COMP 2103 CIN2: Computer Programming 3

Course Syllabus

Instructor

Dr Tomasz Müldner

Visit my page at the school website to learn more about me.

Contact Information:
Dr. Tomasz Müldner, Professor Emeritus
Jodrey School of Computer Science
Acadia University
Wolfville, Nova Scotia
CANADA B4P 2R6

Home: Ottawa
E-mail: tmuldner@gmail.com

Asking questions

Please feel free to contact me at any point throughout the course about this course syllabus.

While you learn a programming language, a programming paradigm, or a single concept, it's good to ask questions. However, to learn well and memorize new concepts, you have to follow these steps:

1. When reading about a specific concept in the textbook, experiment by writing very simple programs, then compiling them, and finally carefully testing. For example, reading about simple I/O, write a program to experiment with I/O, compile it and test.
2. If you do have any questions, **first** try to find answers in your textbook, possibly using the index in this book (found at the end of the book). For example, the term "closing files" appears in the index of the textbook, and refers you to the proper page(s) with the explanation of these concepts and examples. If you **still** need more explanation, then go to the step 3.
3. Use Google. Example: If you enter "**C closing files**" into Google, you will get multiple hits. If you still have questions, go to step 4.
4. Use the forum to ask your questions (do not provide there your solutions). **Only** if you still have questions, go to step 5.
5. Email me your questions (one at a time), **and always provide your suggested answer**.

Course Overview

This course will cover programming in C, and introduction to systems programming; based on **your experience with Java**. The course will be on programming techniques with special attention given to the following topics:

- program correctness
- program robustness
- program portability
- debugging, documentation and maintenance of C programs.

Prerequisite: COMP 1113, or permission of the School of Computer Science.

This course will also cover the introduction to shell programming and writing makefiles.

The course is divided into 10 sections. Each section consists of several of the following parts:

1. A preview
2. A presentation with an overview of the material covered in this module, followed by the references to the related textbook material.
3. One or more videos to help you understand the material (in total, there are 11 videos).
4. A quiz that you will have to pass to continue to the next module.
5. An assignment with one or more programming questions, which you will have to write and submit your solutions (and get it approved) before you can read the following section, note that your solution must be approved by the instructor (i.e., you must receive at least 50% of your mark).

Note: all presentations are in PDF formats and you can view them in your favourite PDF viewer (you can copy them from the browser, but you cannot edit or print them). For viewing, for example in Mac OSX Preview, it is recommended that you choose the "single page view", and also select the "Table of contents" so that you easily move from one part of the section to another.

Course Materials

The textbook for this course are:

- Müldner, T. (2000). *C for Java Programmers*. Boston, MA: Addison Wesley, ISBN: 0201702797. (For more about this book see below).

For shell programming you **may** use the following book:

Unix Shell Examples, by E. Quigley. Prentice Hall

You may also use any other book that covers shell programming as long as it is a standard **bash** shell rather than a version of it, such as C-shell; or simply use Google to find descriptions of bash, in particular this free book.

Note: The textbook covers portable ANSI C and provides a list of typical errors, a list of 59 idioms, and over 100 examples of C programs as well as over 190 exercises. Remember to use the **index** if you need to search for a specific construct, e.g., "function definition".

You can download the code for all examples and solutions to some exercises from the textbook using links below. Also, you should download the **errata** to the book, and use it when you read the book.

See the student handbook for ordering information.

Evaluation

Assignments (10 at 4% each)	40%
Final Exam	60%

Assignments

You are required to complete ten assignments before you can write the final exam. Each assignment includes one or more programming question. With each assignment, you need to submit the following files (compressed using gzip or tar):

- The solution (**source code for all required files, NEVER submit object codes**);
- **Script** file of the sample executions, which carefully tests your implementation (see below).

Using a Unix-like box, the script can be obtained as follows:

```
change the current directory to the directory storing source files and other needed files

script fname          (any file name is OK, the output from the script will be stored in that file)

ls -l                (to show file names and their dates)
```

remove any object files present in the current directory

compile all required source files

ls -l (to show file names and their dates)

next, run all required executable files

finally, press CTRL-D to terminate the script.

Your programs have to be properly structured and follow programming guidelines introduced in the following sections.

It is expected that for each section you will spend at least ten hours studying and programming.

A program consists of one or more files. Each source file **must contain the name of the author, the file name, and the date when this code has been developed.**

The assignments are delivered to the instructor via **assignment drop-boxes**. Please remember to keep a copy in the event the original is lost.

Very important notes:

- Assignments have to be done **in order, *one at a time***, i.e., after you submit the first assignment you have to wait until it is evaluated and **approved** (and you will receive your mark in the email) before you submit the next assignment, and so on.
- For any assignment where the mark would be a failing mark (less than 50%), the assignment will need to be resubmitted, and *you will lose 20% on **this** assignment*. The same procedure will be used for subsequent submissions of the same assignment (when a resubmission is required).
- Avoid typos (a large number of typos will be penalized by up to 10% on the assignment), and avoid warning produced by the compiler (make sure that you modify your code to get rid of warnings because warnings will be penalized by up to 10% on the assignment).
- Word files will **not** be accepted, use **PDF** files instead.
- Copying of assignments is not allowed and ***will result in F (zero) for the course***. The instructor may use Moss - a system for detecting Software Similarity to prevent plagiarism.
- Final exam is **closed-book** and no electronic devices will be allowed.
- ***You have to pass all assignments and have them accepted by the instructor before you will be allowed to write your final exam.***
- ***Note that based on Open Acadia rules your final mark from the final exam must be greater than or equal to 50% to pass the course.***

The last assignment should be received at least 4 weeks prior to the date you wish to write the exam. This will allow adequate processing time for the request and for setting the exam.

Course Contents

Section	Material	Textbook Reading	Textbook Assignment/Exercise
1	1. Development Process. Procedural paradigm versus OO paradigm. Lexical conventions. Portability. Idioms. Introduction to C. Lexical structure. Primitive data types. Main program. 2. Terminal I/O 3. Control structures. Programming idioms, testing and debugging	Chapters 1-4.	**Assignment 1 (2 parts)**
2	1. Text files 2. Preprocessing and Conditional Compilation 3. Functions and documentation 4. Introduction to shell.	Chapters 5-6, Chapter 7.1-7.2. Any bash shell reference.	**Assignment 2 (2 parts)**

3	1. Scope. Introduction to modules. Shared and private variables. Linkage. Interface and implementation 2. Types of modules 3. Extending and modifying modules. Overloading functions. 4. More on shell and the use of makefiles.	Chapter 7.3-7.9. Any bash shell and makefile reference.	**Assignment 3 (2 parts)**
4	1. Pointers: stack and heap based memory management 2. Using Pointers 3. Pointer Arithmetic	Chapter 8.1-8.12	**Assignment 4 (2 parts)**
5	1. Working with memory blocks. Binary blocks 2. Pointers and Functions. Generic pointers. Pointers to blocks containing pointer	Chapter 8.13-8.17	**Assignment 5 (2 parts)**
6	1. Module-based programming. Generic and concrete modules. Homogenous collections. Enumerations 2. Strings. String library	Chapter 8.18 Chapter 9	**Assignment 6 (3 parts)**
7	1. Arrays. Preconditions and Dynamic Arrays	Chapter 10	**Assignment 7 (1 part)**
8	1. Structures and binary files. Introduction to lists 2. Module-based programming: Opaque types	Chapter 11.1-11.10	**Assignment 8 (2 parts)**
9	1. Enumerated Data Types and Unions 2. Bitwise Operations	Chapters 12-13	**Assignment 9 (1 part)**
10	1. Module Based Programming 2. Characterization of Modules	Chapter 14	**Assignment 10 (1 part)**
Final Exam			

Course Schedule

Click to download the suggested schedule for this course: COMP 2103 CIN2 - Suggested Schedule

You may wish to print out this schedule and fill in your start date to use the recommended timeline to plan out when you will do readings and assignments. This is a tool to help you plan and time manage this course. If you get off-track, make sure to revisit your schedule and re-evaluate the dates you've set for yourself.

You have six months to complete this course. You may set your own schedule, but if you intend to complete the course in less than 3 months, you should let me know so that we can arrange a schedule.

Please do not leave all of your course work until a few weeks before your completion date. Although I will make every effort to accommodate your schedule within reason, I need time to grade assignments and mark exams.

Exam

How to apply: Complete the Application for Examination (https://openacadiaexams.acadiau.ca)

Proctored at Acadia

• The final exam in an online course must be passed to successfully pass the course unless otherwise stated in the assessment section of the course syllabus. There are no rewrites or supplemental examinations at Acadia University.

• Examination requests must be received one month prior to the date you wish to write your examination.
• Course requirements must be completed to the satisfaction of your instructor.
• Graduating Students Note: If you are graduating in Spring Convocation you must write by April 15th. If you are graduating in the Fall you must write by September 15th.

Proctored at Another Location

If it isn't practical to take your exam at Acadia, off-campus exams can be written at another university or college. Arrangements for an examination may be made through the Registrar's Office or the Continuing Education office of most universities and colleges. If it is not possible to write your exam at an approved institution, please contact us for assistance.

• All fees associated with examinations written at other locations are your responsibility.
• Some courses may require specific software or internet accessibility at the off-campus examination location.

Student Handbook

You are responsible for becoming familiar with the contents of the Student Handbook. It contains important information about scheduling examinations (if applicable), applying for extensions, withdrawing from your course, ordering books, and computer and library services available to you. If you have questions about the policies outlined in the handbook (https://courseware.acadiau.ca/openacadia/studenthandbook.html), contact:

Open Acadia
• 21 University Avenue (Rhodes Hall)
• Wolfville, NS B4P 2R6
• Phone: 1-800-565-6568
• Fax: 1-902-585-1068
• Email: openacadia@acadiau.ca

Academic Integrity

Academic integrity demands responsible use of the work of other scholars. It is compromised by academic dishonesty such as cheating and plagiarism. A student who is uncertain whether or not a course of action might constitute cheating or plagiarism should seek in advance the advice of the instructor involved.

• Cheating is copying or the use of unauthorized aids or the intentional falsification or invention of information in any academic exercise

• Plagiarism is the act of presenting the ideas or words of another as one's own. Students are required to acknowledge and document the sources of ideas that they use in their written work.

• Self plagiarism is also a form of plagiarism. It is the presentation of the same work in more than one course without the permission of the instructors involved.

• A student who knowingly helps another to commit an act of academic dishonesty is equally guilty.

• Penalties are levied in relation to the degree of the relevant infraction. They range from requiring the student to re-do the piece of work, through failure on that piece of work, to failure in the course, and to dismissal from the university.

Last modified: Tuesday, 3 August 2021, 3:37 PM

Jump to...

COMP 2103 N2 Suggested Schedule ►