

Advanced C++ Programming

Introduction



Background

Goals of this Lecture

- Become literate in *modern* C++ (both reading and writing)
- Become better programmers
- Be prepared for real-world development rather than just “University programs”

I use this as a shorthand for “*a relatively small program which accomplishes a single, well-defined task, and is written in a few days or weeks*”.

Most real programming tasks aren't like that.

Prerequisites

- This lecture assumes a few prerequisites – it is called “*Advanced C++ Programming*” after all
 1. Some general programming experience
 2. Familiarity with the C programming language
 3. Familiarity with at least one object-oriented language

This includes the **pre-processor** and basics of **translation units / linking** – we will touch on some of the finer details of those during the lecture.

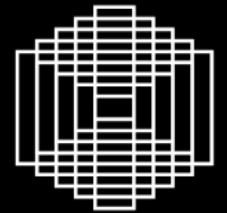
About Me



- C++ programming for >20 years now
- I was working on the Insieme C/C++ source-to-source compiler ~2011-2019
- Industry consulting experience
 - Bachmann electronic
 - Enhance Games
 - XSEED Games
 - NIS America
- Co-founder of PH3



bachmann.



enhance

Xseed
GAMES

NIS
America



Additional Materials – Internet



CORE GUIDELINES

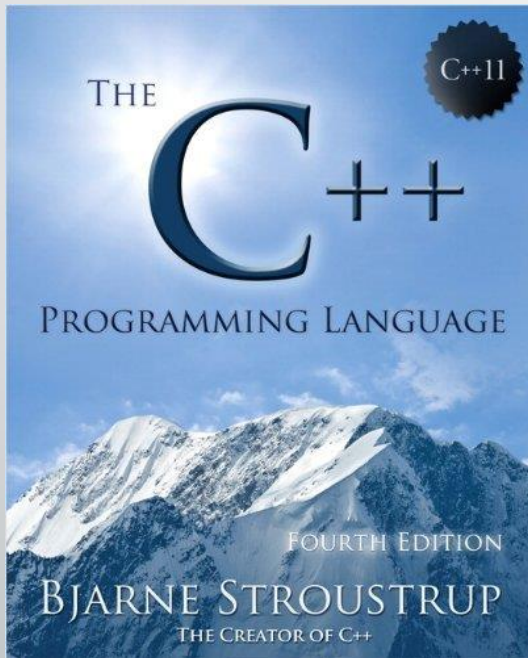
<https://github.com/isocpp/CppCoreGuidelines>

Best practices for how to write good C++

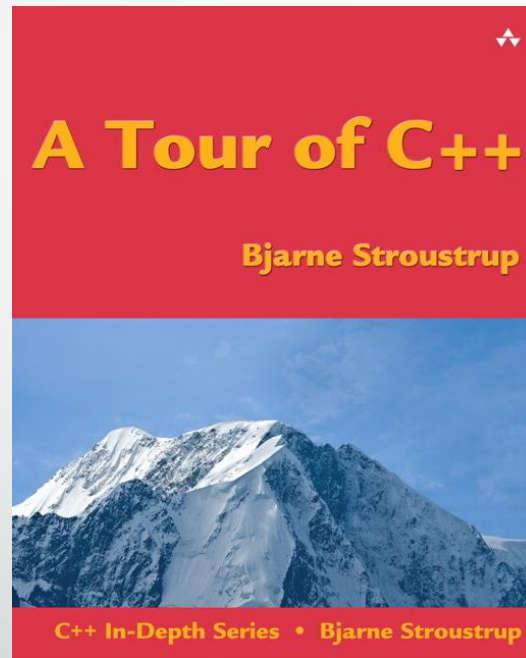
C++ Reference: <http://en.cppreference.com>

Collection: <https://github.com/rigtorp/awesome-modern-cpp>

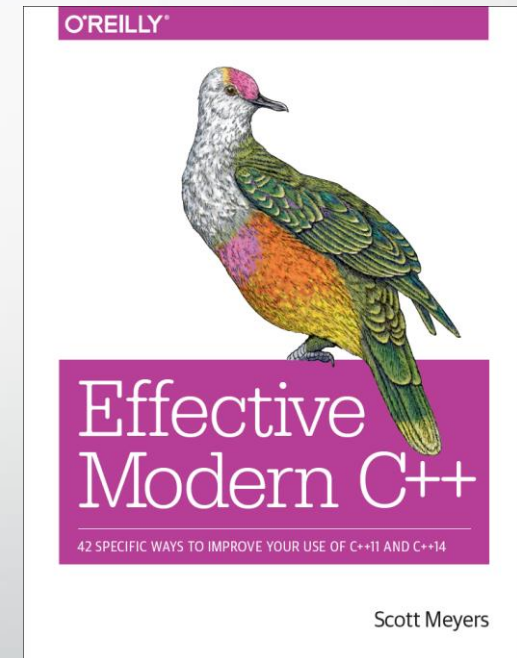
Additional Material – Books



Reference
(but really, use the internet)



(get the second edition)





Organizational

Lecture Style

- I'll try to interleave **theory** and **code** as much and as frequently as possible
- It's a more unusual style, so please *do* provide feedback

Test


- Will consist entirely of reading, understanding, and answering questions about the behaviour of code
- Is designed to be closed-book

Lab

- Combined with the lecture in this VU
- Usually structured as a lecture part first, then the lab
- We'll discuss specifics later today
- **Hands-on programming experience is fundamentally important for getting the most out of this lecture**

URLs

- https://github.com/PeterTh/uibk_cpp
 - Lecture material (Complete, but will be *updated* throughout the semester)
- https://github.com/PeterTh/uibk_cpp_2021
 - Practical/Lab material (Will be *extended* throughout the semester)



Basic C++ Facts

C++ is a Language with many Applications

- Many languages target only one or a few niches
 - Often makes them very well-suited for those, but completely inapplicable to others
- C++ is used in many distinct fields:

Desktop Applications	System-level Tools
Games	High-Performance Computing
Embedded Systems	GPUs
Drivers	Server Software

Why is C++ used in those Scenarios?

- Static Type Safety → Well-specified interfaces
- Resource Safety → Resource Acquisition is Initialization (RAII)
- Abstraction → Often with zero overhead
- Encapsulation → Classes
- Generic Programming → Templates
- Large Ecosystem → Industrial-strength tools, high-quality libraries

- Many languages offer some of these, but rarely all

What are the weaknesses of C++?

- Compile times for large applications
- Hard-to-interpret compiler error messages for highly templated code
- Legacy baggage, both in terms of overall architecture and in terms of language features
- No memory safety
- Steep learning curve; large, complicated language specification



C++ is an ISO standard

- **ISO/IEC JTC1/SC22/WG21**
- This is rather rare – many popular languages are defined by an informal process or just a single implementation
- **Advantages:** many mature implementations, industry trust, well-defined process should weed out destructive changes
- **Disadvantages:** slower adaptation, bureaucracy

Common C++ Myths

- C++ is C with a few extra bits attached
 - No! Idiomatic C++ is often very different from idiomatic C
- C++ is an object-oriented language
 - It can be, but it can also be a functional language, or an imperative language, or ...
 - C++ is a multi-paradigm language, use whatever is appropriate for the problem at hand

*Sometimes, the elegant implementation is just a function.
Not a method. Not a class. Not a framework. Just a function.*

- John Carmack



Q & A