

**Data Structures - CS 271**  
**Fall 2008**  
**Credits: 4 hours**

**Instructor:** David Furcy  
**Email:** [fureyd@uwosh.edu](mailto:fureyd@uwosh.edu)

**Office:** Halsey 220  
**Phone:** 424-1182

**Class Meetings:** 11:30AM-12:30PM on Tuesdays, Wednesdays, and Thursdays in HS 208  
11:30AM-12:30PM on Fridays in HS 101C

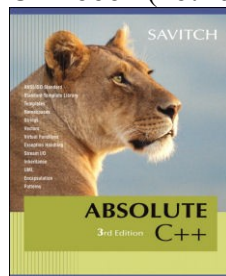
**Office Hours:** **MTuWThF** 8:00-9:00AM  
Feel free to drop in at any other time when my door is open.  
Alternatively, to ensure that I have enough time to answer your questions, I strongly encourage you to make an appointment.

**Prerequisites:** CS 262 with a grade of C or better.

**Required Textbook:** None

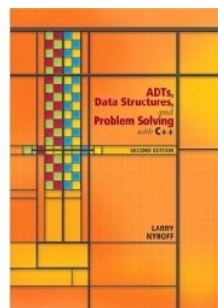
**References:**

- Class handouts. **TAKE NOTES** on and about them. Handouts that are not liberally saturated with your own notes will likely prove useless when you need them most!
- Course web page at: [http://www.uwosh.edu/faculty\\_staff/fureyd/cs271/](http://www.uwosh.edu/faculty_staff/fureyd/cs271/)
- Online algorithm visualizations using JHAVE.
- C++ book (not required):



*ABSOLUTE C++*  
Third Edition  
  
Walter Savitch  
Addison Wesley, 2008

- Data structures book (not required):



*ADTs, Data Structures, and Problem Solving with C++*  
Second Edition  
  
Larry Nyhoff  
Prentice Hall, 2005

**Tests:** Exam #1: Wednesday, October 8<sup>th</sup>, 5:00-7:00PM  
Exam #2: Wednesday, November 12<sup>th</sup>, 5:00-7:00PM  
Exam #3: Wednesday, December 10<sup>th</sup>, 5:00-7:00PM

Exams are scheduled at night to circumvent the one-hour time limit imposed by in-class exams. Please block these time slots at the beginning of the semester. The procedure and criteria of eligibility for making up a missed exam are described below.

**If you have special needs (or ABSOLUTELY cannot make it to one of the above night exams), please come and talk to me RIGHT AWAY so I can accommodate your needs.**

### **Course Overview:**

This course surveys the fundamental methods of representing and structuring data in memory, together with their associated data-processing algorithms. We will discuss classic algorithms for basic tasks such as searching and sorting. We will use and evaluate computational patterns such as recursion and dynamic programming. More generally, we will learn how to analyze, compare, and predict the performance of algorithms.

This course also introduces students to C++. But instead of spending the first 1/4<sup>th</sup> of the course on C++ and the last 3/4<sup>th</sup> on data structures, only selected C++ topics will be discussed in class as needed. From the first day of the semester, students are expected to spend a significant amount of out-of-class time learning C++ and to eventually demonstrate proficiency in this new programming language by completing all programming assignments and labs in C++. In contrast, the written tests will primarily cover data structures concepts, with little programming.

### **Learning Outcomes:**

- Given an algorithmic specification of a process based on decision and iterative control structures, dynamic memory de/allocation, and/or text-based file and console I/O, the student will be able to edit, compile, debug and run in a UNIX/Linux development environment a C++ program that uses pointers, the *new* and *delete* operators, the *iostream* library and/or other predefined classes in the STL to correctly implements the algorithm.
- Given a problem or task description, the student will be able to apply object-oriented design principles to the development of a C++ solution based on user-defined classes and *structs*, the reuse of standard library functions and classes, as well as programming language features not available in Java (e.g., selecting by-value or by-reference parameter passing, taking advantage of operator overloading).
- Given a non-recursive algorithm, the student will be able to examine its loop structures, infer its asymptotic runtime, and express its efficiency using big-O notation.
- Given a recursive algorithm, the student will be able to examine its recursive structure, determine and solve the corresponding recurrence relation, and infer the asymptotic runtime of the algorithm using big-O notation.
- Given the description of a computational problem requiring a mixture of search, insertion, and/or deletion operations on collections of data, the student will be

able to compare the relative advantages of using arrays, vectors, and linked lists in solving the problem efficiently.

- Given a classical computational problem (e.g., infix-to-postfix conversion, postfix-expression evaluation, Huffman data compression, path planning, minimum-spanning tree computation), the student will be able to trace a solution to the problem using appropriate data structures (e.g., stacks, queues, binary trees, binary search trees, red-black trees, graphs) and to predict the asymptotic runtime of the solution based on the selected data structures.
- Given a collection of unordered data, the student will be able to trace the execution of an advanced sorting algorithm (such as quick sort and heap sort) on this data set.
- Given a set of data keys, a hash function, a table size, and a collision-handling strategy, the student will be able to trace through a sequence of key insertions and searches, and to discuss how varying the table size, hash function or collision-handling would affect the execution time of these operations.
- Given a graph data structure, the student will be able to implement it using either adjacency lists or an adjacency matrix, to traverse it using either a depth-first or breadth-first strategy, to identify its structural properties (whether it is directed, cyclic, connected, complete), and to trace the execution of one or more classical graph algorithms (e.g, Dijkstra's, topological sort or minimum-spanning tree computation).

**Topic Coverage:** We will cover the following topics:

- Introduction to C++ and UNIX/Linux.
- Review of arrays, vectors, and linked lists.
- Introduction to classes, structs, and pointers in C++.
- Introduction to C++ templates and the STL.
- Algorithm analysis for non-recursive algorithms.
- Stacks, queues, infix-to-postfix conversion and evaluation.
- Recursion, algorithm analysis for recursive algorithms.
- Dynamic programming.
- Heap sort and quick sort.
- Trees, binary search trees, red-black trees, and heaps.
- Hashing.
- Graphs and their applications.

**Course Grading Policy:** Your final grade for this course will depend on labs, quizzes, programming assignments, and 3 exams. Each assignment and exam will be graded on a scale from 0 to 100. All assignments (labs, quizzes) will carry the same weight when computing your overall assignment (lab, quiz) grade. Your overall numerical grade for the course will be computed as the weighted sum of the component grades using the following weights:

<b>Component</b>	<b>Weight</b>
Labs	8%
Quizzes	8%

Assignments	40%
Exam #1	14%
Exam #2	16%
Exam #3	14%

Finally, your letter grade for the course will be computed using the following mapping:

Numerical Score	Course Grade	Numerical Score	Course Grade
$\geq 92$	A	$\geq 72$	C
$\geq 89$	AB	$\geq 69$	CD
$\geq 82$	B	$\geq 60$	D
$\geq 79$	BC	$< 60$	F

I will be glad to discuss any questions you may have about grades. However, make sure to bring them up right away, upon return of each graded assignment or exam. Last minute requests after the final exam will not be entertained.

**Attendance and Participation:** You are expected to not only attend **every** class meeting but also to come **prepared** for and **participate** actively in it. Necessary preparation requires you to have studied and assimilated the material covered in previous sessions, to have met with me outside of class to discuss any questions you may have, to have done the assigned reading and lab preparation, and to have completed the programming assignments on time.

**It is hard to imagine how a student could do well in this course while missing classes, attending them unprepared, or not participating.**

On the positive side, I have high expectations for my students and will always support and encourage you. I **strongly encourage** you to **ask any question** or raise any issue you have with the course either during or at the end of class, or during my office hours. I will also gladly meet with you by appointment. Send me email or give me a call to make an appointment. While I will meet with you as soon as my schedule permits, do not expect me to be widely available before an assignment is due.

**Late Submissions:** I will describe the submission procedure for your assignments when the time comes. However, let me point out right away that each assignment will come with a deadline (day and time) after which any submission is considered late, **with no exception**. The penalty for late submissions is computed as follows:

Turned in	Penalty
on due date, after deadline	10%
one day late	20%
two days late	40%
three days late	60%
four days late	80%

Note that submissions that are more than 4 days late will receive zero points. Late submissions can easily be avoided by starting to work on the assignment right away and asking me questions early if you get stuck.

The penalty for late submissions can be waived in **only one** scenario, namely if you give me a signed note from the attending physician or a written justification for the extension from the Dean of Students Office. If you miss a scheduled exam, you **may** be able to take a make-up exam provided you give me a valid justification (see above) ahead of time if possible. Only one make-up exam will be given. It will be a comprehensive exam scheduled at the end of the semester.

**Bonus points:** Since late submissions are penalized, it is only fair to reward early ones as well. Therefore, if you submit an assignment at least one full day early, you will earn 3 bonus points for every full 24-hour interval between your submission time and the deadline time. Furthermore, you can double your lab points each week by successfully demoing your lab exercises before the end of the lab session.

**Collaboration versus Cheating:** All submissions must be the work of either one or two students, namely the one(s) whose name appears on the submission (yes, you may choose pair-programming for your assignments, but not for the labs). While it is acceptable and encouraged to discuss the assignments with others, you must submit your own work. You may not look at or “borrow” any piece of code of any length from anyone else, unless you can live with a zero and the other potential academic sanctions of cheating. Check out the UWO Student Discipline Code (UWS 14) at <http://www.uwosh.edu/dean/conduct.htm> for details.

**Final Note:** I expect every committed and hardworking student to do well in this course. I am looking forward to a fun and rewarding semester together. Given my high standards, I could not be more satisfied than if everyone earned an A in this course.