

Price: R8,600.00 excl. VAT
Duration: 5 days
Code: ACPLS

Advanced C++ Programming

Description

The Advanced C++ course provides a deeper look at the C++ language, without reliance on any particular compiler. Advanced topics include special member functions, templates and memory management. It also discussed object-oriented techniques and the difficulties experienced with OOP.

Objectives

Delegates who complete the Advanced C++ Programming course will be able to:

- Understand advanced C++ development techniques.
- Understand and use the standard template library.
- Apply OOP techniques in C++ programs.

Intended Audience

C++ programmers and engineers who require more advanced knowledge of C++ programming techniques.

Prerequisites

Experienced C++ programmers. A minimum would be the Standard C++ Programming course and some practical experience.

Course Contents

The lecturer reserves the right to modify the contents of the course to suit the needs of the delegates.

Fundamentals • Overview of C++ and ANSI C++ Standard. • C and C++ differences, portability issues and using C legacy code in C++. • Structures and classes. • Function overloading. • Operators as functions. • Default parameters. • Variable number of parameters. • Constants and macros. • Inline functions. • Namespaces syntax and usage. • The standard library namespace. • Practical issues and tips for using namespaces. • Understanding and using references. • Run time type information (RTTI). • Name mangling.

Exception Handling • Syntax, structure and usage. • Exception handling options. • Advanced error handling. • Memory allocation error handling. • Classes for exception handling.

Pointers and Memory Management • Performance issues. • Understanding new, delete and other memory allocation options in various environments. • Overloading new and delete. • Dangling pointers. • Smart pointer classes. • Reference counting. • Shallow vs deep copying.

Classes, Inheritance and Polymorphism • Inheritance concepts. • Multiple inheritance. • Virtual functions and the vtable. • Abstract classes. • Pure virtual functions. • Access specifiers and usage. • Private or protected constructors and/or destructors. • Canonical classes. • Iterator and envelope classes. • Class factories.

Templates • Overview. • Template definition syntax. • Function templates. • Template parameters. • Class templates. • Deriving from templates. • Smart pointers with templates.

Standard Template Libraries • Principles and design issues. • Practical use of the STL containers, algorithms, data models, adaptors and allocators. • Tradeoffs.

Object-Oriented Design Principles • Overview. • Unified Modelling Language. • Class relationships and associations. • Containment, composition and aggregation. • HAS, USES and IS relationships. • Interfaces and abstract classes.