Jon Kalb Presents

Templates and Generic Programming

Templates and Generic Programming is an introduction to C++ templates, Generic Programming, and the STL (Standard Template Library).

The class focuses on the fundamental concepts of what **Alex Stepanov**, the creator of the STL, achieved with Generic Programming rather than specific language details, but we do cover important template syntax.

We cover template concepts for function and type templates including argument deduction, two-phase compilation, dependent types, non-type template parameters, and full and partial specialization.

We cover several modern template techniques such as, CRTP, type traits, compile-time conditional code, policy classes, perfect forwarding, and how to view deduced types.

The class introduces Generic Program's goals and methods, both as a valuable programming paradigm in its own right, but also as a key to understanding the STL and how best to use it.

The class ends with a survey of the algorithms in the standard library, with a deep dive on sorting-related algorithms. Portions of this class were developed by **Dave Abrahams** and **Scott Meyers**.

Course Highlights

Participants will gain:

- An understanding template purpose and semantics.
- A working knowledge of function and type template syntax in C++
- An understanding static polymorphism and how it is implemented using CRTP.
- An introduction to type traits, the basis of C++ metaprogramming.
- An understanding of different approaches to compile-time conditional coding.
- An understanding of policy classes.
- An understanding of how to implement perfect forwarding with variadic parameters and forwarding references.
- An understanding of how to determine the types of variables whose types are deduced.
- An understanding of the goals of Generic Programming.
- An understanding of the process of developing code using the Generic Programming paradigm.
- An understanding of how to use the STL to model and solve a real-world problem.
- Familiarity with the algorithms provided in the Standard Library.

Who Should Attend

Systems designers, programmers, and technical managers involved in the design, implementation, and maintenance of libraries and applications in C++ that use templates either directly or indirectly through the Standard Library.

Participants should be familiar with the fundamental concepts of C++, but expertise is not required and no familiarity with templates is required.

Format

Lecture, question/answer, and exercises.

Length

Two full days (six to seven lecture hours).

Topic Outline

- Template Syntax and Mechanics

  - Motivation
  - Template Instantiation
  - Argument Deduction
  - Two-phase Compilation
  - Dependent Types
  - Non-Type Template Parameters
  - Overloading Function Templates
    - Function Template Partial Ordering
  - Class Templates
    - Explicit Specialization
    - Partial Specialization
    - Dependent Base
- Modern Template Techniques
  - The Template Challenge
  - CRTP – Static Polymorphism
  - Type Traits – Basic Metaprogramming
  - Compile-time Conditional Overloading
  - Policy Classes
  - Perfect Forwarding
  - Viewing Deduced Types
- Introduction to Generic Programming
  - Alex Stepanov
  - Programming = Math + Memory
  - Algorithms
  - Abstraction
  - Generic Library Design

    - Generalizing Linear Search
    - Concepts and Models
        - Valid Expressions
        - Associated Types
        - Associate Semantics
    - Fundamental Concepts
    - Iterator Concepts
        - Input Iterator
        - Output Iterator
        - Forward Iterator
        - Bidirectional Iterator
        - Random Access Iterator
- Using the Standard Template Library
    - Example Domain: Graphs
- Survey of the Standard Algorithms
    - Non-modifying and Modyfing
    - Minimum and Maximum
    - Permutation, Sorting, and Partitioning
    - Set, Heap, and Numeric

For more information on this course, contact Jon Kalb at jon@cpp.training.