## **Week 1: C++ Fundamentals**

### **Day 1:** Introduction & Setup

🔹 Learn about **C++ history & features**  
🔹 Install a **C++ compiler & IDE** (VS Code, Code::Blocks, or Dev-C++)  
🔹 Write & run your **first C++ program** (Hello, World!)

### **Day 2:** Basic Syntax & Data Types

🔹 Understand **variables & data types** (int, float, char, string, bool)  
🔹 Learn about **constants (**const**keyword)**  
🔹 Practice **simple input/output** (cin, cout)

### **Day 3:** Operators & Expressions

🔹 Learn **arithmetic, relational, logical, bitwise operators**  
🔹 Understand **precedence & associativity**  
🔹 Solve basic **math-based programs**

### **Day 4:** Control Flow - Conditional Statements

🔹 Learn **if-else & nested if-else**  
🔹 Understand **switch-case**  
🔹 Practice decision-making problems

### **Day 5:** Control Flow - Loops

🔹 Understand **for, while, do-while loops**  
🔹 Use **break & continue** statements  
🔹 Solve **pattern printing programs**

### **Day 6:** Functions & Recursion

🔹 Define **functions & return types**  
🔹 Understand **function parameters & arguments**  
🔹 Learn about **recursion** (Factorial, Fibonacci)

### **Day 7:** Review & Mini Project

✅ Revise everything learned  
✅ Build a **basic calculator**  
✅ Solve **10 coding problems** on loops & functions

## **Week 2: Intermediate C++ Concepts**

### **Day 8:** Arrays & Strings

🔹 Learn **1D & 2D arrays**  
🔹 Understand **string operations (getline, length, concatenation)**  
🔹 Solve array & string-based coding problems

### **Day 9:** Pointers & Dynamic Memory Allocation

🔹 Learn **pointers & memory addresses**  
🔹 Understand **new & delete operators**  
🔹 Practice with **arrays & pointers**

### **Day 10:** Structures & Unions

🔹 Define & use **structures**  
🔹 Learn about **unions & enumerations**  
🔹 Implement a **student record system**

### **Day 11:** File Handling

🔹 Learn **reading/writing files in C++**  
🔹 Implement a **file-based student database**

### **Day 12:** Object-Oriented Programming (OOP) - Basics

🔹 Understand **classes & objects**  
🔹 Learn about **constructors & destructors**  
🔹 Implement a **simple class-based program**

### **Day 13:** OOP - Encapsulation & Inheritance

🔹 Understand **access specifiers (private, public, protected)**  
🔹 Learn **inheritance types**  
🔹 Build an **employee management system**

### **Day 14:** Review & Mini Project

✅ Revise everything learned  
✅ Solve **10 coding problems** on OOP  
✅ Build a **bank account management system**

## **Week 3: Advanced OOP & STL**

### **Day 15:** Polymorphism & Operator Overloading

🔹 Learn **function overloading & overriding**  
🔹 Understand **virtual functions**  
🔹 Implement **operator overloading**

### **Day 16:** Templates & Exception Handling

🔹 Learn **function & class templates**  
🔹 Understand **try-catch exception handling**  
🔹 Solve **template-based problems**

### **Day 17:** Standard Template Library (STL) - Containers

🔹 Learn **vectors, lists, queues, stacks**  
🔹 Solve **STL-based coding problems**

### **Day 18:** STL - Maps & Sets

🔹 Learn **unordered\_map, set, multiset**  
🔹 Solve **map/set-based problems**

### **Day 19:** Multi-threading & Concurrency

🔹 Learn **threads & mutex**  
🔹 Implement **parallel processing in C++**

### **Day 20:** Advanced File Handling & Smart Pointers

🔹 Learn **fstream, file buffers**  
🔹 Understand **unique\_ptr, shared\_ptr**

### **Day 21:** Review & Mini Project

✅ Revise advanced topics  
✅ Solve **10 problems on STL & OOP**  
✅ Build a **task management system**

## **Week 4: Real-World Applications & Expert-Level C++**

### **Day 22:** Design Patterns in C++

🔹 Learn **Singleton, Factory, Observer patterns**  
🔹 Implement **real-world applications**

### **Day 23:** Networking in C++

🔹 Learn **socket programming basics**  
🔹 Build a **client-server chat application**

### **Day 24:** Game Development Basics

🔹 Understand **SFML/SDL for graphics**  
🔹 Build a **simple 2D game**

### **Day 25:** Memory Management & Performance Optimization

🔹 Learn **memory leaks & optimization techniques**  
🔹 Analyze code using **profiling tools**

### **Day 26:** Open-Source Contribution & Best Practices

🔹 Learn **coding standards & best practices**  
🔹 Contribute to **a C++ open-source project**

### **Day 27-28:** Final Project

✅ Build a **complex real-world application** (mini compiler, game, database system)  
✅ Optimize code & add **documentation**

## **Ongoing Practice Plan (Post-Course)**

🔹 Solve **5 coding problems daily** (LeetCode, Codeforces)  
🔹 Read **books & blogs on modern C++**  
🔹 Build **personal projects & contribute to open-source**  
🔹 Stay updated with **C++20/C++23 features**

### **📚 Books**

1. **For Beginners:**
   * Programming: Principles and Practice Using C++ – Bjarne Stroustrup
   * C++ Primer (5th Edition) – Stanley B. Lippman
2. **For Intermediate Learners:**
   * Effective C++ – Scott Meyers
   * The C++ Programming Language – Bjarne Stroustrup
3. **For Advanced Learners:**
   * More Effective C++ – Scott Meyers
   * Modern C++ Design – Andrei Alexandrescu
   * Design Patterns: Elements of Reusable Object-Oriented Software – Gamma, Helm, Johnson, Vlissides

### **🌐 Websites & Online Tutorials**

1. **Interactive & Beginner-Friendly:**
   * GeeksforGeeks C++ Tutorials
   * Cplusplus.com
   * [LearnCpp.com](https://www.learncpp.com/)
2. **Practice & Competitive Coding:**
   * [LeetCode](https://leetcode.com/) – Algorithmic problems
   * [Codeforces](https://codeforces.com/) – Competitive programming
   * HackerRank – Coding challenges
3. **Advanced Concepts & STL:**
   * cppreference.com – C++ documentation
   * [Modern C++ Features](https://github.com/AnthonyCalandra/modern-cpp-features) – C++11, C++14, C++17, C++20

### **🎓 Best Online Courses**

1. **Beginner-Friendly:**
   * Udemy: Beginning C++ Programming – From Beginner to Beyond
   * Codecademy C++ Course
2. **Intermediate & Advanced:**
   * [Coursera: C++ For C Programmers – UC Santa Cruz](https://www.coursera.org/learn/c-plus-plus-a)
   * Pluralsight: C++ Fundamentals
   * MIT OpenCourseWare – Advanced C++

### **🔧 Tools & IDEs**

✅ **Visual Studio Code** – Lightweight & customizable  
✅ **Code::Blocks** – Beginner-friendly  
✅ **CLion (JetBrains)** – Great for advanced development  
✅ **Online Compilers:** [OnlineGDB](https://www.onlinegdb.com/), JDoodle

Here are **C++ project ideas** for each stage of your learning journey:

## **🟢 Beginner-Level Projects (Weeks 1-2)**

✅ **Basic Calculator** – Perform addition, subtraction, multiplication, and division  
✅ **Number Guessing Game** – Generate a random number and let the user guess it  
✅ **Unit Converter** – Convert between temperature, weight, length, etc.  
✅ **Student Grade Calculator** – Take input for marks and calculate the grade  
✅ **To-Do List App (Console-Based)** – Add, remove, and display tasks

## **🔵 Intermediate-Level Projects (Weeks 3-4)**

✅ **Library Management System** – Store book records, issue/return books  
✅ **Bank Account Management System** – Create accounts, deposit, withdraw, and display balance  
✅ **File-Based Contact Manager** – Store and manage contacts in a file  
✅ **Snake Game (Console-Based)** – Implement basic movement using loops and arrays  
✅ **Sudoku Solver** – Solve Sudoku puzzles using backtracking

## **🟣 Advanced-Level Projects (Month 2 & Beyond)**

✅ **Chat Application (Sockets & Networking)** – Implement client-server communication  
✅ **Simple Web Crawler** – Fetch and parse HTML pages  
✅ **Multi-Threaded File Downloader** – Use threading to download multiple files  
✅ **Mini Compiler (Lexical Analyzer)** – Tokenize and analyze a programming language syntax  
✅ **2D Game with SFML/SDL** – Create a basic platformer or puzzle game

## **🟠 Expert-Level Projects (Long-Term)**

✅ **Operating System (Basic Kernel in C++)** – Implement process management, memory allocation  
✅ **AI-Powered Chess Game** – Use Minimax & Alpha-Beta Pruning for AI moves  
✅ **Blockchain-Based Voting System** – Secure voting using cryptography  
✅ **Custom Database Management System (DBMS)** – Build an SQL-like database engine  
✅ **Self-Driving Car Simulation** – Implement machine learning in C++ for obstacle detection