Below is a **32-hour session plan** (16 × 2-hour sessions) built around the Udemy course **“Complete Modern C++ (C++11/14/17)”** and your requirement to start with **MSYS2 + VS Code**. The course covers the right pillars—**move semantics, lambdas, smart pointers, templates, STL, and concurrency**—and presumes learners already know another language, which aligns with your cohort. ([Udemy](https://www.udemy.com/course/beg-modern-cpp))

# Session Plan (32 hours)

**Session 1 — Tooling & Compilation Model (2 h)**  
MSYS2 (UCRT64) + VS Code verification; compile → link → run; translation units; headers vs sources; first build/debug; warnings policy (-std=c++11 -O2 -g -Wall -Wextra -pedantic).

**Session 2 — Types & Initialization (2 h)**  
Fundamental/compound types; uniform initialization; auto/decltype; nullptr; enum class; scopes and using.

**Session 3 — Functions & Lambdas (2 h)**  
Overloading, default args, pass-by-value/reference, const correctness; lambda syntax & captures; std::function (overview).

**Session 4 — Classes I: Fundamentals & Copy Control (2 h)**  
Class layout & invariants; constructors/destructor; rule of 0/3; explicit; =default, =delete; static members; streaming operator basics.

**Session 5 — Move Semantics (2 h)**  
Value categories; rvalue references; move constructor/assignment; std::move/copy elision; swap idiom; when moves help.

**Session 6 — RAII & Smart Pointers (2 h)**  
Ownership patterns; std::unique\_ptr, std::shared\_ptr, std::weak\_ptr; custom deleters; exception-safety levels.

**Session 7 — STL Containers I: Sequences (2 h)**  
std::array, std::vector, std::string, std::deque, std::list; iterator basics; iteration patterns; complexity & locality.

**Session 8 — STL Containers II: Associative/Unordered (2 h)**  
std::map/set, std::multimap/multiset, std::unordered\_map/set; custom comparator/hash; typical use cases.

**Session 9 — Algorithms I: Core Patterns (2 h)**  
Non-modifying (find, count), modifying (copy, remove), sorting (sort, stable\_sort), numeric (accumulate); erase–remove idiom; functors vs lambdas.

**Session 10 — Algorithms II: Iterators & Composition (2 h)**  
Iterator categories; transform, partition, iota; composing pipelines in C++11 style; performance notes.

**Session 11 — I/O Streams & Files (2 h)**  
iostream model; stream state & exceptions; iomanip formatting; text/binary file I/O with fstream; stringstream; locales (overview).

**Session 12 — Templates I: Fundamentals (2 h)**  
Function/class templates; type/non-type parameters; deduction; alias templates; constexpr (C++11) and static\_assert; type-traits overview.

**Session 13 — Templates II: Specialization & SFINAE (2 h)**  
Full/partial specialization; traits-style metaprogramming; std::enable\_if (intro); compile-time computations (basics).

**Session 14 — Error Handling (2 h)**  
Exceptions (throw/try/catch), standard exceptions; noexcept; exception-safety guarantees (basic/strong/nothrow); documenting error contracts.

**Session 15 — Concurrency Basics (2 h)**  
std::thread, std::mutex, lock\_guard/unique\_lock, condition\_variable; futures/std::async (overview); atomics (intro); correctness first.

**Session 16 — Integration & Review Clinic (2 h)**  
Consolidated refactoring of earlier exercises; STL/algorithm patterns review; complexity/performance checklist; Q&A.

**Notes for use with the Udemy course:**

* Select the lectures that correspond to **move semantics, lambdas, smart pointers, templates, STL, concurrency**, and foundational language topics; treat any C++14/17-specific additions as optional enrichment to stay within the C++11 scope if desired. ([Udemy](https://www.udemy.com/course/beg-modern-cpp))
* The course expects learners with **basic programming knowledge** in any language, which matches your audience. ([Udemy](https://www.udemy.com/course/beg-modern-cpp))

If you want, I can annotate each session with the exact Udemy lecture names (as pre-class or post-class viewing) once you confirm which sections you plan to include.