



EXTENDED

Collections and Iterators

STL

Volume 1

MATTHEW WILSON

Praise for *Extended STL*, Volume 1

“Wilson’s menu of STL treatments will no doubt be good eating for generic programming adherents, ardent C programmers just now taking on STL and C++, Java programmers taking a second look at C++, and authors of libraries targeting multiple platforms and languages. Bon appetit!”

—George Frazier, Cadence Design Systems, Inc.

“A thorough treatment of the details and caveats of STL extension.”

—Pablo Aguilar, C++ Software Engineer

“This book is not just about extending STL, it’s also about extending my thinking in C++.”

—Serge Krynine, C++ Software Engineer, RailCorp Australia

“You might not agree 100% with everything Wilson has to say, but as a whole his book is the most valuable, in-depth study of practical STL-like programming.”

—Thorsten Ottosen, M.C.S., Boost Contributor

“Wilson is a master lion tamer, persuading multifarious third-party library beasts to jump through STL hoops. He carefully guides the reader through the design considerations, pointing out the pitfalls and making sure you don’t get your head bitten off.”

—Adi Shavit, Chief Software Architect, EyeTech Co. Ltd

“Wilson’s book provides more than enough information to change the angst/uncertainty level of extending STL from ‘daunting’ to ‘doable.’”

—Garth Lancaster, EDI/Automation Manager, Business Systems Group, MBF Australia

“This book will open up your eyes and uncover just how powerful STL’s abstractions really are.”

—Nevin “:-)” Liber, 19-year veteran of C++

“In the canon of C++ there are very few books that extend the craft. Wilson’s work consistently pushes the limits, showing what can and cannot be done, and the tradeoffs involved.”

—John O’Halloran, Head of Software Development, Mediaproxy

“Essential concepts and practices to take the working programmer beyond the standard library.”

—Greg Peet

“*Extended STL* is not just a book about adapting the STL to fit in with your everyday work, it’s also an odyssey through software design and concepts, C++ power techniques, and the perils of real-world software development—in other words, it’s a Matthew Wilson book. If you’re serious about C++, I think you should read it.”

—Björn Karlsson, Principle Architect, ReadSoft; author of *Beyond the C++ Standard Library: An Introduction to Boost*

Extended STL, Volume 1: Collections and Iterators

Table of Contents

Contents

Preface

Aims

Subject Matter

Structure

Supplementary Material

Acknowledgments

Parachutes: Coda

About the Author

Prologue

A Dichotomy of Character

Principles of UNIX Programming

Seven Signs of Successful C++ Software Libraries

Balancing the Signs: Satisfaction, Dialecticism, and Idioms Old and
New

Example Libraries

Presentation Conventions

Fonts

. . . versus ...

End Iterator Precomputation

Nested Class Type Qualification

NULL

Table of Contents

Template Parameter Names

Member and Namespace-Scope Type Names

Calling Conventions

Endpoint Iterators

Namespace for Standard C Names

Class Adaptors and Instance Adaptors

Header File Names

PART ONE: Foundations

Chapter 1 The Standard Template Library

1.1 Core Concepts

1.2 Containers

1.3 Iterators

1.4 Algorithms

1.5 Function Objects

1.6 Allocators

Chapter 2 Extended STL Concepts, or When STL Meets the Real World

2.1 Terminology

2.2 Collections

2.3 Iterators

Chapter 3 Element Reference Categories

3.1 Introduction

3.2 C++ References

3.3 A Taxonomy of Element Reference Categories

3.4 Using Element Reference Categories

3.5 Defining operator `->()`

3.6 Element Reference Categories: Coda

Chapter 4 The Curious Untemporary Reference

Chapter 5 The DRY SPOT Principle

Table of Contents

5.1 DRY SPOTs in C++

5.2 Not Quite DRY SPOTs in C++

5.3 Closed Namespaces

Chapter 6 The Law of Leaky Abstractions

Chapter 7 Contract Programming

7.1 Enforcement Types

7.2 Enforcement Mechanisms

Chapter 8 Constraints

8.1 Type System Leverage

8.2 Static Assertions

Chapter 9 Shims

9.1 Introduction

9.2 Primary Shims

9.3 Composite Shims

Chapter 10 Duck and Goose, or the Whimsical Bases of Partial Structural Conformance

10.1 Conformance

10.2 Explicit Semantic Conformance

10.3 Intersecting Conformance

Chapter 11 RAI

11.1 Mutability

11.2 Resource Source

Chapter 12 Template Tools

12.1 Traits

12.2 Type Generators

12.3 True Typedefs

Chapter 13 Inferred Interface Adaptation: Compile-Time Adaptation of Interface-Incomplete Types

13.1 Introduction

13.2 Adapting Interface-Incomplete Types

Table of Contents

13.3 Adapting Immutable Collections

13.4 Inferred Interface Adaptation

13.5 Applying IIA to the Range

Chapter 14 Henneys Hypothesis, or When Templates Attack!

Chapter 15 The Independent Autonomies of equal () Friends

15.1 Beware Nonmember Friend Function Abuse

15.2 Collections and Their Iterators

Chapter 16 Essential Components

16.1 Introduction

16.2 auto_buffer

16.3 filesystem_traits

16.4 file_path_buffer

16.5 scoped_handle

16.6 dl_call ()

PART TWO: Collections

Chapter 17 Adapting the glob API

17.1 Introduction

17.2 Decomposition of the Longhand Version

17.3 unixstl : : glob_sequence

17.4 Decomposition of the Shorthand Version

17.5 Summary

Chapter 18 Intermezzo: Constructor Clashes and Design That Is, If
Not Bad, At Least Ill-Conceived for Seamless Evolution

Chapter 19 Adapting the opendir/readdir API

19.1 Introduction

19.2 Decomposition of the Longhand Version

19.3 unixstl : : readdir_sequence

19.4 Alternate Implementations

19.5 Summary

Chapter 20 Adapting the FindFirstFile/FindNextFile API

Table of Contents

20.1 Introduction

20.2 Decomposition of Examples

20.3 Sequence Design

20.4 `winstl : : basic_findfile_sequence`

20.5 `winstl : : basic_findfile_sequence_const_iterator`

20.6 `winstl : : basic_findfile_sequence_value_type`

20.7 Shims

20.8 What, No Shims and Constructor Templates?

20.9 Summary

20.10 File System Enumeration with `recls`: Coda

Chapter 21 Intermezzo: When the Efficiency/Usability Balance Is Tipped: Enumerating FTP Server Directories

21.1 `inetstl : : basic_findfile_sequence`

21.2 `inetstl : : basic_ftplib_sequence`

Chapter 22 Enumerating Processes and Modules

22.1 Collection Characteristics

22.2 `winstl : : pid_sequence`

22.3 `winstl : : process_module_sequence`

22.4 Enumerating All Modules on a System

22.5 Avoiding the System Pseudo Processes

22.6 Handling Optional API Headers

22.7 Summary

Chapter 23 The Fibonacci Sequence

23.1 Introduction

23.2 The Fibonacci Sequence

23.3 Fibonacci as an STL Sequence

23.4 Discoverability Failure

23.5 Defining Finite Bounds

23.6 Summary

Chapter 24 Adapting MFCs CArray Container Family

Table of Contents

- 24.1 Introduction
- 24.2 Motivation
- 24.3 Emulating `std::vector`
- 24.4 Design Considerations
- 24.5 `mfcstl::CArray_adaptor_base` Interface
- 24.6 `mfcstl::CArray_cadaptor`
- 24.7 `mfcstl::CArray_iadaptor`
- 24.8 Construction
- 24.9 Allocator
- 24.10 Element Access Methods
- 24.11 Iteration
- 24.12 Size
- 24.13 Capacity
- 24.14 Comparison
- 24.15 Modifiers
- 24.16 Assignment and `swap()`
- 24.17 Summary
- 24.18 On the CD

Chapter 25 A Map of the Environment

- 25.1 Introduction
- 25.2 Motivation
- 25.3 `getenv()`, `putenv()`, `setenv()/unsetenv()`, and `environ`
- 25.4 `platformstl::environment_variable_traits`
- 25.5 Planning the Interface
- 25.6 Lookup by Name
- 25.7 Inserting, Updating, and Deleting Values by Name
- 25.8 Iteration
- 25.9 Final Iteration Implementation
- 25.10 Heterogeneous Reference Categories?
- 25.11 `size()` and Subscript by Index
- 25.12 Summary

Table of Contents

25.13 On the CD

Chapter 26 Traveling Back and Forth on the Z-Plane

26.1 Prologue

26.2 Introduction

26.3 Version 1: Forward Iteration

26.4 Version 2: Bidirectional Iteration

26.5 Handling External Change

26.6 `winstl` : : `child_window_sequence`

26.7 Bidirectional Iterator Blues

26.8 `winstl` : : `zorder_iterator`: A Reversal of Self

26.9 Finalizing the Window Peer Sequences

26.10 Summary

26.11 Z-Plane: Coda

Chapter 27 String Tokenization

27.1 Introduction

27.2 `strtok` ()

27.3 `SynesisSTL` : : `StringTokeniser`

27.4 Tokenization Use Cases

27.5 Other Tokenization Alternatives

27.6 `stlsoft` : : `string_tokeniser`

27.7 Test Drive

27.8 The Policy Folly

27.9 Performance

27.10 Summary

Chapter 28 Adapting COM Enumerators

28.1 Introduction

28.2 Motivation

28.3 COM Enumerators

28.4 Decomposition of the Longhand Version

28.5 `comstl` : : `enumerator_sequence`

Table of Contents

28.6 `constexpr :: enumerator_sequence :: iterator`

28.7 `constexpr :: enumerator_sequence :: iterator :: enumeration_context`

28.8 Iterator Cloning Policies

28.9 Choosing a Default Cloning Policy: Applying the Principle of Least Surprise

28.10 Summary

28.11 Coming Next

Chapter 29 Intermezzo: Correcting Minor Design Omissions with Member Type Inference

Chapter 30 Adapting COM Collections

30.1 Introduction

30.2 Motivation

30.3 `constexpr :: collection_sequence`

30.4 Enumerator Acquisition Policies

30.5 Summary

Chapter 31 Gathering Scattered I/O

31.1 Introduction

31.2 Scatter/Gather I/O

31.3 Scatter/Gather I/O APIs

31.4 Adapting `ACE_Message_Queue`

31.5 Time for Some Cake

31.6 Summary

Chapter 32 Argument-Dependent Return-Type Variance

32.1 Introduction

32.2 Borrowing a Jewel from Ruby

32.3 Dual-Semantic Subscripting in C++

32.4 Generalized Compatibility via String Access Shims

32.5 A Fly in the int-ment

32.6 Selecting Return Type and Overload

32.7 Summary

Table of Contents

Chapter 33 External Iterator Invalidation

- 33.1 Element-Interface Coherence
- 33.2 Windows ListBox and ComboBox Controls
- 33.3 Enumerating Registry Keys and Values
- 33.4 Summary
- 33.5 On the CD

PART THREE: Iterators

Chapter 34 An Enhanced ostream_iterator

- 34.1 Introduction
- 34.2 std::ostream_iterator
- 34.3 stlsoft::ostream_iterator
- 34.4 Defining Stream Insertion Operators
- 34.5 Summary

Chapter 35 Intermezzo: Proscribing Fatuous Output Iterator Syntax Using the Dereference Proxy Pattern

- 35.1 stlsoft::ostream_iterator::deref_proxy

Chapter 36 Transform Iterator

- 36.1 Introduction
- 36.2 Motivation
- 36.3 Defining Iterator Adaptors
- 36.4 stlsoft::transform_iterator
- 36.5 Composite Transformations
- 36.6 DRY SPOT Violations?
- 36.7 A Spoonful of Sequence Helps the Medicine ... ?
- 36.8 Summary
- 36.9 On the CD

Chapter 37 Intermezzo: Discretion Being the Better Part of Nomenclature . . .

Chapter 38 Member Selector Iterator

Table of Contents

- 38.1 Introduction
- 38.2 Motivation
- 38.3 `stlsoft : : member_selector_iterator`
- 38.4 Creator Function Woes
- 38.5 Summary
- 38.6 On the CD

Chapter 39 C-Style String Concatenation

- 39.1 Motivation
- 39.2 An Inflexible Version
- 39.3 `stlsoft : : cstring_concatenator_iterator`
- 39.4 Creator Functions
- 39.5 Summary
- 39.6 On the CD

Chapter 40 String Object Concatenation

- 40.1 Introduction
- 40.2 `stlsoft : : string_concatenator_iterator`
- 40.3 Heterogeneity of String Types
- 40.4 But ...
- 40.5 Summary

Chapter 41 Adapted Iterators Traits

- 41.1 Introduction
- 41.2 `stlsoft : : adapted_iterator_traits`
- 41.3 Summary
- 41.4 On the CD

Chapter 42 Filtered Iteration

- 42.1 Introduction
- 42.2 An Invalid Version
- 42.3 Member Iterators Define the Range
- 42.4 So ... ?
- 42.5 `stlsoft : : filter_iterator`

Table of Contents

42.6 Constraining the Iterator Category

42.7 Summary

42.8 On the CD

Chapter 43 Composite Iterator Adaptations

43.1 Introduction

43.2 Transforming a Filtered Iterator

43.3 Filtering a Transformed Iterator

43.4 Hedging Our Bets

43.5 Summary

Epilogue

Bibliography

Index