**Khóa học chi tiết 2025**

**Các từ khóa và cú pháp mới trong C++ hiện đại**

**Bài 1:** nullptr và sự khác biệt với NULL

Thực hành viết một hàm xử lý con trỏ mà không sử dụng thư viện (ví dụ: kiểm tra, phân biệt nullptr).

**Bài 2:** auto, decltype và uniform initialization

Thực hành viết các hàm mẫu (template functions) dùng auto để suy diễn kiểu.

Tự xây dựng một ví dụ về khởi tạo theo danh sách, và phân tích các vấn đề “narrowing conversion”.

**Bài 3:** constexpr

Viết hàm constexpr đơn giản và sau đó tự cài đặt một lớp constexpr (ví dụ: một mảng tĩnh với constexpr size).

**Lambda Expressions & Tự Cài Đặt Functor**

**Bài 1:** Lambda Expressions trong C++11

Giới thiệu cú pháp, capture, và ứng dụng trong STL.

**Bài 2:** Tự cài đặt một lớp functor thay thế cho lambda

Học viên sẽ viết một class functor có chức năng tương tự như lambda dùng cho các thuật toán (ví dụ: functor so sánh cho sort).

**Rvalue References, Move Semantics & Perfect Forwarding**

**Bài 1:** Lý thuyết về lvalue và rvalue

Thảo luận các khái niệm cơ bản.

**Bài 2:** Move semantics – Tự cài đặt các class với move constructor và move assignment

Học viên sẽ viết một class “Resource” quản lý bộ nhớ lớn, cài đặt copy và move để thấy rõ sự khác biệt về hiệu năng.

**Bài 3:** Tự cài đặt phiên bản đơn giản của std::move và std::forward

Phân tích “static\_cast<T&&>(t)” và viết macro hoặc hàm mẫu đơn giản mô phỏng chức năng.

**Bài 4:** Perfect forwarding

Viết hàm mẫu “wrapper” chuyển tiếp tham số tới một hàm xử lý và giải thích cách giữ nguyên lvalue/rvalue của đối số.

**Tự Cài Đặt Smart Pointers**

**Bài 1:** Re-implement unique\_ptr

Tự cài đặt một lớp UniquePtr bao gồm các tính năng: không cho phép copy, hỗ trợ chuyển giao (move), reset, release.

**Bài 2:** Re-implement shared\_ptr và weak\_ptr

Tự xây dựng phiên bản đơn giản của SharedPtr với cơ chế đếm tham chiếu (atomic count nếu muốn mở rộng) và WeakPtr để phá vỡ vòng lặp tham chiếu.

So sánh với triển khai của thư viện tiêu chuẩn và thảo luận về các thách thức trong quản lý tài nguyên.

**Tự Cài Đặt Container & STL đơn giản**

**Bài 1:** Re-implement một Vector đơn giản

Viết lại lớp Vector với các thành phần cơ bản: khởi tạo, thêm phần tử (push\_back), truy xuất iterator (begin/end), resize.

So sánh hiệu năng giữa copy, move khi thêm phần tử.

**Bài 2:** Re-implement các hàm non-member begin/end và range-based for loop

Học viên sẽ viết các hàm helper đơn giản để duyệt container tự cài đặt.

**Concurrency & Đồng bộ hóa – Tự Cài Đặt Các Công Cụ Cơ Bản**

**Bài 1:** Cơ chế thread trong C++ hiện đại

Tìm hiểu về std::thread và tự cài đặt một lớp “ThreadWrapper” đơn giản (sử dụng API hệ thống nếu có thể) để khởi tạo và join thread.

**Bài 2:** Tự cài đặt một lớp Mutex và Lock Guard

Mô phỏng một lớp Mutex (giả lập bằng cách dùng các công cụ cơ bản nếu không dùng thư viện chuẩn) để hiểu cơ chế khóa.

**Bài 3:** Re-implement biến atomic đơn giản

Học viên có thể xây dựng một class Atomic<T> rất cơ bản (không cần phải xử lý memory order phức tạp) nhằm hiểu bản chất của atomic operation.

**Thiết kế mẫu – Tự Cài Đặt Singleton và Các Mẫu Thiết Kế Khác**

**Bài 1:** Re-implement Singleton Template Class

Viết lại mẫu Singleton theo kiểu template, đảm bảo thread-safe (sử dụng static local variable) và không cho phép sao chép.

**Bài 2:** Các mẫu thiết kế khác (Overview)

Tự cài đặt mẫu Adapter hoặc Observer đơn giản nhằm hiểu cách tạo ra các giao diện trừu tượng và cài đặt hành vi cụ thể.