



NKU CSC 601 – Fall 2002

Advanced Programming Workshop

A practitioners' workshop on creative approaches to challenging, realistic programming and design problems; emphasizes the use of new C++ programming techniques and the critical evaluation of solutions.

instructor Kevin Kirby, 340 ST (859-572-5377, kirby@nku.edu, www.nku.edu/~kirby/)
Office hours: 10-11am MWF, 5-6pm TTh or by appointment.

website www.nku.edu/~kirby/courses/csc601/

scheduling 7:45 – 9pm Tuesdays and Thursdays.

prerequisites Regular admission to graduate program. CSC 364 or 501 or equivalent.

It is strongly recommended that most 500-level requirements for the MS degree be completed before this course. You should be comfortable programming in C++.

objective The purpose of this workshop is to develop participants' programming and problem solving abilities when confronted with advanced programming challenges "in the small." This complements the software engineering courses (CSC 440, CSC 640), which deal with software systems "in the large" throughout their entire life cycle.

Assuming a mastery of low-level (pointer-based) programming and familiarity with high-level object-oriented programming, as well as a good grounding in data structures, this course consists of a sequence of projects presented as challenges to strengthen participants' adeptness at solving specific programming problems. We will judge the efficacy, generality, portability, clarity and elegance of various solutions. We will compare and employ appropriate contemporary programming paradigms, *with a very strong emphasis on newer uses of C++ language features*. We study the Scheme language to understand the influence of functional programming on the C++ standard library. We will read articles on from recent issues of periodicals such as *Dr. Dobbs' Journal*. Participants will also present tutorials on important new programming technologies to the group.

- | |
|---|
| <p>topics</p> <ol style="list-style-type: none">1. A intensive review of object oriented C++ (\approx 2 weeks)2. Functional programming in Scheme (\approx 2 weeks)3. C++ templates, advanced generic programming and the STL (\approx 2 weeks)4. Patterns via template metaprogramming (\approx 3 weeks)5. Selected advanced topics, by popular demand (\approx 3 weeks)6. Tutorials (\approx 3 weeks) |
|---|

- textbooks**
1. *The C++ Programming Language, Special Edition*. B. Stroustrup (Addison-Wesley, 2000).
 2. *Modern C++ Design*, A. Alexandrescu (Addison-Wesley, 2001).

Optional references:

- *The ANSI Scheme Programming Language*, K. Dybvig (Prentice-Hall, 1996).
- *STL Tutorial and Reference Guide*, 2nd Ed., D.R. Musser et al (Addison-Wesley 2001).

An extensive set of handouts will be available in PDF on the course website.

coursework 4 solo programming assignments (40%)
1 pair programming assignment, in teams of two (15%)
1 tutorial on a contemporary programming topic, prepared & presented by teams of two (15%)
1 midterm, in-class, Thursday October 17 (15%)
1 final exam, take-home (15%).

drop date Friday October 26 is the last day to drop with a grade of “W”.

platform requirements A standard C++ compiler. Visual C++ 6 will be fine for most work, but version 7 is better. The Borland command-line compiler (v. 5.5), available free from www.borland.com/bcppbuilder/freecompiler/ and is quite good.

We will also use the Scheme system for Windows from PLT, also free, from www.cs.rice.edu/CS/PLT/packages/drscheme/ (a 4.6MB download).

- general comments**
1. Attendance is required but not recorded.
 2. Class participation is important; our 75 minutes sessions are meant to be dynamic and productive. After each class, I will put my outline of what we talked about on the course website. Use these outlines to help you review your own notes.
 3. Programs are graded on style and design as well as on nominal correctness. Take pride in your work and polish it to a high gloss. Start early, test continually.
 4. Programs should be submitted via email, as text file attachments (source code only).
 5. I will accept one late program; you may turn it in up to 3 days late with no loss of credit. Graded programs will be returned within one week.
 6. Assignments must be done individually unless stated otherwise on the handout. Evidence of collaboration in coding, or appropriation of code from a source not authorized by the instructor, may result in a grade of F for the course.
 7. Some of your code may be displayed in front of everyone for purposes of discussion. Be supportive but frank in your public comments on fellow participants’ code, and accept comments on your own code with patience and openness. (This is the creative writing workshop model.)
 8. We are in a workshop to share our expertise and experience. If you write code for work, bring in some good stories. Think hard about what you *want* to learn in this area, and let us know.
 9. Ask questions.
 10. Have fun.

Note: “The instructor reserves the right to alter the syllabus if circumstances dictate.”