Carleton University
Department of Systems and Computer Engineering
SYSC 2006 - Foundations of Imperative Programming - Winter 2012
Course Outline

Instructor

Don Bailey, Room 3042 MC, 520-2600 ext. 5748, bailey@sce.carleton.ca.
Office hours: posted on the course Web site.

Undergraduate Calendar Course Description

SYSC 2006 [0.5 credit]
Foundations of Imperative Programming
Modular programming with a procedural language. Compilation and linking, libraries. Memory management and object lifetimes: static allocation, automatic allocation in stack frames, dynamic allocation from the heap. Introduction to data structures: dynamic arrays, linked lists. Collections: lists, stacks, queues. Introduction to recursion.
Precludes additional credit for SYSC 1102 and SYSC 2002.
Prerequisite: ECOR 1606 or SYSC 1005.
Lectures three hours a week, laboratory two hours a week.

Course Aims and Objectives

Aims

After completing this course, students will be able to implement small-scale, modular C programs. Students will be able to visualize how memory is managed by an executing program, and demonstrate this knowledge pictorially. Students will be familiar with the design and implementation of simple abstract linear collections such as the list (vector), queue and stack. Students will be able to construct simple recursive functions. Students will be prepared to undertake a course that covers abstract collections, data structures, and algorithm design and analysis. Students will be prepared to undertake a course that provides thorough coverage of object-oriented programming principles. Students will understand the concepts that underlie most imperative programming languages and be prepared to apply these concepts when learning new langauges.

Objectives

A student will be able to:

- design, code, test and debug small-scale C programs (e.g., up to 500 lines of code) that are partitioned into multiple modules;
- trace a program's execution and draw diagrams that illustrate how memory is allocated by the program; in other words, depict the contents of the program's static variables, its stack frames (containing function arguments and local variables), and memory that has been allocated from the heap and is accessed through pointer variables;
- design, code and test functions that operate on two fundamental data structures: the dynamic (resizable) array and the pointer-based singly-linked list;
- describe, from a client-side perspective, the operations provided by some linear abstract collections; e.g., list (vector), queue, and stack;
- implement these collections; i.e., given the specification of a collection's operations and a description of its underlying data structure, implement the data structure and algorithms that provide the required operations;
- specify simple recursive algorithms, convert these algorithms into recursive functions,

and draw stack-frame diagrams to explain their execution.

Prerequisite
ECOR 1606 or SYSC 1005 are the prerequisites for SYSC 2006. Prerequisite waivers will not normally be granted. Students who have not received credit for ECOR 1606 or SYSC 1005 must withdraw from SYSC 2006 by the last date for registration in Winter term courses; otherwise, they will be deregistered before the end of term. Students who received DEF in the Fall 2011 offerings of ECOR 1606 or SYSC 1005 are eligible to register in SYSC 2006, provided that they write the deferred exam in February 2012. These students can remain in SYSC 2006 if the DEF is changed to a passing grade; otherwise, they must withdraw from SYSC 2006.

SYSC 1100, which was offered for the last time in Fall 2010, is considered equivalent to SYSC 1005 and will be accepted as a prerequisite. Students who have credit for SYSC 1100 do not have to take ECOR 1606 or SYSC 1005 prior to taking SYSC 2006.

Textbook
Problem Solving and Program Design in C, *Sixth Edition*, Jeri R. Hanly, Elliot B. Koffman, Addison-Wesley, 2010, ISBN-10: 0321535421, ISBN-13: 9780321535429.

Printed copies can be purchased at the university bookstore. An online version of the eTextbook can be rented for 180 days from CourseSmart (www.coursesmart.com) for roughly 40% of the list price of a printed copy.

C Reference
The C Programming Language, *Second Edition*, Brian W. Kernighan and Dennis Ritchie, Prentice Hall, 1988, ISBN-10: 0131103628, ISBN-13: 9780131103627.

One of the authors (Ritchie) was the original developer of C. Unfortunately, this book has not been updated to reflect C99 (the most modern dialect of C); however, it's still widely used and is regarded by many as being the authoritative reference on C.

Course Web Site
http://www.sce.carleton.ca/moodle/

Policy on Laptop and Tablet Computers
During scheduled labs, students who prefer to use their own laptop computers instead of the lab computers are permitted to do so; however, all laptop and tablet computers must be turned off during lectures.

Attendance
Students are expected to attend all lectures and lab periods. The University requires students to have a conflict-free timetable. For more information, see the current Undergraduate Calendar, *Academic Regulations of the University*, Section 1.2, Course Selection and Registration and Section 1.5, Deregistration.

Requests to accommodate a missed midterm exam, lab periods, due dates, etc., because of conflicts with jobs or vacation plans will not be considered.

Health and Safety
Every student should have a copy of our Health and Safety Manual. A PDF copy of this manual is available online: http://www.sce.carleton.ca/courses/health-and-safety.pdf.

Evaluation and Grading Scheme
Students will be evaluated by means of laboratory work, assignments, a midterm exam and a final exam.

To pass the course, students must pass the final examination (50% or better). For students who pass the final exam, a numeric mark out of 100 will be calculated using two schemes, which weight the course components as follows:

	Scheme 1	**Scheme** 2
Lab work and assignments	20%	15%
Midterm exam	25%	20%
Final exam	55%	65%

The higher of the two marks will be converted to your final letter grade, using the table of percentage equivalents shown in Section 2.3 of the *Academic Regulations of the University*.

Lab Periods
Attendance at the scheduled laboratory periods is mandatory, and attendance will be taken. During the labs you will work on short programming exercises that are intended to help you understand particular concepts that have been introduced in the lectures. You will normally be required to demonstrate and/or submit your lab work by the end of the lab period (or other specified deadline), as indicated in that week's lab "handout".

Your work in each lab period will be graded *satisfactory*, *marginal*, or *unsatisfactory*.

- *Satisfactory* means that you were present at the lab and made reasonable progress towards completing the lab exercises. Note that you do not have to finish all the exercises to receive a *satisfactory* grade.
- *Marginal* means that you made some progress towards completing the exercises, but your solutions to were not sufficiently complete to warrant a *satisfactory* grade. This grade indicates that you may be falling behind, and should take steps to remedy this situation.
- *Unsatisfactory* means that you were absent from the lab period, or you attended but attempted few of the exercises or did not demonstrate or submit your work, or many of your solutions to the exercises had significant problems. This grade indicates that you are likely having difficulty with an important topic, and should seek help from your instructor as soon as possible.

For each s*atisfactory* or *marginal* grade, you will receive 1/1 towards the lab component of the course. Each *unsatisfactory* grade will receive 0/1.

Your lowest two lab marks will not be counted when calculating your final grade. This means you can have up to two unsatisfactory lab marks and still earn full marks (100%) for the lab component of the course.

If you are absent from a lab period for any reason, you will receive 0/1 for that lab. If you are unable to attend a lab because of illness, you are not required to provide a medical certificate to explain your absence. It is up to you to do the missed lab work on your own time; however, you cannot submit your completed lab work late to receive credit for the missed lab. Please do not ask for exemptions and/or extensions because of illness and so on. You can miss up to two lab periods and still receive full credit for the lab component, but it's up to you to use your "excused absences" wisely. Serious long-term illness will be dealt with on an individual basis; in these circumstances, please contact your instructor to discuss appropriate arrangements.

Assignments

There will be three or four programming assignments. Your lowest assignment mark will not be counted when calculating your final grade. This means you can miss an assignment and still earn full marks (100%) for the assignment component of the course. Late assignments will normally not be accepted. Please do not ask for exemptions and.or exceptions because of illness and so on - you have, in effect, one “sick day” to play with, and it is up to you to use it wise;ly. Serious long-term illness will be dealt with on an individual basis; in these circumstances, please contact your instructor to discuss appropriate arrangements.

Portions of the designs and code from any lab/assignment may be reused and refined in subsequent labs/assignments, and doing these components is the best way to learn the course material and prepare for the exams, so students are encouraged not to "write off" any particular lab/assignment just because of its relatively low weight in the overall grading scheme.

Students can use the Systems and Computer Engineering undergraduate computer labs whenever the Mackenzie Building and Minto CASE are open, except for those times when labs are reserved for specific courses.

Exams

There will be one closed-book midterm exam, which will be held approximately one-half of the way through the term. The date of the exam will be announced in class and posted on the course Web site.

Students who are unable to write the midterm exam because of illness or other circumstances beyond their control must provide in cases of illness a medical certificate dated no later than one working day after the exam, or appropriate documents in other cases. Medical documents must specify the date of the onset of the illness, the (expected) date of recovery, and the extent to which the student was/is incapacitated during the time of the exam. If this information is provided to the instructor no later than five working days after the exam, the weight of the final exam will be increased to cover the missed midterm; otherwise, the mark for the missed midterm exam will be 0.

Requests for accommodation because of poor performance on the midterm exam will not be considered. There will be no “make-up” midterm exam. So, if you are ill on the day of the midterm exam, don't write the exam and later claim that your performance was impaired because you were unwell. You are better off to miss the exam and request that the weight of your final exam be increased, by following the procedure outlined earlier.

A closed-book final exam will be held during the University's April examination period. The *Academic Regulations of the University* permit instructors to specify requirements that must

be satisfied for students to be eligible to write the final examination or, where circumstances warrant, the deferred final examination.

- All students are eligible to write the final examination, regardless of the marks they received during the term.
- Students who miss the final exam, but earned at least 60% in the lab and assignment component and wrote the midterm exam (or provided acceptable documentation to explain their absence from the exam) will receive the grade ABS. These students will be deemed to have performed satisfactorily during the term when their applications for a deferral of the final examination are considered. For more information, see the current Undergraduate Calendar, *Academic Regulations of the University*, Section 2.2, The Course Outline; Section 2.3, Standing in Courses/Grading System; and Section 2.5, Deferred Final Examinations.
- Students who miss the final exam but have not satisfied the conditions for receiving ABS, as listed above, will receive the grade FND. These students are ineligible to write the deferred final exam.

The final examination is for evaluation purposes only and will not be returned to students. You will be able to make arrangements with your instructor to see your marked final examination before June 30, 2012 (the last day for receipt of applications for review of final grades in Winter-term courses). Your exam will not be remarked during this meeting and solutions to the exam questions will not be provided.

Early Feedback
See Section 2.2.1 of the *Academic Regulations of the University*.

The weekly lab exercises will normally be graded during the lab period. Outside of the scheduled labs, you can obtain feedback during office hours or by making an appointment to see your instructor.

The midterm exam will be marked and returned prior to the 40'th teaching day of the term.

Academic Accommodations
You may need special arrangements to meet your academic obligations during the term. To request academic accommodation, the processes are as follows:

Religious Obligations
Email me with any requests for academic accommodation during the first two weeks of class, or as soon as possible after the need for accommodation is known to exist. For more information, refer to the *Guide to Academic Accommodation for Students*, which can be found at:

http://www2.carleton.ca/equity/accommodation/academic

Pregnancy
Email me with any requests for academic accommodation during the first two weeks of class, or as soon as possible after the need for accommodation is known to exist. For more information, refer to the *Guide to Academic Accommodation for Students*, which can be found at:

http://www2.carleton.ca/equity/accommodation/academic

Students with Disabilities
Students with disabilities who require academic accommodations in this course must register with the Paul Menton Centre for Students with Disabilities (PMC) for a formal evaluation of disability-related needs. Documented disabilities include but are not limited to mobility/physical impairments, specific Learning Disabilities (LD), psychiatric/psychological disabilities, sensory disabilities, Attention Deficit Hyperactivity Disorder (ADHD), and chronic medical conditions. Registered PMC students are required to contact the PMC, 613-520-6608, every term to ensure that I am sent a copy of your Letter of Accommodation no later than two weeks before the first assignment is due or the first in-class test/midterm requiring accommodations. If you require accommodations only for the formally scheduled final exam, you must submit your request for accommodations to the PMC by March 7, 2012.

For more information, refer to the *Students* section of the Paul Menton Centre Web site::

http://www1.carleton.ca/pmc/students

Topics

- Fundamental elements of imperative programming languages: types, variables, expressions, control flow: conditional statements, iteration (loops), subroutines.
- Introduction to/review of C as an imperative programming language:
    - types, variables, expressions, `if`, `while`, `for` and `do-while` statements, function definitions vs. function declarations.
- Function calls and the call stack. Visualizing program state by drawing activation records (stack frames) depicting parameters and local variables.
- Brief overview of the standard C library.
- Structuring data: arrays and character strings. Arrays and character strings in C.
- Motivation for modular programming. Modules: interface vs. implementation.
- Modules in C: header (`.h`) and implementation (`.c`) files. The C preprocessor. Compiling and linking C programs comprised of several modules. Brief overview of the standard C library.
- Pointers. Depicting pointers in drawings of activation records.
- C pointers. The `&` and `*` operators. Passing pointers to local variables as function arguments. C arrays and pointers. Drawing activation records to explain how pointers to arrays are passed as function arguments.
- Structuring data: structures.
- C `structs`. Structures vs. pointers to structures as function arguments. The `->` operator.
- Introduction to dynamically-allocated memory and the heap.
- Heap management in C: `malloc` and `free`. Dynamically allocated `structs`. Drawing memory diagrams to explain how parameters and local variables in activation records point to memory blocks allocated on the heap. Memory leaks.
- Dynamically-allocated arrays, dynamic arrays.
- Case study: C implementation of a list collection using a dynamic array.
- Pointer-based singly-linked lists.
- Case study: stack and queue collections implemented in C using a singly-linked list.
- The C preprocessor and macros.
- Introduction to recursion.

Last edited: December 20, 2011.