

Document Generated: 04/02/2026

Learning Style: Virtual Classroom

Technology:

Difficulty: Intermediate

Course Duration: 4 Days

Next Course Date: **April 13, 2026**

## Intermediate C++ 20 Programming / Effective C++ 20 (TTCP2150)



### About This Course:

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.

## **Course Objectives:**

- 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

## **Prerequisites:**

- Students without prior C++ programming background should take the pre-requisite training.

## **Course Outline:**

- 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
  - RAI - 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