

Navigating C++ and Object-Oriented Design

Chapter 1 Getting Started

- 1.1 Object-Oriented Design
 - Abstraction
- 1.2 Object Modeling Technique (OMT)
- 1.3 Object Model Notation
 - Classes
 - Associations
 - Multiplicity
 - Link Attributes and Association Classes
 - Role Names
 - Ordering
 - Qualification
 - Aggregation
 - Generalization
 - Generalization with Multiple Inheritance
 - Virtual Base Classes
- 1.4 Scenarios
 - Error Conditions
- 1.5 Putting It All Together
 - The Modeling Process
 - Library System Description
 - Library System Object Model
 - Initial C++ Class Definitions
 - OMT Notation Summary
- 1.6 Key Point Summary
- 1.7 Exercises

Chapter 2 C++ Basics

- 2.1 Data Representation and Built-in Types
 - Constants
 - Program Variables
 - Constant Variable Types
 - Enumerated Types
 - Arrays
 - Lvalues and Rvalues
 - Pointers
 - Pointers with const
 - Array and pointer Relationship
 - References
 - Typedefs
 - Volatile
 - Casts and Conversions
 - Void Pointers
 - bool
- 2.2 Preprocessor Directives
 - Include File Directive
 - Define and Undefine Directives
 - Macros
 - Conditional Compilation
- 2.3 Comments
- 2.4 Type-Safe I/O

- 2.5 Operators and Expressions
 - Arithmetic Operators
 - Relational Operators
 - Logical Operators
 - Bitwise Operators
 - Assignment Operators
 - Conditional Operator
 - Increment and Decrement Operators
 - Compact Pointer Expressions
 - Comma Operator
 - Sizeof Operator
 - Typeid Operator
 - The Scope Resolution Operator (::)
- 2.6 Control Flow Constructs
 - if and if-else
 - while
 - do while
 - for
 - switch
 - break and continue
 - goto
- 2.7 Putting It All Together
 - Pointer Arrays
- 2.8 Key Point Summary
- 2.9 Exercises

Chapter 3 C++ Program Structure

- 3.1 Functions
 - Inline Functions
 - Recursive Functions
 - Pointers to Functions
 - Arrays as Function Arguments
 - Default Function Arguments
 - Functions with a Variable Number of Arguments
- 3.2 Structures and Unions
 - Structures
 - Member Functions
 - Structure Pointers
 - Arrays with Structures
 - Nested Structures
 - Typedefs with Structures
 - Structure Copy and Assignment
 - Unions
 - Bitfields
- 3.3 References with Functions
 - References as Function Arguments
 - References as Function Return Types
 - References to Pointers
- 3.4 Storage Classes
 - auto
 - static
 - register
 - extern
 - mutable

- 3.5 Exceptions
 - try
 - catch
 - throw
 - Exception Specifications
- 3.6 Namespaces
 - Why Use Namespaces?
 - Namespace Definitions
 - Namespace Extensions
 - Accessing Namespace Members
 - Unnamed Namespaces
 - Nested Namespaces
 - Using Directives
 - Using Declarations
 - Namespace Aliases
 - Namespaces with Program Development
- 3.7 Dynamic Memory Allocation
 - Operator new
 - Error Handling
 - Operator delete
 - new and delete with Pointer Arrays
 - new and delete with Multidimensional Arrays
- 3.8 Putting It All Together
 - A Money Bag
 - An Optimized Block Move Function
- 3.9 Key Point Summary
- 3.10 Exercises

Chapter 4 Classes

- 4.1 What Is Encapsulation?
 - Fifo Objects
 - Controlled Interfaces
 - Private and Public
- 4.2 Classes
 - Using the Scope Operator (::)
- 4.3 Constructors
 - Default Constructors
- 4.4 Destructors
- 4.5 What's this?
- 4.6 Exception Objects
- 4.7 Const Objects
 - Const Member Functions
 - Mutable Data Members
- 4.8 Volatile Objects
 - Volatile Member Functions
- 4.9 Copy Constructors
- 4.10 The Problem with Public Data Members
- 4.11 Data Member Objects
- 4.12 Class Member Initialization
 - Const Data Members
 - Reference Data Members
- 4.13 Putting It All Together
 - A StringN Class
 - A Record Class
- 4.14 Key Point Summary
- 4.15 Exercises

Chapter 5 Working with Classes

- 5.1 Using explicit
- 5.2 Object Lifetimes
 - Containment
 - Pointers to Objects
 - References to Objects
- 5.3 Static Data Members
- 5.4 Static Member Functions
- 5.5 Static Objects
- 5.6 Arrays of Class Objects
 - Constructors with Arrays of Class Objects
 - Destructors with Arrays of Class Objects
- 5.7 Pointers to Class Members
 - Pointers to Static Members
 - Pointers to Nonstatic Data Members
 - Points to Nonstatic Class Member Functions
- 5.8 Friend Classes
- 5.9 Nested Classes
- 5.10 Local Classes
- 5.11 Putting It All Together
 - Namespaces and Classes
 - Namespaces and Libraries
- 5.12 Key Point Summary
- 5.13 Exercises

Chapter 6 Overloading

- 6.1 Why Overload Functions?
- 6.2 Function Overloading
 - Default Arguments
 - Signatures
 - Return Types
- 6.3 Overloading Resolution
 - Single Argument Matching
 - Exact Match
 - Trivial Conversions
 - Promotions
 - Standard Conversions
 - User-Defined Conversions
 - Ellipsis (...)
 - Multiple Argument Matching
 - Intersection Rule
 - Overloading and Namespaces
- 6.4 Why Overload Operators?
- 6.5 Overloadable Operators
- 6.6 Operator Functions
 - Operator Nonmember Functions
 - Operator Member Functions
 - Conversions with Cast Operators
- 6.7 Putting It All Together
 - A Character List Class
 - Smart Pointers
- 6.8 Key Point Summary
- 6.9 Exercises

Chapter 7 Class Design

- 7.1 What Is Class Design?
- 7.2 A Class Design Boilerplate
 - Operator=()
 - A Class Design Checklist
- 7.3 A String Class
 - String Conversion
 - String Assignment
 - String Concatenation
 - String Comparison
 - String Subscripts
 - Substrings
 - Overloading Streams
 - Friend Functions
- 7.4 A Range Integer Class
 - Range Integer Conversion
 - Range Integer Assignment
 - Range Integer Prefix and Postfix Operators
 - Range Integer Input and Output
- 7.5 Putting It All Together
 - Lvalue and Rvalue Separation
- 7.6 Key Point Summary
- 7.7 Exercises

Chapter 8 Object Storage Management

- 8.1 Global new and delete
 - Nothrow Operator new
 - Placement with Operator new
 - Static and Global Buffer Placement
 - Free Store Placement
 - Arena Placement
 - Explicit Destructor Calls
 - User-Defined Operators new and delete
- 8.2 Class-Specific new and delete
 - Memory Pools
- 8.3 Reference Counts
 - A String Class with Reference Counts
 - Copy on Write
 - Adding Reference Counts to Existing Classes
- 8.4 Putting It all Together
 - A Memory Leak Detector
- 8.5 Key Point Summary
- 8.6 Exercises

Chapter 9 Template Functions

- 9.1 Why Should Functions Be Generic?
- 9.2 Template Function Definition
- 9.3 Template Function Instantiation
 - Using export
 - Function Invocation
 - Pointers to Template Functions
- 9.4 Overloading Template Functions
 - Template Function Argument Conversions
- 9.5 Specializing Template Functions
 - Specializing for Correctness
 - Specializing for Performance

- 9.6 Putting It All Together
 - A Generic Two-dimensional Array Function
 - A Generic Transpose() Function
- 9.7 Key Point Summary
- 9.8 Exercises

Chapter 10 Template Classes

- 10.1 Why Should Classes Be Generic?
- 10.2 Template Class Definition
 - A Generic Stack Class
 - Template Default Types
- 10.3 Template Class Instantiation
 - A Generic One-Dimensional Array Class
- 10.4 Specializing Template Classes
 - A Generic Block Class
- 10.5 Containment with Template Classes
 - A Generic Fifo Class
- 10.6 Composite Templates
 - A Generic Two-Dimensional Array Class
 - A Generic Three-Dimensional Array Class
- 10.7 Template Class Static Members
- 10.8 Constant Expression Parameters
 - A Generic Buffer Class
 - Templates with Typedefs and Enumerations
- 10.9 Template Friend Functions
- 10.10 Template Friend Classes
- 10.11 Template Nested Classes
 - A Generic Associative Array Class
- 10.12 Member Templates
 - Specializing Member Templates
- 10.13 Putting It All Together
 - A Generic List Class with Iterators and Value Semantics
- 10.14 Key Point Summary
- 10.15 Exercises

Chapter 11 Inheritance

- 11.1 Why Use Inheritance?
 - A Procedure-Oriented Approach
 - An Object-Oriented Approach
- 11.2 Public Derivation
 - Dominance
 - Constructors and Destructors
 - Base Class Initialization
 - Object Maps
- 11.3 Subtypes
 - Static Binding
- 11.4 Virtual Functions
 - Dynamic Binding
 - Virtual Definitions
 - Virtual Destructors
 - Polymorphic Programming
- 11.5 Protected Access
 - Access Restrictions
 - Two-Dimensional Array Class Revisited
 - Virtual Functions and Access Levels
- 11.6 Using Declarations

- 11.7 Private Derivation
 - A Bounded Array Class
 - Using Declarations
 - Forwarding
- 11.8 Protected Derivation
 - A Grid Class Hierarchy
- 11.9 Summary of Derivation Options
- 11.10 Polymorphism with a Generic PtrList Class
 - A PtrList Class
 - A PtrIterator Class
- 11.11 Abstract Base Classes
- 11.12 Virtual Constructors
 - Virtual Object Functions
 - Virtual Copy Constructors
 - Virtual Functions in Constructors and Destructors
- 11.13 A Class Design Boilerplate with Inheritance
 - A Class Design Checklist
- 11.14 Putting It All Together
 - Container Classes, Pointers, and Copy Semantics
 - Template Classes with Inheritance
 - Virtual Function Implementation
- 11.15 Key Point Summary
- 11.16 Exercises

Chapter 12 Run-Time Type Identification

- 12.1 Why Is RTTI Necessary?
- 12.2 The Dynamic Cast Operator
 - dynamic_cast with Pointers
 - dynamic_cast with References
- 12.3 The typeid Operator
 - typeid Class
- 12.4 RTTI Applications
 - Working with Class Hierarchies
 - Extending Class Libraries
 - Controlling Object Behaviors
- 12.5 Putting It All Together
 - RTTI Implementation
 - Persistence
- 12.6 Key Point Summary
- 12.7 Exercises

Chapter 13 Exception Handling

- 13.1 Why Use Exceptions?
- 13.2 Designing with Exceptions
 - try Blocks
 - Throwing Exceptions
 - Rethrowing Exceptions
 - Catching Exceptions
 - Exception Objects
 - Matching Exceptions to Handlers
- 13.3 Exception Hierarchies
 - Virtual Response Functions
 - Rethrowing Exceptions from Hierarchies
 - An ArrayError Exception Hierarchy
 - Discriminating Exceptions

- 13.4 Resource Management
 - The auto_ptr Class
 - Exceptions Thrown from Constructors
 - Placement Operator delete
 - Exceptions Thrown from Destructors
 - Specialized Pointer Classes
- 13.5 Uncaught Exceptions
 - The terminate() Function
- 13.6 Exception Specifications
 - Designing with Exception Specifications
 - Standard Exception Classes
- 13.7 Unexpected Exceptions
 - The unexpected() Function
 - Handling Unexpected Exceptions
- 13.8 Putting It All Together
 - A RintError Exception Hierarchy
- 13.9 Key Point Summary
- 13.10 Exercises

Chapter 14 Multiple Inheritance

- 14.1 Why Use Multiple Inheritance?
 - Dynamic Classification
- 14.2 Multiple Inheritance Format
 - Base Class Initialization
- 14.3 Multiple Inheritance Characteristics
 - Ambiguities
 - Dominance
 - Polymorphism
 - Structural Patterns with Multiple Inheritance
- 14.4 Distinct Base Classes (Pattern 1)
 - Pattern 1 Example
 - Polymorphism
 - Dynamic Classification
 - Resolving Ambiguities
 - Pattern 1 Object Layout and Conversions
- 14.5 Multiple Inclusion (Pattern 2)
 - Pattern 2 Example
 - Polymorphism
 - Resolving Ambiguities
 - Pattern 2 Object Layout and Conversions
- 14.6 Virtual Base Classes (Pattern 3)
 - Pattern 3 Example
 - Polymorphism
 - Base Class Initializers
 - Pattern 3 Object Layout and Conversions
- 14.7 Interface and Implementation
- 14.8 Putting It All Together
 - Implementing Persistence with Multiple Inheritance
 - Dominance with Virtual Base Classes
 - Virtual Functions Implementation with Multiple Inheritance
- 14.9 Key Point Summary
- 14.10 Exercises

Appendix A IOStream Library

- A.1 Why Use the IOStream Library?
- A.2 IOStream Overview
 - Stream File I/O
 - Random Access with Streams
 - Standard String Streams
 - Delimiters
 - Stream Input
 - Stream Output
 - Manipulators
 - Error Conditions
- A.3 IOStream Examples
 - Hue
 - File Stream Output
 - File Stream Input
 - File Stream Input and Output
 - Appending to a File Stream
 - Word Count Program

Appendix B Standard Template Library

- B.1 Why Use STL?
- B.2 STL Overview
 - Containers
 - Iterators
 - Generic Algorithms
- B.3 STL Examples
 - Vector
 - List
 - Deque
 - Queue
 - Stack
 - Map
 - Pointer Semantics
- B.4 STL References

Appendix C C++ Operator Precedence

- C.1 C++ Operator Precedence (Table)