# C++ Advanced

## Advanced Techniques and Concepts 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 — from embedded systems through to GUI applications.

The *C++ Advanced* course provides developers with an armoury of modern C++ techniques and concepts suitable for serious development. It develops the concepts and syntax through lectures, discussion and hands-on lab exercises.

### Objectives

- Build on existing C++ knowledge to introduce the standard language and library more fully
- Practise and extend object-oriented and generic programming skills in C++
- Emphasise good practice and outline idioms for safe and sensible use of language features

### Audience

The course is suitable for experienced C++ programmers.

### Content

**C++ Review** Classes and objects · Object lifetime management · Exceptions · Inheritance good practice · STL and the standard library · C++ as a multi-paradigm language

**Conversions** Implicit and explicit conversions · Keyword casts · Defining inward and outward conversions · Appropriate use of conversions

**Scope** Defining and using namespaces · Nested types in classes · Class *static* members

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

**Mutability** *const* and its many uses · *mutable* and *const_cast* and their few uses · Overloading access operators such as *operator[]* · Separating classes and objects with respect to *const*-ness

**Proxies** Inheritance-based proxies · Smart pointers · Smart references

**Generic Programming** Template functions · Template parameter requirements · Template classes · Managing long type names · Template specialisation · Template compilation model

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

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

**Encapsulated 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 · Custom function objects · Standard and custom function adaptors · Standard and custom binders

**Dependency Management** Managing inlines and templates · Forward declarations · Nested classes · The Cheshire Cat idiom · Interface classes · Decoupling through templates

**Memory Management** Strategies for using *new* and *delete* · Overloading *new* and *delete*

**Exception Safety** Levels of exception safety · Exceptions in constructors and destructors · Exception-safety idioms · *auto_ptr* · *throw* specs and their pitfalls

**Reference Counting** Sharing objects and object representation · Copy-on-write pros and cons · Reference counting with smart pointers · Intrusive and non-intrusive reference counting

**Policy-Based Design** Compile-time generalisation · Policy parameters · Trait class templates

### Additional Details

**Duration** 5 days

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

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