Advanced C++ Programming

Course Summary

Description

The comprehensive, five-day 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 depending on the background of the students. The second module covers more advanced topics. Advanced issues of inheritance and polymorphism are covered, as are the 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 progressively developed 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.

Topics

- Inheritance and Polymorphism
- ANSI C++ Library
- Templates
- Input/Output in C++
- Practical Aspects of C++ Programming
- Advanced Polymorphism and Inheritance
- Exception Handling
- Runtime Type Information
- Inheritance Hierarchies and Multiple Inheritances
- Applications of C++ Concepts
- An Overview of Templates
- Overview of STL
- Examples from STL
- STL Containers
- STL Iterators

Audience

This course is designed for experienced C++ programmers who wish to deepen their understanding of the language and learn advanced techniques.

Prerequisites

Before taking this course, students should have substantial C++ programming experience.

Duration

Five days

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically

Advanced C++ Programming

Course Outline

I. **Inheritance and Polymorphism**
A. Inheritance Concept
B. Inheritance in C++
C. Protected Members
D. Base Class Initializer List
E. Composition
F. Member Initialization List
G. Order of Initialization
H. Inheritance vs. Composition
I. Summary – Inheritance
J. A Case for Polymorphism
K. Dynamic Binding
L. Pointer Conversion in Inheritance
M. Polymorphism Using Dynamic Binding
N. Virtual Function Specification
O. Invoking Virtual Functions
P. VTable
Q. Virtual Destructors
R. Abstract Class Using Pure Virtual Function
S. Employee as an Abstract Class
T. Heterogeneous Collections
U. Summary – Polymorphism

II. **ANSI C++ Library**
A. ANSI C++ Library
B. Hello ANSI C++
C. Namespaces
D. ANSI C++ String Class
E. Templates

III. **Templates**
A. General Purpose Functions
B. Macros
C. Function Templates
D. Template Parameters
E. Template Parameter Conversion
F. Function Template Problem
G. Generic Programming
H. General Purpose Classes
I. Class Templates
J. Array Class Implementation (array.h)
K. Using the Array Template
L. Template Parameters
M. Class Template Instantiation
N. Non Type Parameter Conversion
O. Standard Template Library
P. STL Components
Q. Generic Programming
R. STL Elements of a Simple Program
S. Simple STL Program
T. Map Container

IV. **Input/Output in C++**
A. Input/Output in C++
B. Built-in Stream Objects
C. Output Operator <<
D. Input Operator >>
E. Character Input
F. String Input
G. Formatted I/O
H. Streams Hierarchy (Simplified)
I. File I/O
J. File Opening
K. Integer File Copy
L. Character File Copy
M. Overloading Stream Operators
N. Implementing Overloaded Stream Operators

V. **Practical Aspects of C++ Programming**
A. Interfacing C++ to Other Languages
B. Calling C from C++
C. _cplusplus Macro
D. Calling C++ from C
E. Interface Module for Stack Class
F. Namespace Collisions
G. ANSI Namespace
H. Reliability Philosophies of Languages
I. Prototypes and Type Checking
J. Constant Types
K. Access Control in C++
L. Reviews and Inspections
M. Inspections and C++
N. Testing Strategies for C++
O. Performance Considerations
P. Class Libraries

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically

Advanced C++ Programming

Course Outline (cont'd)

VI. Advanced Polymorphism and Inheritance
- A. Good Class Design
- B. String Class
- C. Public Inheritance
- D. Public Inheritance Problems
- E. Inheritance and Semantics
- F. Private Inheritance
- G. Composition
- H. Composition vs. Private Inheritance
- I. Templates vs. Inheritance
- J. Protected Inheritance
- K. Implementation Encapsulation
- L. Interface Inheritance

VII. Exception Handling
- A. Exception Handling
- B. try and catch
- C. Exception Flow of Control
- D. Context and Stack Unwinding
- E. Handling Exceptions in best Context
- F. Benefits of Exception Handling
- G. Unhandled Exceptions
- H. Clean Up
- I. Multiple Catch Handlers

VIII. Runtime Type Information
- A. Runtime Type and Polymorphism
- B. type_info Class
- C. typeid Operator
- D. Compiler Options
- E. Safe Pointer Conversions
- F. Dynamic Cast
- G. New C++ Style Casts
- H. Static Cast
- I. Reinterpret Cast
- J. Const Cast

IX. Inheritance Hierarchies and Multiple Inheritances
- A. Class Hierarchy in Smalltalk
- B. Smalltalk Class Hierarchy (Simplified)
- C. Collection Classers
- D. Multiple Inheritance Solution
- E. Basic Derivation
- F. Ambiguities in Multiple Inheritance
- G. Resolving Ambiguity
- H. Duplicate Subobjects
- I. Virtual Base Classes

X. Applications of C++ Concepts
- A. Orthodox Canonical Form (Review)
- B. Object Validation
- C. String Class
- D. Smart Strings
- E. Reference Counting
- F. © ITCourseware, LLC 3
- G. Advanced C++ Programming
- H. Reference Counting Rules
- I. Smart String Pointer
- J. Generic Smart Pointers
- K. Constructing Smart Pointers
- L. Smart Pointer Difficulties

XI. An Overview of Templates
- A. Templates
- B. Overloading Functions
- C. Template Functions
- D. Specializing a Template Function
- E. Disambiguation under Specialization
- F. Template Classes
- G. An Array Template Class
- H. Instantiating a Template Class Object
- I. Friends of Template Classes
- J. Templates with Multiple Type Parameters
- K. Non Class-type Parameters for Template Classes
- L. Comments Regarding Templates
- M. Templates and Inheritance

XII. Overview of STL
- A. Perspective
- B. History and Evolution
- C. New Features
- D. The Standard Template Library
- E. Generic Programming
- F. Design Goals

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically

Advanced C++ Programming

Course Outline (cont'd)

G. Header Files
H. STL Components
I. Containers
J. Algorithms
K. Iterators
L. Compiling STL Code

XIII. Examples from STL
A. vector
B. Vector.cpp
C. list
D. List.cpp
E. map
F. Map.cpp
G. set
H. Set.cpp
I. multiset
J. Multiset.cpp
K. find
L. FindVector.cpp
M. find – list
N. merge
O. Overriding the Default Comparison
P. Iterators
Q. Iterators.cpp
R. Functions
S. Functions.cpp
T. Function Objects
U. FunctionObject.cpp

XIV. STL Containers
A. Vectors
B. Vector.cpp
C. Vector Operations
D. Typedefs
E. Deques
F. deque as Stack
G. deque<T> Functionality
H. Lists
I. Generic Programming
J. Tradeoff with Lists
K. List Memory Allocation
L. list Functionality
M. Associate Containers
N. Sets
O. Sets with User Defined Objects
P. Multisets (Bags)
Q. Maps
R. Multimaps

XV. STL Iterators
A. Pointers
B. Template Version
C. String Version
D. A Generalization of Pointers
E. STL Iterators
F. Input Iterators
G. Output Iterators
H. Forward Iterators
I. Bidirectional Iterators
J. Random Access Iterators

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically