

# Modern C++ Introduction

---

Proto Team 技术培训系列

上海合见工业软件集团有限公司  
Shanghai UniVista Industrial Software Group Co., Ltd.

# About Me

西安交通大学 计算机博士

从业二十年，主要专注于系统软件和数据挖掘 软件

主要编程语言为C++、Python、Java, 玩过一阵Ruby

业余爱好: 游泳、羽毛球

# 目标

- 在读、写C++代码的时候具有充分的信心。
- 学习现代C++的新特性和最佳实践。
  - C++11、C++14、C++17和C++20等标准引入的新功能。
  - 掌握元编程、泛型编程和函数式编程等现代C++的编程技巧。
  - 熟悉C++的STL(标准模板库)以及其他常用的现代C++库。
  - 理解并正确使用智能指针、线程和并发编程等多线程特性。
  - 学会进行性能优化和内存管理, 以提高应用程序的效率。
  - 构建高效、可靠和可维护的C++应用程序。

# C++语言的设计哲学

- Bjarne Stroustrup:

"C++ should be a **general purpose** language, not a special purpose language."

"C++ should be a **powerful** language, but not a language that is so powerful that it is difficult to learn or use."

"C++ should be a **safe** language, but not a language that is so safe that it is inefficient."

"C++ is designed to support **zero-cost abstractions**."

- Herb Sutter:

"C++ should be a **safe, reliable, and extensible** language."

"C++ should be a **modern** language that meets the needs of modern software development."

"C++ should be an **easy-to-learn** and **easy-to-use language**."

# Modern C++的核心理念

- 稳定性和向下兼容性
  - 注重稳定性和向后兼容性，以确保现有代码库的可维护性和扩展性。
- 现代语言特性的引入
  - 引入了新的语言特性，如自动类型推断、范围循环、统一的初始化语法等，以提高代码的简洁性和可读性。
- 模板元编程
  - 鼓励使用模板元编程技术，通过在编译时进行编程，实现更高效和灵活的代码。
- 类型安全和内存安全
  - 加强了类型安全性，提供更严格的类型检查和更好的错误处理机制。同时，引入智能指针等新特性，帮助管理内存和资源，降低内存泄漏和悬挂指针的风险。
- 并发编程支持
  - 提供了丰富的多线程和并发编程库，以支持并行计算和异步编程模型。
- 减少编程焦虑(Anxious Programming)
  - 鼓励对程序的正确性和性能进行关注，避免常见的编程陷阱和错误，以提高应用程序的质量。
  - 越界，指针是否会被误释放，

# 减少编程焦虑

```
int* modifyArray(int* arr, int size);
```

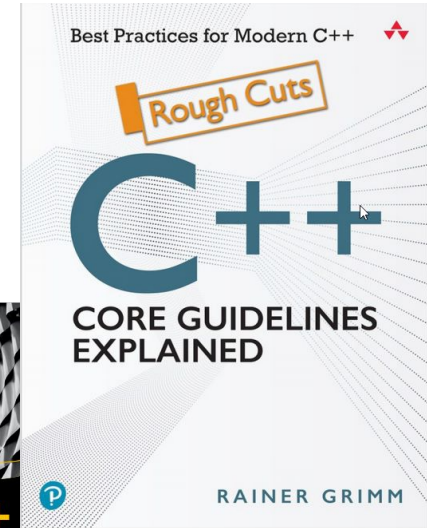
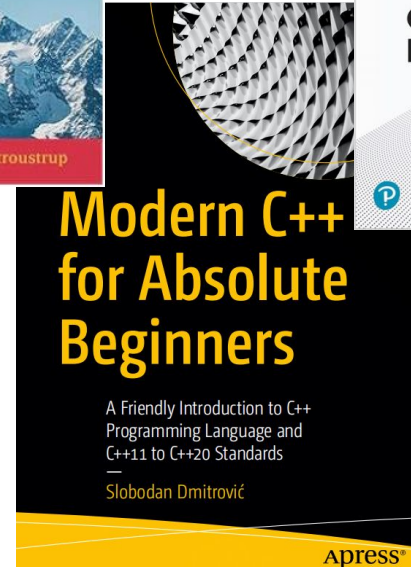
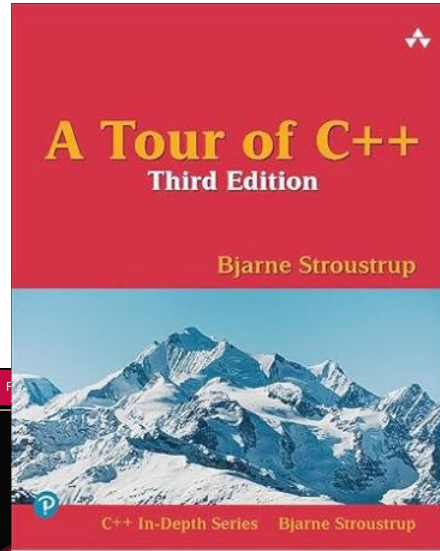
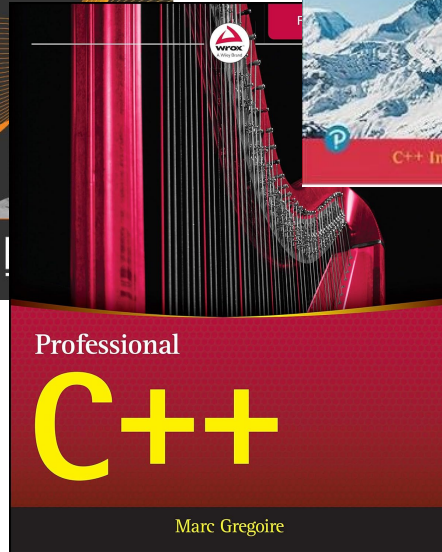
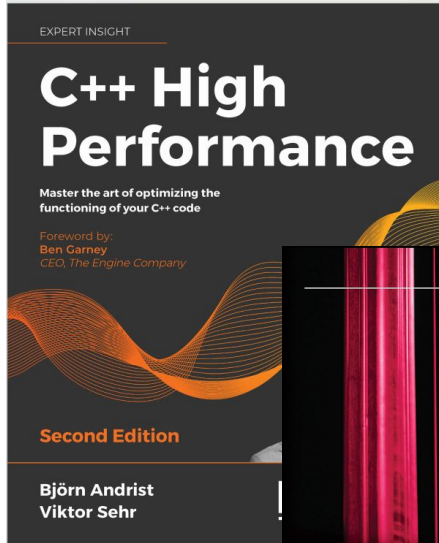
```
std::vector<int> modifyArray(const std::vector<int>& inputArray);
```

Memory Management

Safety

Side Effects

# Books



# Online Resources

- Google C++ Style Guide

Google为了确保C++代码一致性、可维护性和可读性而创建的规范。其中包括命名约定、缩进和空格、注释、类和结构体、异常处理等方面的准则，以帮助开发人员编写高质量的C++代码。这个规范还鼓励使用STL容器和算法、避免宏和全局变量，并推荐使用自动化工具来维护一致的代码格式。它被广泛应用于C++社区中，以提高代码的一致性和质量。

<https://google.github.io/styleguide/cppguide.html>

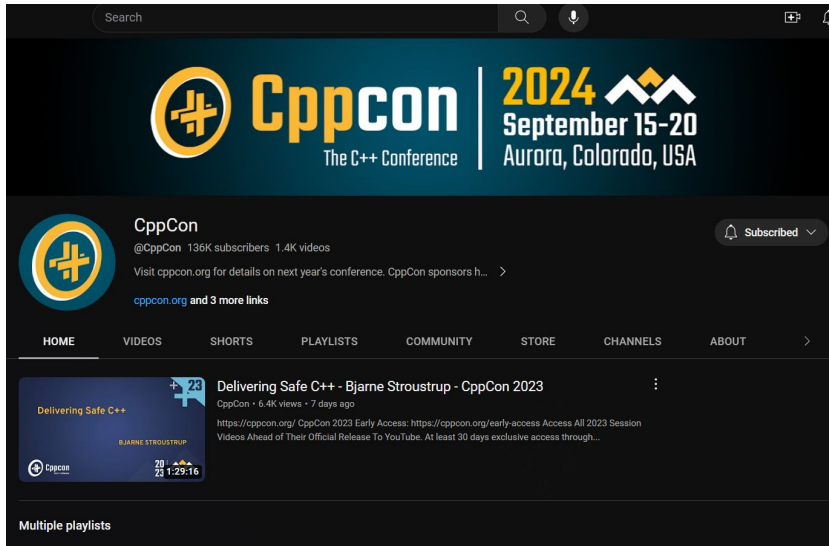
- C++ Core Guidelines

C++ Core Guidelines (C++核心指南) 是由Bjarne Stroustrup (C++的创造者之一) 和Herb Sutter等C++专家共同编写的一份广泛接受的C++编程规范和最佳实践指南。这些指南旨在帮助C++开发人员编写更安全、更可维护、更高效的C++代码。

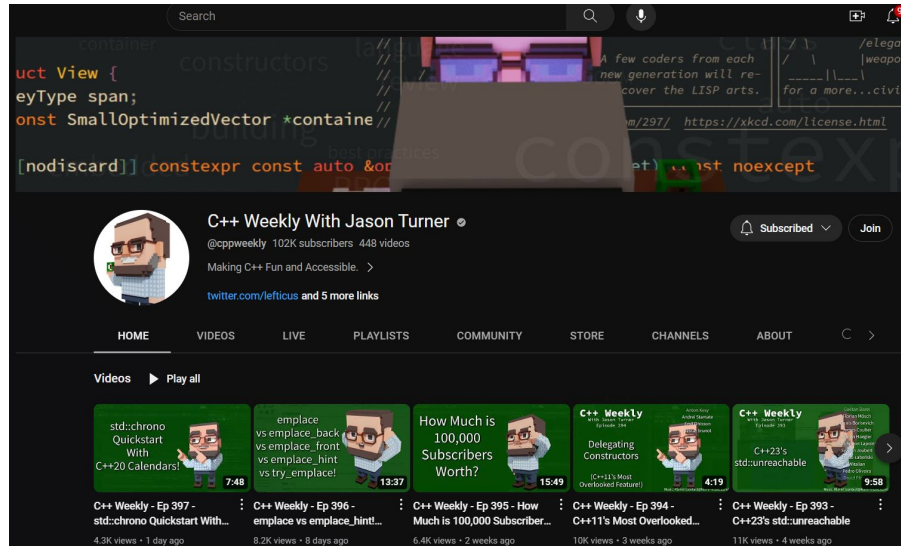
<https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines>



# Video Resources



<https://www.youtube.com/@CppCon>



<https://www.youtube.com/@cppweekly>

# Online tools – Overview

- Compiler Explorer

一个交互式的编译器资源浏览网站。支持大量的编译器和编程语言，可以实时查看正在编写的代码的汇编代码。提供不同的工具和可视化选项，界面非常可定制化。

<https://godbolt.org/>

- Cpp Insights

一种基于clang的工具，它进行源代码到源代码的转换。会去掉语法糖，看到编译器实际看到的内容。

<https://cppinsights.io/>

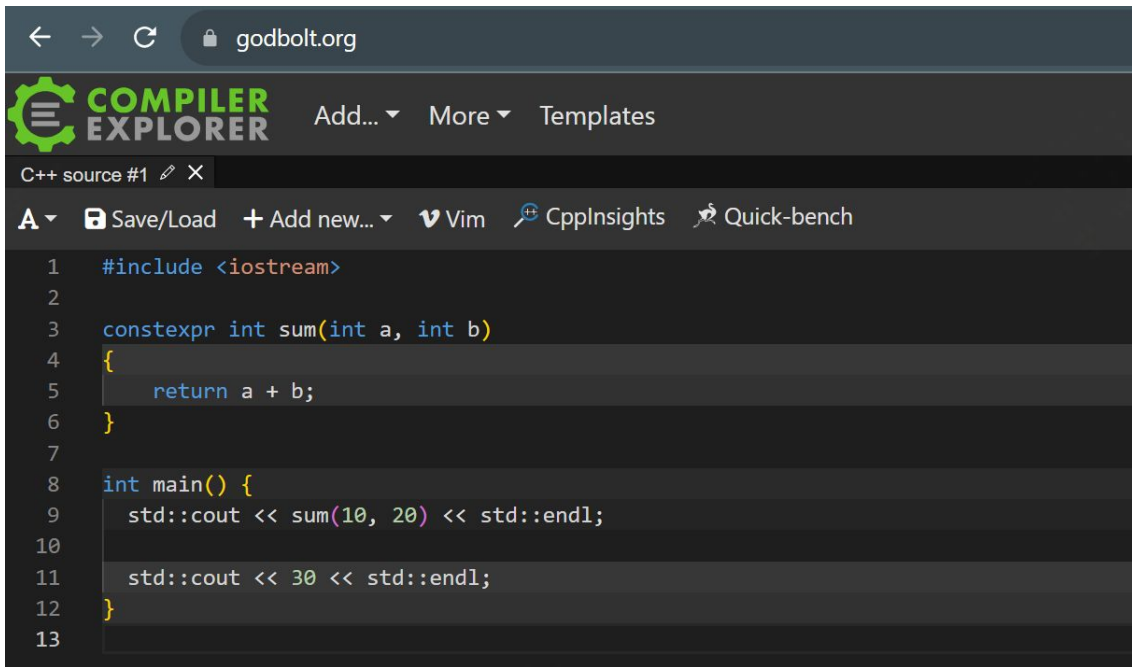
- Quick C++ Benchmark

一个在线基准测试工具，旨在快速简单地比较两个或多个代码片段的性能。它支持不同的编译器(有许多版本可用)，C++从11到20，支持不同的优化级别和其他自定义设置。

<https://quick-bench.com/>

Reference: <https://thamara.dev/posts/online-tools-for-cpp-developers/>

# Godbolt



The screenshot shows the Godbolt Compiler Explorer web application. The browser address bar displays 'godbolt.org'. The application header includes the 'COMPILER EXPLORER' logo and navigation links: 'Add...', 'More', and 'Templates'. Below the header, a tab labeled 'C++ source #1' is active. The main editor area contains the following C++ code:

```
1  #include <iostream>
2
3  constexpr int sum(int a, int b)
4  {
5      return a + b;
6  }
7
8  int main() {
9      std::cout << sum(10, 20) << std::endl;
10
11     std::cout << 30 << std::endl;
12 }
13
```

The interface also features a toolbar with icons for 'Save/Load', 'Add new...', 'Vim', 'CppInsights', and 'Quick-bench'.

第9行和第11行的代码在编译后有区别么？

<https://godbolt.org/z/PKfEfM9j4>

# Quick Bench

智能指针的性能如何？

<https://quick-bench.com/q/5RhrU0667kWfEHkouRx4wRKLI1Q>

<https://gist.github.com/liaozhigang/5b3bda0360027ebac16a7d191e44f66d>

# Cplusplus.io

```
1  #include <iostream>
2  #include <string>
3
4  // Using auto for generic summation
5  auto genericSum(auto a, auto b) {
6      return a + b;
7  }
8
9  // Using specific types for integer summation
10 int addIntegers(int a, int b) {
11     return a + b;
12 }
13
14 // Using specific types for string concatenation
15 std::string concatenateStrings(const std::string& s1, const std::string& s2) {
16     return s1 + s2;
17 }
18
19 int main() {
20     int result1 = genericSum(10, 20);
21     int result2 = addIntegers(11, 22);
22
23     auto result3 = genericSum(std::string{"hello"}, std::string{"world"});
24     auto result4 = concatenateStrings("hello", "world");
25
26     std::cout << result1 << result2 << result3 << result4 << std::endl;
27 }
```

C++11:

- Braced array initialization
- Braced (uniform) Initialization
- auto
- range based for loop
- Lambda with static invoker
- Implicit conversions
- auto, uniform initialization and the equal sign
- Recursive template

C++14:

- Braced return value and decltype(auto)
- Generic lambda

C++17:

- Structured bindings

<https://godbolt.org/z/WMz7o16zM>

<https://gist.github.com/liaozhigang/c90789df846d27f768f573e11a88631c>

# 本讲座与实际工作的关系

- 本讲座的目的
  - 提高大家对C++的理解
  - 提高大家的C++能力
- 适用性 – 有些技术并不适用于我们的工作
  - 对性能的极致要求
  - 团队习惯
  - 使用的类库和框架
- 建议
  - 在学习本讲座内容时, 要结合实际工作进行思考
  - 在实际工作中, 可以根据需要来灵活运用所学到的技术

# Topics

Smart Pointers

Template & Type trait

Concept

Constexpr & Consteval

Auto & Decltype

Range & View

Coroutine & Asynchronous

STL & Algorithms

Concurrency

Const correctness

Move Semantics & RValue

Function Object & Lambda

Utilities

optional, variant, pair, tuple, any, structured bindings...

Module

RAII & Value Semantics

Casting

Misc

string\_view, fmt, span, initializer, deleted & default functions, [nodiscard], [likely] ...

# THANKS



上海合见工业软件集团有限公司  
Shanghai UniVista Industrial Software Group Co., Ltd.