

<b>Instructor:</b>	Derek Harter	Sam Stamport	Jim Majors
<b>Phone:</b>	903-886-5402		
<b>Email:</b>	<a href="mailto:Derek_Harter@tamuc.edu">Derek_Harter@tamuc.edu</a>	<a href="mailto:sam.stamport@gmail.com">sam.stamport@gmail.com</a>	<a href="mailto:mwhiteboard2003@yahoo.com">mwhiteboard2003@yahoo.com</a>
<b>Office Hrs:</b>	T,W 2:00-4:00pm, Jour 208		

**Web Page:** <http://faculty.tamuc.edu/dharter/tamuc/classes/2006summer/csci152>

**Class:** Section 081: T,R 10:00–11:50am, Jour 102

**Course Description:** This course is a continuation of CSci 151. The skills in problem solving and programming acquired in the previous class will be reinforced and enhanced. The concepts of object-oriented programming and design will be introduced. Specific topics include multi-dimension array processing and applications, sort and search algorithms, file processing, string processing, structures, classes, inheritance, pointer variables. This course requires your active involvement in the programming assignments and lab work. Your completion of these is a key to success in this course. If you complete all of them I guarantee that you will find the quizzes and final exam much easier to understand and complete successfully.

**Goals:** After completion of this course, you should be able to code application programs using arrays, sorts and searches, strings, user-defined files, and structures. You should be able to define objects and create and use classes to represent abstract data types.

**Prerequisite:** CSCI 151 or other course covering basic concepts of the C or C++ languages.

**Text:** C++ Programming: From Problem Analysis To Program Design (Second Edition)  
by D.S. Malik  
ISBN: 0-619-16042-X

#### Evaluation:

Your grade for the course will be based on the following approximate numbers and percentages:

- 32% 4 quizzes
- 32% 8 Programming assignments
- 24% 8 Labs
- 12% comprehensive final exam.

**Format for the quizzes** will typically be approximately half coding (usually small segments like functions or parts of functions) and half analyzing the effects of executing code (describing output, completing diagrams, short essay, etc). Quizzes are not cumulative, and will concentrate on the specific material covered from the last quiz to the next.

**Format for the labs and programming assignments** Approximately 30 to 50 minutes of each class period will be spent working on a hands on lab or programming assignment. Each Tuesday of the class, you will work on a small lab which you will be expected to complete yourself in class by the end of the class period. Labs are worth 3 points each towards your final grade in the course (for a total of 24% of your final grade). Each lab will ask you to work with or implement the topic covered in that days reading assignment and lecture presentation. Therefore it is **very important** that you come to class already having read the assigned reading materials, as you will be asked to implement your knowledge of that material in the days lab. Likewise, the second half of class each Thursday will be devoted to a more involved programming assignment. The assignment will be a continuation of the labs from Tuesday, with extra concepts that are discussed in the Thursday class. Again, you need to come to class already having read and understood the assigned reading materials. The programming assignments will still be relatively small, but may not be completely doable in the 30 to 50 minutes of class devoted to the assignments. Programming assignments will be due the Tuesday of the next week after we work on them in class (except for programming assignment #4, which will be due on July 6<sup>th</sup>, due to the Fourth of July holiday).

**Format for the written final exam** will be mostly multiple-choice, true/false, and short answer with minimal coding. The Final will be held on the final Thursday of the class, August 10, during the normal class time from 10-11:50am. Please keep the date and time of the final in mind when planning your travel and end-of-semester activities as makeups cannot be given or scheduled simply to accommodate end of summer travel plans.

A **study guide** will be provided for the final exam listing topics to be covered (or omitted) and recommending selected problems from the text from which the majority of exam questions will be derived. Answers (which are not provided in the text) for these problems will be available on my web page.

Letter grades will be assigned according to the following scale:

- A at least 90% of the total points
- B at least 80% but less than 90% of the total points
- C at least 70% but less than 80% of the total points
- D at least 60% but less than 70% of the total points
- F less than 60% of the total points

You must earn an A on your own. Lower borderline grades may be affected by your class participation and behavior, the pattern of your grades, and the class grade distribution.

**eCollege:** The web page for our class this semester is:

<http://faculty.tamu-commerce.edu/dharter/tamu/classes/2006summer/csci152>

The course syllabus, input data files, sample programs, lecture slides, programming assignments and other miscellaneous handouts and useful materials may be found here. In addition, I will be using the online Texas A&M eCollege system this semester to disseminate course materials and collect assignments. You will need to log onto the eCollege system by going to:

<http://online.tamuc.org>

I will send all e-mail and announcements using your eCollege e-mail addresses, and all submissions of programming assignments will be via eCollege. Please check you account e-mail and announcements frequently.

**Makeups:** Labs can not be made up, they are due at the end of the class period of the lab. All programming assignments are due by the start of class (10am) the Tuesday following the programming assignment.

**Attendance:** You are responsible for everything covered in all class meetings. Since we have labs and programming assignments pretty much each day of class, which can not be made up, you will have trouble passing this class if you do not attend **all** of the class sessions.

**Drops:** If you cannot complete the course, please don't forget to drop. If you are making an obvious effort in the course at the time you drop (still attending class, attempting program assignments), you may drop passing no matter what your actual grade might be. If you just disappear, your grade will be whatever you have actually earned at the end of the semester.

All students should be aware that plagiarism is a serious offense. This is true not only of written essays but also of work written in artificial computer languages such as C++. Copying code for assignments from other students or the internet is not allowed, and will be punished severely. You may discuss general aspects and strategies of the coding assignments with the instructor, the teaching assistants and even fellow students, but you must code the programming assignments on your own. You should never copy code from another persons program, send code or a code snippet to someone via e-mail, even allowing someone to look at your completed code on the monitor should not be done. Copying or reusing other people's programs will not be tolerated, and will result in a 0 for the assignment. Repeated problems or instances of dishonesty will result in disciplinary academic proceedings being sought against the offending student. In short, just don't do it. The goal is for you, yourself, to learn the basics of programming in this course, which you can not do if you copy from other people. Only by doing the assignments and getting down and dirty with the code will you truly begin to understand the concepts we aim to convey in Programming II.

Students requesting accommodations for disabilities must go through the Academic Support Committee. For more information, please contact the Director of Disability Resources & Services, Halladay Student Services Bldg., Room 303D, (903) 886-5835.

"All students enrolled at the University shall follow the tenets of common decency and acceptable behavior conducive to a positive learning environment." (See Student's Guide Handbook, Policies and Procedures, Conduct)

## TENTATIVE SCHEDULE

Date	Question / Topic	Reading	Quiz	Lab	Prog
Jun 6/9	What is it to “program” and what are programming fundamentals? Review of selected 151 topics.	Review Ch 1-8		#1	#1
Jun 13/15	What is an array used for, and how do we define them? Using arrays of characters in C++ to represent and process alphanumeric strings.	Ch 9: 423-442 Ch9: 442-450		#2	#2
Jun 20/22	Using arrays to represent multi-dimensional tables. <b>Quiz #1, review of I/O, control structures (selection &amp; repetition), functions, arrays.</b>	Ch 9: 450-468	#1	#3	#3
Jun 27/29	How can we sort a collection of items and put them in some order? How can we search a collection of items more efficiently than simply examining every item in the collection?	Ch 10: 497-515 Ch 10: 515-523		#4	#4
Jul 4/6	<b>Fourth of July Holiday</b> Using user defined data types to represent collections of items of different types. <b>Quiz #2, applications of arrays, searching and sorting, structures and user defined collections</b>	Ch 11: 533-571	#2		
Jul 11/13	Fundamentals of classes, non-homogeneous collections of both data and behavior. How do we add behavior to a collection? Defining member functions, accessors and mutators.	Ch 12: 595-605 Ch 12: 605-617		#5	#5
Jul 18/20	Specialize behaviors of classes, using constructors to initialize data members. What use are classes, data abstraction and information hiding as concepts for building programming solutions? <b>Quiz #3, structures and classes, how to define data members and member functions</b>	Ch 12: 630-647 Ch 12: 630-647	#3	#6	#6
Jul 25/27	Functional vs. Object Oriented programming: the OO paradigm shift. How do we use Inheritance in OO programming and what is it good for? What is object composition and how is it useful in programming?	Ch 13: 705-708 Ch 13: 673-700 Ch 13: 700-705		#7	#7
Aug 1/3	Introduction to using basic pointers in C and C++. How are pointers related to C arrays? An introduction to some advanced concepts in C++ OO programming: overloading, templates and virtual functions. <b>Quiz #4 concepts of OO programming and using inheritance and composition, pointers, overloading and templates.</b>	Ch 14: 741-756 Ch 15: 802-843, Ch 15: 865-876	#4	#8	#8
Aug 8/10	<b>Final Exam, Thursday Aug 10, 10:00-11:50 am, EDS 131</b>				

## Course Objectives

1. Be able to use one-dimensional arrays.
  - a. Be able to declare, initialize, and manipulate individual elements of a one-dimensional array.
  - b. Be able to pass one-dimensional arrays to functions.
2. Be able to use at least one (preferably at least two) sorting technique(s) to rearrange data in an array.
3. Be able to search an array using both linear and binary searching techniques.
4. Be able to use multiple-dimensional arrays.
  - a. Be able to declare, initialize, and manipulate individual elements of a two-dimensional array.
  - b. Be able to pass two-dimensional arrays to functions.
  - c. Be able to pass one row of a two-dimensional array to a function.
5. Be able to use traditional C-style strings.
  - a. Be able to declare and initialize C-strings.
  - b. Be able to manipulate C-strings one character at a time.
  - c. Be able to assign and compare C-strings and to determine their length.
  - d. Be able to pass C-strings to functions.
6. Be able to use the ANSI C++ string class.
  - a. Be able to declare and initialize strings.
  - b. Be able to manipulate strings one character at a time.
  - c. Be able to assign and compare strings and to determine their length.
  - d. Be able to pass strings to functions and to have a function return a string.
  - e. Understand the use of logical operators with string class instances
7. Be able to use address variables.
  - a. Be able to declare a pointer variable, initialize it, and dereference it.
  - b. Be able to pass a variable by address using a pointer parameter.
  - c. Understand the relationship between pointers and arrays.
  - d. Be able to use pointer arithmetic to access elements of an array.
  - e. Be able to declare, initialize and use a reference variable.
  - f. Be able to use a reference variable as a function parameter.
  - g. Be able to use a reference variable as the return data type of a function.
  - h. Be able to pass an argument to a function that uses a reference parameter to accept the argument's address.
  - i. Understand the differences between reference variables and pointer variables.
8. Be able to use structs.
  - a. Be able to declare a struct and manipulate its fields.
  - b. Be able to declare an array of structs and manipulate the elements.
  - c. Understand the differences between structs and classes.
9. Be able to use classes.
  - a. Be able to create and use class objects.
  - b. Be able to declare a class and control access to class data members.
  - c. Be able to create and use constructor and destructor functions.
  - d. Be able to create and use friend functions.
  - e. Be able to overload functions.
  - f. Be able to overload operator functions.
10. Be able to design and code a program which includes a user-created class.