



POSTCPP - Version: 2

01 May 2021

Post Modern C++



Post Modern C++

POSTCPP - Version: 2

 3 days Course

Description:

The modern C++ era arrived with the STL, the C++98 standard and a host of techniques and guidelines. Unfortunately, it also came with a lot of complexity of techniques, syntactic noise and unfulfilled potential. C++11 represented a fundamental shift in both language features and supported programming styles, a postmodern tradition continued in C++14 and the forthcoming C++17 standard.

This course spans lectures and hands-on exercises to explore language and library features and the implications for programming style, everything from code that is easier on the eye to code that is easier on the processor, from SOLID guidelines to more fluid code, from cleaner object-oriented programming to a functional-programming style, from event-driven to concurrent code.

Intended audience:

- C++ developers, software architects and team leads.

Prerequisites:

Comfortable with C++98 and a few advanced techniques.

C++11/14 experience is not assumed, but can be beneficial.

Objectives:

Appreciate the brevity, convenience and basic shifts in style that recent standards permit and encourage in C++.

Get a better appreciation for what functional programming means for C++ style.

Understand what the SOLID principles do (and don't) mean and how they apply in C++11/14.

Aim for a more compositional approach to creating C++ functions and classes.

Topics:

From Early to Postmodern C++

- How C++ has changed over different formal and informal standards
- A quick tour of what's changed and the influence on style
- Conveniences for declarations, loops, enums, functions, etc.; new types in the core language and library

Good Unit Tests: unit testing C++ code

- Unit testing C++ code
- Tests using ordinary functions and assert
- Tests using lambdas and exceptions
- Given–when–then structure for tests

Multiparadigm Programming

- Structured programming revisited
- Block structure and reduced mutability
- Anatomy of C++ lambdas; bind and function
- Simplified event-based code beyond the Observer pattern
- Constexpr versus template metaprogramming
- STL and generic programming

Data Abstraction and Objects

- Abstraction and naming
- Modularity in C++
- Abstract data type (ADT) approaches in C++
- Value-based programming
- Operator overloading
- Anatomy of copyable and moveable value types
- Object-oriented programming (OOP) revisited



- OO guidance and common pitfalls
- Layered class hierarchies

Functional C++

- Functions as objects
- Partial application
- Generic programming and functional style
- Mapping, reduction and other functional primitives
- Referential transparency, immutability and copying
- Copy and move optimisations
- Persistent data structures
- Append-only data structures
- Pitfalls and challenges in for FP in C++