Advanced C++ Programming

Course Summary

Description

This course will teach the student how to solve problems arising from subtleties of the C++ language as well as techniques for improving performance and efficiency. Students are invited to bring their current ideas and question to the classroom for discussion

Objectives

At the completion of this course, the student will be able to:

- Point out the areas where hidden class members can cause hard-to-find bugs
- Write easier-to-read and easier-to-code using operator overloading
- Perform file input-output and describe the concepts of persistent objects
- Understand issues in using virtual base classes and multiple inheritance
- Better utilize the features of inheritance and polymorphism in program design
- Understand how templates can reduce code replication when used properly
- Use and understand the techniques of exception handling in C++
- Describe when/how to use run-time type identification, mutable keyword/other language features

Topics

- Why object oriented programming?
- A review of C++ and classes
- Nested classes
- Const and static members
- Pointers to members
- Hidden class details
- Resolution of overloaded functions
- Operator overloading in detail
- Streams-based file I/O
- Multiple inheritance and virtual base classes
- Protected methods: the good and the bad
- New-style casts
- Template classes and functions
- Exception handling
- Run-time type identification
- Namespaces

Audience

This course is intended for engineers, programmers and analysts who have been designing and creating programs using C++ and desire a higher level of technical detail to allow them more flexibility and productivity when writing C++ programs. Anyone desiring the ability to read and understand complex C++ programs for maintenance or learning purposes.

Prerequisites

Programming experience in C++ for at least three months is required for students to gain maximum benefit, although experience in other object oriented programming languages may suffice.

Duration

Four days

Advanced C++ Programming

Course Summary

I. Why Object Oriented Programming?
- A. Modularity
- B. Data Hiding
- C. Abstraction
- D. Encapsulation

II. Review of C++ and Classes
- A. Nested classes
- B. Const and static members
- C. Pointers to members
- D. Introduction to namespaces
- E. Introduction to standard template Library (STL)
- F. The this pointer
- G. Nested classes

III. Polymorphism
- A. What any why
- B. Virtual functions
- C. Costs involved in use
- D. Common problems

IV. Name Mangling
- A. What and why
- B. How this can help decipher compiler error messages

V. Operator Overloading
- A. What and why
- B. Operators that cannot be overloaded
- C. Prefix vs. postfix for ++ / --
- D. Overloading []
- E. Using a global function vs. member method
- F. Best practices
- G. Implicit conversions
- H. Explicit conversions

VI. Templates
- A. What and why
- B. How to use
- C. How to create
- D. Template instantiation
- E. Function templates
- F. User specializations
- G. Source code organization
- H. More STL

VII. Stream I/O
- A. Stream I/O vs. file I/O
- B. Class hierarchy
- C. I/O manipulators
- D. Formatting using manipulators
- E. Buffering issues
- F. Adding overloaded I/O operators
- G. String stream
- H. Disk file I/O

VIII. Inheritance
- A. What and why
- B. Composition
- C. Upcasting vs. downcasting
- D. Virtual functions
- E. Abstract base classes
- F. Multiple inheritance (MI)
- G. MI Issues / solutions

IX. Standard Template Library
- A. What and why
- B. Tour of included classes
- C. How to use
- D. Iterators

X. Exception Handlers
- A. What and why
- B. Overhead issues
- C. When not to use exceptions
- D. How to write and use

XI. New Style Casts
- A. What and why
- B. Const cast
- C. Static cast
- D. Dynamic cast
- E. Reinterpret cast

XII. Run-Time Type Identification (RTTI): What and Why?