LEVEL 1: FOUNDATION & BASIC C++ (Beginner)

Goal: Solid understanding of C++ syntax, control flow, functions, and basic OOP.

Topics:

1. C++ introduction: history, standards (C++17/20/23)
2. Compilation process: preprocessor, compiler, linker
3. Variables, constants, data types
4. Operators: arithmetic, relational, logical, bitwise
5. Control structures: if, switch, loops (for, while, do-while)
6. Functions: declaration, definition, parameter passing (by value/ref)
7. Header files & modular source organization
8. Arrays (1D, 2D), strings (C-style & std::string)
9. Basic I/O: cin, cout
10. Recursion basics
11. Basic debugging & error messages
12. File I/O: reading/writing text files

DSA Integration:

- Simple array/string problems: sum, reverse, palindrome
- Recursion practice: factorial, Fibonacci, basic backtracking

Skills: Syntax mastery, basic problem-solving, compiling programs

---

LEVEL 2: OBJECT-ORIENTED PROGRAMMING (Intermediate)

Goal: Master OOP, modular design, and modern C++ basics.

Topics:

1. Classes & objects, constructors & destructors
2. Copy & move semantics
3. Inheritance: single, multiple, multilevel
4. Polymorphism: overloading & virtual functions
5. Abstract classes & interfaces
6. Operator overloading
7. Friend functions & classes
8. Static members & const correctness
9. Namespaces
10. Encapsulation: getters/setters
11. Exception handling: try/catch/throw
12. Function overloading & default arguments
13. Basic templates: function templates

DSA Integration:

- Object-oriented data structures: Stack, Queue, Linked List
- Implement basic algorithms using classes (e.g., Queue with arrays or linked list)

Skills: Object-oriented design, modular coding, error handling

---

LEVEL 3: MEMORY MANAGEMENT & STL (Intermediate-Advanced)

Goal: Efficient memory usage, STL mastery, and intermediate algorithm practice.

Topics:

1. Stack vs heap memory
2. Pointers & references, pointer arithmetic
3. Dynamic memory allocation: new/delete
4. Smart pointers: unique_ptr, shared_ptr, weak_ptr
5. Move semantics & std::move
6. Memory alignment and padding
7. STL containers: vector, list, deque, set, map, unordered_map
8. Iterators and iterator categories
9. STL algorithms: sort, find, transform, copy
10. Lambda functions & std::function
11. Strings & string_view
12. Ranges (C++20)
13. Variadic templates & fold expressions
14. Constexpr & compile-time computation
15. Type traits & SFINAE

DSA Integration:

- Array, string, stack, queue, list, map/set problems
- STL algorithm-based problems (sort, binary_search, lower_bound/upper_bound)
- Practice small coding challenges on LeetCode/HackerRank

Skills: Memory safety, STL usage, modern C++ features

LEVEL 4: CONCURRENCY, SYSTEMS & PERFORMANCE (Advanced)

Goal: Build high-performance, multithreaded, system-level applications.

Topics:

1. Multithreading: std::thread, std::mutex, lock_guard
2. Futures, promises, async programming
3. Thread pools & task-based parallelism
4. Atomic operations & memory models
5. Lock-free programming basics
6. File I/O streams & binary files
7. Network programming: sockets, Boost.Asio
8. Inter-process communication & signals
9. Profiling & debugging: Valgrind, perf, VTune
10. CPU cache optimization
11. SIMD/vectorization basics
12. Inline functions & branch prediction

13. Zero-cost abstractions
14. Advanced templates & concepts (C++20)

DSA & Coding Practice:

- Multithreaded algorithm implementation (parallel sum, producer-consumer)
- Graphs & trees: BFS, DFS, shortest path (STL-based)
- Complexity analysis & optimization problems

Skills: Multithreading, system programming, performance optimization

---

LEVEL 5: MODERN C++ ECOSYSTEM & DESIGN PATTERNS (Expert)

Goal: Professional software development, clean code, and scalable architecture.

Topics:

1. Build systems: CMake, Bazel
2. Package managers: vcpkg, Conan
3. Debugging: GDB, LLDB
4. Static analysis: Clang-Tidy, SonarQube
5. Unit testing: Google Test, Catch2
6. CI/CD pipelines
7. Design patterns: Factory, Singleton, Observer
8. Dependency injection
9. Event-driven architecture
10. Plugin & modular architectures
11. Microservices in C++
12. Boost, Abseil, Folly, Qt libraries
13. Cross-platform considerations

DSA & Coding Practice:

- Design pattern problems: implement observer, singleton, factory
- STL + OOP coding problems for scalable architecture
- System design coding exercises

Skills: Modern software engineering practices, code quality, scalable design

---

LEVEL 6: DOMAIN-SPECIFIC SKILLS, PROJECTS & INTERVIEW PREP (Expert-Professional)

Goal: Domain expertise, portfolio readiness, and interview mastery.

Topics:

1. Finance / HFT: low-latency programming, lock-free data structures, QuantLib
2. Game Development: engine architecture, real-time physics, graphics API integration
3. Embedded Systems: RTOS, memory-constrained coding, hardware interfacing, MISRA C++
4. Open-source contribution & code review practices
5. Legacy code modernization

6. High-performance data pipelines
7. System design in C++
8. Smart pointer implementation puzzles
9. Virtual table mechanism & template metaprogramming
10. Algorithm optimization with STL & Big O analysis
11. Portfolio projects:
    - Custom STL container
    - Multi-threaded web server
    - Game engine components
    - Trading system simulator
    - Embedded systems controller
12. Certifications: C++ Institute (CPA/CPA+), LinkedIn skill assessments
13. Interview prep:
    - Arrays, strings, linked lists, stacks/queues
    - Trees, graphs, heaps, tries
    - Dynamic programming & backtracking
    - Problem-solving patterns & LeetCode/HackerRank challenges

Skills: Domain expertise, high-level problem-solving, portfolio-ready applications, coding interview mastery

---

✅ **Summary:**

This **updated 6-level syllabus** now covers:

- **Core C++ knowledge** (syntax, OOP, modern C++)
- **Advanced C++** (memory, STL, concurrency, performance)
- **Domain-specific skills** (finance, game, embedded)
- **Coding interview & DSA practice** (arrays, strings, trees, graphs, DP)
- **Software engineering skills** (design patterns, CI/CD, testing, tools)
- **Portfolio and projects** for practical demonstration

This is **100% aligned with international C++ job interview requirements**, from entry-level to senior roles.