



ModCpp - Version: 1
01 May 2021

Modern C++



Modern C++

ModCpp - Version: 1

 **4 days Course**

Description:

Modern C++ is not just about new language features or a bunch of library improvements. Modern C++ is how we write code, how we design interfaces, and how we make the best use of modern features while respecting existing investments. This is an exciting time to be a C++ developer, and this course will show you how to make the most of C++ 11/14, how to use advanced template features and design for exception safety, and how to work with parallelism and concurrency in standard and non-standard C++ code. In this course you will experience the spirit of true modern C++.

Intended audience:

C++ developers, team leaders, and software architects.

Prerequisites:

At least two years of C++ programming experience, including with templates and the STL

Familiarity with operating systems concepts, such as processes and threads

Objectives:

Understand the productivity features of C++ 11/14 and how they can be used to modernize existing code

Manage resources with RAII and smart pointers in an exception-safe fashion

Use the C++ 11 standard threading features and the Microsoft-specific Concurrency Runtime to parallelize code

Apply move semantics to improve performance in resource-owning classes

Understand advanced templates, type traits, and metaprogramming issues

Topics:

Module 01 - Visual Studio Goodies

- Runtime checks
- Static code analysis
- The Visual Studio Profiler
- WPO
- PGO

Module 02 - C++ Technical Report 1 (TR1)

- Function objects
- `std::function`
- `std::bind` and `std::mem_fn`
- New containers
- Miscellaneous features

Module 03 - C++ 11/14 Productivity Features

- Non-static data member initializers
- `override` and `final` functions
- Delegating constructors
- Alias templates
- Type inference for local variables (`auto`)
- Type inference for expressions (`decltype`)
- Uniform initialization and list initialization
- Extended literals
- Compile-time constant expressions (`constexpr`)
- Deleted and defaulted functions
- Range-based `for` loop
- Lambda functions
- Lambdas and function pointers
- LAB: `auto`, range-based `for`, lambdas

Module 04 - C++ 11 Move Semantics

- Temporary objects and copies
- Rvalue references
- Move semantics (constructor and assignment)
- `std::move`
- Universal references
- Perfect forwarding
- Guidance for move semantics
- LAB: move support

Module 05 - STL in C++ 11

- `std::atomic`
- Threads
- Futures and promises
- `std::async`
- RAI and smart pointers
- Guidance for smart pointers
- Unordered containers
- Regular expressions
- LAB: smart pointers

Module 06 - Microsoft Concurrency Runtime

- Task parallelism vs. data parallelism
- Task groups
- Tasks and continuations
- Parallel loops
- Parallel algorithms
- Concurrent containers
- Synchronization mechanisms
- LAB: parallelism and asynchrony

Module 07 - Advanced Templates

- Recap of function templates and class templates
- Template specialization and partial specialization
- Type traits
- Detecting nested type and member at compile-time
- `std::enable_if`
- Variadic templates

Module 08 - Exception Safety

- Error handling
- Recap of C++ exception syntax
- C++ 11 nested exceptions and exception capturing
- Exception safety guarantees
- Guidance for constructors and operator=
- Obtaining strong exception safety by move or swap
- LAB: analyzing exception safety

° Module 09 - C++ AMP (optional, by request)

° Module 10 - C++/CX (optional, by request)