
Contents

<i>Preface</i>	xiii
Structure of This Book	xiv
Changes to the Third Edition	xviii
The Future of C++	xix
Acknowledgments	xx
Acknowledgments to the Second Edition	xx
Bibliography	xxi
Part I: C++, An Overview	1
<i>Chapter 1: Getting Started</i>	5
1.1: Problem Solving	5
1.2: The C++ Program	6
1.3: Preprocessor Directives	13
1.4: A Word About Comments	17
1.5: A First Look at Input/Output	19
<i>Chapter 2: A Tour of C++</i>	23
2.1: The Built-In Array Data Type	23
2.2: Dynamic Memory Allocation and Pointers	26
2.3: An Object-Based Design	30
2.4: An Object-Oriented Design	41
2.5: A Generic Design	51
2.6: An Exception-Based Design	58
2.7: An Array by Any Other Name	62
2.8: The Standard Array Is a Vector	68
Part II: The Basic Language	73
<i>Chapter 3: The C++ Data Types</i>	75
3.1: Literal Constant	75
3.2: Variables	79
3.3: Pointer Types	88
3.4: String Types	93
3.5: const Qualifier	102
3.6: Reference Types	105
3.7: The bool Type	110
3.8: Enumeration Types	111
3.9: Array Types	114

3.10:	The vector Container Type	121
3.11:	complex Number Types	125
3.12:	Typedef Names	126
3.13:	volatile Qualifier	127
3.14:	The pair Type	128
3.15:	Class Types	129
<i>Chapter 4: Expressions</i>		141
4.1:	What Is an Expression?	141
4.2:	Arithmetic Operators	143
4.3:	Equality, Relational, and Logical Operators	146
4.4:	Assignment Operators	149
4.5:	Increment and Decrement Operators	154
4.6:	Complex Number Operations	155
4.7:	The Conditional Operator	159
4.8:	The sizeof Operator	160
4.9:	The new and delete Expressions	162
4.10:	Comma Operator	164
4.11:	The Bitwise Operators	164
4.12:	bitset Operations	168
4.13:	Precedence	172
4.14:	Type Conversions	175
4.15:	A Stack Class Example	185
<i>Chapter 5: Statements</i>		189
5.1:	Simple and Compound Statements	189
5.2:	Declaration Statement	191
5.3:	The if Statement	194
5.4:	The switch Statement	202
5.5:	The for Loop Statement	210
5.6:	The while Statement	214
5.7:	The do while Statement	216
5.8:	The break Statement	218
5.9:	The continue Statement	220
5.10:	The goto Statement	220
5.11:	A Linked List Example	222
<i>Chapter 6: Abstract Container Types</i>		249
6.1:	Our Text Query System	250
6.2:	A vector or a list?	254
6.3:	How a vector Grows Itself	256
6.4:	Defining a Sequence Container	260
6.5:	Iterators	265

6.6:	Sequence Container Operations	269
6.7:	Storing Lines of Text	273
6.8:	Finding a Substring	276
6.9:	Handling Punctuation	282
6.10:	A String by Any Other Format	285
6.11:	Additional String Operations	288
6.12:	Building a Text Location Map	294
6.13:	Building a Word Exclusion Set	305
6.14:	The Complete Program	308
6.15:	Multimap and Multiset	318
6.16:	Stack	321
6.17:	Queue and Priority Queue	323
6.18:	Revisiting Our iStack Class	324
Part III:	Procedural-Based Programming	329
<i>Chapter 7:</i>	<i>Functions</i>	<i>331</i>
7.1:	Overview	331
7.2:	Function Prototype	334
7.3:	Argument Passing	338
7.4:	Returning a Value	356
7.5:	Recursion	361
7.6:	Inline Functions	363
7.7:	Linkage Directives: extern "C"	364
7.8:	main(): Handling Command Line Options	367
7.9:	Pointers to Functions	378
<i>Chapter 8:</i>	<i>Scope and Lifetime</i>	<i>389</i>
8.1:	Scope	389
8.2:	Global Objects and Functions	395
8.3:	Local Objects	402
8.4:	Dynamically Allocated Objects	405
8.5:	Namespace Definitions	420
8.6:	Using Namespace Members	434
<i>Chapter 9:</i>	<i>Overloaded Functions</i>	<i>443</i>
9.1:	Overloaded Function Declarations	443
9.2:	The Three Steps of Overload Resolution	456
9.3:	Argument Type Conversions	458
9.4:	Details of Function Overload Resolution	474
<i>Chapter 10:</i>	<i>Function Templates</i>	<i>489</i>
10.1:	Function Template Definition	489
10.2:	Function Template Instantiation	497
10.3:	Template Argument Deduction	500

10.4:	Explicit Template Arguments	505
10.5:	Template Compilation Models	509
10.6:	Template Explicit Specialization	514
10.7:	Overloading Function Templates	520
10.8:	Overload Resolution with Instantiations	522
10.9:	Name Resolution in Template Definitions	531
10.10:	Namespaces and Function Templates	538
10.11:	Function Template Example	542
<i>Chapter 11:</i>	<i>Exception Handling</i>	547
11.1:	Throwing an Exception	547
11.2:	The Try Block	551
11.3:	Catching an Exception	555
11.4:	Exception Specifications	564
11.5:	Exceptions and Design Issues	568
<i>Chapter 12:</i>	<i>The Generic Algorithms</i>	571
12.1:	Overview	571
12.2:	Using the Generic Algorithms	575
12.3:	Function Objects	586
12.4:	Revisiting Iterators	594
12.5:	The Generic Algorithms	603
12.6:	When Not to Use the Generic Algorithms	606
Part IV:	Object-Based Programming	611
<i>Chapter 13:</i>	<i>Classes</i>	613
13.1:	Class Definition	614
13.2:	Class Objects	621
13.3:	Class Member Functions	624
13.4:	The Implicit this Pointer	636
13.5:	Static Class Members	641
13.6:	Pointer to Class Member	649
13.7:	Union: A Space-Saving Class	658
13.8:	Bit-field: A Space-Saving Member	663
13.9:	Class Scope	665
13.10:	Nested Classes	672
13.11:	Classes as Namespace Members	683
13.12:	Local Classes	687
<i>Chapter 14:</i>	<i>Class Initialization, Assignment, and Destruction</i>	689
14.1:	Class Initialization	689
14.2:	The Class Constructor	691
14.3:	The Class Destructor	703
14.4:	Class Object Arrays and Vectors	709

14.5: The Member Initialization List	716
14.6: Memberwise Initialization	723
14.7: Memberwise Assignment	729
14.8: Efficiency Considerations	732
<i>Chapter 15: Overloaded Operators and User-Defined Conversions.</i>	737
15.1: Operator Overloading	737
15.2: Friends.	748
15.3: Operator =	751
15.4: Operator []	754
15.5: Operator ()	755
15.6: Operator ->	756
15.7: Operators ++ and --	759
15.8: Operators new and delete	763
15.9: User-Defined Conversions.	772
15.10: Selecting a Conversion	782
15.11: Overload Resolution and Member Functions	795
15.12: Overload Resolution and Operators	801
<i>Chapter 16: Class Templates</i>	811
16.1: Class Template Definition	812
16.2: Class Template Instantiation	820
16.3: Member Functions of Class Templates.	829
16.4: Friend Declarations in Class Templates.	833
16.5: Static Data Members of Class Templates	839
16.6: Nested Types of Class Templates	841
16.7: Member Templates	844
16.8: Class Templates and Compilation Model	849
16.9: Class Template Specializations	856
16.10: Class Template Partial Specializations	860
16.11: Name Resolution in Class Templates	862
16.12: Namespaces and Class Templates	865
16.13: A Template Array Class	867
Part V: Object-Oriented Programming.	877
<i>Chapter 17: Class Inheritance and Subtyping.</i>	879
17.1: Defining a Class Hierarchy	882
17.2: Identifying the Members of the Hierarchy	890
17.3: Base Class Member Access.	900
17.4: Base and Derived Class Construction	908
17.5: Base and Derived Class Virtual Functions.	919
17.6: Memberwise Initialization and Assignment	943

17.7: A UserQuery Manager Class.....	948
17.8: Putting It Together	958
<i>Chapter 18: Multiple and Virtual Inheritance</i>	<i>965</i>
18.1: Setting the Stage	965
18.2: Multiple Inheritance.....	970
18.3: Public, Private, and Protected Inheritance	977
18.4: Class Scope under Inheritance	985
18.5: Virtual Inheritance	993
18.6: A Multiple, Virtual Inheritance Example	1005
<i>Chapter 19: Uses of Inheritance in C++.....</i>	<i>1021</i>
19.1: Run-Time Type Identification	1021
19.2: Exceptions and Inheritance.....	1033
19.3: Overload Resolution and Inheritance	1051
<i>Chapter 20: The iostream Library</i>	<i>1063</i>
20.1: The Output Operator<<.....	1067
20.2: Input	1072
20.3: Additional Input/Output Operators.....	1083
20.4: Overloading the Output Operator <<	1090
20.5: Overloading the Input Operator >>.....	1095
20.6: File Input and Output	1097
20.7: Condition States	1107
20.8: String Streams	1109
20.9: Format State.....	1112
20.10: A Strongly Typed Library	1121
<i>Appendix: The Generic Algorithms Alphabetically.....</i>	<i>1123</i>
accumulate()	1125
adjacent_difference()	1126
adjacent_find()	1127
binary_search()	1128
copy()	1129
copy_backward()	1130
count()	1131
count_if()	1133
equal().....	1134
equal_range()	1136
fill()	1138
fill_n().....	1139
find().....	1140
find_if()	1141
find_end()	1143

<code>find_first_of()</code>	1144
<code>for_each()</code>	1145
<code>generate()</code>	1146
<code>generate_n()</code>	1147
<code>includes()</code>	1148
<code>inner_product()</code>	1149
<code>inplace_merge()</code>	1150
<code>iter_swap()</code>	1152
<code>lexicographical_compare()</code>	1153
<code>lower_bound()</code>	1154
<code>max()</code>	1156
<code>max_element()</code>	1156
<code>min()</code>	1156
<code>min_element()</code>	1157
<code>merge()</code>	1158
<code>mismatch()</code>	1159
<code>next_permutation()</code>	1161
<code>nth_element()</code>	1162
<code>partial_sort()</code>	1163
<code>partial_sort_copy()</code>	1164
<code>partial_sum()</code>	1165
<code>partition()</code>	1167
<code>prev_permutation()</code>	1168
<code>random_shuffle()</code>	1169
<code>remove()</code>	1170
<code>remove_copy()</code>	1170
<code>remove_if()</code>	1171
<code>remove_copy_if()</code>	1172
<code>replace()</code>	1173
<code>replace_copy()</code>	1173
<code>replace_if()</code>	1174
<code>replace_copy_if()</code>	1174
<code>reverse()</code>	1175
<code>reverse_copy()</code>	1176
<code>rotate()</code>	1177
<code>rotate_copy()</code>	1177
<code>search()</code>	1178
<code>search_n()</code>	1180
<code>set_difference()</code>	1181
<code>set_intersection()</code>	1181
<code>set_symmetric_difference()</code>	1182

set_union()	1182
sort()	1184
stable_partition()	1185
stable_sort()	1186
swap()	1187
swap_range()	1188
transform()	1189
unique()	1190
unique_copy()	1191
upper_bound()	1193
Heap Algorithms	1194
make_heap()	1194
pop_heap()	1195
push_heap()	1195
sort_heap()	1195
<i>Index</i>	1199

