Introduction to C/C++

C Language

C was developed in 1972 at Bell Labs as a **systems language** to write operating systems. The goal was to create a minimal language that was:

- Easy to compile.
- Produced memory-efficient code.
- Self-contained.

C was designed to give the programmer a lot of control while supporting **cross-platform development**. It proved to be so effective that Dennis Ritchie and Ken Thompson rewrote the Unix operating system using C. This made C's popularity closely tied to the success of Unix.

Unlike other operating systems that relied on assembly language, C allowed portability across different hardware.

K&R and ANSI Standards

The book *The C Programming Language* by Brian Kernighan and Dennis Ritchie (**K&R**) set the initial specification for the language, and most compilers implemented these standards.

Later, the **American National Standards Institute (ANSI)** formed a committee to create a formal standard for C. The resulting standards include:

- **C89**: ANSI C standard, released in 1989.
- **C90**: ISO adoption of ANSI C.
- **C99**: Released in 1999, incorporating many features implemented in C++.

C++ Language

C++ was developed by Bjarne Stroustrup at Bell Labs in 1979 as an extension to C. C++ added many features and can be thought of as a **superset of C** (though not entirely true). Its most notable extension was the introduction of **Object-Oriented Programming (OOP)**.

In 1998, the ISO committee standardized C++ to ensure that C++ code was:

- Portable.
- Behaved consistently across compilers and platforms.

C++ Standards and Updates

Since its standardization, C++ has received five major updates:

- **C++11**: Introduced a large number of capabilities and is considered the baseline for modern C++.
- **C++14**.
- **C++17**.
- **C++20**.
- **C++23**.

New updates are typically released every three years. The standards are named after the year they are published (e.g., C++20 for 2020).

C/C++ Philosophy

The design philosophy of C/C++ can be summarized as:

> **Trust the programmer.**

This philosophy provides both great power and potential danger, as it assumes the programmer knows what they are doing.

What is C++ Good For?

C++ is well-suited for applications that require performance and fine-grained control, such as:

- Video games.
- Real-time systems.
- Financial applications.
- Graphical applications.
- Embedded software.
- Artificial Intelligence (AI).