C++ 30-Day Learning Plan (Basic → Advanced)

Week 1 — C++ Fundamentals (Basics & Logic Building)

- Day 1: Setup, first program (Hello World), input/output.
- Day 2: Data types, variables, operators — create calculator.
- Day 3: Conditional statements (if, else if, switch).
- Day 4: Loops — pattern printing & multiplication table.
- Day 5: Functions (with return, parameters, recursion).
- Day 6: Arrays (1D, 2D), operations like sum, max, min.
- Day 7: Strings (char[], string class), palindrome check.

Week 2 — Object-Oriented Programming (OOP Concepts)

- Day 8: Classes & Objects — Student class example.
- Day 9: Constructors & Destructors — BankAccount example.
- Day 10: Access specifiers, Encapsulation, Abstraction.
- Day 11: Inheritance — Shape base class example.
- Day 12: Polymorphism — Function & Operator overloading.
- Day 13: Virtual functions & Runtime polymorphism.
- Day 14: Static members, Friend functions, this pointer — OOP Mini Project.

Week 3 — Intermediate Concepts & Data Handling

- Day 15: Pointers, pointer arithmetic, swapping using pointers.
- Day 16: Dynamic memory (new/delete), references.
- Day 17: File Handling (ofstream, ifstream, fstream).
- Day 18: Exception handling (try, catch, throw).
- Day 19: Templates (Function & Class templates).
- Day 20: Namespaces, Enumerations, Inline functions — Mini Project.

Week 4 — Data Structures & STL

- Day 21: Structures, Linked List basics.
- Day 22: Stack & Queue concepts (using arrays & STL).
- Day 23: Searching (Linear, Binary) & Sorting (Bubble, Quick).
- Day 24: Introduction to STL — vector, set, map, pair.
- Day 25: STL algorithms (sort, find, count).
- Day 26: Trees & Graphs basics, BST example.

Week 5 — Modern & Advanced C++

- Day 27: Auto keyword, Range-based loops, Lambda functions.
- Day 28: Smart pointers (unique_ptr, shared_ptr, weak_ptr).
- Day 29: Multithreading basics (std::thread, mutex).
- Day 30: Final Project — Banking System / Inventory / Student DB.

Tips for Success

- ✔■ Practice coding daily — even 30 minutes helps.
- ✔■ Revise weekly and focus on problem-solving logic.
- ✔■ Use HackerRank, LeetCode, and GeeksforGeeks for practice.
- ✔■ Read others’ C++ code to improve your style and approach.
- ✔■ Keep your code clean with comments and indentation.