C++ for C Programmers

Course 6056 – 32 Hours

Overview

C++ is undoubtedly one of the most popular programming languages for software development. It brings language enhancements and object-oriented programming support to C++. However, C++ is a large and sometimes difficult language, and even with a C background, a programmer needs to understand C++ programming style as well as C++ constructs to get the best out of it. Our highly successful C++ training for C Programmers course has a thorough theoretical and practical coverage of the language. This helps to eliminate the misconceptions and poor programming practice that can cause so many problems.
For experienced C programmers, the course will illustrate how to get the benefits of good software engineering and code reuse by using standard C++ and object-oriented programming techniques in real-world programming situations. There will be ample opportunity to apply these techniques during the practical sessions.
This is a hands on course with a mix of tuition and practical sessions for each technical chapter which reinforce the C++ syntax and object-oriented programming techniques covered in the course.

On Completion, Delegates will be able to

- Define classes.
- Write class member functions.
- Use constructors, destructors and dynamic memory.
- Write code that is efficient and robust.
- Build new classes from other classes using inheritance, composition and association.
- Use template classes.
- Use operator overloading.
- Design and write classes with polymorphic behavior.

Who Should Attend

- C Programmers wishing to learn or improve in C++.

Prerequisites

- Delegates must have solid experience of C including structures (i.e. struct and/or class); Declaration and use of pointers; Function declaration, definition and use with call by value or call by pointer; Dynamic memory allocation (i.e. malloc and free, or new and delete); Multiple source file projects (requiring project files or makes files). You should

also have a good appreciation of object-oriented principles or have attended our Object-Oriented Primer course.
- Delegates with less than four months or no recent experience of C should attend the Programming with C++ course instead.

Course Contents

Course Introduction
- Course Prerequisites
- Course Objectives
- Course Delivery
- Course Practical's
- Course Structure

An Overview of OO Programming and C++
- Review of OOP principles
- Behavior, state, identity, inheritance, polymorphism
- History and evolution of C++
- Key features of C++
- C++ as a better and safer C

The Class Approach
- Grouping of data and functionality
- Syntax of a class declaration
- Syntax of use
- Public and private
- Abstract Data Types
- Program structure

Providing Class Functionality
- Member functions
- Function overloading
- Default arguments
- Ambiguities
- Anonymous arguments
- Resolving scope conflicts
- The Scope resolution operator
- The this pointer

Object birth and death
- Life of an object
- Constructors
- operator new
- Death of an object
- Destructors
- operator delete
- Dynamic arrays

Efficiency and Integrity Issues

- Enumerations
- Const declarations
- Const member functions
- Const member data
- Inline function mechanism
- Reference variables

Composite Classes

- An opportunity for reuse
- Scoping and initialisation
- Order of construction
- Member Initialisation lists
- Use of fundamental classes

Associative Classes

- Delegating class functionality
- Dynamic associations
- Custody and lifetime
- Constant associations

Operator Overloading

- Operator functions
- Unary operators
- Binary operators
- Global operators
- Member operators
- Subscript operators
- Input operators
- Output operators
- Guidelines

Class Properties

- Static data members
- Static member functions
- Nested types
- Forward declarations
- Friend classes

Container Classes

- Organising collections of objects
- Template classes
- vector
- list
- Iterators
- Template functions
- Algorithms

- Using the Standard Library

Copying and Conversions

- The copy assignment operator
- Copy constructors
- Conversions to a class object
- Conversions from a class object

Inheritance

- Extension of existing classes
- Notation, syntax, terminology
- Protected members
- Scoping and initialisation
- Multiple inheritance
- Abstract base classes
- Guidelines

Polymorphism

- Modified class behavior
- Virtual functions
- virtual destructors
- Late binding
- Inside the virtual function mechanism
- Pure virtual functions
- Use of pointers to base type
- Guidelines

C++ 11 Standard Enhancements

- Namespaces
- Overview of exception handling
- Smart pointers
- Customized memory management
- Interfacing with C
- Class design recommendations