CMSC 10600
Fundamentals of Computer Programming II (C++)

Department of Computer Science
University of Chicago

Winter 2011 Quarter

Dates: January 3 through March 19, 2011
Lectures: TuTh 12:00-13:20 in Ryerson 251
Labs: Thu 3:00-4:20 and 4:30-5:50 in Maclab (A-level of Regenstein)
Website: `http://www.classes.cs.uchicago.edu/archive/2011/winter/10600-1/`

Lecturer: Borja Sotomayor
E-mail: borja@cs.uchicago.edu
Office: Searle 211-A
Office hours: Open door policy (see page 6)

Lab TA: Sonjia Waxmonsky
E-mail: wax@cs.uchicago.edu
Office: Ryerson 177

TA: Tatiana Orlova
E-mail: orlova@cs.uchicago.edu
Office: Ryerson 178

Contents of this Document

Course description

This course provides an introduction to computer programming and to software development. The first portion of the course introduces students to computer programming, using the C/C++ language, and covers fundamental topics such as flow of control, function definition, data structures, and object-oriented design and programming. The second portion of the course provides a more holistic view of software development and introduces students to a selection of tools, libraries, and additional languages that programmers should be proficient in to become effective software developers, with an emphasis on the Python programming language. Topics in this portion include using build tools, third-party libraries, scripting languages, and data storage.

This course takes a practical approach to teaching computer programming. Learning a programming language is similar, in some respects, to learning how to speak a new language. To become a fluent speaker, it is rarely enough to learn the syntax, grammar, and vocabulary of the language, even if we can memorize them letter by letter. It is generally necessary to converse and interact frequently with native speakers to pick up all the nuances of the language. Similarly, the best way to learn computer programming is by *writing programs*. This course will involve homework assignments, labs, and a two exams, all of which require solving programming problems on a computer. There will be no "paper and pencil" exercises.

Prerequisites

Some students may need to take CMSC 10500 to meet general education requirements. However, this course is designed as a standalone introduction to computer programming, and can be taken by students from any background who need to develop programming skills.

Course goals

By the end of this course, students should be able to

— Write, compile, and run C/C++ programs on their own.

— Write and run Python programs on their own.

— Given a programming problem, compare and contrast the various programming languages, tools, and technologies that could be used to solve that problem, and choose the combination that is best suited for that problem.

— Learn new programming languages on their own.

Course organization

The class meets two times a week for lectures, where fundamental concepts will be explained, and once a week in a computer lab where each student will be able to apply these concepts in front of a computer. The course calendar, including the contents of each lecture, is shown in Table 1.

Homework

There will be a weekly homework assignment handed out in class on Thursday and due on Wednesday before 5pm. These assignments involve solving programming problems related to the topics covered in class so far.

Labs

There will be a weekly lab session to complement the class lectures. Attendance to the lab sessions is mandatory, and students will be expected to hand in the results of an exercise at the end of the lab.

Exams

There will be a midterm and a final exam.

The midterm will be a take-home exam that will require solving a substantial programming exercise in ∼48 hours. It will be assigned on February 8 at 1:20pm (at the end of class), and is due February 10 at 12:00pm (at the start of class). There will be no homework due that week. Although you are allowed to consult external references (books, Internet resources, etc.) to solve this exercise, you are expected to do the exercise *entirely on your own*. Unlike homework assignments, you must not discuss the exercise with anyone.

The final exam will take place in the Maclab and will involve solving a set of programming problems on a machine (there will be no paper-and-pencil exercises). We will not know the exact logistics of the final exam until we have an accurate registration count for the course, as this will affect whether we need to do the exam in two batches, or can fit everyone in the Maclab for a single exam session. We will announce the exact date and time of the exam by the end of the second week, once all course add/drop requests have been processed. We are *tentatively* considering holding the exam during the last class of the quarter, or during the time allocated for this course during Finals Week (Thursday 10:30-12:30).

Books

The text for this course is Absolute C++, 4th edition, by Walter Savitch, published by Addison-Wesley. The book will be available for purchase from the Seminary Co-op Bookstore (5757 South University Avenue).

Grading

Grading will be based on homework (30%), labs (20%), midterm exam (20%) and final exam (30%).

Types of grades

Students may take this course for a quality grade (a "letter" grade), a pass/fail grade, or as an auditor. Students will declare on the final exam whether, depending on their final grade, they want to receive a letter grade, a pass/fail grade or withdraw from the course (a *W* grade). For example,

Table 1: CMSC 10600 Winter 2011 Calendar

Week	Date	Lecture	Lab
1	4 January	Introduction to computer programming. Variables, data types, expressions, assignment.	
	6 January	Operators, basic I/O, invoking functions, writing functions	Using gcc. Simple programs.
2	11 January	Branching structures, looping structures	
	13 January	Pointers, arrays, memory allocation	Debugging with Eclipse.
3	18 January	Complex data types.	
	20 January	File I/O.	Files
4	25 January	Make. Separate Compilation.	
	27 January	Linked lists.	Data structures
5	1 February	Introduction to object-oriented programming. Classes, constructors, encapsulation.	
	3 February	Inheritance, polymorphism	Object orientation
6	8 February	Third-party libraries.	
	10 February	C++0x	Installing and using a third-party library
7	15 February	Python	
	17 February	Python	Python
8	22 February	Storing data: Flat files	
	24 February	Storing data: Structured formats	Working with data
9	1 March	Introduction to databases.	
	3 March	SQL. SQLite. Pysqlite.	SQL
10	8 March	Python 3	

students can declare "If my final grade is a C+ or lower, I will take a P (Pass) instead of a letter grade and, if my grade is an F, I wish to take a W".

Students are welcome to audit this course. However, be aware that a student who is auditing the course cannot switch to a letter grade or a pass/fail grade, and will receive an R (Registered) grade.

> Note: *Students taking this course to meet general education requirements must take the course for a letter grade.*

Late assignments

There will be no deadline extensions and late submissions will receive no points. Nonetheless, you are allowed two 24-hour extensions, without any penalty, to be used at your discretion on any of the homeworks. Before the deadline, you must inform the TA that you plan to use an extension or your assignment will be marked as late.

Failing to submit three homeworks will result in a failing grade for the course.

Policy on academic honesty

The University of Chicago has a formal policy on academic honesty which you are expected to adhere to:

`http://www.uchicago.edu/docs/studentmanual/academic_honesty.shtml`

In brief, academic dishonesty (handing in someone else's work as your own, taking existing code and not citing its origin, etc.) will *not* be tolerated in this course. Depending on the severity of the offense, you risk getting a hefty point penalty or being dismissed altogether from the course.

Even so, collaboration between students is certainly allowed (and encouraged) *as long as you don't hand someone else's work as your own.* If you have discussed parts of an assignment, then make sure to say so.

As for consulting other sources, this can be a thorny issue, as programmers are always encouraged to not reinvent the wheel and reuse as much code as possible. However, in this course you are also required to show that you are capable of effectively coding a solution to a given programming problem. So, in no case will you be allowed to hand in an existing solution to a problem (e.g. taken from a website) even if you do cite its origin. In general, you are allowed to look at existing code in search of inspiration, as long as you cite the sources you consulted. For large programming problems (which will turn up later in the course), you will be allowed (and encouraged) to directly use code taken from other sources (web sites, programming books, etc.) as long as this code is used to solve a particular subproblem and not the entire problem.

If you have any questions regarding what would or would not be considered academic dishonesty in this course, please don't hesitate to ask the instructor.

Asking questions

This course has an *open door policy* for asking questions. Instead of setting fixed office hours, you are welcome to consult with the instructor at any time. Nonetheless, you should try to give the instructor, whenever possible, some advance warning of your visit (by e-mail) to make sure that he will be in the office at that time.

The preferred form of support for this course is though the *course mailing list*, which can be used to ask questions and share useful information with your classmates. In fact, we encourage that all questions about homework assignments, labs, and programming in general be sent to the mailing list, and not directly to the instructor. The reason for this is that, this way, all your classmates will be able to benefit from the reply to your question. Furthermore, the instructor will usually be able to reply to your e-mail faster than it would take you to walk to his office and back. In some cases, some of your classmates might even pitch in to provide their insights into questions or issues discussed in the mailing list.

You can subscribe to the mailing list in the following web page:

`http://mailman.cs.uchicago.edu/mailman/listinfo/cmsc10600`

How to hand in assignments

Assignments must be submitted electronically through `hwsubmit` and as a hard copy. You must electronically submit your assignment before the stated deadline, but you *do not* have to submit your hard copy by that deadline. The hard copy can be submitted in the first lecture after the deadline. The hard copy must correspond to the version you submitted electronically.

To submit an assignment, you must perform these steps:

1. Submit your code through `hwsubmit` before the due date.

2. Print a hard copy of your code. An easy way to print code from a UNIX system is using the `enscript` command. This command will automatically format the code for you, and can handle most programming languages. For example, to print out C++ code, you could run enscript like this:

   ```
   enscript foo.cpp -P printer_name -Ecpp
   ```

   Where *printer_name* is the printer you want to send the code to. See the enscript man page for more details on using this command, and for a list of languages that enscript can handle.

3. Each homework/lab handout includes a grading sheet where you will need to write down your name and student ID. Take this grading sheet, fill it out, and keep the rest of the handout.

4. Staple all your pages in order: grading sheet first, code second. Make sure your code printout includes your full name on it.

5. The hard copy must be handed in at the beginning of Thursday's lecture.

For each assignment, you will be expected to hand in the following items:

Source code and documentation. All the code you've written, and any accompanying documentation (if required by the assignment). You will hand this in using the `hwsubmit` command (described below)

Printout of source code [Optional] If you provide a printout of your code and documentation, the instructor will return it with annotations and comments, along with the grading sheet. If you do not, you will only get back your grading sheet (which will make it difficult for you to see what you did right and what you did wrong)

The due date

hwsubmit

This command is available if you log into any of the Linux machines in the Maclab. Make sure all the files you want to hand in are inside a directory, and then run hwsubmit like this (where dir_name refers to the directory you want to submit):

```
hwsubmit cmsc10600 dir_name
```

For example, assuming that you are currently inside your home directory, and that you placed all the files for a particular assignment in directory /home/myusername/hw01, you would run hwsubmit like this:

```
hwsubmit cmsc10600 hw01
```

Practical information

There are certain things you will need to do before you can start using the Maclab computers, work on homeworks and labs, etc.

Obtain a CS account

Lab sessions will take place in the Linux section of the Maclab. Before using those machines, you need to request a CS account. This account will allow you to access certain computing resources in the Department of Computer Science, most notably the Linux machines in the Maclab. You can claim your CS account here:

`https://www.cs.uchicago.edu/info/services/account_request`

Knowing your way around a UNIX system

Although you will have the option of doing your homeworks and labs on UNIX or Mac in the Maclab, all lab instructions will be given from a UNIX system. If you are completely new to UNIX, you will find that the default graphical interface in the Maclab machines is very similar to the ones found on Windows and Mac systems. However, at a certain point, you will need to perform certain actions from the UNIX command line interface (or "console"). The first lab in the course will provide a basic introduction to the UNIX console. However, if you want a more complete introduction, you can take a look at the tutorials in the following page:

`http://itservices.uchicago.edu/docs/os/unix/`

Working from home

Although the Maclab provides an excellent work environment, with all the software you need to complete the lab exercises, you are certainly free to work from home. However, take into account that you will need to hand your homework in using the `hwsubmit` command described above. Since this command is only available in your CS account, you will need to log in remotely to a CS machine using SSH. Instructions on how to do this are available here:

`http://www.cs.uchicago.edu/info/services/new_users_guide`

Also, take into account that there are two ways of working from home:

1. If you have a UNIX system at home (such as a computer with GNU/Linux installed in it), you can do your homework assignment entirely in your machine, using whatever tools you prefer. Once your assignment is complete, you simply have to copy your files to your CS account home directory and submit them using hwsubmit. Just in case, you might want to make sure that your code works fine in your CS account, as it will be graded in one of the CS Linux machines.

2. Regardless of having a UNIX system or not at home, you can do all your work using your CS account. To do this, you will have to log into your account using SSH and then do the assignment using the tools and commands available in your CS account.