

Course Outline

Advanced C++ 20 Programming (Modern C++) Course TTCP2175: 3 days Instructor Led

About this course

Geared for experienced C++ developers, Advanced C++ 20 Programming / Modern C++ is a three-day hands-on course designed to provide you with skills required to write faster, robust C++ code, enhancing your ability to create performance-critical applications ranging from system software, game development, to real-time systems and AI programming.

Working in a lab-focused learning environment guided by our experienced Instructor, you'll explore a broad spectrum of 'next-level' topics such as SOLID design principles, operator overloading, functional programming, and template usage, all aimed at refining your programming craft. The hands-on work will mirror real-world scenarios, including implementing design patterns and managing threads and tasks. You'll also discover the realms of multi-threading and asynchronous programming, invaluable skills for creating efficient, high-performance applications. These advanced skills have significant application in industries such as finance for high-frequency trading systems, in gaming for building high-performance game engines, or in tech companies for building large scale distributed systems.

By the end of this unique and intensive course, you will be well-equipped to tackle complex coding challenges, contribute more effectively to your team's projects, and deliver high-quality, efficient applications that meet modern business demands.

Audience profile

This is an intermediate and beyond level development course designed for developers with prior C++ programming experience. Students without prior C++ programming background should take the pre-requisite training.

Take Before: Incoming students should have practical skills equivalent to the topics in, or should have recently attended, one of these courses as a prerequisite:

- TTCP2100 Introduction to C++ Programming / C++ 20

At course completion

After completing this course, students will be able to:

- Become a Pro at SOLID Design: You'll delve into SOLID design principles, mastering how to write clean, maintainable code. By the end, you'll be able to identify and avoid design smells, enhancing the overall quality of your projects.
- Master the Art of Factory Implementation: Get hands-on with factories in C++. We'll guide you through the basics and options, including Singleton, to help you understand the critical role of factories in object-oriented design.
- Up Your Game with Operator Overloading: You'll learn about operator overloading and its applications. By understanding how to enhance the readability and flexibility of your code, you'll streamline your programming tasks.
- Unlock the Power of Templates: We'll dive into the intriguing world of templates, exploring variance, concepts, and the 'auto' keyword. You'll get to implement covariant and contravariant templates, broadening your C++ expertise.
- Ace Multithreading and Asynchronous Programming: We'll explore the realms of multithreading and asynchronous programming, equipping you with the tools to create efficient, high-performance applications. You'll get to practice with mutexes, semaphores, atomics, and coroutines, gaining invaluable experience for your future projects.

Course Outline

Course Outline

C++ Quick Review

- Modern C++ - an introduction to new features in C++ 11-20

SOLID Design

- Design Smells
- Project Overview
- Single Responsibility
- Open/Close
- Liskov's Substitution
- Interface Segregation
- Dependency Inversion

Implementing a Factory in C++

- Factory Basics
- Options
- Singleton
- A C++ Object Factory

Operator Overloading

- Commonly Overloaded operators
- Conversions
- Constructor Conversions
- Implicit vs Explicit

Templates

- Understanding variance
- Implementing covariant templates
- Implementing contravariant templates
- <concepts>
- Auto

Functional Programming

- Lambda Expressions
- Functors
- <functional>

Structural Patterns

- Adapter
- Bridge
- Composite
- Decorator
- RAII and Proxy Pattern – Smart Pointers
- Strategies for Smart Pointers & Raw Pointers
- Other patterns

Course Outline

Behavioral Patterns

- Solving common design smells with behavioral patterns
- Template Method – issues initializing C++ objects
- State Pattern
- Strategy Pattern
- Command Pattern
- Other behavioral Patterns

Threads, Tasks, Async

- All about threads
- Mutex
- Semaphores
- Latch & barrier
- atomics
- All about Tasks
- <future>
- Coroutines (async)