Practical STL

A Hands-on Introduction to Generic Programming for C++ Programmers

ISO standard C++ is a general-purpose language that bridges different styles of programming — from procedural, to object-oriented and generic programming with templates — and spans different platforms and different application styles. A standard library supports the language.

The *Practical STL* course focuses on the part of the C++ standard library known commonly as the *Standard Template Library* along with the generic programming practices associated with it. Generic programming forms part of the modern OO design approach used by many C++ programmers. It differs from traditional object-oriented programming in many respects, most notably in using value-based rather than heap-based objects, parametric polymorphism rather than inclusion polymorphism, and separating algorithms from data structures.

The course introduces generic programming and the STL through lectures, discussion and lab exercises to reinforce the taught material.

Objectives

- Build on existing C++ knowledge to introduce the templates and the Standard Template Library more fully
- Describe the principles and practices behind C++ generic programming
- Understand the design elements of the STL, how to use the standard components in the library, and how to write application-specific code that works with the STL
- Emphasise good practice and outline idioms for safe and sensible use of templates and the STL

Audience

The course is suitable for experienced C++ programmers with at least some basic knowledge of templates.

Content

A Tour of the STL STL and the standard library · C++ as a multi-paradigm language · Forms of polymorphism · Generic programming · Convenient containers · Decoupling data structures and algorithms through iterators · Parameterising functionality with function objects

Value-Based Objects Values versus other kinds of objects · Passing values around · Defining copy and assignment for values · Operator overloading · Encapsulated memory management · Exception safety

Working with Templates Template functions · Template parameter requirements · Template classes · Managing long type names · Template specialisation · Template compilation model · Decoupling through templates

Containers The standard containers · Sequences and their requirements · Associative containers and their requirements · Customising associative containers · Container adaptors · Defining containers

Iterators Iterator categories · Iterator adaptors · I/O iterators · Defining iterators

Algorithms Factoring out common loop control flow · Generalisation through iterators · Standard algorithms · Defining algorithms for briefer and clearer code

Function Objects Functions and function objects · Standard function object types · Defining function object types · Standard and custom function adaptors · Standard and custom binders

Moving to the STL Common pitfalls and practices · Moving from C-style collections to STL containers · Refactoring loops into algorithm templates · Performance and memory usage

Additional Details

Duration 4 days

Setup Projection facilities for a laptop · Whiteboards and/or flip charts · One workstation per delegate with C++ compiler installed (configuration to be agreed)

Contact Kevlin Henney · kevlin@curbralan.com · Curbralan Limited · +44 117 942 2990