

C++ Primer
Fifth Edition

Stanley B. Lippman
Josée Lajoie
Barbara E. Moo

◆◆ Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sidney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U. S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the U. S., please contact:

International Sales
international@pearsoned.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Lippman, Stanley B.

C++ primer / Stanley B. Lippman, Josée Lajoie, Barbara E. Moo. – 5th ed.

p. cm.

Includes index.

ISBN 0-321-71411-3 (pbk. : alk. paper) 1. C++ (Computer program language) I. Lajoie, Josée. II. Moo, Barbara E. III. Title.

QA76.73.C153L57697 2013

005.13'3–dc23

2012020184

Copyright © 2013 Objectwrite Inc., Josée Lajoie and Barbara E. Moo

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

C++ Primer, Fifth Edition, features an enhanced, layflat binding, which allows the book to stay open more easily when placed on a flat surface. This special binding method—notable by a small space inside the spine—also increases durability.

ISBN-13: 978-0-321-71411-4

ISBN-10: 0-321-71411-3

Text printed in the United States on recycled paper at Courier in Westford, Massachusetts.

Third printing, February 2013

PEARSON

Contents

Preface	xxiii
Chapter 1 Getting Started	1
1.1 Writing a Simple C++ Program	2
1.1.1 Compiling and Executing Our Program	3
1.2 A First Look at Input/Output	5
1.3 A Word about Comments	9
1.4 Flow of Control	11
1.4.1 The while Statement	11
1.4.2 The for Statement	13
1.4.3 Reading an Unknown Number of Inputs	14
1.4.4 The if Statement	17
1.5 Introducing Classes	19
1.5.1 The Sales_item Class	20
1.5.2 A First Look at Member Functions	23
1.6 The Bookstore Program	24
Chapter Summary	26
Defined Terms	26
Part I The Basics	29
Chapter 2 Variables and Basic Types	31
2.1 Primitive Built-in Types	32
2.1.1 Arithmetic Types	32
2.1.2 Type Conversions	35
2.1.3 Literals	38
2.2 Variables	41
2.2.1 Variable Definitions	41
2.2.2 Variable Declarations and Definitions	44
2.2.3 Identifiers	46
2.2.4 Scope of a Name	48
2.3 Compound Types	50
2.3.1 References	50
2.3.2 Pointers	52

2.3.3	Understanding Compound Type Declarations	57
2.4	<code>const</code> Qualifier	59
2.4.1	References to <code>const</code>	61
2.4.2	Pointers and <code>const</code>	62
2.4.3	Top-Level <code>const</code>	63
2.4.4	<code>constexpr</code> and Constant Expressions	65
2.5	Dealing with Types	67
2.5.1	Type Aliases	67
2.5.2	The <code>auto</code> Type Specifier	68
2.5.3	The <code>decltype</code> Type Specifier	70
2.6	Defining Our Own Data Structures	72
2.6.1	Defining the <code>Sales_data</code> Type	72
2.6.2	Using the <code>Sales_data</code> Class	74
2.6.3	Writing Our Own Header Files	76
	Chapter Summary	78
	Defined Terms	78
Chapter 3 Strings, Vectors, and Arrays		81
3.1	Namespace using Declarations	82
3.2	Library <code>string</code> Type	84
3.2.1	Defining and Initializing <code>strings</code>	84
3.2.2	Operations on <code>strings</code>	85
3.2.3	Dealing with the Characters in a <code>string</code>	90
3.3	Library <code>vector</code> Type	96
3.3.1	Defining and Initializing <code>vectors</code>	97
3.3.2	Adding Elements to a <code>vector</code>	100
3.3.3	Other <code>vector</code> Operations	102
3.4	Introducing Iterators	106
3.4.1	Using Iterators	106
3.4.2	Iterator Arithmetic	111
3.5	Arrays	113
3.5.1	Defining and Initializing Built-in Arrays	113
3.5.2	Accessing the Elements of an Array	116
3.5.3	Pointers and Arrays	117
3.5.4	C-Style Character Strings	122
3.5.5	Interfacing to Older Code	124
3.6	Multidimensional Arrays	125
	Chapter Summary	131
	Defined Terms	131
Chapter 4 Expressions		133
4.1	Fundamentals	134
4.1.1	Basic Concepts	134
4.1.2	Precedence and Associativity	136
4.1.3	Order of Evaluation	137
4.2	Arithmetic Operators	139
4.3	Logical and Relational Operators	141

4.4	Assignment Operators	144
4.5	Increment and Decrement Operators	147
4.6	The Member Access Operators	150
4.7	The Conditional Operator	151
4.8	The Bitwise Operators	152
4.9	The sizeof Operator	156
4.10	Comma Operator	157
4.11	Type Conversions	159
4.11.1	The Arithmetic Conversions	159
4.11.2	Other Implicit Conversions	161
4.11.3	Explicit Conversions	162
4.12	Operator Precedence Table	166
	Chapter Summary	168
	Defined Terms	168
Chapter 5	Statements	171
5.1	Simple Statements	172
5.2	Statement Scope	174
5.3	Conditional Statements	174
5.3.1	The if Statement	175
5.3.2	The switch Statement	178
5.4	Iterative Statements	183
5.4.1	The while Statement	183
5.4.2	Traditional for Statement	185
5.4.3	Range for Statement	187
5.4.4	The do while Statement	189
5.5	Jump Statements	190
5.5.1	The break Statement	190
5.5.2	The continue Statement	191
5.5.3	The goto Statement	192
5.6	try Blocks and Exception Handling	193
5.6.1	A throw Expression	193
5.6.2	The try Block	194
5.6.3	Standard Exceptions	197
	Chapter Summary	199
	Defined Terms	199
Chapter 6	Functions	201
6.1	Function Basics	202
6.1.1	Local Objects	204
6.1.2	Function Declarations	206
6.1.3	Separate Compilation	207
6.2	Argument Passing	208
6.2.1	Passing Arguments by Value	209
6.2.2	Passing Arguments by Reference	210
6.2.3	const Parameters and Arguments	212
6.2.4	Array Parameters	214

6.2.5	main: Handling Command-Line Options	218
6.2.6	Functions with Varying Parameters	220
6.3	Return Types and the <code>return</code> Statement	222
6.3.1	Functions with No Return Value	223
6.3.2	Functions That Return a Value	223
6.3.3	Returning a Pointer to an Array	228
6.4	Overloaded Functions	230
6.4.1	Overloading and Scope	234
6.5	Features for Specialized Uses	236
6.5.1	Default Arguments	236
6.5.2	Inline and <code>constexpr</code> Functions	238
6.5.3	Aids for Debugging	240
6.6	Function Matching	242
6.6.1	Argument Type Conversions	245
6.7	Pointers to Functions	247
	Chapter Summary	251
	Defined Terms	251
Chapter 7	Classes	253
7.1	Defining Abstract Data Types	254
7.1.1	Designing the <code>Sales_data</code> Class	254
7.1.2	Defining the Revised <code>Sales_data</code> Class	256
7.1.3	Defining Nonmember Class-Related Functions	260
7.1.4	Constructors	262
7.1.5	Copy, Assignment, and Destruction	267
7.2	Access Control and Encapsulation	268
7.2.1	Friends	269
7.3	Additional Class Features	271
7.3.1	Class Members Revisited	271
7.3.2	Functions That Return <code>*this</code>	275
7.3.3	Class Types	277
7.3.4	Friendship Revisited	279
7.4	Class Scope	282
7.4.1	Name Lookup and Class Scope	283
7.5	Constructors Revisited	288
7.5.1	Constructor Initializer List	288
7.5.2	Delegating Constructors	291
7.5.3	The Role of the Default Constructor	293
7.5.4	Implicit Class-Type Conversions	294
7.5.5	Aggregate Classes	298
7.5.6	Literal Classes	299
7.6	<code>static</code> Class Members	300
	Chapter Summary	305
	Defined Terms	305

Part II	The C++ Library	307
Chapter 8	The IO Library	309
8.1	The IO Classes	310
8.1.1	No Copy or Assign for IO Objects	311
8.1.2	Condition States	312
8.1.3	Managing the Output Buffer	314
8.2	File Input and Output	316
8.2.1	Using File Stream Objects	317
8.2.2	File Modes	319
8.3	string Streams	321
8.3.1	Using an <code>istringstream</code>	321
8.3.2	Using <code>ostringstreams</code>	323
	Chapter Summary	324
	Defined Terms	324
Chapter 9	Sequential Containers	325
9.1	Overview of the Sequential Containers	326
9.2	Container Library Overview	328
9.2.1	Iterators	331
9.2.2	Container Type Members	332
9.2.3	<code>begin</code> and <code>end</code> Members	333
9.2.4	Defining and Initializing a Container	334
9.2.5	Assignment and <code>swap</code>	337
9.2.6	Container Size Operations	340
9.2.7	Relational Operators	340
9.3	Sequential Container Operations	341
9.3.1	Adding Elements to a Sequential Container	341
9.3.2	Accessing Elements	346
9.3.3	Erasing Elements	348
9.3.4	Specialized <code>forward_list</code> Operations	350
9.3.5	Resizing a Container	352
9.3.6	Container Operations May Invalidate Iterators	353
9.4	How a <code>vector</code> Grows	355
9.5	Additional <code>string</code> Operations	360
9.5.1	Other Ways to Construct <code>strings</code>	360
9.5.2	Other Ways to Change a <code>string</code>	361
9.5.3	<code>string</code> Search Operations	364
9.5.4	The <code>compare</code> Functions	366
9.5.5	Numeric Conversions	367
9.6	Container Adaptors	368
	Chapter Summary	372
	Defined Terms	372

Chapter 10 Generic Algorithms	375
10.1 Overview	376
10.2 A First Look at the Algorithms	378
10.2.1 Read-Only Algorithms	379
10.2.2 Algorithms That Write Container Elements	380
10.2.3 Algorithms That Reorder Container Elements	383
10.3 Customizing Operations	385
10.3.1 Passing a Function to an Algorithm	386
10.3.2 Lambda Expressions	387
10.3.3 Lambda Captures and Returns	392
10.3.4 Binding Arguments	397
10.4 Revisiting Iterators	401
10.4.1 Insert Iterators	401
10.4.2 <code>iostream</code> Iterators	403
10.4.3 Reverse Iterators	407
10.5 Structure of Generic Algorithms	410
10.5.1 The Five Iterator Categories	410
10.5.2 Algorithm Parameter Patterns	412
10.5.3 Algorithm Naming Conventions	413
10.6 Container-Specific Algorithms	415
Chapter Summary	417
Defined Terms	417
Chapter 11 Associative Containers	419
11.1 Using an Associative Container	420
11.2 Overview of the Associative Containers	423
11.2.1 Defining an Associative Container	423
11.2.2 Requirements on Key Type	424
11.2.3 The <code>pair</code> Type	426
11.3 Operations on Associative Containers	428
11.3.1 Associative Container Iterators	429
11.3.2 Adding Elements	431
11.3.3 Erasing Elements	434
11.3.4 Subscripting a <code>map</code>	435
11.3.5 Accessing Elements	436
11.3.6 A Word Transformation Map	440
11.4 The Unordered Containers	443
Chapter Summary	447
Defined Terms	447
Chapter 12 Dynamic Memory	449
12.1 Dynamic Memory and Smart Pointers	450
12.1.1 The <code>shared_ptr</code> Class	450
12.1.2 Managing Memory Directly	458
12.1.3 Using <code>shared_ptrs</code> with <code>new</code>	464
12.1.4 Smart Pointers and Exceptions	467
12.1.5 <code>unique_ptr</code>	470

- 12.1.6 `weak_ptr` 473
- 12.2 Dynamic Arrays 476
 - 12.2.1 `new` and Arrays 477
 - 12.2.2 The `allocator` Class 481
- 12.3 Using the Library: A Text-Query Program 484
 - 12.3.1 Design of the Query Program 485
 - 12.3.2 Defining the Query Program Classes 487
- Chapter Summary 491
- Defined Terms 491

Part III Tools for Class Authors 493

Chapter 13 Copy Control 495

- 13.1 Copy, Assign, and Destroy 496
 - 13.1.1 The Copy Constructor 496
 - 13.1.2 The Copy-Assignment Operator 500
 - 13.1.3 The Destructor 501
 - 13.1.4 The Rule of Three/Five 503
 - 13.1.5 Using `= default` 506
 - 13.1.6 Preventing Copies 507
- 13.2 Copy Control and Resource Management 510
 - 13.2.1 Classes That Act Like Values 511
 - 13.2.2 Defining Classes That Act Like Pointers 513
- 13.3 Swap 516
- 13.4 A Copy-Control Example 519
- 13.5 Classes That Manage Dynamic Memory 524
- 13.6 Moving Objects 531
 - 13.6.1 Rvalue References 532
 - 13.6.2 Move Constructor and Move Assignment 534
 - 13.6.3 Rvalue References and Member Functions 544
- Chapter Summary 549
- Defined Terms 549

Chapter 14 Overloaded Operations and Conversions 551

- 14.1 Basic Concepts 552
- 14.2 Input and Output Operators 556
 - 14.2.1 Overloading the Output Operator `<<` 557
 - 14.2.2 Overloading the Input Operator `>>` 558
- 14.3 Arithmetic and Relational Operators 560
 - 14.3.1 Equality Operators 561
 - 14.3.2 Relational Operators 562
- 14.4 Assignment Operators 563
- 14.5 Subscript Operator 564
- 14.6 Increment and Decrement Operators 566
- 14.7 Member Access Operators 569
- 14.8 Function-Call Operator 571

14.8.1	Lambdas Are Function Objects	572
14.8.2	Library-Defined Function Objects	574
14.8.3	Callable Objects and <code>function</code>	576
14.9	Overloading, Conversions, and Operators	579
14.9.1	Conversion Operators	580
14.9.2	Avoiding Ambiguous Conversions	583
14.9.3	Function Matching and Overloaded Operators	587
	Chapter Summary	590
	Defined Terms	590
Chapter 15	Object-Oriented Programming	591
15.1	OOP: An Overview	592
15.2	Defining Base and Derived Classes	594
15.2.1	Defining a Base Class	594
15.2.2	Defining a Derived Class	596
15.2.3	Conversions and Inheritance	601
15.3	Virtual Functions	603
15.4	Abstract Base Classes	608
15.5	Access Control and Inheritance	611
15.6	Class Scope under Inheritance	617
15.7	Constructors and Copy Control	622
15.7.1	Virtual Destructors	622
15.7.2	Synthesized Copy Control and Inheritance	623
15.7.3	Derived-Class Copy-Control Members	625
15.7.4	Inherited Constructors	628
15.8	Containers and Inheritance	630
15.8.1	Writing a Basket Class	631
15.9	Text Queries Revisited	634
15.9.1	An Object-Oriented Solution	636
15.9.2	The <code>query_base</code> and <code>query</code> Classes	639
15.9.3	The Derived Classes	642
15.9.4	The <code>eval</code> Functions	645
	Chapter Summary	649
	Defined Terms	649
Chapter 16	Templates and Generic Programming	651
16.1	Defining a Template	652
16.1.1	Function Templates	652
16.1.2	Class Templates	658
16.1.3	Template Parameters	668
16.1.4	Member Templates	672
16.1.5	Controlling Instantiations	675
16.1.6	Efficiency and Flexibility	676
16.2	Template Argument Deduction	678
16.2.1	Conversions and Template Type Parameters	679
16.2.2	Function-Template Explicit Arguments	681
16.2.3	Trailing Return Types and Type Transformation	683

- 16.2.4 Function Pointers and Argument Deduction 686
- 16.2.5 Template Argument Deduction and References 687
- 16.2.6 Understanding `std::move` 690
- 16.2.7 Forwarding 692
- 16.3 Overloading and Templates 694
- 16.4 Variadic Templates 699
 - 16.4.1 Writing a Variadic Function Template 701
 - 16.4.2 Pack Expansion 702
 - 16.4.3 Forwarding Parameter Packs 704
- 16.5 Template Specializations 706
- Chapter Summary 713
- Defined Terms 713

Part IV Advanced Topics 715

Chapter 17 Specialized Library Facilities 717

- 17.1 The tuple Type 718
 - 17.1.1 Defining and Initializing tuples 718
 - 17.1.2 Using a tuple to Return Multiple Values 721
- 17.2 The bitset Type 723
 - 17.2.1 Defining and Initializing bitsets 723
 - 17.2.2 Operations on bitsets 725
- 17.3 Regular Expressions 728
 - 17.3.1 Using the Regular Expression Library 729
 - 17.3.2 The Match and Regex Iterator Types 734
 - 17.3.3 Using Subexpressions 738
 - 17.3.4 Using `regex_replace` 741
- 17.4 Random Numbers 745
 - 17.4.1 Random-Number Engines and Distribution 745
 - 17.4.2 Other Kinds of Distributions 749
- 17.5 The IO Library Revisited 752
 - 17.5.1 Formatted Input and Output 753
 - 17.5.2 Unformatted Input/Output Operations 761
 - 17.5.3 Random Access to a Stream 763
- Chapter Summary 769
- Defined Terms 769

Chapter 18 Tools for Large Programs 771

- 18.1 Exception Handling 772
 - 18.1.1 Throwing an Exception 772
 - 18.1.2 Catching an Exception 775
 - 18.1.3 Function `try` Blocks and Constructors 777
 - 18.1.4 The `noexcept` Exception Specification 779
 - 18.1.5 Exception Class Hierarchies 782
- 18.2 Namespaces 785
 - 18.2.1 Namespace Definitions 785

18.2.2	Using Namespace Members	792
18.2.3	Classes, Namespaces, and Scope	796
18.2.4	Overloading and Namespaces	800
18.3	Multiple and Virtual Inheritance	802
18.3.1	Multiple Inheritance	803
18.3.2	Conversions and Multiple Base Classes	805
18.3.3	Class Scope under Multiple Inheritance	807
18.3.4	Virtual Inheritance	810
18.3.5	Constructors and Virtual Inheritance	813
	Chapter Summary	816
	Defined Terms	816
Chapter 19 Specialized Tools and Techniques		819
19.1	Controlling Memory Allocation	820
19.1.1	Overloading new and delete	820
19.1.2	Placement new Expressions	823
19.2	Run-Time Type Identification	825
19.2.1	The dynamic_cast Operator	825
19.2.2	The typeid Operator	826
19.2.3	Using RTTI	828
19.2.4	The type_info Class	831
19.3	Enumerations	832
19.4	Pointer to Class Member	835
19.4.1	Pointers to Data Members	836
19.4.2	Pointers to Member Functions	838
19.4.3	Using Member Functions as Callable Objects	841
19.5	Nested Classes	843
19.6	union: A Space-Saving Class	847
19.7	Local Classes	852
19.8	Inherently Nonportable Features	854
19.8.1	Bit-fields	854
19.8.2	volatile Qualifier	856
19.8.3	Linkage Directives: extern "C"	857
	Chapter Summary	862
	Defined Terms	862
Appendix A The Library		865
A.1	Library Names and Headers	866
A.2	A Brief Tour of the Algorithms	870
A.2.1	Algorithms to Find an Object	871
A.2.2	Other Read-Only Algorithms	872
A.2.3	Binary Search Algorithms	873
A.2.4	Algorithms That Write Container Elements	873
A.2.5	Partitioning and Sorting Algorithms	875
A.2.6	General Reordering Operations	877
A.2.7	Permutation Algorithms	879
A.2.8	Set Algorithms for Sorted Sequences	880

- A.2.9 Minimum and Maximum Values 880
- A.2.10 Numeric Algorithms 881
- A.3 Random Numbers 882
 - A.3.1 Random Number Distributions 883
 - A.3.2 Random Number Engines 884

Index **887**

This page intentionally left blank

New Features in C++11

2.1.1	long long Type	33
2.2.1	List Initialization	43
2.3.2	nullptr Literal	54
2.4.4	constexpr Variables	66
2.5.1	Type Alias Declarations	68
2.5.2	The auto Type Specifier	68
2.5.3	The decltype Type Specifier	70
2.6.1	In-Class Initializers	73
3.2.2	Using auto or decltype for Type Abbreviation	88
3.2.3	Range for Statement	91
3.3	Defining a vector of vectors	97
3.3.1	List Initialization for vectors	98
3.4.1	Container cbegin and cend Functions	109
3.5.3	Library begin and end Functions	118
3.6	Using auto or decltype to Simplify Declarations	129
4.2	Rounding Rules for Division	141
4.4	Assignment from a Braced List of Values	145
4.9	sizeof Applied to a Class Member	157
5.4.3	Range for Statement	187
6.2.6	Library initializer_list Class	220
6.3.2	List Initializing a Return Value	226
6.3.3	Declaring a Trailing Return Type	229
6.3.3	Using decltype to Simplify Return Type Declarations	230
6.5.2	constexpr Functions	239
7.1.4	Using = default to Generate a Default Constructor	265
7.3.1	In-class Initializers for Members of Class Type	274
7.5.2	Delegating Constructors	291
7.5.6	constexpr Constructors	299
8.2.1	Using strings for File Names	317
9.1	The array and forward_list Containers	327
9.2.3	Container cbegin and cend Functions	334
9.2.4	List Initialization for Containers	336
9.2.5	Container Nonmember swap Functions	339
9.3.1	Return Type for Container insert Members	344
9.3.1	Container emplace Members	345

9.4	<code>shrink_to_fit</code>	357
9.5.5	Numeric Conversion Functions for <code>strings</code>	367
10.3.2	Lambda Expressions	388
10.3.3	Trailing Return Type in Lambda Expressions	396
10.3.4	The Library <code>bind</code> Function	397
11.2.1	List Initialization of an Associative Container	423
11.2.3	List Initializing <code>pair</code> Return Type	427
11.3.2	List Initialization of a <code>pair</code>	431
11.4	The Unordered Containers	443
12.1	Smart Pointers	450
12.1.1	The <code>shared_ptr</code> Class	450
12.1.2	List Initialization of Dynamically Allocated Objects	459
12.1.2	<code>auto</code> and Dynamic Allocation	459
12.1.5	The <code>unique_ptr</code> Class	470
12.1.6	The <code>weak_ptr</code> Class	473
12.2.1	Range <code>for</code> Doesn't Apply to Dynamically Allocated Arrays	477
12.2.1	List Initialization of Dynamically Allocated Arrays	478
12.2.1	<code>auto</code> Can't Be Used to Allocate an Array	478
12.2.2	<code>allocator::construct</code> Can Use any Constructor	482
13.1.5	Using <code>= default</code> for Copy-Control Members	506
13.1.6	Using <code>= delete</code> to Prevent Copying Class Objects	507
13.5	Moving Instead of Copying Class Objects	529
13.6.1	Rvalue References	532
13.6.1	The Library <code>move</code> Function	533
13.6.2	Move Constructor and Move Assignment	534
13.6.2	Move Constructors Usually Should Be <code>noexcept</code>	535
13.6.2	Move Iterators	543
13.6.3	Reference Qualified Member Functions	546
14.8.3	The <code>function</code> Class Template	577
14.9.1	<code>explicit</code> Conversion Operators	582
15.2.2	<code>override</code> Specifier for Virtual Functions	596
15.2.2	Preventing Inheritance by Defining a Class as <code>final</code>	600
15.3	<code>override</code> and <code>final</code> Specifiers for Virtual Functions	606
15.7.2	Deleted Copy Control and Inheritance	624
15.7.4	Inherited Constructors	628
16.1.2	Declaring a Template Type Parameter as a Friend	666
16.1.2	Template Type Aliases	666
16.1.3	Default Template Arguments for Template Functions	670
16.1.5	Explicit Control of Instantiation	675
16.2.3	Template Functions and Trailing Return Types	684
16.2.5	Reference Collapsing Rules	688
16.2.6	<code>static_cast</code> from an Lvalue to an Rvalue	691
16.2.7	The Library <code>forward</code> Function	694
16.4	Variadic Templates	699
16.4	The <code>sizeof... Operator</code>	700
16.4.3	Variadic Templates and Forwarding	704

17.1	The Library <code>tuple</code> Class Template	718
17.2.2	New <code>bitset</code> Operations	726
17.3	The Regular Expression Library	728
17.4	The Random Number Library	745
17.5.1	Floating-Point Format Control	757
18.1.4	The <code>noexcept</code> Exception Specifier	779
18.1.4	The <code>noexcept</code> Operator	780
18.2.1	Inline Namespaces	790
18.3.1	Inherited Constructors and Multiple Inheritance	804
19.3	Scoped enums	832
19.3	Specifying the Type Used to Hold an enum	834
19.3	Forward Declarations for enums	834
19.4.3	The Library <code>mem_fn</code> Class Template	843
19.6	Union Members of Class Types	848