



Course AC++-501: Advanced C++ for Embedded Systems

Course Description:

Since its standardisation in 1998, the support and use of C++ as an embedded programming language has grown significantly. However, until recently most C++ cross compilers were actually "Embedded C++" (EC++) compilers or were typically inefficient in the more advanced areas of the language. To date, a high proportion of programmers using C++ for embedded development have either been unable or have chosen not to use certain features of C++ owing to a fear of bloated code with poor performance.

This course addresses the "fear, uncertainty and doubt" of using full C++. Specifically the course deals with: performance and memory considerations of polymorphic functions, exceptions and templates. In addition, coverage of the Standard Library, including the STL, is provided, again with the specifics of performance and memory models.

Overview:

A five-day course which reviews C++ in the light of real-time systems, addresses the application of C++ in a real-time/embedded environment, and then focuses on the advanced parts of the language. Fifty percent of the course is spent on practical work and the course includes the use of target hardware.

Course Objectives:

- To provide an understanding of the advanced aspects of the C++ programming language.
- To give you practical experience of writing efficient C++ code for real-time and embedded systems.
- To understand the impact different compilers have on performance.

Delegates Will Learn:

- About memory and performance issues associated with C++
- How to access hardware, write device drivers, and program interrupt handlers in C++.
- The overheads of exceptions
- How templates work
- Different approaches to integrating real-time operating systems with C++.

Prerequisites:

- Some experience of programming with C++.

Who Should Attend?

The course is designed for real-time engineers who either (a) have a working knowledge of C++ but are embarking on a real-time project using C++ for the first time, or (b) have been using EC++ to date and want to extend their knowledge of full C++ for embedded systems programming.

Duration:

Five days.

Course Materials:

Delegate Handbook.

Related Courses:

- OO-503 Real-Time Software Design with UML 2.0
- C++-501 C++ for Embedded Developers
- RTOS-201 Fundamentals of Real-Time Operating Systems.

Course Workshop:

The course makes use of target hardware during the real-time practical exercises. The development board is based around a NXP LPC2129 ARM7-based microcontroller.

Course Outline:

Introduction

C++ Performance

- Member functions
- Static functions
- Inheritance and virtual tables
- Run-time type information (RTTI)

Embedded C++

- Why Embedded C++ was developed
- Embedded C++ features.
- Migrating from EC++ to full C++

Real-Time Specifics

- Accessing hardware
- Manipulating information at the bit level

Interrupts

- Different interrupt models
- Function model
 - Name encoding
- Class model
 - Device "has an" interrupt
 - Memory overhead

Functions and Operators

- Class defined conversions
- Overloading and function selection
- Friend functions and classes
- Overloading operators
- Dynamic memory allocation revisited
 - Assignment
 - Copy constructors

Exception Handling

- What are exceptions?
- Throwing an exception
- The try block
- Catching an exception
- Rethrowing exceptions
- Catch-all handlers
- Exception specifications
- Exception models and overheads

Templates

- Introduce parameterised types and functions:
 - function templates
 - class templates
- Performance implications

The Standard Library

- Introduction to the Standard Library.
- The STL
- Using the STL efficiently

Software Structuring

- Consider how to structure large scale software systems
- Separate implementation from interface header files
- Dealing with name conflicts
- Linking with other languages

Target Specific Considerations

- Portability considerations
- Non-standard C++ language features
- Assembly language interfacing.
- Designing ROMable objects

Design Patterns

- What patterns are and are not
- Pattern types
 - architectural
 - design
 - idioms
- Pattern examples

Concurrency

- Concurrency
- Scheduling strategies
- Task-Is-Polymorphic
- Task-Runs-Polymorphic
- Sharing resources in multi-tasking systems
- Mutex objects
- Synchronising tasks
- Transferring data between tasks

Course Summary

FEABHAS

Feabhas Ltd

5, Lowesden Works
Lambourn Woodlands
Hungerford, Berkshire
RG17 7RY, UK

Tel: +44 (0) 1488 73050

Fax: +44 (0) 1488 73051

Email: info@feabhas.com

Web: www.feabhas.com