The C++ Programming Language

Historical Development, Design Principles, and Practical Application

---

Khadija Azzouzi

January 23, 2026

University of Siegen

Introduction

- C++ is often viewed as either outdated or indispensable
- Still widely used in performance- and resource-critical domains
- Requires historical and practical context to be understood

Historical Development

- Originated in the late 1970s as *C with Classes*
- Designed as an extension of C without sacrificing performance
- Development driven by real systems programming needs

Standardization and Philosophy

- ISO standardization began in the 1990s
- Evolution preserved backward compatibility
- Core philosophy remained stable over time

Core Concepts and Design Principles

- Static type system with strong compile-time guarantees
- Support for multiple programming paradigms
- Zero-overhead abstraction principle

Applications and Use Cases

- Systems software and operating systems
- Infrastructure and foundational libraries
- Performance-critical and embedded systems

Tooling and Ecosystem

- Strong reliance on external tooling
- Build systems are central to development
- Tooling reflects flexibility and complexity

Implementation Overview

- Command-line search tool similar to `grep`
- Recursive directory traversal
- Regex-based line matching
- Optional context output

Key Implementation Aspect

```
for (auto& e : fs::recursive_directory_iterator(root)) {
    if (!e.is_regular_file()) continue;
    if (is_binary(e.path())) continue;
}
```

- Uses `std::filesystem`
- Explicit exclusion of binary files

Performance Observations

- Dominated by file traversal and I/O
- Regex matching introduces additional overhead
- Design favors clarity over maximal optimization

Conclusion

- C++ remains relevant in carefully chosen domains
- Strengths and complexity are closely linked
- Effective use requires deliberate design choices

References

Stroustrup, B. (1984). *The C++ Programming Language — Reference Manual*. AT&T Bell Laboratories.

Stroustrup, B. (1995). *Why C++ is not just an object-oriented programming language*. OOPSLA.

Stroustrup, B. (1999). *An overview of the C++ programming language*. Macmillan.

Stroustrup, B. (2013). *The C++ Programming Language* (4th ed.). Addison-Wesley.

Brunner, T., Porkoláb, Z. (2017). *Programming language history: Experiences based on the evolution of C++*. ICAI.

Nienhuis, K., Memarian, K., Sewell, P. (2016). *An operational semantics for C/C++11 concurrency*. OOPSLA.

Stroustrup, B. (2025). *21st Century C++*. Technical Report P3650R0.

Workshop: Programming Languages (2024). Programming task specification.