Programming Objects with C++, Advanced

Description

The course starts with basic OO concepts, and then a quick introduction to the language. Except OOP, topics covered include Templates, Standard Template Library (STL) and Exceptions. The course also covers some advanced Object-Oriented design techniques in C++ such as Design Heuristics, Design by Contract, Interfaced-based programming, Composition and Delegation Patterns, Memory Management and Smart Pointers, Subtyping, Design for efficiency and Meta programming in C++.

Classroom Registration Price (CHF)
3800
Virtual Classroom Registration Price (CHF)
3550
Course Content
Module 1: Language History and C++11/14

- Lesson 1: History of C++
- Lesson 2: Versions
- Lesson 3: New in TR1
- Lesson 4: New in C++11
- Lesson 5: New in C++14

Module 2: Language Review and Best Practice: Part I

- Lesson 1: Object-Oriented Programming
- Lesson 2: Constructors and Destructors
- Lesson 3: New and delete

Module 3: Basic Inheritance

- Lesson 1: Interface vs. Implementation
- Lesson 2: Type Inheritance
- Lesson 3: Implementation Inheritance
- Lesson 4: Proper Use of C++11 final and override
- Lesson 5: Virtual Destructors: When and Why?
- Lesson 6: Inheritance Guidelines

Module 4: Correct Use of Well Known Language Features

- Lesson 1: Pointers vs. References vs. Value
- Lesson 2: Proper Use of const
- Lesson 3: Proper Use of Inline Functions
- Lesson 4: Proper Use of static
- Lesson 5: Proper Use of Default Parameters
- Lesson 6: Proper Use of friend
- Lesson 7: Proper Use of namespace
- Lesson 8: The C++ Way to Cast
- Lesson 9: Proper Use of Operator Overloading

- Lesson 10: Copy Constructor: Why/When?
- Lesson 11: Assignment Operator: Why/When?
- Lesson 12: The Law of The Big Three

Module 5: Exceptions

- Lesson 1: Lessons from Traditional Error Handling
- Lesson 2: Object-Oriented Error Handling
- Lesson 3: Throw, try and catch
- Lesson 4: Design of Exception Hierarchies
- Lesson 5: Proper use of Rethrow
- Lesson 6: using unexpected
- Lesson 7: Exception Pitfalls
- Lesson 8: Exception Guidelines

Module 6: Templates

- Lesson 1: Template Classes Definition
- Lesson 2: Template Classes Implementation
- Lesson 3: Parametrized Classes
- Lesson 4: Templates and Non-Type Parameters
- Lesson 5: Template Guidelines
- Lesson 6: Templates and Plain Functions
- Lesson 7: C++ Template Function

Module 7: Standard Template Library (STL)

- Lesson 1: STL String
- Lesson 2: STL Components
- Lesson 3: Sequence Containers
- Lesson 4: Use of Iterators in STL
- Lesson 5: Example of Algorithms
- Lesson 6: Initialization of Containers
- Lesson 7: Performance Profiles of Sequence Containers

Module 8: STL Algorithms

- Lesson 1: STL vs Boost
- Lesson 2: Parameterization of Algorithms
- Lesson 3: Using Functions
- Lesson 4: Using Function Objects
- Lesson 5: Using Lambda Expressions
- Lesson 6: Library of Selected Algorithms
- Lesson 7: Contributing Algorithms

Module 9: STL Associative Containers

- Lesson 1: Set
- Lesson 2: Multiset
- Lesson 3: Map
- Lesson 4: Multimaps

Module 10: STL Functors, Allocators and More

- Lesson 1: Standard Exception
- Lesson 2: Functors
- Lesson 3: Library Provided Function Objects
- Lesson 4: Using STL Function Objects and Binders
- Lesson 5: Negators
- Lesson 6: Allocators
- Lesson 7: Complex Number
- Lesson 8: Smart Pointers

Module 11: Efficiency: Temporary Objects

- Lesson 1: Temporary Objects: The Problem
- Lesson 2: Various Techniques to Avoid Temporaries
- Lesson 3: STL String: How to Avoid Creation of Temporaries
- Lesson 4: C++11: Move Semantics
- Lesson 5: Miscellaneous Techniques to Avoid Temporaries

Module 12: Memory Management

- Lesson 1: How Does C++ Use Memory?
- Lesson 2: Basic Guidelines
- Lesson 3: Implementation of Singletons in C++
- Lesson 4: Efficient Use of Smart Pointers
- Lesson 5: Overloading new and delete
- Lesson 6: Memory Management
- Lesson 7: How Does C++ Use Memory?
- Lesson 8: Basic Guidelines
- Lesson 9: Implementation of Singletons in C++
- Lesson 10: Efficient Use of Smart Pointers
- Lesson 11: Overloading new and delete
- Lesson 12: The Importance of Data Layout

Module 13: Hot vs. Cold Memory

- Lesson 1: Miscellaneous Efficiency Techniques
- Lesson 2: Design Concerns
- Lesson 3: Flexibility vs Performance
- Lesson 4: Lazy Evaluation
- Lesson 5: Eager Evaluation
- Lesson 6: Copy on Write Techniques
- Lesson 7: Data Layout Revisited
- Lesson 8: Modern Hardware and Cache Pipelines
- Lesson 9: The Effect of Data Structures and Algorithms
- Lesson 10: Postcondition and C++
- Lesson 11: Efficiency Profiles of Libraries
- Lesson 12: STL and Performance
- Lesson 13: Latency
- Lesson 14: Cost and Benefits of Threads
- Lesson 15: Asynchronous Programming
- Lesson 16: Futures

Module 14: Delegation

- Lesson 1: Concept of Delegation
- Lesson 2: Delegation in C++
- Lesson 3: Simple Delegation
- Lesson 4: Static Delegation
- Lesson 5: Superclass Delegation
- Lesson 6: Subclass Delegation
- Lesson 7: Issues With Subclass Delegation in C++
- Lesson 8: Object Delegation
- Lesson 9: Strategy Pattern
- Lesson 10: C++ and Strategy
- Lesson 11: State Pattern
- Lesson 12: C++ and State
- Lesson 13: Design of Composite
- Lesson 14: Composite and Delegation
- Lesson 15: Other Delegation Patterns

Module 15: Decoupling

- Lesson 1: What is Coupling?
- Lesson 2: Kinds of Coupling
- Lesson 3: Identity Coupling
- Lesson 4: Identity Coupling: Object Lifetimes
- Lesson 5: Change of Identity
- Lesson 6: Type Coupling
- Lesson 7: Implementation Coupling
- Lesson 8: Interfaces and Implementations
- Lesson 9: Decoupling by Example

Module 16: Advanced Inheritance

- Lesson 1: Multiple Inheritance of Interfaces
- Lesson 2: Multiple Inheritance of Implementation

- Lesson 3: Shared Properties
- Lesson 4: Resolving Ambiguity
- Lesson 5: Virtual Inheritance
- Lesson 6: Multi Methods
- Lesson 7: Double Dispatch
- Lesson 8: Use of RTTI
- Lesson 9: Rules and Guidelines
- Lesson 10: Inheritance of Baseclass Methods
- Lesson 11: Change of Methods
- Lesson 12: Contracts and Inheritance
- Lesson 13: Contracts and Subtyping
- Lesson 14: Variations of Method Arguments
- Lesson 15: Rules for Method Arguments
- Lesson 16: Rules for Changing Return Types
- Lesson 17: Cancellations of Methods

Module 17: Design Heuristics

- Lesson 1: Object-Oriented Design Guidelines
- Lesson 2: Reflecting Client's View
- Lesson 3: Polling
- Lesson 4: Express Interfaces Through Objects
- Lesson 5: Value Objects
- Lesson 6: Class Invariants
- Lesson 7: Abstract Base Classes
- Lesson 8: Classes and Interfaces
- Lesson 9: Design Interfaces Between Base and Derived Classes
- Lesson 10: Classes Cohesiveness

Lab / Exercises

- During the course participants are encouraged to actively participate in the learning experience by running example files during lectures and performing coding challenges during labs. Each lab session allows you to compare your solution to the instructor's

Documentation

- Digital courseware included

Participant profiles

- Application developers
- Programmers
- System Designers

Prerequisites

- Having followed or have knowledge covered by: Programming Objects with C++ Fundamentals

Objectives

- Apply advanced concepts of OO designs
- Be able to write and maintain C++ programs
- Write robust, maintainable, elegant and efficient C++ code

- Be able to deploy good C++ programming practices
- Be able to use the advanced features of the C++ programming language
- Be able to implement advanced Object-Oriented techniques in C++ to realize efficient and flexible applications
- Leave with skills needed to develop industrial C++ applications

Niveau
Avancé
Duration (in Days)
5
Reference
CPP-02