

CONTENTS

Foreword	xv
Preface	xvii
Guide for Readers	xxi

PART I: STREAM INPUT AND OUTPUT

1. IOStreams Basics	3
1.1 <i>Input and Output</i>	3
1.2 <i>Formatted Input/Output</i>	12
1.2.1 The Predefined Global Streams	12
1.2.2 The Input and Output Operators	13
1.2.3 The Format Parameters of a Stream	16
1.2.4 Manipulators	22
1.2.5 The Locale of a Stream	27
1.2.6 Comparison Between Formatted Input and Output	28
1.2.7 Peculiarities of Formatted Input	29
1.3 <i>The Stream State</i>	31
1.3.1 The Stream State Flags	31
1.3.2 Checking the Stream State	34
1.3.3 Catching Stream Exceptions	36
1.3.4 Resetting the Stream State	38
1.4 <i>File Input/Output</i>	39
1.4.1 Creating, Opening, Closing and Destroying File Streams	39
1.4.2 The Open Modes	41
1.4.3 Bidirectional File Streams	45

1.5 In-Memory Input/Output	47
1.6 Unformatted Input/Output	48
1.7 Stream Positioning	51
1.8 Synchronization of Streams	54
1.8.1 Means of Synchronization	54
1.8.1.1 Synchronization via <code>flush()</code> and <code>sync()</code>	54
1.8.1.2 Synchronization using the <code>unitbuf</code> Format Flag	55
1.8.1.3 Synchronization by Tying Streams	55
1.8.2 Synchronizing the Predefined Standard Streams	56
1.8.2.1 Synchronization Among the Predefined Standard Streams	57
1.8.2.2 Synchronization with the C Standard I/O	57
1.8.2.3 Synchronization with the External Device	58
1.8.2.4 Synchronization Between Predefined Standard Streams for Narrow and Wide Characters	58
2. The Architecture of IOStreams	61
2.1 The Stream Classes	63
2.1.1 Class Hierarchy	63
2.1.1.1 The Stream Base Classes	63
2.1.1.2 The General Stream Classes	66
2.1.1.3 The Concrete Stream Classes	68
2.1.2 How Streams Maintain Their Stream Buffer	69
2.1.3 Copying and Assignment of Streams	72
2.1.4 How Streams Maintain Their Locale	76
2.1.5 Collaboration Among Streams, Stream Buffers, and Locales	79
2.2 The Stream Buffer Classes	84
2.2.1 Class Hierarchy	85
2.2.2 The Stream Buffer Abstraction	86
2.2.3 String Stream Buffers	91
2.2.4 File Stream Buffers	97
2.3 Character Types and Character Traits	109
2.3.1 Character Representations	109
2.3.2 Character Traits	114
2.3.2.1 Requirements of a Character Traits Type	114
2.3.2.2 The Predefined Standard Character Traits	118
2.3.3 Character Types	119
2.3.3.1 Requirements for Character Types	120
2.4 Stream Iterators and Stream Buffer Iterators	123
2.4.1 The Concepts of Iterators in the Standard Library	123
2.4.2 Stream Iterators	126
2.4.2.1 Output Stream Iterator	127
2.4.2.2 Input Stream Iterator	129

2.4.2.3	Stream Iterators Are One-Pass Iterators	132
2.4.3	Stream Buffer Iterators	134
2.4.3.1	Output Stream Buffer Iterator	134
2.4.3.2	Input Stream Buffer Iterator	136
2.5	<i>Additional Stream Storage and Stream Callbacks</i>	140
2.5.1	Additional Stream Storage	140
2.5.2	Stream Callbacks	142
3.	Advanced iostreams Usage	145
3.1	<i>Input and Output of User-Defined Types</i>	145
3.1.1	The Signature of Inserters and Extractors	146
3.1.2	First Inserters and Extractors	146
3.1.3	Refinements	150
3.1.3.1	Format Control	150
3.1.3.2	Prefix and Suffix Operations	151
3.1.3.3	Error Indication	152
3.1.3.4	Internationalization	154
3.1.3.5	I/O Operations	155
3.1.4	Refined Inserters and Extractors	156
3.1.4.1	Internationalization	156
3.1.4.2	Prefix and Suffix Operations	158
3.1.4.3	Format Control	159
3.1.4.4	Error Indication	161
3.1.4.5	Using the Refined Inserter and Extractor	168
3.1.5	Generic Inserters and Extractors	169
3.1.6	Simple Versus Refined Approach	174
3.2	<i>User-Defined Manipulators</i>	176
3.2.1	Manipulators Without Parameters	177
3.2.2	Manipulators with Parameters	179
3.2.2.1	Straightforward Manipulator Implementations	179
3.2.2.2	Generalized Technique: Using a Manipulator Base Template	180
3.2.2.3	Variants of a Manipulator Implementation	184
3.2.2.4	Refinements	186
3.2.2.5	The Standard Manipulator Base Type <code>smanip</code>	191
3.3	<i>Extending Stream Functionality</i>	191
3.3.1	Using Stream Storage for Private Use: <code>iwor</code> d, <code>pwor</code> d, and <code>xalloc</code>	192
3.3.1.1	Initializing and Maintaining the <code>iwor</code> d/ <code>pwor</code> d Index	195
3.3.1.2	Implementing the Date Inserter	195
3.3.1.3	Implementing the Manipulator	197
3.3.1.4	Using Stream Callbacks for Memory Management	200
3.3.1.5	Error Indication of Stream Callback Functions	204
3.3.1.6	Using the New Functionality	207
3.3.1.7	Evaluation of the <code>iwor</code> d/ <code>pwor</code> d Approach	207

3.3.2	Creating New Stream Classes by Derivation	209
3.3.2.1	Deriving the New Stream Class	210
3.3.2.2	Implementing the Date Inserter and the Manipulator	218
3.3.2.3	Summary	222
3.3.3	Comparing Both Solutions—iword/pword versus Derivation	223
3.4	<i>Adding Stream Buffer Functionality</i>	225
3.4.1	Deriving from the Stream Buffer Base Class	225
3.4.1.1	Core Functionality of Stream Buffers: Character Transportation	226
3.4.1.2	Optional Functionality of Stream Buffers	244
3.4.1.3	Providing New Stream Classes Along with New Stream Buffer Classes	245
3.4.2	Deriving from Concrete Stream Buffer Classes	246
 PART II: INTERNATIONALIZATION		
4.	Introduction to Internationalization and Localization	251
4.1	<i>Internationalization and Localization</i>	252
4.2	<i>Cultural Conventions</i>	253
4.2.1	Language	253
4.2.2	Numerical Values	253
4.2.3	Monetary Values	254
4.2.4	Time and Date	254
4.2.5	Sorting Words	255
4.2.6	Messages	256
4.2.7	Character Encodings	257
4.2.7.1	Terms and Definitions	257
4.2.7.2	Character Codesets	258
4.2.7.3	Character Encoding Schemes	259
4.2.7.4	Uses of Multibyte Encodings and Wide Characters	262
4.2.7.5	Code Conversions	263
5.	Locales	265
5.1	<i>Creating Locale Objects</i>	267
5.1.1	Named Locales	267
5.1.2	Combined Locales	268
5.1.3	The Global Locale	270
5.2	<i>Retrieving Facets from a Locale</i>	271
5.2.1	has_facet	272
5.2.2	use_facet	272
6.	Standard Facets	275
6.1	<i>Alphabet- and Language-Related Facets</i>	276
6.1.1	Character Classification	276
6.1.1.1	Character Classification	276

6.1.1.2	Character Conversion to Upper- and Lowercase	278
6.1.1.3	Character Conversion Between <code>charT</code> and <code>char</code>	279
6.1.1.4	Special Properties of <code>ctype<char></code>	279
6.1.2	String Collation	280
6.1.3	Code Conversion	282
6.1.4	Message Catalogs	285
6.2	<i>Formatting and Parsing Facets</i>	286
6.2.1	Numerical and Boolean Values	286
6.2.1.1	The <code>num_punct</code> Facet	287
6.2.1.2	The <code>num_put</code> Facet	287
6.2.1.3	The <code>num_get</code> Facet	289
6.2.2	Monetary Values	290
6.2.2.1	The <code>money_punct</code> Facet	291
6.2.2.2	The <code>money_put</code> Facet	292
6.2.2.3	The <code>money_get</code> Facet	294
6.2.3	Date and Time Values	295
6.2.3.1	The <code>time_put</code> Facet	295
6.2.3.2	The <code>time_get</code> Facet	297
6.3	<i>Grouping of Standard Facets in a Locale</i>	299
6.3.1	The Standard Facet Families	299
6.3.1.1	The Standard Facet Base Class Templates	299
6.3.1.2	The Derived by Name Facets	300
6.3.1.3	Behavior of the Base Class Facets	300
6.3.1.4	Mandatory Facet Types	302
6.3.2	Locale Categories	304
6.3.3	Diagram: Facets and Categories	305
6.4	<i>Advanced Usage of the Standard Facets</i>	306
6.4.1	Indirect Use of a Facet Through a Stream	306
6.4.2	Use of a Facet Through a Locale	307
6.4.3	Direct Use of the Facet Independent of a Locale	309
7	The Architecture of the Locale Framework	311
7.1	<i>Class Hierarchy</i>	311
7.2	<i>Identification and Lookup of Facets in a Locale</i>	311
7.2.1	Facet Identification	312
7.2.2	Facet Lookup	315
7.2.2.1	Retrieval of Facets from a Locale	315
7.2.2.2	Storing Facets in a Locale	319
7.2.2.3	The Rationale Behind the Use of the Two-Phase Polymorphism	320
7.3	<i>Memory Management of Facets in a Locale</i>	320
7.3.1	The Facet Reference Counter	321
7.3.2	Immutability of Facets in a Locale	326

8. User-Defined Facets	327
8.1 Adding a User-Defined Facet to an Existing Facet Family	327
8.2 Defining a New Facet Family	331

REFERENCE GUIDE

Introduction	343
1. Locale	346
header file <locale>	346
global functions	349
codecvt<internT,externT,stateT>	352
codecvt_base	358
codecvt_byname<internT,externT,stateT>	360
collate<charT>	362
collate_byname<charT>	365
ctype<charT>	367
ctype<char>	373
ctype_base	377
ctype_byname<charT>	379
locale	381
messages<charT>	387
messages_base	390
messages_byname<charT>	391
money_base	393
money_get<charT,InputIterator>	395
moneypunct<charT,Inter>	398
moneypunct_byname<charT,Inter>	403
money_put<charT,OutputIterator>	405
num_get<charT,InputIterator>	408
numpunct<charT>	414
numpunct_byname<charT>	417
num_put<charT,OutputIterator>	419
time_base	423
time_get<charT,InputIterator>	425
time_get_byname<charT,InputIterator>	430
time_put<charT,OutputIterator>	431
time_put_byname<charT,OutputIterator>	434

<i>time_base</i>	436
<i>tm</i>	438
2. Character Traits	440
<i>header file</i> <string>	440
<i>char_traits</i> <charT>	442
<i>char_traits</i> <char>	443
<i>char_traits</i> <wchar_t>	447
3. IOStreams	451
<i>header file</i> <iosfwd>	451
global type definitions	464
global objects	465
<i>basic_filebuf</i> <charT,traits>	467
<i>basic_fstream</i> <charT,traits>	471
<i>basic_ifstream</i> <charT,traits>	473
<i>basic_ios</i> <charT,traits>	475
<i>basic_iostream</i> <charT,traits>	480
<i>basic_istream</i> <charTtraits>	481
<i>basic_istreamstream</i> <charT,traits,Allocator>	492
<i>basic_ofstream</i> <charT,traits>	494
<i>basic_ostream</i> <charT,traits>	496
<i>basic_ostreamstream</i> <charT,traits,Allocator>	506
<i>basic_streambuf</i> <charT,traits>	508
<i>basic_stringbuf</i> <charT,traits,Allocator>	517
<i>basic_stringstream</i> <charT,traits,Allocator>	521
<i>fpos</i> <stateT>	523
<i>ios_base</i>	524
manipulators	538
4. Stream Iterators	540
<i>header file</i> <iterator>	540
<i>istreambuf_iterator</i> <charT,traits>	542
<i>istream_iterator</i> <T,charT,traits,Distance>	546
<i>iterator</i> <Category,T,Distance,Pointer,Reference>	549
iterator category tags	550
<i>ostreambuf_iterator</i> <charT,traits>	551
<i>ostream_iterator</i> <T,charT,traits,Distance>	553

5. Other I/O Operations	555
<i>bitset</i> < <i>N</i> >	555
<i>complex</i> < <i>T</i> >	557
<i>basic_string</i> < <i>charT</i> , <i>traits</i> , <i>Allocator</i> >	558

APPENDICES

Appendix A: Parsing and Extraction of Numerical and <code>bool</code> Values	563
A.1 Parsing Numerical Values	564
A.2 Parsing <code>bool</code> Values	566
A.3 Conversion Specifier and Length Modifier	566
Appendix B: Formatting of Numerical and <code>bool</code> Values	569
B.1 Formatting Numerical Values	570
B.2 Formatting <code>bool</code> Values	572
B.3 Conversion Specifiers, Qualifiers, and Length Modifiers	573
Appendix C: <code>strftime()</code> Conversion Specifiers	575
Appendix D: Correspondences Between C Stdio and C++ <code>IOStreams</code>	579
D.1 File Open Modes	579
D.2 Stream Positions	580
Appendix E: Differences Between Classic and Standard <code>IOStreams</code>	581
E.1 Templating the <code>IOStreams</code> Classes	582
E.2 Splitting Base Class <code>ios</code>	583
E.3 Indicating Errors	584
E.4 Internationalizing <code>IOStreams</code>	585
E.5 Removing <code>_withassign</code> Classes	585
E.6 Removing File Descriptors	586
E.7 String Streams: Replacing <code>strstream</code> by <code>stringstream</code>	587
E.8 Changes to the Stream Buffer Classes	588
E.9 Minor Changes	590
Appendix F: The Relationship Between C and C++ Locales	591
F.1 Locale Categories in C and C++	591
F.2 The Global Locale in C and C++	592
Appendix G: New C++ Features and Idioms	595
G.1 Bitmask Types	595
G.2 POD—Plain Old Data	596

<i>G.3 Explicit Constructors</i>	596
<i>G.4 Template Specialization</i>	599
<i>G.5 Default Template Arguments</i>	604
<i>G.6 Explicit Template Argument Specification</i>	607
<i>G.7 The Typename Keyword</i>	609
<i>G.8 Dynamic Cast</i>	612
<i>G.9 Function try Blocks</i>	616
<i>G.10 Standard Exceptions</i>	619
<i>G.11 Numeric Limits</i>	620
<i>G.12 C++ Strings</i>	620
Bibliography	623
Index	625