ProTech Professional Technical Services, Inc.

protechtraining.com

Modern Advanced C++

Course Summary

Description

Advanced C++ will give you the confidence to efficiently tackle any C++ project. You will learn how to structure your code for readability, streamline code for speed and functionality, and seamlessly generate and incorporate libraries. C++ is one of the most widely used programming languages and is applied in a variety of domains, right from gaming to graphical user interface (GUI) programming and even operating systems. If you're looking to expand your career opportunities, mastering the advanced features of C++ is key.

The course begins with advanced C++ concepts by helping you decipher the sophisticated C++ type system and understand how various stages of compilation convert source code to object code. You'll then learn how to recognize the tools that need to be used in order to control the flow of execution, capture data, and pass data around. By creating small models, you'll even discover how to use advanced lambdas and captures and express common API design patterns in C++. As you cover later lessons, you'll explore ways to optimize your code by learning about memory alignment, cache access, and the time a program takes to run. The concluding lesson will help you to maximize performance by understanding modern CPU branch prediction and how to make your code cache-friendly.

Objectives

At the end of this course, students will be able to:

- Delve into the anatomy and workflow of C++
- Study the pros and cons of different approaches to coding in C++
- Test, run, and debug your programs
- Link object files as a dynamic library
- Use templates, SFINAE, constexpr if expressions and variadic templates
- Apply best practice to resource management

Topics

- Anatomy of Portable C++ Software
- No Ducks Allowed – Types and Deduction
- No Ducks Allowed – Templates and Deduction
- No Leaks Allowed – Exceptions and Resources
- Separation of Concerns – Software Architecture, Functions, and Variadic Templates
- The Philosophers' Dinner – Threads and Concurrency
- Streams and I/O
- Everybody Falls, It's How You Get Back Up – Testing and Debugging
- Need for Speed – Performance and Optimization

Audience

If you have worked in C++ but want to learn how to make the most of this language, especially for large projects, this course is for you. A general understanding of programming and knowledge of using an editor to produce code files in project directories is a must. Some experience with strongly typed languages, such as C and C++, is also recommended.

Prerequisites

If you have worked in C++ but want to learn how to make the most of this language, especially for large projects, this course is for you. A general understanding of programming and knowledge of using an editor to produce code files in project directories is a must. Some experience with strongly typed languages, such as C and C++, is also recommended.

Duration

Five days

Modern Advanced C++

Course Outline

I. Anatomy of Portable C++ Software
- A. Managing C++ Projects
- B. Writing Readable Code

II. No Ducks Allowed – Types and Deduction
- A. C++ Types
- B. Creating User Types
- C. Structuring our Code

III. No Ducks Allowed – Templates and Deduction
- A. Inheritance, Polymorphism, and Interfaces
- B. Templates – Generic Programming
- C. Type Aliases – typedef and using
- D. Class Templates

IV. No Leaks Allowed – Exceptions and Resources
- A. Exceptions in C++
- B. RAII and the STL
- C. Move Semantics
- D. Name Lookup
- E. Caveat Emptor

V. Separation of Concerns – Software Architecture, Functions, and Variadic Templates
- A. Function Objects and Lambda Expressions
- B. Variadic Templates

VI. The Philosophers' Dinner – Threads and Concurrency
- A. Synchronous, Asynchronous, and Threaded Execution
- B. Review Synchronization, Data Hazards, and Race Conditions
- C. Future, Promises, and Async

VII. Streams and I/O
- A. File I/O Implementation Classes
- B. String I/O Implementation
- C. I/O Manipulators
- D. Making Additional Streams
- E. Using Macros

VIII. Everybody Falls, It's How You Get Back Up – Testing and Debugging
- A. Assertions
- B. Unit Testing and Mock Testing
- C. Understanding Exception Handling
- D. Breakpoints, Watchpoints, and Data Visualization

IX. Need for Speed – Performance and Optimization
- A. Performance Measurement
- B. Runtime Profiling
- C. Optimization Strategies
- D. Cache Friendly Code