exercises. The rest of the time is for you to work on the currently assigned programming assignment. **You are expected to use the time to the end of the scheduled class period to work on the assignment.** The only valid reason for leaving early is if you have already completed the outstanding assignment. During this time the instructor is available to answer your questions regarding the program and its design. This is incorporated into the design of this course and if you do not take advantage of this, several key concepts may be missed (and your grade will probably end up reflecting this).

**Credit Hour Policy**
This course meets the federal requirements of 45-75 total hours of student work (combined classroom plus out-of-class work) per credit hour.

**Disability Policy**
It is the policy and practice of the University of Evansville to make reasonable accommodations for students with properly documented disabilities. Students should contact the Office of Counseling and Health Education at 812-488-2663 to seek services or accommodations for disabilities. Written notification to the instructor from the Office of Counseling and Health Education is required for academic accommodations.

**Honor Code**
All students are expected to adhere to the University's Honor Code regarding receiving and giving unauthorized assistance. Specific guidelines in force for this course are:

- Written homework and in-class exercises are for you to gain experience and practice. You may collaborate with your classmates, but each student should submit a solution in his/her own words that reflect his/her understanding of the solution. Ultimately you will be required to demonstrate your proficiency of the material on exams. Therefore, it is highly recommended that you attempt all homework and in-class problems on your own before finding a solution from another source.

- Programming assignments (including analysis and design) are to be your own work unless otherwise noted. Discussing the meaning and general solution techniques of an assignment with other students is permitted. For example, discussing "How is this assignment similar or different from problems presented in the text or in lecture?" is acceptable.

Asking another person for assistance on specific items in your own analysis and design or code is also permitted, but **you may not observe another person's solution or code in its entirety for the purposes of studying or copying it, with or without that student's permission.** Examples of acceptable assistance include:

- "The compiler is complaining about a missing semicolon. I can't see where I'm missing one. Could you look at my code and see if you can find where it is?" "Here is it. Remember, it's usually the line above the error that's missing the semicolon."

- "What's the design step for loops again? I want a loop that accesses each element of an array." "The format is 'For each element in the items array ...'."

- "What's the C code for array accessing loops again? I want a loop that accesses each element." "The code is 'for (index = 0; index < n; index++)'."
"My function doesn't compute the correct answer. It seems to be going through the loop one too many times. Could you look at this?" "Your loop condition is not quite correct; it should be \( i < n \), not \( i \leq n \). Remember, array indexes start at 0 and end at \( n - 1 \).

Note in the last case, the assistance is acceptable because the requester has asked about something specific. Examples of unacceptable assistance include:

- "How did you write the Sort function?" "I passed an array and a size, and wrote nested for loops..."
- "Can I look at your analysis and design (or program) and see what you did?", "Sure..."
- "Hey, here's Jane Student's CS 210 program listing. Let's look at her program..."
- "Hey, Joe Student left himself logged into this machine. Let's look at his program..."

Note in the last two cases that just looking at another student's code, even if he/she left it unprotected, is not permitted.

- Solutions shall not be copied from Internet or other sources. Discovery of such will result in a 0 for that assignment for all parties involved for the first offense. A second offense will result in failure of the course. A third offense will result in dismissal from the program. Please note, each instance will recorded and that this penalty is cumulative throughout your career at UE.

- **Any** attempt to compromise the automatic testing system will result in immediate failure of the course.

If there is any doubt as to whether assistance is acceptable, consult the instructor.
# Reading Schedule

This is a tentative schedule of topics for this course. You are expected to have read the assigned material before coming to class.

<table>
<thead>
<tr>
<th>Week of</th>
<th>Monday / Tuesday</th>
<th>Wednesday / Thursday</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/12</td>
<td>Chapter 1 &amp; 2: Introduction, Overview of C</td>
<td>Chapter 2 Overview of C</td>
<td></td>
</tr>
<tr>
<td>01/19</td>
<td>MLK, JR. DAY NO CLASS</td>
<td>CLASS CANCELLED</td>
<td>Chapter 3: Functions</td>
</tr>
<tr>
<td>01/26</td>
<td>Chapter 3: More Functions</td>
<td>Chapter 4: Selection</td>
<td></td>
</tr>
<tr>
<td>02/02</td>
<td>Chapter 5: Repetition, aka Loops</td>
<td>Chapter 5: More Repetition</td>
<td></td>
</tr>
<tr>
<td>02/09</td>
<td>Chapter 5: Iterative Approximation</td>
<td>Practical Exam 1</td>
<td></td>
</tr>
<tr>
<td>02/16</td>
<td>Chapter 6: Pointers, Reference Parameters</td>
<td>Chapter 11: File pointers, Textfiles</td>
<td></td>
</tr>
<tr>
<td>02/23</td>
<td>Chapter 7: Arrays</td>
<td>Chapter 7: More Arrays</td>
<td></td>
</tr>
<tr>
<td>03/02</td>
<td>Written Midterm Exam 1 Review</td>
<td>Written Midterm Exam 1</td>
<td></td>
</tr>
</tbody>
</table>
| 03/09   | SPRING BREAK – NO CLASSES | | |}
| 03/16   | Chapter 8: Strings | Chapter 8: More Strings |     |
| 03/23   | Chapter 7: Multi-dimensional Arrays | Chapter 10: Structs |     |
| 03/30   | Practical Exam 2 | Chapter 12: Separate Compilation | EASTER BREAK |
| 04/06   | Chapter 13: Dynamic Allocation | Chapter 13: Dynamic Arrays |     |
| 04/13   | Chapter 9: Recursion | Chapter 9: More Recursion |     |
| 04/20   | Parallel Computing | Parallel Computing |     |
| 04/27   | Course Wrap-up Final Exam Review | Reading/Study Day |     |
| 05/04   | Final Exam, Sec. 02, 2pm-4pm | Final Exam, Sec. 01, 12pm-2pm |     |