+44 7989 401397
info@olsensoft.com

Advanced C++ Programming (5 days)

Course overview

C++ is a powerful and complex language. This course covers advanced C++ language features and development techniques, to help you get the most out of the language. The course also provides thorough coverage of the C++ Standard Template Library (STL), and explains how to implement OO design patterns and C++ programming idioms to reinforce best practice in your code.

What you'll learn

- Using casting and conversion techniques
- Overloading [], (), and ->
- Defining template functions and template classes
- Using STL iterators and algorithms
- Managing resource acquisition and release
- Utilizing OO patterns and C++ idioms effectively
- Using advanced template techniques

Prerequisites

- 3-6 months C++ programming experience

Course details

- Setting the Scene: Recap of C++ and OO features and techniques; ISO C++; Core language additions; Recap of the standard library
- Copying and Conversions: staticcast, dynamiccast, constcast and reinterpretcast; The mutable keyword; The explicit keyword; User defined conversion operators; Copy construction and assignment
- Scope and Related Patterns/Idioms: Recap of static class members; The Singleton pattern; Defining nested classes; The Handle/Body idiom; Using namespaces effectively
- Using Delegation: Recap of association and delegation; The Object Adapter pattern; The Proxy pattern; The Null Object pattern; Defining smart pointers; Lazy loading
- Overloading the subscript operator: How to overload []; Why to overload []; Creating multi-dimensional containers
- Template Functions: Overview of template functions; Implementing generic algorithms using template functions
- Template Classes: Overview of template classes; Specifying multiple type parameters; Using the standard container classes

- Using Iterators and Algorithms: What is an iterator; Using standard iterators; Creating generic algorithms using iterators; Function objects
- Exception Handling Techniques: Recap of exceptions; The standard exception class hierarchy; Uncaught exceptions; Strategies for handling exceptions; Exception safety
- Effective Memory Management: Handling allocation failures; Overriding the new operator to customise memory allocation; Caching; Controlling timing of construction and destruction
- Reference Counting Techniques: Defining classes that use shared representation objects; Reference-counted strings; Defining smart pointers for garbage collection
- Inheritance Techniques: Defining interfaces; Multiple inheritance; Virtual base classes; Interface classes; Mixin classes; Runtime type information (RTTI); Private and protected inheritance; The Class Adapter pattern
- Advanced Template Techniques: Defining non-type template parameters; Defining template adapters; Specifying default template parameters; Specializing templates; Defining trait classes
- Call-back Techniques: Implementing call-backs using function pointers; The Command pattern; Function objects; Member function pointers