

## COURSE DESCRIPTION

<b>Department and Course Number</b>	CMPS 260	<b>Course Coordinator</b>	Jim Etheredge
Course Title	Introduction to Data Structures and Software Design	<b>Total Credits</b>	3
URL	<a href="http://www.ucs.louisiana.edu/~jne1390/cs260/cs260.html">http://www.ucs.louisiana.edu/~jne1390/cs260/cs260.html</a> <a href="http://fidelio.cacs.louisiana.edu/fdd5501/260/">http://fidelio.cacs.louisiana.edu/fdd5501/260/</a> (due to copyright issues, some links require “cmps260” as a username and “guest” as a password)	<b>Semester hours</b>	3

### Current Bulletin Description

Integrated software engineering principles, fundamental data structures and algorithm design and development. Focus on requirements, specifications, design and testing. Prerequisites: CMPS 150 and MATH 110 or 201 with a grade of C or better. Co-requisite: Math 250, 270 or 272.

### Textbook

C++ Plus Programming: Program Design Including Data Structures (Third Edition), 2007 by D. S. Malik. ISBN-13: 978-1-4188-3640-5, ISBN-10: 1-4188-3640-0 (It's the same as the CMPS150 textbook).

### References

Programming documentation standards (on web site:  
<http://www.ucs.louisiana.edu/~jne1390/cs260/cs260.html>)

### Course Goals

- To gain programming experience in C++ on Unix.
- To gain experience with abstract data types and object-oriented development.
- To understand software design principles. The documentation standards for this class direct students in the specification and design of object-oriented software systems.
- To develop a working knowledge of lists, stacks, and queues. The programming projects help students understand the use and implementation of these data structures.

### Course Outcomes

- Students are able to apply object-oriented design and development principles to the creation of software systems.
- Students have an understanding of the mechanism by which storage can be dynamically allocated and de-allocated. They also understand the tradeoffs between dynamic storage allocation and static storage allocation. Finally, they are able to identify design situations that are best handled using dynamic storage.
- Students understand and are able to apply software design principles.
- Students have a working knowledge of list, stack, and queue data structures. They are able to readily identify program development situations where these data structures are appropriate.

## Prerequisites by Topic

- Experience in developing computer solutions to problems.
- Working knowledge of object oriented programming techniques and C++.
- Working knowledge of the Unix platform and its C++ programming facilities.
- Working knowledge of the program development process (problem definition, requirements specification, design, implementation, testing, maintenance).

## Major Topics Covered in the Course

- Abstract data types, classes, object oriented design and programming (15 hours)
- Software design documentation (2 hours)
- Fundamental data structures, their operations and applications (3 hours)
- Array implementations of lists, stacks, and queues (4 hours)
- Pointers and dynamic storage allocation and de-allocation (3 hours)
- Linked lists, stacks, and queues (10 hours)
- Continued study of C++, separate compilation units, makefiles (4 hours)
- Template classes (3 hours)
- Operator overloading (3 hours)
- Algorithm analysis (1 hour)
- Searching techniques: sequential & binary (1 hour)

## Laboratory projects (specify number of weeks on each)

- ADTs and classes (2 weeks)
- Makefiles (1 week)
- Dynamic lists (1 week)
- Overloading (1 week)
- Template classes (1 week)
- Pointers and dynamic variables (1 week)
- Linked lists (2 weeks)
- Linked list implementation of linked lists, stacks and queues (1 week)
- Requirements documentation (1 week)
- Design documentation (2 weeks)

## Oral and Written Communications

Every student is required to submit at least 2 written reports (not including exams, tests, quizzes, or commented programs) of typically 2 pages and to make 0 oral presentations of typically 0 minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

## Social and Ethical Issues

Social and ethical issues are not covered specifically in this course. They are discussed in general as part of classroom discussions of the responsibilities of system analysts and programmers to

produce computer systems that fulfill requirements and adhere to accepted legal, moral and ethical standards. Students are not tested directly on these issues.

### **Theoretical Content**

- Abstract data types, classes, object oriented design and programming (10 hours)
- Dynamic storage allocation and de-allocation (3 hours)
- Program development and debugging (3 hours)

### **Problem Analysis**

Students in this course are required to perform the analysis necessary to design, create, debug, and test programs to implement the concepts covered. The design process is an integral part of the class since this skill is necessary for computer scientists. Students learn the analysis process through classroom lectures, laboratory work, and guidance provided by faculty and teaching assistants in the understanding and completion of programming and design assignments.

### **Solution Design**

Students in this course are required to design all programming assignments. Assignments are made in the form of requirements documents and the students are responsible for designing the solution prior to implementation and testing.