# Chapter 1

## The C++ language – introduction (1 week)

- History,
- Comparison to other langages,
- C++ in industry,
- Standards and versions of C++ (C++98 • C++03 • C++11 • C++14 • C++17 • C++20),
- new language features

The C++ programming language has a rich history that dates back to the late 1970s. Here's an overview of its development:

**1. Origins (1979-1983):**

**1979:** Bjarne Stroustrup, a Danish computer scientist, started working at Bell Labs. He was interested in the C programming language but found it lacking in features for large-scale software development.

**1980:** Stroustrup started working on "C with Classes," an extension of the C language that introduced classes to support object-oriented programming (OOP) concepts.

**1983:** The first edition of "The C++ Programming Language" was published, introducing the name "C++" for the first time.

**2. Standardization (1989-1998):**

**1985:** The first commercial implementation of C++ was released by AT&T.

**1989:** The first official standard for C++ was released, known as C++98. This standardization helped in ensuring the portability of C++ programs across different platforms.

**1990s:** C++ gained popularity rapidly, especially in the field of systems programming, game development, and other performance-critical applications.

**3. Evolution and Updates (2003-2017):**

**2003:** The C++ language underwent significant updates with the release of C++03, addressing various issues and adding new features.

**2011:** A major update, C++11, was released. It introduced features like auto keyword, lambda expressions, range-based for loops, and smart pointers, making the language more modern and expressive.

**2014:** The C++14 standard brought additional improvements without introducing major breaking changes.

**2017:** C++17 was released, featuring enhancements such as filesystem library, parallel algorithms, and various language improvements.

**4. Current State (2018 Onwards):**

**2018:** C++20 was released, introducing modules, concepts, ranges, coroutines, and other features to enhance code readability, performance, and safety.

**2020s:** The C++ standard continues to evolve, with ongoing discussions about future features and improvements. Concepts like reflection and contracts are being considered for future standards.

C++ has become a versatile and powerful programming language, widely used in various domains, including system programming, game development, embedded systems, and high-performance computing. Its continued evolution is driven by the C++ community and the need to address new challenges in software development.

C++ is a powerful and versatile programming language, but it's not the only language available for developers. Here's a comparison of C++ with some other similar languages:

| | Similarities | Differences |
|---|---|---|
| C | C++ is an extension of C, inheriting its syntax and low-level features. Both languages provide low-level memory manipulation capabilities. | C++ supports object-oriented programming (OOP) with features like classes and inheritance, which C lacks.<br>C++ has a richer standard library compared to C, offering more high-level abstractions. |
| **Java** | Both languages are statically typed. They share a similar syntax for basic control structures like loops and conditionals. | Java is platform-independent and relies on a virtual machine (JVM), whereas C++ code is compiled directly to machine code.<br>C++ allows manual memory management, while Java has automatic garbage collection.<br>Java is more focused on OOP, while C++ supports both procedural and object-oriented programming. |

| | | |
|---|---|---|
| **C# (C Sharp)** | **C# shares some syntax similarities with C++ due to their common ancestry (both influenced by C). Both support OOP principles.** | **C# is primarily used in the context of the .NET framework and is not as platform-independent as C++. C# has automatic memory management (garbage collection) and does not allow direct memory manipulation like C++. C# has language features like properties and events that are not present in C++.** |
| **Python** | Both languages support high-level abstractions, making them easier to use than lower-level languages. They share some syntax similarities, like loops and conditionals. | Python is dynamically typed, while C++ is statically typed. Python is an interpreted language, whereas C++ is compiled. C++ is often chosen for performance-critical applications, while Python is known for its simplicity and readability. |

Choosing between these languages depends on various factors, including the nature of the project, performance requirements, and the developer's preferences and expertise. Each language has its strengths and weaknesses, making them suitable for different use cases.

**C++ in industry:**

C++ is widely used in various industries due to its versatility, performance, and ability to handle low-level programming tasks. Here are some common applications of C++ in industry:

**1. System Programming:**

C++ is frequently used for system-level programming, such as developing operating systems, device drivers, and firmware.

**2. Game Development:**

Many video games, including both desktop and console games, are developed using C++. Its performance and ability to access low-level hardware features make it a preferred choice for resource-intensive applications.

**3. Embedded Systems:**

C++ is commonly employed in the development of embedded systems, where efficient use of hardware resources is crucial. This includes applications in automotive, aerospace, and IoT devices.

**4. Graphics and Multimedia:**

C++ is utilized in graphics libraries and multimedia applications due to its efficiency in handling complex graphics operations and real-time processing.

**5. Finance and High-Frequency Trading:**

In the financial industry, C++ is preferred for building applications that require high-performance computing, such as algorithmic trading platforms and financial modeling software.

**6. Telecommunications:**

C++ is used in the development of telecommunications software, including networking protocols, routers, and communication systems.

**7. Scientific Computing:**

Applications in scientific research, simulations, and data analysis often leverage C++ for its computational efficiency and ability to handle complex algorithms.

**8. Database Systems:**

C++ is employed in the development of database management systems and software that require efficient data processing.

**9. Operating Systems Development:**

C++ is used in the development of components within operating systems, taking advantage of its low-level capabilities.

**10. Middleware and Libraries:**

Many middleware components and libraries, including popular ones like Boost, are written in C++. These libraries provide reusable and optimized code for various applications.

**11. Performance-Critical Applications:**

C++ is chosen for applications where performance is critical, and low-level control over hardware resources is necessary. This includes applications in aerospace, defense, and high-performance computing.

**12. Cross-Platform Development:**

C++ is suitable for cross-platform development, allowing developers to write code that can be compiled and run on different operating systems with minimal modification.

In summary, C++ is a versatile language that is well-suited for a wide range of applications, particularly those that require high performance, low-level control, and efficient use of system resources. It is a reliable choice for a variety of industrial applications.