CS 215 - Fundamentals of Programming II
Fall 2019 - Syllabus

Instructor

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

Office Hours: See instructor's home page.

Course Home Page

Handouts and assignments will be available only at the course home page (https://csserver.evansville.edu/~hwang/f19-courses/cs215.html). Although announcements regarding handouts and assignments will be made in class, 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 Description

Project and problem-solving course emphasizing the use of classes for encapsulation of abstract data types and abstract data structures. Topics include classes, templates, dynamic allocation, searching and sorting, recursion, and exception handling.

Objectives and Outcomes

The main objective of this course is to continue the study of problem solving techniques used in programming software solutions with emphasis on abstract data types, and to become proficient in the use of the UNIX operating system and development tools. Specific outcomes for this course include:

- Students will be able to write programs using selection and repetition constructs, functions, and arrays.
- Students will be able to use dynamic allocation and recursion to solve problems.
- Students will be able to use design and implement abstract data types (ADTs) using simpler data structures such as multi-dimensional arrays and vectors, and use these ADTs to solve problems.
- Students will be able to design and implement abstract data types (ADTs) using linked data structures such as linked lists and binary trees, and use these ADTs to solve problems.
- Students will be able to implement several sorting algorithms.
- Students will be introduced to rudimentary algorithm analysis.
- Students will be able to do basic generic programming and use library structures such as stacks and queues to solve problems.
- Students will complete at least 4 assignments requiring design of an application using one or more data structures, testing, and debugging.
- Students will be proficient using the UNIX operating system.
- Students will be proficient using a UNIX editor, such as emacs or vim.
- Students will be proficient using relevant GNU tools for software development, including g++ (compiler) and make (program build utility).
- Students will be introduced to contemporary professional issues.

Prerequisites: Grade of C- or better in CS 210

Required Textbook

Wittenberg, *Data Structures and Algorithms in C++*, Mercury Learning and Information, LLC, 2018, ISBN: 978-1-68392-084-7.

On-line references

Available at https://csserver.evansville.edu/~hwang/f19-courses/cs215/references.html

Daily Requirements

Assigned daily reading assignments. In-class exercises, and homework assignments as needed. Homework assignments may include both written exercises and short programming problems.

Programming Projects

There will be 7-8 programming projects of 1-2 weeks in duration each. See handout *A C++ Programming Guideline for CS 215* for appropriate code format used in this course.

Programming projects will be graded using the following criteria with the weights as shown.

65%	Correct results, including command line arguments and file I/O
10%	Error checking, including proper use of exceptions
25%	Style, observed coding guidelines, originality, makefile

Programming projects must be submitted electronically as explained in the handout *Using GitKeeper for Submitting Assignments*.

Exams and Evaluation

There will be one in-class **written** midterm exam, two 2-hour practical **programming** exams, and a (2-hour) comprehensive **written** final exam, tentatively scheduled as shown below. The purpose of the practical **programming** exams is to demonstrate mastery in using the C++ programming language and the UNIX environment. **Therefore, it is necessary to score a minimum of 70% on the second practical programming exam to pass the course (grade of C- or better) regardless of performance on other assignments.** Students who fail to do so and are otherwise passing the course will be allowed an opportunity to substitute a makeup second practical programming exam at the end of the term with a 10% penalty.

Final grades will be based on the following weighted distribution with some adjustments depending on class distribution. Historically, the A/B line falls around 88% +/- 2% with subsequent grade levels every 10%.

20%	2-hour comprehensive written final exam
15%	In-class written midterm exam
20%	Two 2-hour practical programming exams (5% and 15%, respectively)
5%	Homework and in-class exercises (weighted as indicated in assignment)
40%	Programming projects (weighted as indicated in assignment)

Graded work scores will be posted to BlackBoard. The total weighted score reported is the weighted percentage of the currently graded work based on the assignments for the **entire** course. Thus as the course proceeds and

more work is graded, the maximum weight percentage available increases until it is 100% after the final exam. After each exam is graded, the instructor will announce the current maximum weighted percentage available.

Missed Classes, Missed Exams, Late Homework, Late Projects

There are no makeups for in-class exercises, if any. Missing in-class work due to excused absences will be taken in consideration when assigning final grades.

Written homework is due at the instructor's office **by 4:30pm** on the date specified unless otherwise noted. Any assignments arriving after 4:30pm are considered late. **Programming homework and projects** must be submitted electronically by 11:59pm on the date specified. The following automatic late penalties will be applied:

10%	if handed in or submitted one day late
20%	if handed in or submitted two days late
30%	if handed in or submitted 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 to Monday is considered one day. 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 result in a lower final grade. Note that the submission system will report the number calendar day an assignment is late.

Valid excuses for missing exams 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.

Generally, 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.** 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. In cases of excessive absences, the instructor reserves the right to require official documentation.

Excused work must be made up within one calendar week from the original due date for full credit unless prior arrangements are made. 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

Attendance is important and expected. Students that do not attend class regularly generally do not pass this course. Attendance records will be maintained in accordance with Federal Law, but will not be used in the determination of grades, except in borderline cases. However, the instructor reserves the right to reduce a final grade in this course for excessive absences. Students will be warned prior to such action. 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.

Credit Hour Policy

This course meets the federal requirements of 15 in-class hours plus an expected 30 hours of 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 assistance. The following specific guidelines are in force for this course.

- **Homework** (including programming exercises) 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. This includes the programming exercises, which are to be the result of your own typing. Ultimately you will be required to demonstrate your proficiency of the material on exams. Therefore, it is highly recommended that you attempt all homework problems on your own before finding a solution from another source.

- **Programming projects 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 also is permitted, but you may not observe another person's solution or code for the purposes of studying or copying it, with or without that student's permission. This includes, but is not limited to, other students in the class, previous students in the class, and the Internet. For example, asking, "What does this compiler error mean?" or "Do I have the correct class syntax here?" is acceptable. Whereas asking "Can I see how you coded your stack?" is not acceptable.

  In particular, since UNIX systems tend to be open by default, it is absolutely forbidden to "rummage" around the csserver file system looking at anyone else's work even if they have not set the file permissions to prevent such observation. (For those that would rather not rely on the integrity of others, it is suggested that all work for this class be put into a subdirectory that has its permissions set to owner only.) Also, studying printouts left in CS Lab (or any other lab) is not acceptable.

  Giving or receiving unauthorized aid on a programming project will result in a 0 for the project for *both* the giver and the receiver on the first offense. Any subsequent violations will result in an F for the course and possibly formal disciplinary action.

- **Exams and quizzes, of course, are to be solely your own work.** Giving or receiving any type of unauthorized aid on any exam will result in a final grade of F and possibly formal disciplinary action.

Also note that violation of the Honor Code are recorded by the EECS department and multiple violations across courses and years also may result in formal disciplinary action. If there is any doubt as to whether assistance is acceptable, consult the instructor.

Course Schedule

Here is a tentative schedule showing the daily reading assignments and exams for this term. Adjustments will be made as needed.

Week of	*Monday*	*Wednesday*	*Friday*
08/19		Overview, basic UNIX, `g++`, console I/O, namespaces	Command lines, `argv`, `argc`
08/26	Filestreams, version control, `git/gitkeeper`	Reference parameters, output formatting	Ch. 5.1-5.3: Recursion
09/02	**Labor Day – no class**	Ch. 5.3-5.5: Recursion	C++ strings Ch. 1.2: Classes
09/08	Ch. 1.2: Classes Separate compilation, `make`	Ch. 1.2: Overloaded operators	Assertions and exceptions
09/16	Ch. 1.5-1.6.2: Templates, STL vector, iterators	Ch. 1.1, 3.1: `auto`, range-based for loops, STL list	Ch. 1.3: Pointers and dynamic arrays
09/23	Ch. 1.6.3, 1.7: STL algorithms, vector implementation	Ch. 1.7: Vector implementation	Ch. 1.7: Vector implementation
09/30	**Practical Exam I** **Time: TBD**	Midterm Exam Review	**Instructor at a conference** **No class**
10/07	**Fall Break – no class**	**Written Midterm Exam**	Testing and debugging, `gdb`
10/14	Ch. 2, 5.6: Algorithm analysis	Ch. 3.1-3.2: Linked lists	Ch. 3.1-3.2: Linked lists
10/21	Ch. 3.2-3.3: Linked lists	Ch. 4.1: STL stack and queue	Ch. 4.2: Stack applications
10/28	Ch. 6.1: Binary tree	Ch. 6.2-6.3: Binary search trees (BST)	Ch. 6.7: BST implementation
11/04	**Practical Exam II** **Time: TBD**	Ch. 6.7: BST implementation	Ch. 6.7: BST implementation **Last day to drop**
11/11	Ch. 6.4-6.5: STL set and map	Ch. 7.3: Heaps	Ch. 7.4-7.5: Priority queue
11/18	Ch. 8.5: Heapsort	Ch. 8.1-8.3 : Bubble, selection, insertion sort	Ch. 8.7: Quick sort
11/25	Ch. 8.6: Merge sort	**Thanksgiving Break – no classes**	
12/02	Ch. 8.8: STL sort Comparing sort algorithms	Final Exam Review	***Thursday:*** Reading/Study Day
12/09	**(Written) Final Exam** **2:00-4:00pm**		