

Astro 503: Astronomical Methods and Instrumentation

Special Edition 2006: Programming for Physicists

Professor Gary Bernstein
4N1 DRL, 3-6252
garyb@physics.upenn.edu

Course hours: TBA
Course website:
www.physics.upenn.edu/~garyb/Astro503

Summary

In this course you will learn a modern object-oriented programming language (C++); techniques for producing reliable, reusable, and efficient code; and some algorithms that are ubiquitous in numerical solutions of problems in astronomy and physics, such as numerical integration, matrix operations, fast Fourier transforms, and model-fitting. By the end of the semester, you should have the skills and knowledge to solve the numerical problems that arise in your own research. I will *not* assume previous experience with any programming language, though of course this will help you.

Textbooks:

There are thousands of books on the topics of programming and numerical analysis. I have selected two that I have found particularly useful:

Thinking in C++: Introduction to Standard C++, Volume One (2nd ed) by Bruce Eckel (Prentice Hall) 2000

Numerical Recipes in C: The Art of Scientific Computing (2nd ed.) by William H. Press, Saul A. Teukolsky, William T. Vetterling, & Brian P. Flannery (Cambridge University Press) 1992

Both of these books are available for free online: mindview.net/Books and www.nr.com. I will post links to additional useful resources and books on the course website – let me know if you find any.

Grading

The course grade will be based on problem sets, once per week or so, and 1 or 2 large programming projects. There will be no final exam but there will be a final programming project. Weekly assignments will be primarily in the form of programming tasks. Grades will be based on the correctness of your code, with speed and elegance being designated as criteria on some assignments. Problem sets should be entirely your own work, but the large projects will be done collaboratively.

Computing

I will expect that you have your own computer, equipped with a C++ compiler that is compatible with the current ANSI standard. This would likely mean either the GNU compiler (g++ version 3.x or above), the Intel compiler (icc), or the Microsoft compiler. I will also expect that you have a text editor (or some program editor) and know how to use it. You'll be responsible for maintaining a functional configuration on your own machine.

Syllabus

Week of	Material
Jan 9	Machines and abstractions; intro to object-oriented programming; basic C/C++ syntax and compile/link/run sequence
Jan 16	C/C++ syntax, libraries, headers
Jan 23	Objects, access control, constructors/destructors
Jan 30	C++ extras: const, pointers, inline, namespaces, references
Feb 6	Overloading; dynamic allocation
Feb 13	Inheritance, composition, polymorphism
Feb 20	Template programming; STL basics
Feb 27	Data structures; STL
Mar 6	<i>Spring Break</i>
Mar 13	Exceptions, debugging, good programming practices
Mar 20	Linear algebra: algorithms and efficiency
Mar 27	Searches, sorts, integration, differential equations
Apr 3	Optimization and model-fitting
Apr 10	FFTs and image processing
Apr 17	The Final Project??

The first half of the course will cover the C++ programming language, with an emphasis on the concepts of object-oriented programming and how they make life easier for scientific programmers. In these sections we will mostly follow the Eckel book, until we reach the Standard Template Library (STL). We will use common scientific problems as our examples and exercises to keep things a little interesting.

The second half of the course will emphasize numerical methods, primarily following *Numerical Recipes*. Experts in the field (I'm not one of them!) often complain that this book does not give the best algorithms, and it certainly has poorly implementations in many cases. But it does serve as a good place to start for non-specialists like us.

There is an endless range of topics to cover, so I have necessarily picked only a few. Missing are: hardware, device, or systems programming; finite-element methods; multithreaded, distributed, or high-performance computing. The choice of topics and projects will be guided by the interests of the students in the course.