

CSC 2430: Data Structures I Winter 2012

Professor: Mike Tindall
Office: OMH 240
Office Phone: 206-281-2945
Email: mht@spu.edu
Website: Blackboard (and <http://myhome.spu.edu/mht/csc2430>)
Office hours: Mondays and Wednesdays 12:30 – 1:30pm or by appointment

Meeting Times: MWF 11:00am – 12:20pm in OMH 245

Final Exam: Wednesday March 14, 10:30 – 12:30

Course Description and Learning Objectives

Prerequisite: CSC 1230 or equivalent.

Develops discipline in program design, style, debugging, testing. Introduces object-oriented design, with Classes, Methods and encapsulation. Introduces dynamic storage allocation and pointers. Examines arrays, linked linear data structures, and recursion. (Currently uses the C++ programming language.)

How does this class help prepare you for future success in the computing sciences?

This course will introduce the student to the design and use of data structures and algorithms, including their implementation in the object-oriented C++ programming language. Students who successfully complete this course (grade of C¹ or better) will be able to:

- 1) Apply problem-solving skills to design well-structured algorithms from problem statements.
- 2) Code algorithms in single- and multi-file C++ programs using good coding style and code documentation.
- 3) Compile, execute, test, debug, and informally verify correct operation of programs.
- 4) Articulate the concept of an abstract data type (ADT).
- 5) Articulate and apply the object-oriented concepts of encapsulation, information hiding, classes and methods.
- 6) Articulate the design, implementation, and use of linear ADTs: arrays, stacks, queues, and lists.
- 7) Articulate the concept of Big-O notation and compare the efficiency of some common algorithms.
- 8) Utilize arrays and strings in programs, and describe string implementation alternatives.
- 9) Make use of recursive algorithms in problem solving and programming.
- 10) Utilize dynamic storage allocation and pointers.

Required Text

C++ Programming: Program Design Including Data Structures, 5th edition
by D.S. Malik
publisher: Course Technology, Cengage Learning, 2011
ISBN: 978-0-538-79809-9 (0-538-79809-2)

Additional information and student resources for the text are available by clicking the Access Now button at:

<http://www.cengagebrain.com/shop/ISBN/9780538798099?cid=APL1>



¹ While a C- or better is considered “passing”, any grade less than B- in this course should be understood to denote minimal (at best) preparation for the follow-on CSC 2431 course.

Software

We will be using Microsoft Visual C++ .NET 2010 (which is part of Visual Studio .NET 2010) for this class. SPU'S Microsoft Campus License Agreement makes it possible for you to purchase Visual Studio .NET for the truly remarkable price of \$30. (Similar bargains are available for Windows 7 and Microsoft Office 2010) To buy your CDs, please visit CIS in Lower Marston Hall and be sure to bring your student ID with you. Complete information on media availability and purchase procedures can be found at: <http://www.spu.edu/CISHelpDesk/software/software.asp>.

Computers

Programming assignments may be written and run on the Windows PCs in the Otto Miller Hall Open Computer Lab (OMH 253) or in the SPU Library or on your own computer.

BE SURE TO ALWAYS BACK-UP YOUR WORK and store incremental copies in multiple places (see "Storage Media" below). Drives and disks only fail just prior to the due date of programming assignments. The instructor is very unlikely to perceive a dead or full disk (or even a system crash) as adequate grounds for allowing an extension of the due date!

Storage Media

You should invest in some storage device so that you can back up your personal files. The best option is a USB storage device. USB flash drives are available for purchase in the basement of the University Library.

ALWAYS BRING YOUR TEXTBOOK AND STORAGE TO CLASS. In-class labs / practice assignments will not always be announced ahead of time. Come prepared!

Grading

Grades will be based on performance in three areas:

Homework/Labs/Quizzes/In-class activities	50%
1 Mid-term Exam	25%
Final Exam	25%

The (approximate) grading scale is:

90% and above	⇒ A, A-
80% - 89%	⇒ B+, B, B-
70% - 79%	⇒ C+, C, C-
60% - 69%	⇒ D
Below 60%	⇒ E

NOTE: Exams are measures of individual ability and knowledge whereas homework may be a collaborative effort. Therefore, the highest grade that you could possibly receive in this class is the highest grade you earn on any of the exams. For example, if your final cumulative percentage in the class is 74% (a C) but the highest grade you earned on any one of the 3 exams is 65% (a D), you will receive a 'D' for the course. On the other hand, if your final cumulative percentage is 74% and your exams were 62%, 80%, and 54%, you will receive a 'C' in the class.

Exams

There will be 1 mid-term exam plus a final exam in this class.

The **mid-term exam** will be: **Wednesday, February 1** (tentative schedule date)

A scheduling change will be announced in class at least 1 week prior to the change.

NO make-ups of missed mid-term exams will be given unless *prior* arrangements are made, and then only for reasons of *serious* illness or emergency.

The **Final Exam** is set for: **Wednesday March 14, 10:30 – 12:30.**

It will be closed-book and comprehensive. Policies for rescheduling final exams are set by the university and may be found at: <http://www.spu.edu/acad/UGCatalog/20101/finalexam.asp>. There will be no exceptions to this policy.

PLAN AHEAD! Mark these dates on your calendar NOW!

Assignment Policies

One learns to program by programming. That is why programming assignments are a significant portion of your grade in this class. We can discuss ideas, demonstrate how things work and, occasionally, try things out in class, but most of the actual “learning” and skill-building will occur outside of class as you apply these concepts to assigned problems.

The following procedures apply to *all* assignments (unless specifically directed otherwise):

- Except for in-class assignments due at the end of the period, ***hardcopy assignments are due at the beginning of class on the due date.*** Assignments being turned in during class time should be placed under the instructor’s door. DO NOT place any assignments in the instructor’s mailbox (unless you don’t mind if someone copies your work behind your back!). ***Assignments that are to be electronically submitted will have an announced (and enforced) cut-off time.***
- Assignments not turned in when they are due are considered late. ***Late assignments will not be accepted or graded.*** A “0” score will be recorded for the assignment.
- ***Each student will be allowed one exception to the late-assignment policy during this course.*** If you contact your instructor and request it (on or before the due date), you may turn in one assignment late *within one week of the original due date*. At the end of the quarter, all assigned work (including the last homework assignment) must be received before the start of the final exam.
- Unless an assignment is specifically designated as a team effort (and some in-class labs might be), ***all work is to be done on an individual basis.*** At the same time, it is understood that learning from your peers is valid and you are encouraged to talk among yourselves about programming in general and current assignments in particular. Keep in mind, however, that each individual student must do the work in order to learn. Hence, the following guidelines are established:
 - Feel free to discuss any and all *programming* assignments but do not allow other students to copy your code. Do not give any student an electronic or printed copy of any program you write for this class.
 - Gaining the ability to properly analyze common programming errors is an important experience. Do not deprive a fellow student of his/her opportunity to practice problem solving: control the urge to show them what to do by writing the code for them.
 - If you’ve given the assignment a fair effort and still need help, see the tutor or instructor.
 - If there is any evidence that a program or other written assignment was copied from another student, neither student may receive any credit for it (or each may be allowed ½ credit for their "shared" effort).
 - Protect yourself: ***Handle throw-away program listings carefully.***

Policy Notes

1) **Academic Integrity:** The current edition of the SPU *Undergraduate Catalog* describes the University's commitment to academic integrity, which is breached by academic dishonesty of various kinds. Among these is turning in another's work as your own and committing plagiarism, which is the copying of portions of another's words from a published or electronic source without acknowledgement of that source. The penalty for a breach of academic integrity is a failing grade for the work in question on the first offense and a failing grade for the course as a whole with repeated offenses.

2) **Classroom Environment:** Mutual respect and consideration for others make for an effective classroom environment. By following a few sensible guidelines, we can make the classroom a more comfortable place for all.

- **Please turn off your cell phones** at the beginning of class, or set them to silent mode, so you don't disturb our time together.
- **Do NOT use your computer during the class period for anything that is not class-related;** that is rude and disrespectful.
- **Eating and drinking in class**
 - **If your class is scheduled for a lab classroom such as OMH 244,** food and drink are prohibited to protect the equipment. The only exception will be for water bottles, which are to be kept tightly closed when not being used.
 - **If your class is held in a regular classroom** (such as OMH 215 or 245), food and drink may be allowed during class at the discretion of the professor. However, avoid foods with strong aromas and try not to be too distracting when consuming them.
- Always **take care of bathroom visits and other personal needs before class** begins.
- **If you know ahead of time that you need to leave class early,** please let the instructor know and sit near the door so as to cause minimal distraction.
- **Please come to class on time** (better yet, be early!). This shows respect not only for the instructor but for your classmates as well.

3) **Inclement Weather:** Inclement weather or an emergency may on occasion affect SPU's schedule. In some cases, classes and campus offices may shut-down entirely; in others there may be a late start or an early closure of the campus. Two campus contacts will always carry the most up-to-date information on the campus schedule: the Emergency Closure Hotline (206-281-2800) and the SPU Home Page. Since weather in the Seattle area can change rapidly, check these sites often for updates. If SPU is open but you are unable to travel to campus due to inclement weather, please let your instructor know as soon as possible. Finally, be sure to check your email for any messages from your instructor regarding class activities.

4) **Emergency procedure:** Note the emergency procedures posted in the classroom or laboratory, and note the emergency exits. In case of an emergency (fire, earthquake, hazardous material spillage, bomb threat, etc.), the class will evacuate the building and gather in the Alumni parking lot to the south of OMH. Please stay together as you exit, and be sure to check in with your instructor once you reach the gathering place.

5) **Disability statement:** In accordance with Section 504 of the Rehabilitation Act of 1973 and the Americans with Disabilities Act of 1990, students with specific disabilities that qualify for academic accommodations should contact Disabled Student Services (DSS) in the Center for Learning. DSS in turn will send a Disability Verification Letter to the course instructor indicating what accommodations have been approved.

6) **Course Evaluation:** I hope that you will participate in an online evaluation of this course and its instructor in a thoughtful and constructive manner. The evaluation data is used to make improvements in the course, and your feedback is considered when selecting textbooks, designing teaching methods and preparing assignments. Courses are evaluated using the Banner Course Evaluation System. All answers are completely confidential - your name is not stored with your answers in any way. In addition, I will not see any results of the evaluation until after final grades are submitted to the University.

About the Introductory Programming Sequence

CSC 1230, CSC 2430 and CSC 2431 should be thought of as a Programming 1, 2, 3, sequence. In this class, CSC 1230, you will learn the basics of the C++ programming language, problem solving skills, and how to correctly program a solution to a complex problem. Good solutions to problems are emphasized. In this class, learning syntax is very important.

In CSC 2430, Data Structures I, you will learn some more advanced C++ programming and techniques for internal memory management. This includes more complex data structures and algorithms that work efficiently on those data structures. The emphasis in this class will be making the “correct” choice of data structure depending on needs and the efficiency (in both time and space) of various algorithms.

CSC 2431, Data Structures II, focuses on object-oriented design and programming. More complex data structures such as trees, networks and graphs are discussed. Working with these issues forces us to use even more advanced C / C++ programming methods.

HOW TO READ A PROGRAMMING TEXT

(I'm not your mother, but please read this anyway)

By this point in your education, you already know that a mathematics or science text is not read in the same way that you read a novel. In particular, they must be read *s-l-o-w-l-y* and in small chunks, a section or two at a time rather than a whole chapter in one sitting. Programming texts should be read in this way. Saving your computer science reading for the night before the exam is one sure way to avoid success!

Programming texts also need to be read with a pencil in hand (I usually keep *both* a highlighter pen and a pencil handy!). Reading through a section is a good first step, but you won't know if you understand it until you try a few of the Exercises at the end of the section.

Finally, make an effort to read the example C++ functions and programs that your author has included in the text. Code reading takes a little effort at first, and it is tempting to skip over the code and its analysis. *Yield not to this temptation!* Take the time to look at the examples and make sure you understand what they do. As with working in any language, you must be able to *read* in C++ before you can *write* in C++. The good news is, reading practice improves writing ability, which improves reading understanding, which makes it easier to write, which (you get the idea).

Mission Statements

The mission of the Computing Sciences department is:

Programs in the Computing Sciences provide knowledge and problem-solving skills in the theoretical and applied aspects of computing-related disciplines. Informed by a Christian worldview, our students learn to utilize computing technologies in a socially responsible manner and apply their expertise wherever they serve in the world.

The mission of Seattle Pacific University is:

As a community of learners, Seattle Pacific University seeks to educate and prepare students for service and leadership. We are committed to evangelical Christian faith and values, and to excellence in teaching and scholarship for the intellectual, personal and spiritual growth of students.

Seattle Pacific University seeks to change the world and engage the culture by graduating students of competence and character, cultivating people of wisdom, and modeling a grace-filled community.

Department of Computing Sciences – Goals and Learning Objectives

The Department of Computing Sciences has identified several goals for students enrolled in our programs. Each of the courses in our curriculum are designed to help students achieve specific learning objectives that will help them progress towards these goals. As noted below (\Rightarrow), several of these learning objectives are particularly relevant to your studies in CSC 2430:

Goal 1: Help students develop problem-solving skills, especially those required to analyze, design and implement solutions involving the use of a computer.

- \Rightarrow Objective 1: Successful students will acquire the up-to-date technical knowledge and develop the skills needed for a successful start to careers in the computing industry.
- \Rightarrow Objective 2: Successful students will be able to develop solutions to problems that are new to them, and implement these solutions efficiently.
- Objective 3: Successful students will be able to implement solutions utilizing different computer platforms and programming languages.
- Objective 4: Successful students will develop the skills needed to work in small groups on medium to large scale projects.
- \Rightarrow Objective 5: Successful students will develop the ability to write technical documents that include specification, design, and implementation of major projects.
- Objective 6: Successful students will be able to effectively disseminate information and results using both oral and written communication.

Goal 2: Provide a background in modern computing systems and the theoretical aspects of computer science.

- Objective 1: Successful students will acquire the computer science knowledge required for graduate studies.
- \Rightarrow Objective 2: Successful students will understand the architecture, organization and programming of modern computing systems.
- Objective 3: Successful students will understand the mathematical foundations of computer science, algorithm efficiency and computational complexity.

Goal 3: Challenge students to consider the ethical and social impacts of technology, enabling them to take responsible action informed by a Christian world view.

- \Rightarrow Objective 1: Successful students will be aware of ethical and social issues related to technology and recognize their impact.
- Objective 2: Successful students will be able to evaluate potential ethical dilemmas and apply decision-making techniques to resolve them.

Goal 4: Prepare students for continued learning in a rapidly changing discipline.

- \Rightarrow Objective 1: Successful students will be aware of the rapid rate of change of technology and methodologies in computer science.
- \Rightarrow Objective 2: Successful students will be familiar with ways to gain knowledge and understanding of new developments in computer science and technology.
- Objective 3: Successful students will be aware of alternatives for continuing education in computer science.