

Advanced C++

Contents

0. Prelude

- Introductions
- Course Overview
- Knowledge Assessment “Quiz”

1. 11th-Hour C++98 Language Additions and Selected New (C++0x) Library Facilities

- New-style Casts
 - The four new cast operators
 - `static_cast`
 - `const_cast`
 - `reinterpret_cast`
 - `dynamic_cast`
 - Capability Queries
- Runtime Type Information
 - `typeid` and `type_info`
 - `typeid` and polymorphism
- Namespaces
 - `using` declarations and `using` directives
 - Anonymous namespaces
 - Argument-dependent lookup
- Boost and TR1
 - `boost::scoped_pointer`
 - Containers of pointers
 - `tr1::shared_ptr`

2. Template Mechanics

- Class templates
 - Class template specialization
 - Template implementation
 - Member function templates
- Function templates
 - Function templates and conversions
 - Overloading function templates

- Mixing template and non-templates
- Implicit template requirements and the role of convention
- Function template argument deduction and overloading
 - Argument deduction
 - Function template explicit arguments
- Class template specialization
 - Explicit specialization
 - Partial specialization
 - Member function explicit specialization
- Non-type template arguments
- SFINAE and INFINI
- Explicit instantiation

3a. Introduction to the Standard Template Library

- Purpose and structure of the STL: containers, algorithms, and iterators
- Review: class and function templates
- Container classification, characteristics, and content
- Iterator classification and behavior
- Generic algorithms and iterators
- Design of generic algorithms and performance guarantees
- iostream iterators

3b. Generic Algorithms

- Review: generic algorithms and helper templates
- Interaction between algorithms and iterators
- Generic algorithm goals, documentation, and design
- Sequences, subsequences, and sequence errors
- Descriptions of STL algorithms

3c. Containers

- Properties, insertion and deletion effects, specific functionality
- vectors
- lists
- maps
- Containers and exceptions
- Choosing an appropriate container
- Container adapters

3d. Function Objects

- Functions and generic algorithms
- Function objects and generic algorithms

- Algorithm families
- Function objects and containers, strict weak ordering
- Standard function objects

4. Exception Handling / Exception Safety

- The purpose of exception handling
- Syntax and mechanics of the exception handling mechanism
- Throwing exceptions
 - `throw` expressions
- Handling exceptions
 - `try` blocks and handlers
 - `catch` clauses
- Exception specifications
 - Exceptions and inheritance
 - MI for exception types
 - RAII
 - Hierarchical exceptions
- Designing exception types
- Designing for exceptions
 - Catch by reference
 - Exception safety
 - Levels of exception safety
 - `auto_ptr`
 - Plugging resource leaks
 - Partially constructed objects
 - Resource leaks in constructors
 - Function `try` blocks
 - Copy assignment idioms

5. Advanced Memory Management

- `new` and `delete` operators
- New handlers
- Exceptions and memory management
- Placement `new` and explicit destruction
- Member operator `new` and operator `delete`
- Array `new` and `delete` operators

6. Copying, Conversions and Temporaries

- Assignment and initialization
- Copy operations and class mechanism

- Deep vs. shallow copy
 - Compiler-supplied default copy semantics
- Copying problems
 - Compiler-supplied copy operations
- Implementing copy operations
 - Bitwise copy
 - Coding copy constructors
 - Assignment in a hierarchy
- User-defined conversions
- Unintended conversions and `explicit`
- Conversions, temporaries, and efficiency
 - Computational constructors
- Temporaries and copy construction
 - Direct vs. copy initialization
- Return value optimizations
- Conversions, temporaries, and references
- Temporary lifetime and correctness

7. Inheritance and Object-Oriented Design (Special Unit by Scott Meyers)

- Make Sure Public Inheritance Models “is a”
 - Public inheritance and intuition
 - Runtime errors vs. compile-time errors
 - Substitutability
- Differentiate Between Inheritance of Interface and Inheritance of Implementation
 - Pure virtual, non-pure virtual and non-virtual functions
 - Coupling mandatory interface with default implementation
 - Redefining non-virtual functions
- Avoid Casts Down the Interface Hierarchy
 - RTTI and safe downcasting
- Model “Has-a” or “Is-implemented in terms of” Through Composition
 - The meaning of composition
 - “Is a” vs. “is-implemented in terms of”
- Use Private Inheritance Judiciously
 - The behavior/meaning of private inheritance
 - Type-safe interfaces
- Use Multiple Inheritance Judiciously
 - Ambiguity and multiple inheritance
 - Virtual base classes
- Software Evolution and Multiple Inheritance

8. C++ Pointers, References and Addresses

(Time permitting – provided in PDF form only)

- References
 - References are aliases, not pointers
 - Reference initialization
 - Pass by reference
 - References and casting
- `const` and pointers
 - Safe and unsafe conversions
 - `const` formal parameters
- `const` member functions
 - Physical vs. logical constness
 - mutable data members
 - `volatile`
 - Casting away `const`
 - Overloading on `const`
- `const` and references
- Multilevel pointers and references to pointers
- Arrays
 - Pointer/array duality
 - References to arrays
 - Arrays of class objects
- `typedef` and type sinks
- The meaning of pointer comparison
- `void *` and casting