CPP_MASTERY

> "C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows your whole leg off."

- Author: Kintsugi-Programmer

> Disclaimer: The content presented here is a curated blend of my personal learning journey, experiences, open-source documentation, and invaluable knowledge gained from diverse sources. I do not claim sole ownership over all the material; this is a community-driven effort to learn, share, and grow together.

Table of Contents

Chapter 1: C++ Introduction

1. Series Announcement and Introduction

- Resource : Chai aur C++

In this series, **we will write a lot of C++ code**, understand it, and go in-depth. C++ is a very fun and interesting language.

By the time the series concludes, you will be experts in C++. You will enjoy writing code. Crucially, you will understand the **code flow, architecture, and how to convert thoughts into code**.

2. Instructor and Channel Background

The instructor's name is Hitesh. He has been coding and teaching for the last 12–15 years. He has taught and explained code to millions (lakhs) of students. He has worked at several companies (though he will not display FAANG logos, he has worked extensively). Students are in good hands for learning C++.

3. Course Logistics and Engagement

The videos will be long. They will include:

- Stories.
- Projects.
- Assignments and questions.
- A lot of content will be included, as always.

The first video covers the initial story, installation steps, and other miscellaneous topics. Today, the instructional team is sitting with **cold tea** (*thandi chai*).

4. Why C++? Language Features and Use Cases

C++ is already a very popular language.

- Reason for Popularity:
    - Academics
    - Platform Independence
    - Efficiency and Large Scale Applications
    - Object Oriented Programming (OOP)
    - Statically Typed
    - Speed and System Proximity
    - Abstraction Layer and Use in Other Languages
    - Inner System Knowledge
    - Pointers and Manual Memory Management

4.1. Reason for Popularity: Academics

The number one reason for C++'s popularity is **academics**. When academic curricula were designed many years ago, the structure followed the historical timeline of languages. They started with C, then C++, and then languages like Java (an object-based language popular at the time) were added. PHP is also included in some curricula. Many college students and even school students study C++.

4.2. Platform Independence

C++ is **purely platform independent**.

- The code can execute anywhere.
- Libraries and binaries are required for compilation.
- Once the code is compiled, the executable code (binaries) can be run anywhere.
- Most software seen on Windows, such as `.exe` files, are easily built using C++.

4.3. Efficiency and Large Scale Applications

- C++ is used for building large-scale applications.
- Building software in C++ is **memory efficient** than other Langs, js etc.
- It allows for efficient memory management.
- It is suitable for **high-end applications**.

4.4. Object Oriented Programming (OOP)

C++ was specifically built as an Object Oriented language.

- The object-oriented path originated from a **PhD thesis**.
- C++ is one of the first languages that properly defines object-oriented concepts.
- It provides all the base structures **openly**.
- It does not hide or abstract away many concepts, leading to strong foundational knowledge in OOP.

4.5. Statically Typed

C++ is a **statically typed** language.

- In statically typed languages, the data type of a variable is specified beforehand.
- This is analogous to filling out a form, where you know specifically where to write numbers or words.
- Knowing the data type (number, string, letter) beforehand simplifies the work.
- This leads to **fewer mistakes** and increases predictability.

4.6. Speed and System Proximity

C++ is one of the **fastest languages** because it operates very close to the system.

- *Note:* If speed is the only reason for learning C++, Assembly language is faster.
- C++ remains closer to the system but offers enough abstraction, making it comparatively easier to write code than Assembly.
- Modern APIs and libraries are often built using C++.

> Many Games , Softwares(Like Dropbox, Adobe Ps,etc.) and Even Servers Are built in C++ and are Very Efficient !!!

> C++ is completly capable to interact with MordernDBs like MongoDB, Postgres, etc., Almost Most Have Drivers in C++.

4.7. Abstraction Layer and Use in Other Languages

C++ was used in genesis of Many Many stuff. C++ is often abstracted and used to design APIs for other, higher-level languages.

Abstracted Language	C++ Role	Details
Python	**Underlying Engine**	Major Machine Learning (ML) libraries like NumPy and Pandas were originally designed in C++. Programmers preferred an easier language (Python), so an exposure layer was created to allow them to use these libraries via Python.
JavaScript (JS)	**V8 Engine**	The original JavaScript V8 engine is built in C++. JS is used because it is easier.
Mobile Applications (React Native)	**Core Functionality**	A large chunk of React Native mobile application development is built entirely in C++.

4.8. Inner System Knowledge

Learning C++ provides **inner system knowledge**.

- It helps you understand what happens to your data when it enters the memory.
- This system understanding is the primary reason for learning C++, not just speed.
- This knowledge helps in extraordinary cases, such as performing **inner optimization** or tweaking at a large-scale company.
- C++ is the language where hardware, graphics, and device drivers are available and written, allowing users to see how they are written and installed.

> During Big Projects and Systems, Having Good Background of C++ helps in tweaking, which mayn't be possibly done by other High-level frameworks !!!

4.9. Pointers and Manual Memory Management

C++ is one of the languages that allows **direct, manual memory management**.

- Other languages typically use automated Garbage Collection (where unused variables are removed automatically).
- C++ gives the capability to `manually` manage memory.
- It was the first language to introduce the concept of **Pointers**.
- Pointers provide a direct memory reference.
- This allows the developer to manipulate or read data directly in memory, bypassing the variable pathway.
- This reduces an abstraction layer, which improves system understanding and allows for better command over abstracted languages.

5. The Journey of C++: Bjarne Stroustrup

5.1. The Creator

We must acknowledge **Bjarne Stroustrup**, the creator of C++.

- He has created a great language and done extensive work.
- He has videos on YouTube and attends conferences (e.g., in Europe).
- He still provides services and his homepage is available.
- He publishes content like "Tour of C++" and is active in development.

5.2. Birth of the Language

Stroustrup was heavily influenced by the Object Oriented Programming paradigm and wanted to bring its principles to commercial software development.

1. **Initial Attempt (Simula):** During his PhD, Stroustrup used the language **Simula**. His major thesis portion focused on attaching OOP principles to Simula.
2. **Failure:** This thesis failed. When OOP principles were attached to Simula, the speed dropped dramatically, making the language almost unusable.
3. **New Idea (C Integration):** Stroustrup decided to integrate OOP principles with the existing C language(like Right now TypeScript is integrating with JavaScript). He took the code base and detached some parts of Simula.
4. **Early Language and Compiler:** Object Oriented principles were integrated with C.
    - A new compiler named **Cfront** (pronounced C-font or C-fawn, possibly French-inspired) was introduced.
    - The language was initially called **C with Classes (CP)**.
5. **Cfront Functionality:** Similar to how TypeScript is run in Node.js, Cfront would strip out the Object Oriented principles and libraries when the code executed. It would run the C code and inject OOP features where necessary.
6. **Success:** This concept was interesting because **performance did not drop**.
7. **Final Product:** After achieving this successful integration, Stroustrup designed a new compiler, packaged everything, and named it **C++**.

C++ is still under active development with frequent versions and updates. The foundation of the language is very strong, so concepts learned remain applicable in all subsequent versions.

> "There are only two kinds of languages: the ones people complain about and the ones nobody uses." - Bjarne Stroustrup

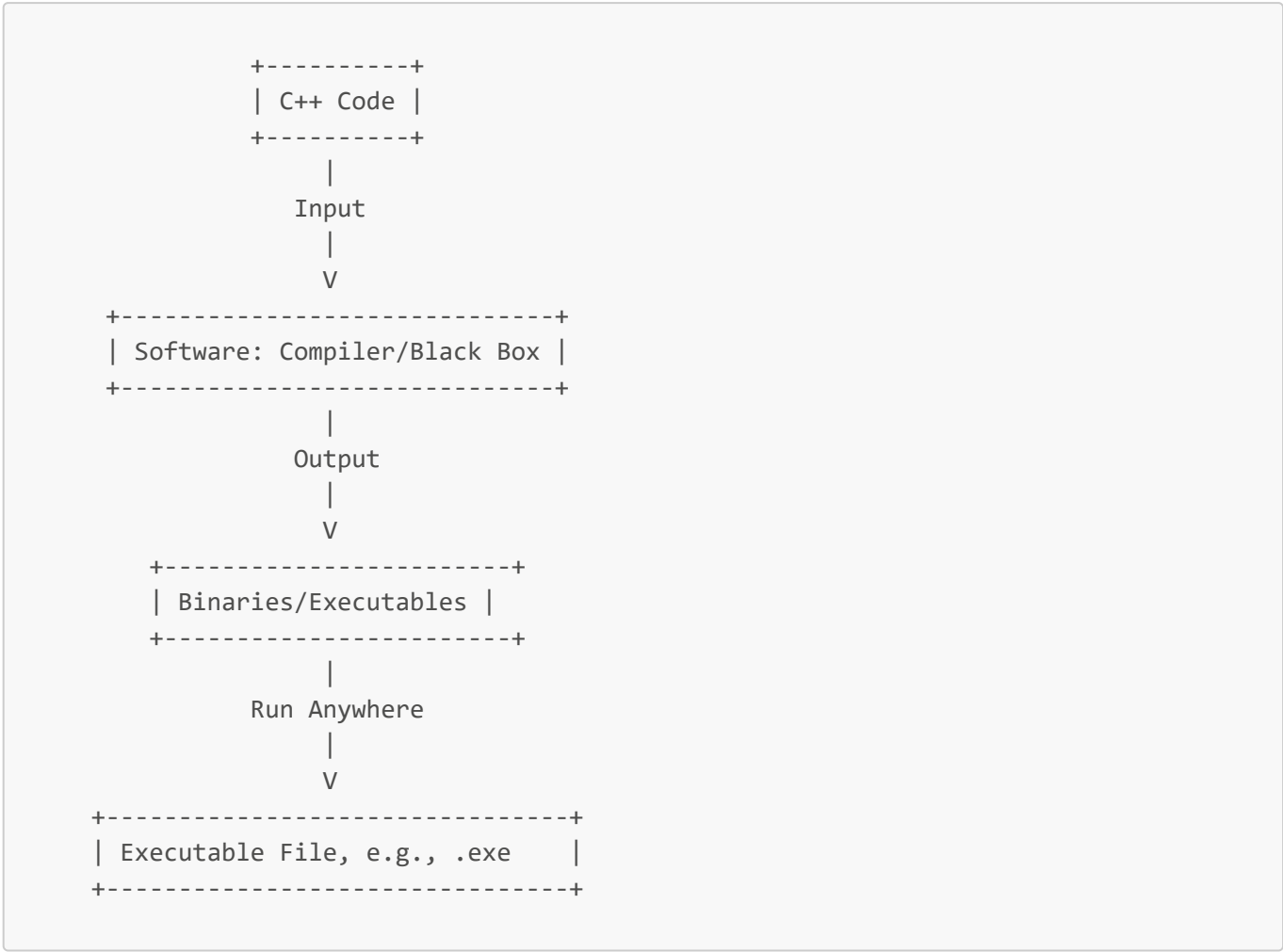
5.3. C++ Major Versions

Version	Key Features/Notes
C++11	Considered the most major version by the instructor. Introduced concepts to modernize the language, such as **Lambda functions** and **Smart Pointers**.
C++14	Introduced **Generics**.
C++17	Included minor updates.
C++20	Released.
C++23	Released. Showed influence from Rust in its error handling style.

6. C++ Environment Setup

6.1. Compiled Language Concept

C++ is a **compiled language**. Unlike scripting languages, C++ code requires a compiler to run.



The process is as follows:

1. The C++ code file is fed into the compiler software.
2. The compiler produces **binaries** (executables).
3. Examples of executables include `.exe` files (easy to understand).
4. This compiled approach ensures **portability** (binaries run anywhere) and **speed** (calculations are finalized beforehand).

6.2. Official Documentation

It is important to read documentation. The official C++ standard documentation is **paid**.

- **Website:** ISO CPP https://isocpp.org/ .
- **Purchase Location:** The official standard documentation must be purchased at a National Body Store, such as an ANSI Store.
- *Note:* The website explains why working materials are free on GitHub but the standard must be purchased through ISO.
- *Microsoft C++, C, and Assembler documentation:* https://learn.microsoft.com/en-us/cpp/?view=msvc-170, Free.

6.3. Compilers and Tools

We require C++ binaries (compilers) to write, compile, and execute our code.

Operating System	Recommended Tooling	Core Compilers	Alternatives
Mac	**Xcode** (Provides everything needed for C++ development).	**Clang** and **CMake** (these are the core compilers/tools that create the application).	-
Windows	**Visual Studio** (Community Edition is free; Professional requires payment).	**Clang** and **CMake**.	**g++ compiler**; **MinGW** (very famous among software engineers, can be downloaded from SourceForge).

Note: Turbo C is available but should not be downloaded.

6.4. Setting up VS Code (Step-by-Step)

The series will use **VS Code**.

1. **Folder Structure:** Bring up a VS Code instance. Create an empty folder for C++, e.g., `test`.
2. **File Creation:** Create a new file inside the folder, naming it `hello.cpp`. The extension for C++ files is `.cpp`.
3. **Install C++ Extension Pack:** Upon creation, VS Code will offer suggestions. Install the C++ Extension Pack. This pack automatically installs development environment components like `c++ theme`, `cmake`, and `cmake tools`.
4. **Install Code Runner:** Install the `Code Runner` extension (by Jun Han). This tool is highly recommended and provides the "Run Code" option.
5. **Write Code:** C++ code requires semicolons (`;`).

```
#include <iostream>

using namespace std; // Use this line, followed by a semicolon

int main() {
    // Parentheses and curly braces are required
    cout << "Hello Chai From Hitesh"; // Use cout (not count). The arrows (<<)
must point toward cout to send the output

    // The semicolon must be placed outside the string

} // Semicolon must be outside the string.
```

6. **Execute Code:** Go to the drop-down menu and click **"Run Code"**. The output will appear: `Hello Chai From Hitesh`.

Note: If you need to change the compiler, you can go to settings and choose the binary (e.g., C Lang, C++, g++).

6.5. Alternative Code Environments

Environment	Pros	Cons / Restrictions
Online Compilers (e.g., Online GDB)	They are good and readily available.	Run on someone else's server, leading to restrictions. You cannot read/upload/download files. Time segmentation readings will show the standard server time, which can cause confusion. Use only if installation is impossible (e.g., company laptop).
GitHub Code Spaces	Available for C++ development.	-
Replit	Available (offers 2-3 free repls). You can create a new repl using the C++ template (e.g., name it `Chai aur CPP`).	The environment is acceptable, but RAM/CPU are limited, requiring more power for faster work.

7. Next Steps

The primary goal is that your code runs, regardless of the tools you use.

A separate channel has been created on Discord for C++ help.

After finishing this initial setup:

1. You should have a working code that prints "Hello Chai From...".
2. Take a selfie with the output displayed.
3. Post the selfie on Instagram and tag the channel.

The first phase (gaining interest, learning history, and installation) is complete.

The instructor will use VS Code throughout the series. Users are free to use any editor (Vim, Sublime, Xcode, online compiler). Code files will be pushed to GitHub at the end of the series.

It is highly recommended to **code along** because typing is crucial for learning. C++ will be easily understood, and we will cover more than what is strictly required.

End-of-File

The god-stack repository, authored by Kintsugi-Programmer, is less a comprehensive resource and more an Artifact of Continuous Research and Deep Inquiry into Computer Science and Software Engineering. It serves as a transparent ledger of the author's relentless pursuit of mastery, from the foundational algorithms to modern full-stack implementation.

> Made with ♡ Kintsugi-Programmer