C++ Learning Checklist

Detailed Checklist for Mastering C++ by Section

Below is a detailed **checklist** for each phase of your C++ learning path. It's designed to help you ensure that **you know exactly what to learn** and **test your understanding** as you go along.

PHASE 1: C++ BASICS (1-2 Months)

Goal: Master C++ syntax, data types, control flow, and functions.

1. C++ Syntax & Basics

Can you **set up** a C++ environment (IDE, compiler)?

Can you **compile** a C++ program and understand **error messages**?

What is the purpose of `#include <iostream>`?

Can you explain how the `main()` **function** works and why it is necessary in every C++ program?

Do you understand the difference between **statements** and **expressions** in C++?

Can you use `std::cout` to **output** text and `std::cin` to **input** values?

Can you explain what `endl` does vs `\n`?

Can you write **multi-line comments** and **single-line comments**?

Mini-Project: *CLI Calculator*
Test yourself:

Do you know how to output results in the console correctly?

Can you handle basic arithmetic operations (addition, subtraction, etc.)?

2. Data Types, Operators, and Type Conversions

Can you **define** and **use** the primary data types: `int`, `float`, `double`, `char`, `bool`?

Can you **describe** and **use arithmetic operators** (`+`, `-`, `*`, `/`, `%`)?

Do you know **comparison operators** (`==`, `!=`, `<`, `>`, `<=`, `>=`)?

Can you **use logical operators** (, `||`, `!`)?

What is **type casting** (implicit vs explicit)?

Can you explain **type promotion** (how `int` converts to `float`, etc.)?

What is the difference between **integer division** and **floating-point division**?

Can you define **constants** with `const` and `constexpr` and explain when to use each?

Can you demonstrate **typecasting** between `int`, `float`, and `char`?

What is **integer overflow**, and how can you prevent it?

Mini-Project: *Unit Converter*
Test yourself:

Can you convert between **different units** (like km to miles, kg to pounds)?

Can you handle the **precision of floating-point numbers**?

3. Control Flow (Loops & Conditionals)

Can you **write** and **explain** `if-else` and `switch-case` statements?

How do **logical conditions** work inside `if-else`?

How do you write and use `for`, `while`, and `do-while` loops?

Can you explain the difference between `for` and `while` loops?

How do you use **break** and **continue** statements inside loops?

Can you create a **nested loop** and explain when it's appropriate to use it?

What happens if a loop has a **condition that's always true**? How do you prevent infinite loops?

Mini-Project: *Basic Budget Calculator*
Test yourself:

Can you handle looping through user input to calculate and display the **final result**?

Can you use a `switch-case` statement for selecting options in a menu?

4. Functions & Scope

What is the **syntax** for declaring and defining a function in C++?

What are **function arguments** and **return types**?

Can you explain the difference between **pass-by-value** and **pass-by-reference**?

What is **function overloading**, and how does C++ resolve overloaded functions?

Can you explain **default arguments** and how to use them in functions?

What is **recursion**, and can you give an example?

How do **local and global variables** differ, and where can they be accessed?

Can you explain **function prototypes**?

Can you use a **void function** (one that returns nothing)?

Mini-Project: *Math Function Library (factorial, power, prime check, GCD, etc.)*
Test yourself:

Can you write a function that **calculates the factorial** of a number?

Can you **pass an array** or **vector** as a function parameter?

Can you write **recursive functions** for problems like factorial or Fibonacci?

PHASE 2: INTERMEDIATE C++ (1-2 Months)

Goal: Master memory management, OOP, and file handling to prepare for more complex applications.

5. Arrays, Vectors & Strings

What are **static arrays**, and how are they different from **dynamic arrays**?

Can you declare and initialize an **array** of integers?

How do you iterate through an **array** using **pointers**?

What is the advantage of using `std::vector` over arrays?

Can you explain **array bounds checking** and **segmentation faults**?

How do you use `std::string` and manipulate strings in C++?

What are **C-style strings** (`char[]`) and how do they differ from `std::string`?

Mini-Project: *To-Do List App (store tasks in an array/vector)*
Test yourself:

Can you create a **dynamic array** that grows when needed?

Can you manipulate strings (concatenate, compare, find substrings)?

6. Pointers, Memory Management & Smart Pointers

What are **pointers**, and how are they declared?

How do you access memory addresses with **&** (address-of operator) and ***** (dereference operator)?

What is the difference between **pointer dereferencing** and **pointer arithmetic**?

How do you **allocate** and **deallocate memory** with `new` and `delete`?

What are **dangling pointers**, and how can they be avoided?

What are **smart pointers** in C++?

Can you explain the difference between `std::unique_ptr`, `std::shared_ptr`, and `std::weak_ptr`?

How does **RAII (Resource Acquisition Is Initialization)** help manage memory?

Mini-Project: *Dynamic Memory Allocator*
Test yourself:

Can you **dynamically allocate memory** for objects (arrays, structures)?

Can you use **smart pointers** to manage memory automatically?

7. Object-Oriented Programming (OOP)

What are the four principles of OOP in C++: **Encapsulation**, **Abstraction**, **Inheritance**, and **Polymorphism**?

How do you define **classes** and **objects** in C++?

What are **member functions** and **member variables**?

How does **constructor overloading** work in C++?

What is the difference between **public**, **private**, and **protected** access modifiers?

How do you implement **inheritance** and **method overriding**?

What is **virtual inheritance**, and why is it important in C++?

How does **operator overloading** work in C++?

Mini-Project: *Library Management System*
Test yourself:

Can you create **base and derived classes** for different object types (like books, authors)?

Can you implement **dynamic polymorphism** using **virtual functions**?

8. File Handling (Reading/Writing Files)

How do you open and close a file in C++ using `ifstream` and `ofstream`?

Can you **read** from and **write** to text and binary files?

How do you check if a file was **successfully opened**?

What is **file buffering**, and how does it affect reading and writing performance?

How do you handle **file errors** and **exceptions** in C++?

Mini-Project: *Expense Tracker (store data in a file)*
Test yourself:

Can you read a file and **process** its data (e.g., calculate total expenses)?

Can you append data to an **existing file**?