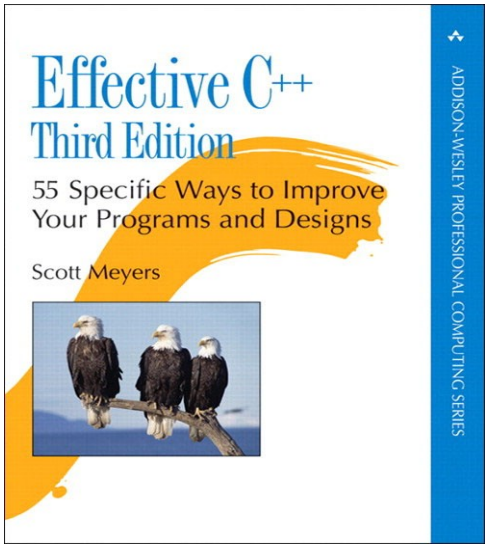
Chapter 15: Where to Go From Here

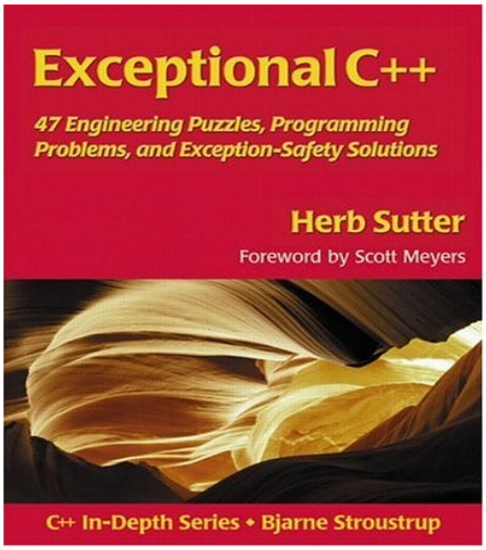
Congratulations! You've made it through CS106L. You've taken the first step on your journey toward a mastery of the C++ programming language. This is no easy feat! In the course of reading through this far, you now have a command of the following concepts:

- The streams library, including how to interface with it through operator overloading.
- STL containers, iterators, algorithms, adapters, and functional programming constructs, including a working knowledge of how these objects are put together.
- Pointer arithmetic and how objects are laid out in memory.
- The preprocessor and how to harness it to automatically generate C++ code.
- Generic programming in C++ and just how powerful the C++ template system can be.
- The `const` keyword and how to use it to communicate function side-effects to other programmers.
- Object layout and in-memory representation.
- Copy semantics and how to define implicit conversions between types.
- Operator overloading and how to make a C++ class act like a primitive type.
- What a functor is and how surprisingly useful and flexible they are.
- Exception handling and how to use objects to automatically manage resources.
- C++0x and what C++ will look like in the future.
- ... and a whole host of real-world examples of each of these techniques.

Despite all of the material we've covered here, there is *much* more to learn in the world of C++ and your journey has just begun. I feel that it is a fitting conclusion to this course reader to direct you toward other C++ resources that will prove invaluable along your journey into the wondrous realm of this language. In particular, there are several excellent C++ resources I would be remiss to omit:

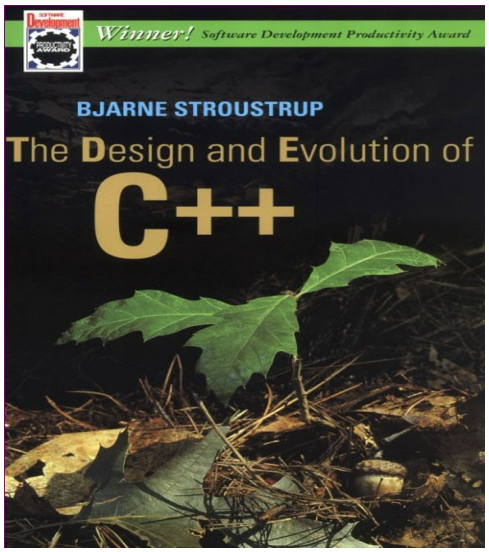


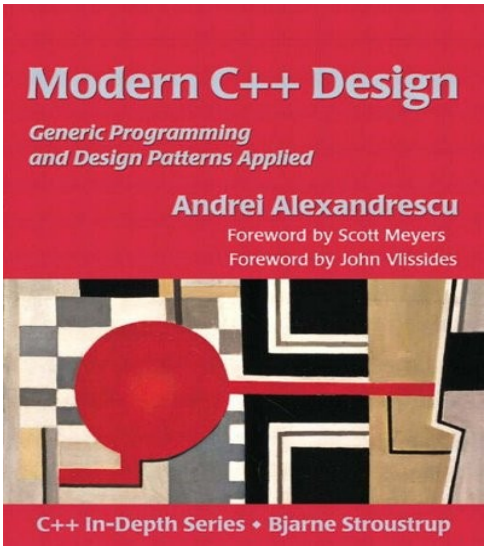
Effective C++, *More Effective C++*, and *Effective STL* by Scott Meyers. Picking up and reading this trio of books is perhaps the best thing you can do for yourself as a C++ programmer. The books in the *Effective C++* series will help you transition from a solid C++ programmer into an excellent C++ programmer and are widely regarded as among the best C++ books on the market. What separates the *Effective C++* series from most other C++ books is that *Effective C++* focuses almost exclusively on correct usage of core C++ language features and how to avoid common pitfalls. If you plan on using C++ in the professional world, you should own copies of this book.



Exceptional C++ by Herb Sutter. This book is an invaluable tool in any C++ programmer's arsenal. The book is largely organized as a set of puzzles that give you a chance to think about the best way to solve a problem and what C++ issues you'll encounter in the process. Along with *Effective C++*, *Exceptional C++* is one of the most highly-recommended C++ books out there. Herb Sutter's personal website is also an excellent resource for all your C++ needs.

The Design and Evolution of C++ by Bjarne Stroustrup. This book, affectionately known to hardcore C++ programmers as *D&E*, is a glimpse into Bjarne Stroustrup's thought processes as he went about designing C++. D&E is not a programming guide, but rather a history of the evolution of C++ from the small language C with Classes into the modern language we know and love today. D&E was written before C++ had been ISO standardized and even predates the STL, meaning that it can offer a new perspective on some of the language features and libraries you may take for granted. If you want an interesting glimpse into the mind of the man behind C++, this is the book for you.





Modern C++ Design: Generic Programming and Design Patterns Applied by Andrei Alexandrescu. Considered the seminal work in modern C++ programming, this book is an excellent introduction into an entirely new way of thinking in C++. Alexandrescu takes many advanced language features like templates and multiple inheritance, then shows how to harness them to achieve synergistic effects that are far more powerful than any of the individual features used. As an example, the first chapter shows how to write a single smart pointer class that is capable of storing any type of value, performing any sort of resource management, and having any copy behavior that the client desires. The book is very language-intensive and requires you to have a grasp of C++ slightly beyond the scope of this reader, but is a most wonderful text for all who are interested.

Final Thoughts

It's been quite a trip since we first started with the streams library. You now know how to program with the STL, write well-behaved C++ objects, and even how to use functional programming constructs. But despite the immense amount of material we've covered, we have barely scratched the surface of C++. There are volumes of articles and books out there that cover all sorts of amazing C++ tips and tricks, and by taking the initiative and exploring what's out there you can hone your C++ skills until problem solving in C++ transforms from "how do I solve this problem?" to "which of these many options is best for solving this problem?"

C++ is an amazing language. It has some of the most expressive syntax of any modern programming language, and affords an enormous latitude in programming styles. Of course, it has its flaws, as critics are eager to point out, but despite the advent of more modern languages like Java and Python C++ still occupies a prime position in the software world.

I hope that you've enjoyed reading this course reader as much as I enjoyed writing it. If you have any comments, suggestions, or criticisms, feel free to email me at htiek@cs.stanford.edu. Like the C++ language, CS106L and this course reader are constantly evolving, and if there's anything I can do to make the class more enjoyable, be sure to let me know!

Have fun with C++, and I wish you the best of luck wherever it takes you!

- Keith