C++ from non-C

An Introduction to C++ for non-C Programmers

ISO standard C++ is a general-purpose language that bridges different styles of programming and spans different platforms and different application styles.

The *C++ from non-C* course presents C++ by starting with modern C++ concepts and techniques rather than assuming, or seeking to provide, a C background. It develops the concepts and syntax through lectures, discussion and lab exercises.

Objectives

- Introduce C++ as a modern programming language, whilst appreciating its relationship with C
- Understand object-oriented and generic programming in C++
- Emphasise good practice and outline idioms for safe and sensible use of language features

Audience

The course is suitable for experienced programmers with little or no C background. Some previous exposure to object-oriented concepts is required.

Content

A Brief Tour The *main* function · Header file inclusion · The standard library and *namespace std* · Statements, blocks and scope · Comments · Functions · Stream I/O · Containers · Classes

Fundamental Types C++ built-in arithmetic types · Declaration and initialisation · Defining constants · Conversions between types · Arithmetic operators · Boolean operators

User-Defined Types *typedef*, *enum* and *struct* · Using *string*, *vector* and *map* types

Control Flow Selection using *if else* and *switch* · Loops · *try*, *throw* and *catch* · Exception safety

Functions Declaring and defining functions · Pass by copy · Pass by reference and *const* reference · Guidelines on argument passing · Function overloading · Operator overloading

Pointers and Arrays Pointer declaration and usage · Array declaration and usage · Strings

Classes and Objects Encapsulation · Classes, member functions and data · *public* and *private* · *const* member functions · *class* versus *struct* · The *this* pointer

Object Relationships Delegation and forwarding · Composition and association · Managing objects dynamically with *new* and *delete*

Construction and Destruction Constructors and destructors · Default constructors · Member initialiser lists · Construction and destruction order

Value Objects Conversions for value objects · *explicit* · Default copying · Copy constructors · Member and non-member operators · Copy assignment · Copy prevention for non-value types

Exception Objects Using objects for exceptions · Safe copy assignment · Safe resource release

Templates Generic programming · Function templates · Class templates · Templates in the standard library · *auto_ptr* for exception safety · Managing long type names

Containers The standard sequence containers · The standard associative containers

Iterators Iterator categories · Container iterators · *iterator* and *const_iterator*

Algorithms Sorting, searching and copying ranges · Refactoring loops with generic algorithms

Interface-Based Programming Interface classes and pure *virtual* functions · *virtual* destructors · Implementing interfaces with *public* derivation and overridden functions · Multiple base classes

Inheritance-Based Programming Generalisation, specialisation and substitutability · Abstract classes with some implementation · *protected* · Base class construction · Exceptions

Additional Details

Duration 5 days

Setup Projection facilities for a laptop · One workstation per delegate with C++ compiler installed (configuration to be agreed) · Whiteboards and/or flip charts

Contact Kevlin Henney · kevlin@curbralan.com · Curbralan Limited · +44 117 942 2990