

Updated for C++ 20



Discovering Modern C++

*An Intensive Course for Scientists, Engineers,
and Programmers*

Peter Gottschling

SECOND EDITION



C++ In-Depth Series Bjarne Stroustrup

Discovering Modern C++

Second Edition

Discovering Modern C++

Table of Contents

Cover

Half Title

Title Page

Copyright Page

Contents

Preface

- Reasons to Learn C ++

- Reasons to Read This Book

- The Beauty and the Beast

- Languages in Science and Engineering

- Typographical Conventions

Acknowledgments

About the Author

Chapter 1 C++ Basics

- 1.1 Our First Program

- 1.2 Variables

 - 1.2.1 Intrinsic Types

 - 1.2.2 Characters and Strings

 - 1.2.3 Declaring Variables

 - 1.2.4 Constants

 - 1.2.5 Literals

 - 1.2.6 Non-narrowing Initialization

 - 1.2.7 Scopes

Table of Contents

1.3 Operators

1.3.1 Arithmetic Operators

1.3.2 Boolean Operators

1.3.3 Bitwise Operators

1.3.4 Assignment

1.3.5 Program Flow

1.3.6 Memory Handling

1.3.7 Access Operators

1.3.8 Type Handling

1.3.9 Error Handling

1.3.10 Overloading

1.3.12 Avoid Side Effects!

1.3.11 Operator Precedence

1.4 Expressions and Statements

1.4.1 Expressions

1.4.2 Statements

1.4.3 Branching

1.4.4 Loops

1.4.5 goto

1.5 Functions

1.5.1 Arguments

1.5.2 Returning Results

1.5.3 Inlining

1.5.4 Overloading

1.5.5 main Function

1.6 Error Handling

1.6.1 Assertions

1.6.2 Exceptions

1.6.3 Static Assertions

Table of Contents

1.7 I/O

- 1.7.1 Standard Output
- 1.7.2 Standard Input
- 1.7.3 Input/Output with Files
- 1.7.4 Generic Stream Concept
- 1.7.5 Formatting
- 1.7.6 New Formatting
- 1.7.7 Dealing with I/O Errors
- 1.7.8 Filesystem

1.8 Arrays, Pointers, and References

- 1.8.1 Arrays
- 1.8.2 Pointers
- 1.8.3 Smart Pointers
- 1.8.4 References
- 1.8.5 Comparison between Pointers and References
- 1.8.6 Do Not Refer to Outdated Data!
- 1.8.7 Containers for Arrays

1.9 Structuring Software Projects

- 1.9.1 Comments
- 1.9.2 Preprocessor Directives

1.10 Exercises

- 1.10.1 Narrowing
- 1.10.2 Literals
- 1.10.3 Operators
- 1.10.4 Branching
- 1.10.5 Loops
- 1.10.6 I/O
- 1.10.7 Arrays and Pointers
- 1.10.8 Functions

Table of Contents

Chapter 2 Classes

2.1 Program for Universal Meaning, Not Technical Details

2.2 Members

2.2.1 Member Variables

2.2.2 Accessibility

2.2.3 Access Operators

2.2.4 The Static Declarator for Classes

2.2.5 Member Functions

2.3 Setting Values: Constructors and Assignments

2.3.1 Constructors

2.3.2 Assignment

2.3.3 Initializer Lists

2.3.4 Uniform Initialization

2.3.5 Move Semantics

2.3.6 Construct Objects from Literals

2.4 Destructors

2.4.1 Implementation Rules

2.4.2 Dealing with Resources Properly

2.5 Method Generation Summary

2.6 Accessing Member Variables

2.6.1 Access Functions

2.6.2 Subscript Operator

2.6.3 Constant Member Functions

2.6.4 Reference-Qualified Members

2.7 Operator Overloading Design

2.7.1 Be Consistent!

2.7.2 Respect the Priority

2.7.3 Member or Free Function

2.7.4 Overloading Equality

Table of Contents

- 2.7.5 Overloading a Spaceship
- 2.7.6 Explore the Type System in Overloading

2.8 Exercises

- 2.8.1 Polynomial
- 2.8.2 Rational
- 2.8.3 Move Assignment
- 2.8.4_INITIALIZER List
- 2.8.5 Resource Rescue

Chapter 3 Generic Programming

3.1 Function Templates

- 3.1.1 Instantiation
- 3.1.2 Parameter Type Deduction
- 3.1.3 Dealing with Errors in Templates
- 3.1.4 Mixing Types
- 3.1.5 Uniform Initialization
- 3.1.6 Automatic Return Type
- 3.1.7 Terse Template Parameters

3.2 Namespaces and Function Lookup

- 3.2.1 Namespaces
- 3.2.2 Argument-Dependent Lookup
- 3.2.3 Namespace Qualification or ADL

3.3 Class Templates

- 3.3.1 A Container Example
- 3.3.2 Designing Uniform Class and Function Interfaces

3.4 Type Deduction and Definition

- 3.4.1 Automatic Variable Type
- 3.4.2 Type of an Expression
- 3.4.3 decltype(auto)
- 3.4.4 Deduced Class Template Parameters

Table of Contents

3.4.5 Deducing Multiple Types

3.4.6 Defining Types

3.5 Template Specialization

3.5.1 Specializing a Class for One Type

3.5.2 Specializing and Overloading Functions

3.5.3 Partial Specialization of Classes

3.5.4 Partially Specializing Functions

3.5.5 Structured Bindings with User Types

3.5.6 User-Defined Formatting

3.6 Non-Type Parameters for Templates

3.6.1 Fixed-Size Containers

3.6.2 Deducing Non-Type Parameters

3.7 Functors

3.7.1 Function-Like Parameters

3.7.2 Composing Functors

3.7.3 Recursion

3.7.4 Generic Reduction

3.8 Lambda

3.8.1 Capture

3.8.2 Generic Lambdas

3.9 Variable Templates

3.10 Programming with Concept(s)

3.10.1 Defining Concepts

3.10.2 Dispatching by Concepts

3.10.3 Concepts in Classes

3.10.4 Concept Design

3.11 Variadic Templates

3.11.1 Recursive Functions

3.11.2 Direct Expansion

Table of Contents

3.11.3 Index Sequences

3.11.4 Folding

3.11.5 Type Generators

3.11.6 Growing Tests

3.12 Exercises

3.12.1 String Representation

3.12.2 String Representation of Tuples

3.12.3 Generic Stack

3.12.4 Rationals with Type Parameter

3.12.5 Iterator of a Vector

3.12.6 Odd Iterator

3.12.7 Odd Range

3.12.8 Stack of bool

3.12.9 Stack with Custom Size

3.12.10 Trapezoid Rule

3.12.11 Partial Specialization with a static Function

3.12.12 Functor

3.12.13 Lambda

3.12.14 Implement make_unique

Chapter 4 Libraries

4.1 Standard Template Library

4.1.1 Introductory Example

4.1.2 Iterators

4.1.3 Containers

4.1.4 Algorithms

4.1.5 Ranges

4.1.6 Parallel Computation

4.2 Numerics

4.2.1 Complex Numbers

Table of Contents

4.2.2 Random Number Generators

4.2.3 Mathematical Special Functions

4.2.4 Mathematical Constants

4.3 Meta-programming

4.3.1 Limits

4.3.2 Type Traits

4.4 Utilities

4.4.1 optional

4.4.2 tuple

4.4.3 variant

4.4.4 any

4.4.5 string_view

4.4.6 span

4.4.7 function

4.4.8 Reference Wrapper

4.5 The Time Is Now

4.6 Concurrency

4.6.1 Terminology

4.6.2 Overview

4.6.3 thread

4.6.4 Talking Back to the Caller

4.6.5 Asynchronous Calls

4.6.6 Asynchronous Solver

4.6.7 Variadic Mutex Lock

4.6.8 Coroutines

4.6.9 Other New Concurrency Features

4.7 Scientific Libraries Beyond the Standard

4.7.1 Alternative Arithmetic

4.7.2 Interval Arithmetic

Table of Contents

- 4.7.3 Linear Algebra
- 4.7.4 Ordinary Differential Equations
- 4.7.5 Partial Differential Equations
- 4.7.6 Graph Algorithms

4.8 Exercises

- 4.8.1 Sorting by Magnitude
- 4.8.2 Finding with a Lambda as Predicate
- 4.8.3 STL Container
- 4.8.4 Complex Numbers
- 4.8.5 Parallel Vector Addition
- 4.8.6 Refactor Parallel Addition

Chapter 5 Meta-Programming

5.1 Let the Compiler Compute

- 5.1.1 Compile-Time Functions
- 5.1.2 Extended Compile-Time Functions
- 5.1.3 Primeness
- 5.1.4 How Constant Are Our Constants?
- 5.1.5 Compile-Time Lambdas

5.2 Providing and Using Type Information

- 5.2.1 Type Traits
- 5.2.2 Conditional Exception Handling
- 5.2.3 A const-Clean View Example
- 5.2.4 Parameterized Rational Numbers
- 5.2.5 Domain-Specific Type Properties
- 5.2.6 `enable_if`
- 5.2.7 Variadic Templates Revised

5.3 Expression Templates

- 5.3.1 Simple Operator Implementation
- 5.3.2 An Expression Template Class

Table of Contents

5.3.3 Generic Expression Templates

5.3.4 Copy Before Its Stale

5.4 Meta-Tuning: Write Your Own Compiler Optimization

5.4.1 Classical Fixed-Size Unrolling

5.4.3 Dynamic Unrolling: Warm-up

5.4.2 Nested Unrolling

5.4.4 Unrolling Vector Expressions

5.4.5 Tuning an Expression Template

5.4.6 Tuning Reduction Operations

5.4.7 Tuning Nested Loops

5.4.8 Tuning Summary

5.5 Optimizing with Semantic Concepts

5.5.1 Semantic Tuning Requirements

5.5.2 Semantic Concept Hierarchy

5.6 Turing Completeness

5.7 Exercises

5.7.1 Type Traits

5.7.2 Fibonacci Sequence

5.7.3 Meta-Program for Greatest Common Divisor

5.7.4 Rational Numbers with Mixed Types

5.7.5 Vector Expression Template

5.7.6 Meta-List

Chapter 6 Object-Oriented Programming

6.1 Basic Principles

6.1.1 Base and Derived Classes

6.1.2 Inheriting Constructors

6.1.3 Virtual Functions and Polymorphic Classes

6.1.4 Functors via Inheritance

6.1.5 Derived Exception Classes

Table of Contents

6.2 Removing Redundancy

6.3 Multiple Inheritance

6.3.1 Multiple Parents

6.3.2 Common Grandparents

6.4 Dynamic Selection by Sub-typing

6.5 Conversion

6.5.1 Casting between Base and Derived Classes

6.5.2 Const-Cast

6.5.3 Reinterpretation Cast

6.5.4 Function-Style Conversion

6.5.5 Implicit Conversions

6.6 Advanced Techniques

6.6.1 CRTP

6.6.2 Type Traits with Overloading

6.7 Exercises

6.7.1 Non-redundant Diamond Shape

6.7.2 Inheritance Vector Class

6.7.3 Refactor Exceptions in Vector

6.7.4 Test for Thrown Exception

6.7.5 Clone Function

Chapter 7 Scientific Projects

7.1 Implementation of ODE Solvers

7.1.1 Ordinary Differential Equations

7.1.2 Runge-Kutta Algorithms

7.1.3 Generic Implementation

7.1.4 Outlook

7.2 Creating Projects

7.2.1 Build Process

7.2.2 Build Tools

Table of Contents

7.2.3 Separate Compilation

7.3 Modules

7.4 Some Final Words

Appendix A Clumsy Stuff

A.1 More Good and Bad Scientific Software

A.2 Basics in Detail

A.2.1 static Variables

A.2.2 More about if

A.2.3 Duffs Device

A.2.4 Program Calls

A.2.5 Assertion or Exception?

A.2.6 Binary I/O

A.2.7 C-Style I/O

A.2.8 Garbage Collection

A.2.9 Trouble with Macros

A.3 Real-World Example: Matrix Inversion

A.4 Class Details

A.4.1 Pointer to Member

A.4.2 More Initialization Examples

A.4.3 Accessing Multi-Dimensional Data Structures

A.5 Method Generation

A.5.1 Automatic Generation

A.5.2 Controlling the Generation

A.5.3 Generation Rules

A.5.4 Pitfalls and Design Guides

A.6 Template Details

A.6.1 Uniform Initialization

A.6.2 Which Function Is Called?

A.6.3 Specializing for Specific Hardware

Table of Contents

A.6.4 Variadic Binary I/O

A.7 More on Libraries

A.7.1 Using `std::vector` in C++03

A.7.2 `variant`

A.8 Dynamic Selection in Old Style

A.9 More about Meta-Programming

A.9.1 First Meta-Program in History

A.9.2 Meta-Functions

A.9.3 Backward-Compatible Static Assertion

A.9.4 Anonymous Type Parameters

A.10 Linking to C Code

Appendix B Programming Tools

B.1 g++

B.2 Debugging

B.2.1 Text-Based Debugger

B.2.2 Debugging with Graphical Interface: DDD

B.3 Memory Analysis

B.4 gnuplot

B.5 Unix, Linux, and Mac OS

Appendix C Language Definitions

C.1 Value Categories

C.2 Operator Overview

C.3 Conversion Rules

C.3.1 Promotion

C.3.2 Other Conversions

C.3.3 Usual Arithmetic Conversions

C.3.4 Narrowing

Bibliography

Table of Contents

Subject Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X