

Generic Software Design

Rensselaer CSCI-6090

September 28, 2006

This course is being offered in the Fall 2006 semester. It meets Mondays and Thursdays 2-3:20 pm in Voorhees North, and on the Web at <http://www.cs.rpi.edu/~musser/gsd>. The lectures and class discussions are an important part of the course but so is the Web site. Check it frequently for the latest information about the course and for material not covered in the lectures or textbooks.

General Information

Instructor

Dave Musser, 276-8660, [musser\[at\]cs.rpi.edu](mailto:musser[at]cs.rpi.edu)
Office Hours: Lally 313, Tuesday & Wednesday 2:30-4 pm or
by appointment

Catalog description

Study of the generic programming approach to design and systematic classification of software components. Techniques for achieving correctness, efficiency, and generality of algorithms, data structures, and memory management. Methods of structuring a library of generic software components for maximum usability are practiced in a significant design and implementation project. Prerequisite: CSCI-2300 or equivalent. 3 credit hours

Prerequisites

CSCI-2300 Data Structures and Algorithms

From the Rensselaer Catalog: Data structures and algorithms, and the mathematical techniques necessary to design and analyze them. Basic data structures: lists, associative structures,

trees. Mathematical techniques for designing algorithms and analyzing worst-case and expected-case algorithm efficiency. Advanced data structures: balanced trees, tries, heaps, priority queues, graphs. Searching, sorting. Algorithm design techniques: dynamic programming, greedy algorithms, divide-and-conquer, backtracking. Example graph, string, geometric, and numeric algorithms. Prerequisites: CSCI-1200 and MATH-1010. Fall and spring terms annually. 4 credit hours.

The main background needed is programming experience and familiarity with data structures and algorithms, including basic efficiency analysis techniques. Programming will be done primarily in C++ and G (a research language designed specifically for generic programming). The more advanced C++ and G language and library features will be thoroughly discussed and illustrated with example programs.

What it's not

This course is not on, but does draw ideas and goals from

- mathematics
- software engineering
- analysis of algorithms
- programming language semantics
- object-oriented programming
- functional and imperative programming

Homework, projects, exams, and grading

There will be in-class assignments, homework assignments, a programming project, and two exams. Only a few of the in-class and homework assignments will be graded; they will count about 10% of the course grade.

The programming project (which will be assigned in several parts) will count about 40% of the course grade.

The two exams will each count about 25% of the course grade.

Collaboration

Except in cases in which collaboration is specifically permitted, you must work on assignments individually. Particularly for some parts of the project, collaboration may be permitted or required, but in those cases the instructions given with the assignment will say so.

On the *exams*, collaboration is never permitted; all work must be your own.

I don't expect any incidents of academic dishonesty in this course, but any that do occur will be dealt with according to the policies stated in the *Rensselaer Handbook*.

Restricted access to some files

Some files, including most of the homework and exam solutions, have restricted access. These are the files that reside in the codebase or exercises directories in the course subversion repository, which requires you to log in with your CSLab username and password. You are permitted to download any of these files for your own use, and you may share them with others in the Rensselaer community, subject to restrictions against collaboration as specified in homework or project assignments. Such files should not be made available outside the Rensselaer community; in particular, don't repost them on any public Web page. Access to some files is restricted because textbook authors and other course instructors generally do not want exercise or problem solutions widely published, for obvious reasons.

Topics

This schedule is subject to change—check the course Web site frequently for updates.

The reading assignment for each date should be done *before* class on that date.

LGP = *A Language for Generic Programming*, NP = *Notes on Programming*

Week	Date	Topic	Reading	Online Material	HW
1	8/28	Organization, Introduction			
	8/31	Generic algorithms and STL concepts	LGP 1-1.4, 2-2.2.1		
2	9/04	Holiday: NO CLASS			
	9/07	Tools: C++ compilers, Emacs, Subversion	NP 1		
3	9/11	Generic containers and adaptors in the STL	LGP 2.2.2-2.4		
	9/14	Designing fvector_int	NP 2		HW 1 due 9/14 Feedback
4	9/18	Continuing fvector_int	NP 3		
	9/21	Implementing swap	NP 4		HW 2 due 9/21 Feedback
5	9/25	Types and type functions	NP 5		
	9/28	Regular types and equality	NP 6		
6	10/02	Language design for generics	LGP 3-3.3		HW 3 due 10/2
	10/05	Concepts: organizing type requirements	LGP 3.4-3.7		
7	10/10	[Tues] Design of G, part 1	LGP 4-4.5		
	10/12	Design of G, part 2	LGP 4.6-4.10		
8	10/16	Case study - STL in G	LGP 6-6.1.8		
	10/19	Case study - BGL in G	LGP 6.2		
9	10/23	EXAM I			
	10/26	TBA			
10	10/30	Unit testing: correctness	Handout		
	11/02	Unit testing: performance	Handout		
11	11/06	Ordering and related algorithms	NP 7		
	11/09	Function objects	NP 9		

Week	Date	Topic	Reading	Online Material	HW
12	11/13	Locations and addresses	NP 11		
	11/16	Iterators	NP 13		
13	11/20	Iterator type-functions	NP 15		
	11/23	Holiday: NO CLASS			
14	11/27	Permutation algorithms	NP 17		
	11/30	Reverse	NP 18		
15	12/04	Rotate	NP 19		
	12/07	Exam II			

Resources

Textbooks and language references

There are two required texts, but you don't need to purchase them; they are available online, and hard copies of assigned reading from them will be provided as handouts (in installments).

1. Jeremy G. Siek,¹ *A Language for Generic Programming*.² This is the author's Ph.D. thesis, completed in 2005 at Indiana University, in which he describes the design and implementation of a language called G.
2. Alex Stepanov,³ *Notes on Programming*.⁴ These notes are being developed by the author for a course he teaches at Adobe Systems, San Jose, CA; he has plans to eventually turn them into a textbook.

You don't need to print out these files or any portion of them; hard copies of individual chapters will be provided in class as needed for reading assignments. We will not cover all chapters in this course.

You should also have at hand at least one up-to-date and comprehensive C++ language reference. The most authoritative such reference is, of course, the C++ standard itself:

Document#: INCITS/ISO/IEC 14882-2003
Title: Programming languages - C++

¹<http://www.cs.colorado.edu/~siek>

²<http://www.cs.rpi.edu/~musser/gsd/siek05...thesis.pdf>

³<http://www.stepanovpapers.com>

⁴http://www.cs.rpi.edu/~musser/gsd/notes_on_programming.pdf

Abstract: This International Standard specifies requirements for implementations of the C++ programming language. The first such requirement is that they implement the language, and so this International Standard also defines C++. Other requirements and relaxations of the first requirement appear at various places within this International Standard.

It is available from American National Standards Institute eStandards Store.⁵ Note that this is the *2003 revision* of the standard (the first official C++ standard was released in 1998). The price for the electronic version is \$30, for the print version . . . unless you have an Uncle Midas, you'll want to choose the electronic version.

But the standard's legalistic style and lack of tutorial examples make for difficult and sometimes frustrating reading. Instead you might prefer one of the following:

Stanley Lippman, Josée Lajoie, and Barbara Moo, *C++ Primer*, Fourth Edition, 885 pages, Addison-Wesley, 2005, ISBN 0-201-72184-1;

Bjarne Stroustrup, *The C++ Programming Language*, 3rd edition, 991 pages, Addison-Wesley Pub Co, 1998, ISBN 0-201-88954-4.

The following book is less thorough and can't take the place of a complete reference, but it has lots of examples and emphasizes the advantages of understanding and using the C++ standard library from the beginning:

Andrew Koenig and Barbara E. Moo, *Accelerated C++*, 336 pages, Addison Wesley, 2000, ISBN 0-201-70353-X.

C++ Style Guide

In programming assignments and projects for the course, adhere to rules in this C++ Style Guide.

Cygwin Tools (including the GNU C++ compiler, X Windows, and Emacs)

The Cygwin Tools are a set of freeware compilers (including for GNU C/C++), other programming tools (such as debuggers), editors (Emacs in particular),

⁵<http://webstore.ansi.org/ansidocstore/product.asp?sku=INCITS%2FISO%2FIEC+14882%2D2003>

and utility programs (such as the BASH shell and X Windows) for Windows 95/98/NT/2000/XP. They provide many of the same useful tools you would find on a typical Unix or Linux platform.

If you have a Windows box you should install the Cygwin Tools on it. Download the installation file for your OS version from the Cygwin⁶ Web page.

GNU C++ compiler and standard library

This is a just a command-line compiler, but in combination with the compilation support commands of Emacs you can have a reasonable edit/compile/execute environment. For this course, version 3.3 or later is required: invoke it with `g++` on CSLab's FreeBSD machines and with `g++3` on Solaris machines (amazingly, version 2.95.3 is still available on those machines and is what you get with `g++`.)

Emacs

GNU Emacs, “the extensible, customizable, self-documenting real-time display editor,” is available on most campus computers. For Windows, either install Cygwin's implementation (it's under the Editors category; you'll also need Cygwin's X Windows implementation) or the native Windows version from GNU Emacs⁷ Web page.

[**Added 9/7/2006:** Emacs has a built-in tutorial (type `Esc-x help t`) and “info system” (type `Esc-x info`).

You can tailor Emacs to your preferences with a file called `.emacs` in your home directory. For use in this course, you should use the following `.emacs` file or add its contents to your existing `.emacs` file. See the comments (lines that begin with semicolon) in the file for explanation of each part. Some parts are optional.

```
.emacs
]
```

Subversion

From the Subversion⁸ project home page: “The goal of the Subversion project is to build a version control system that is a compelling replacement

⁶<http://www.cygwin.com>

⁷<http://www.gnu.org/software/emacs/emacs.html>

⁸<http://subversion.tigris.org>

for CVS in the open source community.” Subversion is fully documented in the Subversion Book,⁹ which is available in various electronic formats. The main part of the book needed for use in this course is “Basic Work Cycle: Chapter 3. Guided Tour” (a printed copy was handed out in the 9/7/2006 class), but you should also go through enough of the rest of the book to get an overview and to know where to look for answers to questions that come up later.

STL Web resources

1. Silicon Graphics STL¹⁰ (including a complete online STL reference manual)
2. The Dinkum C++ Library Reference¹¹ (including a complete online STL reference manual, but also covering other important parts of the library such as the `string` class).

Boost Libraries

The Boost Organization¹² provides many useful C++ Libraries. Not all are generic libraries in the full sense of the term; the largest library that is generic is the Boost Graph Library, BGL.¹³

On CSLab machines the Boost libraries (this is not quite the latest version) are installed at

```
/software/boost-1.32.0-0/pkg
```

So if you just put the option

```
-I/software/boost-1.32.0-0/pkg
```

on your compilation command line, the compiler will be able to find any of the Boost header files you are using in your source files.

L^AT_EX

From *L^AT_EX: A Document Preparation System*:¹⁴ “L^AT_EX is a high-quality typesetting system, with features designed for the production of technical

⁹<http://svnbook.red-bean.com>

¹⁰<http://www.sgi.com/tech/stl>

¹¹<http://www.dinkumware.com/manuals/#Standard%20C++%20Library>

¹²<http://www.boost.org>

¹³<http://www.boost.org/libs/graph/doc/index.html>

¹⁴<http://www.latex-project.org/>

and scientific documentation. LaTeX is the de facto standard for the communication and publication of scientific documents.” If that’s not enough motivation, see *Word Processors: Stupid and Inefficient*,¹⁵ by Allin Cottrell.

L^AT_EX is available on CSLab, RCS, and probably most other campus computers. To apply it to latex source file `mydoc.tex`, run

```
latex mydoc
```

This produces a DVI file, `mydoc.dvi`, which can be converted to a Postscript file, `mydoc.ps`, with

```
dvips -o mydoc.ps mydoc
```

or to a PDF file, `mydoc.pdf`, with

```
dvipdfm mydoc
```

Pdflatex is a tool that produces a PDF file directly from a L^AT_EX source file `mydoc.tex`:

```
pdflatex mydoc
```

produces `mydoc.pdf`. Using a L^AT_EX package called `hyperref`, e.g., with

```
\usepackage[plainpages=false,pdftex,colorlinks,backref]{hyperref}
```

causes cross-references and citations to appear as hyperlinks in the resulting PDF file.

To download a L^AT_EX system (free) for use on a Windows box, visit

MiK_TE_X¹⁶

The MiK_TE_X distribution includes utilities `dvips`, `dvipdfm`, and `pdflatex`.

If you are not familiar with L^AT_EX, a good place to get started is the brief *Introduction to L^AT_EX*¹⁷ by A. J. Hildebrand.

A not so short introduction is *The Not So Short Introduction to L^AT_EX2_ε*¹⁸ (PDF).

The original L^AT_EX manual, by the system’s creator, is

Leslie Lamport, *L^AT_EX: A Document Preparation System User’s Guide & Reference Manual*, Addison-Wesley, 1986, ISBN 0-201-15790.

¹⁵<http://ricardo.ecn.wfu.edu/~cottrell/wp.html>

¹⁶<http://www.miktex.org/>

¹⁷<http://www.math.uiuc.edu/~hildebr/tex/course>

¹⁸<http://ctan.tug.org/tex-archive/info/lshort/english/lshort.pdf?action=/starter/>

Another possibility is *First Steps in L^AT_EX*, by George Gratzer.

Beginning L^AT_EX users (and some veterans) sometimes make errors that cause no complaint from the L^AT_EX processor but do result in poor formatting. The following short note discusses some common problems and how to avoid them:

D. R. Musser, Elements of L^AT_EX Style,¹⁹ November 24, 1998.

¹⁹<http://www.cs.rpi.edu/~musser/gsd/formatting.pdf>