Syllabus for

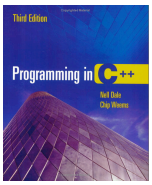
Robots, Games, and Problem Solving
COMP 115

Instructor: Mark LeBlanc (mleblanc@wheatoncollege.edu)
http://cs.wheatoncollege.edu/mleblanc
Office: SC-B103
Phone: 286-3970 (on campus: x3970)

Office Hours: by appt. *or*
T/W 2:30-3:30

Lecture: MWF 9:30-10:20
Lab: W 3:30-5:20

Required Text:



Programming in C++ (3rd edition), by Dale and Weems, Jones and Bartlett, 2005.

I also *strongly recommend* that you buy a USB flash drive to store your work as as well as a 3-ring binder. I'll pass out lots of handouts.

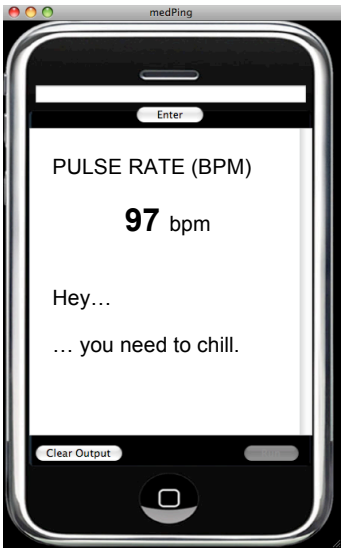
Preamble:

Programming. There seems to be no end of the urgent need for more and more people to know how to script, how to write software, how to program. Mobile phones, mathematical models, embedded medical devices, video-cams, scientific experimentation, and modern desktop computers need software to do the things they do. This course is about learning how to program and how to do it well. We use the programming language C++ because we feel it is still the backbone of the industry and offers a full range of rigor and elegance. Glad you are on board; like riding a bike, once you learn to program, you'll never forget. Let's cut code ....

Content:

Problem-solving techniques and algorithm development with emphasis on elementary objects, software reuse, and numerical methods. Topics covered include abstraction, design and decomposition, elementary data structures of arrays and records, and an understanding of how these features form the building blocks of user-defined classes of objects. Avatars (virtual

robots) in the ALICE environment will help you begin begin to think in terms of objects and the algorithms that operate on those objects. Games will provide examples for some programming concepts. A number of assignments will use a software emulation of an embedded medical chip under the skin of a person. Called "medPing", the chip performs real-time monitoring of one's "health" (e.g., blood pressure, pulse rate, glucose level) and automatically beams the medical status to an iPhone display window.



No previous programming experience is assumed. Out-of-class assignments and in-class labs emphasize the physical limitations of problem-solving-machines as well as techniques to write programs that are both safe and correct.

In computer science, if you are almost correct you are a liability.
Fred Kollett (1941-1997), Math/CS, Wheaton College

Your Grade:

Things to do	Grading Percents	Frequency
Labs	10%	weekly
7 Programs	50%	TBA
Quizzes (approximately 10)	10%	TBA
Exams (two)	15%	Exam1 Feb 23rd Exam2 April 10
Final Exam	15%	Sat, May 9th, 2pm

Late Submissions:
Due is due. Always turn in whatever you have on time. Something turned in on time is much better than not having it accepted because it is late. Late is not an option. (Good, glad we can all agree with this). Note that the Blackboard (Bb) page associated with this course will have submission areas that are time triggered; if you are late, submissions will be blocked. Also, Bb only allows one submission per assignment. If you need to resubmit material and the due date has not passed, email the instructor and I can clear your previous submission.

Honor Code Revisited:
It goes without saying that all submitted work will be the student's own, in keeping with the Wheaton Honor Code, unless the assignment has assigned groups. For labs, you may get "help" from fellow classmates, but remember that all completed work must be your own. Use discretion; don't ask your colleague for "the" answer or for lines of code. However, I do encourage you to discuss the problem in general, such as the type of statements or functions one might use. For homework, your answers and software must be your own from beginning to end. Here is an analogy. Almost no one would every "use/steal" a line or two from another person's poem. Consider it the same with your programs. Don't "borrow/use" lines or sections of code from another classmate. Your program is (like) your poem; everyone's program should be unique. Be wise. If a colleague is asking you for too much help, be honest and remind them your program is just that, *your* program.

Homework: It is expected that you spend at least 2-3 hours on reading and practice problems for every 50 minutes of lecture. This computes to at least 6 hours of work in the text per week. This should be done throughout the semester and not just when studying for quizzes or exams. The material is cumulative in a big way; for example, week 5 depends heavily on weeks 1 through 4. It is expected that you spend at least 6 hours per week on your current programming assignment. WARNING: Programmers typically underestimate the time it takes to complete a software project; 6 hours per week on your programming assignment may be one of those "underestimations."

Labs: The labs are a critical part of the course. In almost all cases, the current lab will be preparing you for the current programming assignment. That is, if you complete and understand the lab, you should be well on your way to a solution for the programming assignment. In order to best grasp the material in some labs, I strongly suggest that you completely redo any labs that you find difficult. (Read that last sentence again, unless of course you've already reread it once).

Quizzes: In addition to two exams spaced evenly over the semester, you will be quizzed on a (near) weekly basis. Quiz dates will be announced in lecture, e.g., in Friday's lecture I may announce a Quiz on Monday. Each quiz may address material that has been assigned in the text, but not yet presented by the instructor. NOTE: you'll have to read the material and practice the sample problems BEFORE coming to class in order to be successful with the quizzes. Quizzes start at the beginning of a lecture and last approximately 10-15 minutes. After some quizzes, immediate (peer) grading will be followed by a discussion of a question on the quiz in order to provide you with immediate feedback on how you are mastering the material.

Quizzes and Exams: There will be no make-ups, nor will the lowest grades be dropped. If you are an athlete and/or you have a conflict with a quiz or exam date, please see me. When studying for quizzes and exams, have someone pick problems from the notes and text, mix them up, and present them to you as if you were taking a quiz or exam (not telling you from what section a problem comes). While practicing, do not peek at the textbook or an earlier solution. NOTE: its easy to "read" code; its much more difficulty to "write" code from scratch. Quizzes and exams will sometimes ask you to write code from scratch; practice ownership of the concepts!

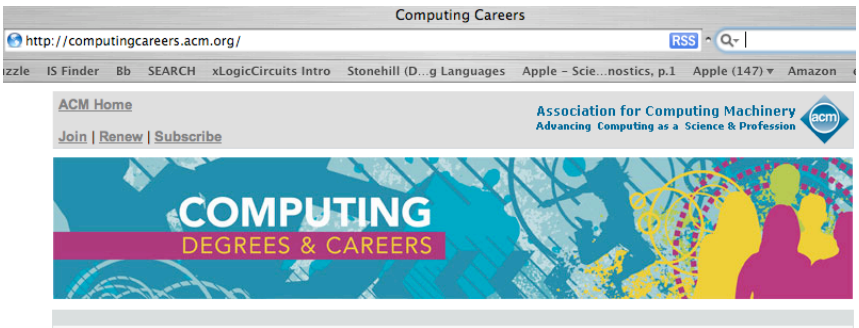
Working with **MacOS X and Xcode**: C++ programming assignments will often require the use of the medPing API and thus you must work on a computer with MacOS 10.5 and Xcode v3.1. The following labs are available for after-hours work:

A102 (our lab room)	Sun/Mon/Wed 7-11pm, Tue/Thur 9-11pm
ICUC (2nd floor science center)	Sun-Thur, 7-10:30pm
Graphics Design Lab (2nd floor Meneely)	Sun-Thur, 12noon - midnight

HELP
I have listed my office hours on the syllabus, but be bold: schedule a time to meet. Study, study, study and talk about the material with me as often as you can.

Please don't wait too long before you see me;
a quick chat in my office can often clear things up.
I'm here a lot...

Wondering what you can do with a major or minor in computer science?
Check out this website to learn of the exciting potential of a career in computing ....
`http://www.computingcareers.acm.org`



Week	Topic Note: At the start of each lecture, I will write on the board the exact pages that are associated with the material covered in this class meeting.
1	**Jan 21/23** Introduction Working in ALICE: - breathing life into your avatar avatar, *n.* "computer user's representation of himself/herself or alter ego" - object-oriented lingo; algorithms (computing "recipes") - control structures: sequential, conditional, and repetition (Did I say, repetition? Did I …)
2	**Jan 26/28/30** Problem decomposition The importance of documentation and good coding style Working with random number generators `if-else` and `while` statements handling events
3	**Feb 2/4/6** Introduction to C++ Welcome to "medPing" variables, types, arithmetic operators, expressions a peek at BNF rules for valid variable names working with the Xcode IDE and the Debugger *Monday, Feb. 2: a0 due – Your ALICE Avatar*
4	**Feb 9/11/13** mathematical functions, writing mathematical expressions casting more Input/Output (I/O) - from the keyboard, to the console - from the medPing chip, to/from the iPhone Syntax, Linking, Logical, Runtime "Bugs" *Wednesday, Feb. 11: a1 due*
5	**Feb 16/18/20** selectional control – `if-else` statements in-depth dangling else, trapping potential errors, lazy evaluation English to boolean conversions the `switch` statement *Friday, Feb. 20: a2 due*

6	**Feb 23 25 27** Exam1 – Monday, Feb 23rd Repetitional control – `while`-, `for`-loops
7	**Mar 2 4 6** Introduction to numerical methods ✓ algebraic results are not necessarily equivalent to computational results ✓ never compare two real's for equality, rather agree on "close enough" ✓ always trap o division by zero o overflow o bad arguments to library functions, e.g., `ln(-2), sqrt(-3)` o use explicit casting Fencepost problems Introduction to file I/O *Friday, March 6: a3 due*
8	**Spring Break** – Monday, March 9 – Friday, March 13 – **Spring Break**
9	**Mar 16/18/20** More file I/O: reading files of DNA Working with/on strings Manipulating characters as data: encryption
10	**Mar 23/25/27** User-defined functions/procedures and methods Pass by value vs. pass by reference *Friday, March 27: a4 due*
11	**Mar 30 Apr 1/3** Intro to Data Structures: Arrays Parallel arrays Passing arrays to functions
12	**Apr 6/8/10** Arrays revisited Linear Search Inside strings: arrays of characters Scope Introduction to "Big Oh" notation: O(lg(n)), O(n), $O(n^2)$, $O(n^3)$, …, $O(2^n)$ Exam2 – Friday, April 10

13	**Apr 13/15/17** Vectors in the Standard Template Library (STL) *Wednesday, April 15: a5 due*
14	**Apr 20/22/24** Intro to Data Structures: Records (called "`structs`" in C/C++)
15	**Apr 27/29 May 1** Intro to Data Structures: two-dimensional (2D) arrays: matrices Summary of introductory numerical methods: *"the curse of being almost correct"* *Friday, May 1: a6 due* Evaluations