+44 7989 401397
info@olsensoft.com

Modern C++ Development (5 days)

Course overview

Thought you knew C++? Think again! The C++11 standard saw a huge leap forward in the language and STL. Largely inspired by Boost, the C++11 standard introduced support for lambdas, multithreading, vastly improved object construction mechanisms, and much more beside. This course takes a deep and detailed look at all the new features on offer in C++11, along with the incremental additional changes in C++14, C++17, and C++20.

What you'll learn

- Working with C++11 smart pointers
- Functional programming in C++11
- Defining and using lambda expressions
- Using C++11 container classes
- Implementing code applications in C++11
- Using miscellaneous C++ language features
- What's new in C++ 14, C++17, and C++20

Prerequisites

- 3-6 months C++ programming experience

Course details

- General Language Enhancements in C++11: auto variables; Using auto in template definitions; Using decltype; New return syntax; Range-based for loops; Making your own classes iterable; Generalised constant expressions; Strongly-typed enums; Null pointers; Explicit overrides; Static asserts
- Additional Language Features in C++11: Lvalues, rvalues, and rrvalues; Movability; Reference binding rules; Support for movability in the STL; Improved initialization syntax; Inheriting and delegating constructors; Regular expressions; Date and time; Chrono; Explicit conversions; Variadic templates
- Miscellaneous New Language Features in C++14: Function return type deduction; Alternate type deduction in declarations; Relaxed constexpr restrictions; Variable templates; Aggregate member initialization; Standard user-defined literals
- Smart Pointers: Recap of smart pointer concepts; Shared pointers; Weak pointers; Unique pointers; Techniques and patterns
- Introduction to Functional Programming: Overview of functional programming; Using std bind to bind parameters; Using placeholders with for_each(); Passing by reference; Using std function to represent free functions and member functions

- Lambda Expressions: Overview of lambda expressions; Lambda syntax in C++11; Defining lambdas with arguments and a return value; Variable capture; Using lambdas with the STL; Performance considerations; Generic lambdas and lambda capture expressions in C++14
- C++11 and C++14 Containers: Overview of new STL features; Using std array and forward_list; Using unordered containers; Understanding hashing; Defining a custom hash function; Understanding buckets; In-place construction; Heterogeneous lookup in associative containers in C++ 14
- C++11 and C++14 Multithreading: Creating simple threads using std thread; Using lambda expressions with threading; Accessing the current thread; Using mutexes; Lock management and lock strategies; Atomic variables; Condition variables; Calling functions asynchronously; Working with future values; Shared mutexes and locking in C++14
- What's New in C++17: Nested namespaces; Attributes; Fold expressions in variadic templates; Aggregate initialization with inheritance; Lambda enhancements; Template class type deduction; Inline variables; Library enhancements; Parallel algorithms; Miscellaneous enhancements and additions
- What's New in C++20: Concepts; Ranges; Lambda improvements; The spaceship operator; Atomic smart pointers; Concurrency and synchronization improvements; Co-routines; Modules; The consteval and constinit keywords; Miscellaneous enhancements and additions