



ADVANCED C++ PROGRAMMING

Course Duration: 5 days

Course Overview: This course is for experienced C++ programmers who wish to deepen their understanding of the language and learn advanced techniques. The course consists of three modules.

A preliminary module reviews topics, including inheritance, the ANSI C++ Standard Library, templates, I/O streams, and practical issues of C++ programming, such as reliability, testing, efficiency and interfacing to C. This material is covered as needed if the students are missing the proper background for the course. If a student has taken Object Innovations Course 156, this section can be skipped.

The second module covers more advanced topics. Advanced issues of inheritance and polymorphism are covered. Principles of effective class design, including the orthodox canonical form, use of composition, templates and interface inheritance. The course covers exception handling and runtime type information (RTTI). Multiple inheritance is covered, including the complications that are introduced by this powerful feature. Advanced applications of C++ concepts are studied, including smart pointers and reference counting.

The third module introduces the Standard Template Library (STL). The main components of data structures, algorithms and iterators are covered. Illustrations are provided of a number of important containers, such as vectors, stacks, queues, lists and sets.

Extensive programming examples and exercises are provided. A number of larger scale case studies are used to illustrate object oriented programming techniques and to give the student practical experience in putting together features of C++ learned in the course. A file is provided containing all the examples and laboratory exercises in the course.

LEARNING OBJECTIVES

- Review intermediate features of C++ such as inheritance and templates
- Gain a more complete understanding of the issues involved in effective class design
- Learn important features of C++ such as exception handling, RTTI and multiple inheritance
- Learn advanced techniques such as smart pointers and reference counting
- Gain a working knowledge of the Standard Template Library

Prerequisites: C++ programming experience

COURSE OUTLINE

Module 0. Intermediate C++ Topics

Inheritance and Polymorphism

- Class Derivation
- Access Control
- Base Class Initialization
- Initializing Class Type Members
- Polymorphism and Virtual Functions
- Pointer Conversion
- Virtual Destructors
- Abstract Classes and Pure Virtual Functions

ANSI C++ Library

- New Header Files
- Namespaces in the Standard Library
- ANSI C++ String Classes
- Templates in the Standard Library

Templates

- C++ Template Mechanism
- Function Templates
- Class Templates
- Generic Programming
- Implementing a General Array Class
- Standard Template Library

Input/Output in C++

- Streams I/O Library
- Formatted Stream I/O
- File I/O
- I/O in User Defined Classes

Practical Aspects of C++ Programming

- Interfacing C and C++
- Namespaces
- Reliability in C++ Programs
- Testing Considerations
- Efficiency Considerations

Module 1. Advanced C++ Topics

Advanced Polymorphism and Inheritance

- Orthodox Canonical Form
- Public, Private and Protected Inheritance
- Composition vs. Inheritance
- Templates vs. Inheritance
- Interface Encapsulation

Exception Handling

- C++ Exception Mechanism
- Exceptions Compared to Other Error Handling Techniques
- throw, try and catch
- Exception Context and Stack Unwinding
- Uncaught Exceptions
- Automatic Cleanup in Exception Handling

Runtime Type Information

- Runtime Type Information (RTTI) Mechanism
- type_info Class and typeid Operator
- Type Safe Pointer Conversion
- New C++ Cast Syntax

Inheritance Hierarchies and Multiple Inheritance

- Smalltalk Style Class Hierarchies
- Collection Classes in Object-Based Hierarchies
- Independent Class Hierarchies in C++
- Multiple Inheritance
- Resolving Ambiguities
- Duplicate Subobjects
- Virtual Base Classes
- RTTI in Multiple Inheritance

Applications of C++ Concepts

- Object Validation
- Smart Pointers
- Reference Counting
- Generic Smart Pointers

Module 2. Fundamentals of STL

An Overview of Templates

- Templates
- Overloading functions
- Template functions
- Specializing a template function
- Disambiguation under specialization
- Template classes
- An array template class
- Instantiating a template class object
- Rules for templates
- Non member function with a template argument
- Friends of template classes
- Templates with multiple type parameters
- Non type parameters for template classes
- Comments regarding templates

Overview of the Standard Template Library

- Perspective
- History and evolution
- New features in C++
- The Standard Template Library
- Design goals
- Header files
- STL components
- Containers
- Algorithms
- Iterators

Examples from STL

- Example: vectors, lists
- Example: maps
- Example: sets
- Example: multiset
- Example: find with a vector
- Example: find with a list
- Example: merge
- Iterators
- Function objects
- Adaptors

STL Containers

- Vector
- Deque
- List
- The beauty of STL
- Associative Containers
- Set
- Multiset
- Map
- Multimap

STL Iterators

- Input iterators
- Output iterators
- Forward iterators
- Backward iterators