

High Performance C++

Duration

1 day

Instructor

Scott Meyers

Class Limit

20 students

Prerequisite

Participants should already know the basic features of C++ (e.g., classes, inheritance, virtual functions, templates), but expertise is not required.

Price

On-site:

Please contact SPC for pricing (contact information on page 2)

Public Training:

\$495 (1 day)

*Discount available for early registration

Materials Provided

- Student manual containing the course slides
- Student handouts with class exercises and class studies

This seminar has three primary goals:

1. Educate participants regarding the behavior of C++ compilers to understand what is happening behind the scenes, especially with respect to the generation of temporary objects. (Temporary objects are not present in C++ source code, but, because they must be constructed and destructed, they can be a source of significant run-time friction.)
2. Demonstrate how to avoid constructs in C++ and its standard library that tend to decrease speed and increase memory usage.
3. Describe alternative design and implementation techniques for C++ software, techniques that typically perform better than the “obvious” approaches.

The seminar discusses not just C++ itself, but also focuses on the STL portion of the C++ standard library.

Please note that there is no hands-on programming in this course.



Intended Audience

Systems designers, programmers, and technical managers involved in the design, implementation, and maintenance of production libraries and applications using C++. Participants should already know the basic features of C++ (e.g., classes, inheritance, virtual functions, templates), but expertise is not required. People who have learned C++ recently, as well as people who have been programming in C++ for many years, will come away from this seminar with useful, practical, proven information.

TRAINING

High Performance C++

Instructor

Scott Meyers is one of the world's foremost experts on C++ software development. He is the author of the best-selling *Effective C++*, *More Effective C++*, and *Effective STL*; author and designer of the acclaimed *Effective C++ CD*; a former columnist for *C++ Report*; a frequent contributor to *C/C++ Users Journal*; and a member of the Advisory Boards of several software start-ups. In addition to nearly three decades of programming experience, he has an M.S. in Computer Science from Stanford University and a Ph.D. in Computer Science from Brown University.

Scott's consulting work has spanned a wide range of industries, including CAD/CAE applications, video games, newspaper layout, medical diagnostics, and nuclear simulations.

For more information on this or other SPC Springboard courses, please visit www.spcspringboard.com or e-mail SPC at info@spc.ca

Software Productivity Center
Suite 460—1122 Mainland Street
Vancouver, BC V8M 4T8
www.spc.ca

Toll Free:	Fax:
1.877.548.1948	604.689.0141
Vancouver:	Toronto:
604.662.8181	416.885.0512

Outline

80-20 Rule and Program Profiling

Language Issues

- Eliminating unnecessary temporary objects
- Avoiding superfluous constructor calls
- Facilitating the return value optimization
- Understanding inlining
- Pros and cons of writing custom memory managers

Library Issues

- Using reserve to avoid unnecessary reallocations in vector and string
- Why range member function calls are superior to single-element calls
- How vector-based data structures can be superior to sets and maps for lookup-intensive applications
- Functors vs. functions, including why sort is typically faster than qsort
- Why the erase-remove idiom is faster than handwritten loops
- Choices for non-standard containers based on hash tables

Reference Counting

- Possible string implementation
- “Soft underbelly” of reference-counting, techniques for coping, and how these issues affect class interfaces
- How threading issues can turn an optimization into a pessimization

The material in this seminar is primarily taken from Scott's landmark books, *Effective C++*, *More Effective C++* and *Effective STL*.

