



EMBRACING MODERN C++ SAFELY



JOHN LAKOS | VITTORIO ROMEO | ROSTISLAV KHLEBNIKOV | ALISDAIR MEREDITH

Embracing Modern C++ *Safely*

Embracing Modern C++ Safely

Table of Contents

Cover

Half Title

Title Page

Copyright Page

Contents

Foreword

Foreword

Acknowledgments

About the Authors

Chapter 0 Introduction

- What Makes This Book Different

- Scope for the First Edition

- The EMC++S Guiding Principles

- What Do We Mean by Safely?

- A Safe Feature

- A Conditionally Safe Feature

- An Unsafe Feature

- Modern C++ Feature Catalog

- How to Use This Book

Chapter 1 Safe Features

- 1.1 C++11

 - Attribute Syntax Generalized Attribute Support

Table of Contents

Consecutive >s Consecutive Right-Angle Brackets
decltype Operator for Extracting Expression Types
Defaulted Functions Using = default for Special Member Functions
Delegating Ctors Constructors Calling Other Constructors
Deleted Functions Using = delete for Arbitrary Functions
explicit Operators Explicit Conversion Operators
Function static '11 Thread-Safe Function-Scope static Variables
Local Types '11 Local/Unnamed Types as Template Arguments
long long The long long (64 bits) Integral Type
noreturn The [[noreturn]] Attribute
nullptr The Null-Pointer-Literal Keyword
override The override Member-Function Specifier
Raw String Literals Syntax for Unprocessed String Contents
static_assert Compile-Time Assertions
Trailing Return Trailing Function Return Types
Unicode Literals Unicode String Literals
using Aliases Type/Template Aliases (Extended typedef)

1.2 C++14

Aggregate Init '14 Aggregates Having Default Member Initializers
Binary Literals Binary Literals: The 0b Prefix
deprecated The [[deprecated]] Attribute
Digit Separators The Digit Separator (')
Variable Templates Templated Variable Declarations/Definitions

Chapter 2 Conditionally Safe Features

2.1 C++11

alignas The alignas Specifier
alignof The alignof Operator
auto Variables Variables of Automatically Deduced Type
Braced Init Braced-Initialization Syntax: {}

Table of Contents

constexpr Functions Compile-Time Invocable Functions
constexpr Variables Compile-Time Accessible Variables
Default Member Init Default class/union Member Initializers
enum class Strongly Typed, Scoped Enumerations
extern template Explicit-Instantiation Declarations
Forwarding References Forwarding References (T&&)
Generalized PODs '11 Trivial and Standard-Layout Types
Inheriting Constructors Inheriting Base-Class Constructors
initializer_list List Initialization: std::initializer_list<T>
Lambdas Anonymous Function Objects (Closures
noexcept Operator Asking if an Expression Cannot throw
Opaque enums Opaque Enumeration Declarations
Range for Range-Based for Loops
Rvalue References Move Semantics and Rvalue References (&&)
Underlying Type '11 Explicit Enumeration Underlying Type
User-Defined Literals User-Defined Literal Operators
Variadic Templates Variable-Argument-Count Templates

2.2 C++14

constexpr Functions '14 Relaxed Restrictions on constexpr Functions
Generic Lambdas Lambdas Having a Templated Call Operator
Lambda Init-Captures Lambda-Capture Expressions

Chapter 3 Unsafe Features

3.1 C++11

carries_dependency The [[carries_dependency]] Attribute
final Prohibiting Overriding and Derivation
friend '11 Extended friend Declarations
inline namespace Transparently Nested Namespaces
noexcept Specifier The Nonthrowing-Function Specifier
Ref-Qualifiers Reference-Qualified Member Functions

Table of Contents

union '11 Unions Having Non-Trivial Members

3.2 C++14

auto Return Function (auto) Return-Type Deduction

decltype(auto) Deducing Types Using decltype Semantics

Afterword: Looking Back and Looking Forward

Glossary

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

Table of Contents

W

X

Y

Z

Bibliography

Index