Advanced C++11 and C++14 (code CPPA-11)

DATASIM EDUCATION BV

Course Contents

A Foundations

Quick Review of C++98

- Function pointers
- Function overloading and virtual functions
- The categories of polymorphic behaviour
- Using (and misusing) inheritance to realise subtype polymorphism

Fundamentals of Functional Programming (FP)

- Short history of FP
- Higher-order functions
- Recursion; passing a function to itself
- Strict and non-strict (delayed) evaluation
- Pure functions and lambda functions

Functions and Data

- Function composition
- Closures
- Currying and uncurrying
- Partial function application
- Fold and continuations
- Functional Programming in C++

Overview C++ as a multi-paradigm programming language

- Universal function type (polymorphic) wrappers (std::function)
- Binders and predefined function objects (std::bind)
- Lambda functions versus binders
- A uniform function framework

Lambda Functions

- What is a lambda function?
- The closure of a lambda function
- Using lambda functions with auto
- The mutable keyword

Using Lambda Functions

- Configuring applications
- With algorithms
- As sorting criteria
- As hash function
- Lambda functions versus function objects
- A Taxonomy of Functions in C++

Function Pointers and free Functions

- Object and static member functions
- Function objects
- Lambda functions
- Events and signals (Boost signals2 library)

B Core Topics

IEEE 754

- Overview of IEEE 754
- Numerics and IEEE 754
- Rounding rules and exception handling
- Normal, subnormal and infinite numbers; NaN
- Machine precision
- Rounding and cancellation errors

Numerics in C++

- `std::numeric_limits<>`
- Directed roundings
- Floating-point decomposition functions
- Error analysis
- Comparing floating-point numbers

More on Lambda Functions

- What is a lambda function A-Z?
- Stored lambda functions
- Using lambda functions to create higher-order functions
- Lambda versus `std::bind`

Advanced Lambda Functions

- Generic lambda functions
- Generic lambda functions versus templates
- Capture modes
- Using lambda with `decltype` and `std::forward`

Applications

- Creating an algebra of higher-order functions
- Using lambda functions to configure applications
- Generalised lambda capture
- Lambda functions and software design patterns
- Advanced Features

Introduction to Type Traits
- Introduction to metaprogramming
- Defining behaviour based on type
- Type categories
- Using type traits in applications and libraries

Type Categories
- Primary (is a generic type of a given type?)
- Composite (is a type scalar, compound or object?)
- Properties (e.g. is a class abstract)
- Relationships (comparing types In some way)

Some Applications of Type Traits
- Robust numerics libraries
- Compile-time *Bridge* design pattern
- Type-independent code

C Data Structures, Libraries and STL

Review of STL Containers
- Sequence containers
- Associative containers
- Unordered containers
- Container adapters
- User-defined containers
- Hashing

Hash function and hash table
- Categories of hash function
- Creating custom hash
- Applications
- Boost and STL Heap

Heap ADT
- Variants (Fibonacci, skew, priority queue, etc.)
- Heap and computational efficiency
- Boost Heap versus STL heap
- Unordered Containers

Differences with (ordered) associative containers
- Abilities of unordered containers
- Complexity analysis
- Integration with STL and other Boost libraries
- The Bucket interface
- Tuples

Modelling n-tuples (pair is a 2-tuple)
- Using tuples as function arguments and return types
- Accessing the elements of a tuple
- Advantages and applications of tuples
- Tuple member functions

Fixed-sized Arrays `std:array<>`
- Why do we need `std:array<>` ?
- Operations and abilities
- Using arrays as C-Style arrays
- Combining arrays and tuples

D Parallel Programming

The new C++ Memory Model
- Sequential consistency
- Ordering non-atomic operations
- Relaxed consistency models
- Total order

Introduction to C++ Threads
- What is a thread?
- Creating a thread with various callable objects
- Thread function: pros and cons
- Waiting on a thread; detaching a thread
- Using lambda functions

Atomics
- Atomic types and atomic operations
- Atomic load, store, increment, decrement
- Atomic flags
- Smart pointers and thread-safe pointer interface

How Threads Cooperate, I
- Thread synchronisation
- Locks and mutex
- Exception-safe lock
- Sleep and yield

How Threads Cooperate, II
- Thread notification
- Condition variables
- Wait and notify
- Example: *Producer-Consumer* pattern

C++ Concurrency: Tasks
- Motivation
- Data Dependency graph
- Tasks versus Threads
- Concurrency versus Parallelism

C++ Tasking in Detail
- Futures and shared futures
- Promises
- Packaged tasks
- Waiting on tasks to complete

Your Trainer

Daniel J. Duffy started the company Datasim in 1987 to promote C++ as a new object-oriented language for developing applications in the roles of developer, architect and requirements analyst to help clients design and analyse software systems for Computer Aided Design (CAD), process control and hardware-software systems, logistics, holography (optical technology) and computational finance. He used a combination of top-down functional decomposition and bottom-up object-oriented programming techniques to create stable and extendible applications (for a discussion, see Duffy 2004 where we have grouped applications into domain categories). Previous to Datasim he worked on engineering applications in oil and gas and semiconductor industries using a range of numerical methods (for example, the finite element method (FEM)) on mainframe and mini-computers.

Daniel Duffy has BA (Mod), MSc and PhD degrees in pure and applied mathematics and has been active in promoting partial differential equation (PDE) and finite difference methods (FDM) for applications in computational finance. He was responsible for the introduction of the Fractional Step (Soviet Splitting) method and the Alternating Direction Explicit (ADE) method in computational finance. He is also the originator of the exponential fitting method for time-dependent partial differential equations.

He is also the originator of two very popular C++ online courses (both C++98 and C++11/14) on www.quantnet.com in cooperation with Quantnet LLC and Baruch College (CUNY), NYC. He also trains developers and designers around the world. He can be contacted dduffy@datasim.nl for queries, information and course venues, in-company course and course dates