Intermediate C++ 20 Programming | Effective C++ 20 - TTCP2150

Explore C++ Templates, Memory Management, Functional Programming, Unit Testing & Modern Features & More

Duration: 4 Days
Skill Level: Intermediate
Available Format: Instructor-Led Online; Instructor-Led, Onsite In Person ; Blended; On Public Schedule

Geared for experienced C++ developers, Intermediate C++ 20 / Effective C++ 20 is a four day, hands-on program that dives covers a broad spectrum of topics – from the quick review of C++ essentials to modern C++ features, memory management, unit testing, and more. Our expert instructors will walk you through a comprehensive journey, investigating cutting-edge concepts such as RAII, copy and move semantics, namespaces, templates, and C++ 20 Concepts & auto Templates.

What You'll Learn

Overview

C++ is a powerful, high-performance programming language that offers an ideal blend of low-level memory manipulation and high-level abstraction capabilities. Learning C++ is a valuable investment for developers, as it opens the door to creating efficient, versatile, and complex applications that run on a variety of platforms. Modern companies across diverse industries - including finance, gaming, automotive, and telecommunications - rely on C++ for developing performance-critical applications, system software, and embedded systems. Renowned organizations like Google, Facebook, and Microsoft continue to leverage the power of C++ in their development practices, solidifying its status as a crucial skill for developers seeking lucrative and challenging career opportunities.

Geared for experienced C++ developers, **Intermediate C++ 20 / Effective C++ 20** is a four day, hands-on program that dives covers a broad spectrum of topics – from the quick review of C++ essentials to modern C++ features, memory management, unit testing, and more. Our expert instructors will walk you through a comprehensive journey, investigating cutting-edge concepts such as RAII, copy and move semantics, namespaces, templates, and C++ 20 Concepts & auto Templates. You'll learn to leverage the power of modern C++ and unravel the intricacies of memory management, including the handle/body pattern, smart pointers, and move constructors. By the end of this course, you'll have an in-depth understanding of C++ memory, pointers, and complexity.

Working in a hands-on environment, explore the art of functional programming and discover how the IoC pattern, dependency injection, functors, and lambda expressions can bring about significant enhancements to your code. With a strong emphasis on SOLID principles, inheritance, polymorphism, exceptions, and operator overloading, this course will help you design robust, maintainable, and scalable modern applications. You'll also expand your C++ toolset by exploring the rich offerings of the Standard Library, mastering the essentials of containers, algorithms, numerics, dates, and times. Gain a solid introduction to multitasking with threads, tasks, and async. As a bonus, you'll also learn how to implement effective unit testing in C++ using GTest, ensuring your code is reliable and bug-free.

Join our immersive training experience and become an adept C++ developer with unparalleled skills in the latest C++ 20 programming techniques. This fast-paced, lab-intensive course is designed to equip you with the knowledge and confidence to tackle the most challenging C++ development projects.

Objectives

Working in a hands-on learning environment, guided by our expert team you'll learn to:

- Master intermediate to advanced C++ 20 programming techniques, enabling the development of efficient and maintainable applications using the latest features and best practices.

- Acquire in-depth knowledge of memory management in C++, including the handle/body pattern, smart pointers, and move constructors, to optimize performance and minimize memory-related issues.
- Develop proficiency in functional programming with C++, incorporating concepts such as dependency injection, functors, and lambda expressions to enhance code flexibility and modularity.
- Gain expertise in utilizing the C++ Standard Library for generic programming, mastering the use of containers, algorithms, numerics, and other features to create powerful, reusable code components.
- Learn to implement effective unit testing in C++ using GTest, ensuring the reliability and robustness of your applications through rigorous testing methodologies.
- Understand the basics of multitasking in C++, exploring threads, tasks, and async for concurrent programming, empowering developers to create scalable and high-performance applications.

Audience

This is an **intermediate 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 pre-reqsuite:

- TTCP2100: Introduction to C++ Programming

Pre-Requisites

This is an **intermediate 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 pre-reqsuite:

- TTCP2100: Introduction to C++ Programming

TTCP2100 Introduction to C++ Programming Essentials

Agenda

Please note that this list of topics is based on our standard course offering, evolved from typical industry uses and trends. We will work with you to tune this course and level of coverage to target the skills you need most. Course agenda, topics and labs are subject to adjust during live delivery in response to student skill level, interests and participation.

Quick Review of C++

- Implementing a basic O-O design
- Implementing Classes
- Visibility & friends
- File organization
- C++ types – structs, classes, interfaces, enums

Modern C++

- New features in C++ 11,14,17,20
- RAII - Modern memory management in C++ - overview
- Copy vs Move semantics
- Namespaces
- Strings
- Input & Output
- Implementing a linked-list - a demonstration of class, memory, pointers and complexity

Templates

- General Purpose Functions
- Function Templates
- Template Parameters

- Template Parameter Conversion
- Function Template Problem
- Generic Programming
- General Purpose Classes
- Class Templates
- Class Template Instantiation
- Non-Type Parameter
- C++ Containers overview
- C++ 20 **concepts** & **auto** Templates

Memory Management

- The handle/body (Bridge) pattern
- Using strings effectively
- Smart Pointers
- Move constructor in depth
- Other <memory> features

Unit Testing in C++

- Unit testing - Quick Overview
- Unit testing in C++
- Using GTest

Inheritance and Polymorphism

- Inheritance Concept
- Inheritance in C++
- Virtual Function Specification
- Invoking Virtual Functions
- VTable
- Virtual Destructors
- Abstract Class Using Pure Virtual Function
- Design for Polymorphism
- Interfaces
- Design for Interface

- A SOLID introduction

Exceptions

- Review of the basics: **try, catch, throw**
- The throws declaration in modern C++
- Using **noexcept**
- Overriding **terminate**

Operator Overloading & Conversion

- Basics
- Essential Operators
- Conversion Operators
- Constructor as conversion
- Explicit vs Implicit conversion

Functional Programming

- The IoC pattern
- Dependency Injection
- Functions as objects
- IoC via interface
- Functors
- IoC with Functors
- Implementing Functors
- Function Pointers
- IoC with Function Pointers
- Lambda Expressions
- Lambda Syntax
- IoC with Lambdas

Standard Library

- Perspective

- History and Evolution
- New Features
- Generic Programming
- Containers
- Algorithms
- Numerics
- Dates & Times
- Initializer List

Introduction to Multitasking

- Threads
- Tasks
- Async

Related Courses

TTCP2000	Introduction to Programming and C++ Basics for Non-Developers
TTCP2100	Introduction to C++ Programming Essentials
TTCP2150	Intermediate C++ 20 Programming | Effective C++ 20
TTCP2175	Advanced C++ 20 Programming

There are many options for the student development platform. IDE's from Visual Studio, Visual Studio Code, Eclipse and others are acceptable. A compiler compatible with C++ 20 is also required. GCC version 12+ or Visual Studio 2022 (MSVC) are compatible. GTest is also required for the course and comes with VS 2022. For Linux and other platforms, it must be installed. Students may configure their own environment or, ask us about a virtual lab setup with everything already installed

For More Information

Please contact us or call 844-475-4559 toll free for more information about our training services (instructor-led, self-paced or blended), coaching and mentoring services, public course enrollment or questions, partner programs, courseware licensing options and more.