



Object-Oriented Programming in C++

Nicolai M. Josuttis

Solutions in Time



JOHN WILEY & SONS, LTD

Contents

Preface	1
1 About this Book	3
1.1 Why Did I Write this Book?	3
1.2 Prerequisites	4
1.3 Organization of the Book	4
1.4 How Should You Read this Book?	6
1.5 Example Code and Additional Informations	6
1.6 Feedback	6
2 Introduction: C++ and Object-Oriented Programming	7
2.1 The C++ Language	7
2.1.1 Design Criteria	7
2.1.2 History of the Language	8
2.2 C++ as an Object-Oriented Programming Language	8
2.2.1 Objects, Classes, and Instances	9
2.2.2 Classes in C++	11
2.2.3 Data Encapsulation	13
2.2.4 Inheritance	15
2.2.5 Polymorphism	16
2.3 Other Concepts of C++	18
2.3.1 Exception Handling	18
2.3.2 Templates	18
2.3.3 Namespaces	20
2.4 Terminology	20
3 Basic Concepts of C++ Programs	23
3.1 The First Program	24

3.1.1	'Hello, World!'	24
3.1.2	Comments in C++	25
3.1.3	Function <code>main()</code>	26
3.1.4	Input and Output	27
3.1.5	Namespaces	28
3.1.6	Summary	29
3.2	Types, Operators, and Control Constructs	30
3.2.1	A First Program that Actually Calculates Something	30
3.2.2	Fundamental Types	33
3.2.3	Operators	36
3.2.4	Control Constructs	42
3.2.5	Summary	45
3.3	Functions and Modules	46
3.3.1	Header Files	46
3.3.2	Source Files with the Definitions	48
3.3.3	Source Files with the Calls	48
3.3.4	Compiling and Linking	49
3.3.5	Filename Endings	50
3.3.6	System Files and Libraries	51
3.3.7	The Preprocessor	51
3.3.8	Namespaces	54
3.3.9	The <code>static</code> Keyword	55
3.3.10	Summary	58
3.4	Strings	59
3.4.1	A First Simple Sample Program with Strings	59
3.4.2	Another Sample Program Using Strings	62
3.4.3	An Overview of String Operations	67
3.4.4	Strings and C-Strings	68
3.4.5	Summary	68
3.5	Collections	70
3.5.1	A Sample Program with Vectors	70
3.5.2	A Sample Program with Deques	72
3.5.3	Vectors versus Deques	73
3.5.4	Iterators	74
3.5.5	A Sample Program with Lists	76
3.5.6	Sample Programs with Associative Containers	78
3.5.7	Algorithms	82

3.5.8	Algorithms with Multiple Ranges	86
3.5.9	Stream Iterators	89
3.5.10	Endnotes	91
3.5.11	Summary	92
3.6	Exception Handling	93
3.6.1	Motivation for the Concept of Exception Handling	93
3.6.2	The Concept of Exception Handling	94
3.6.3	Standard Exception Classes	95
3.6.4	An Example of Exception Handling	96
3.6.5	Handling of Unexpected Exceptions	100
3.6.6	Auxiliary Functions for Exception Handling	101
3.6.7	Summary	102
3.7	Pointers, Arrays, and C-Strings	104
3.7.1	Pointers	104
3.7.2	Arrays	106
3.7.3	C-Strings	109
3.7.4	Summary	113
3.8	Memory Management Using <code>new</code> and <code>delete</code>	114
3.8.1	The <code>new</code> Operator	115
3.8.2	The <code>delete</code> Operator	115
3.8.3	Dynamic Memory Management for Arrays	116
3.8.4	Error Handling for <code>new</code>	118
3.8.5	Summary	119
3.9	Communication with the Outside World	120
3.9.1	Arguments from the Program Call	120
3.9.2	Access to Environment Variables	121
3.9.3	Aborting Programs	122
3.9.4	Calling Other Programs	123
3.9.5	Summary	123
4	Class Programming	125
4.1	The First Class: <code>Fraction</code>	126
4.1.1	Food for Thought Before Implementation	126
4.1.2	Declaration of the <code>Fraction</code> Class	129
4.1.3	The Class Structure	130
4.1.4	Member Functions	132
4.1.5	Constructors	133

4.1.6	Function Overloading	134
4.1.7	Implementation of the Fraction Class	136
4.1.8	Application of the Fraction Class	141
4.1.9	Creation of Temporary Objects	147
4.1.10	UML Notation	147
4.1.11	Summary	147
4.2	Operators for Classes	149
4.2.1	Declaring Operator Functions	149
4.2.2	Implementation of Operator Functions	152
4.2.3	Using Operator Functions	159
4.2.4	Global Operator Functions	161
4.2.5	Limitations in Defining Operators	162
4.2.6	Special Features of Certain Operators	163
4.2.7	Summary	166
4.3	Running Time and Code Optimization	168
4.3.1	The Fraction Class with Initial Optimizations	168
4.3.2	Default Arguments	171
4.3.3	Inline Functions	173
4.3.4	Optimizations from the User's Point of View	175
4.3.5	Using Statements	175
4.3.6	Declarations Between Statements	177
4.3.7	Copy Constructors	179
4.3.8	Summary	180
4.4	References and Constants	181
4.4.1	Copy Constructors and Argument Passing	181
4.4.2	References	182
4.4.3	Constants	185
4.4.4	Constant Member Functions	187
4.4.5	The Fraction Class with References	187
4.4.6	Pointers to Constants versus Pointer Constants	191
4.4.7	Summary	194
4.5	Input and Output Using Streams	195
4.5.1	Streams	195
4.5.2	Using Streams	196
4.5.3	Status of Streams	203
4.5.4	I/O Operators for User-Defined Types	204
4.5.5	Summary	216

4.6	Friends and Other Types	217
4.6.1	Automatic Type Conversions	217
4.6.2	The <code>explicit</code> Keyword	219
4.6.3	Friend Functions	220
4.6.4	Conversion Functions	227
4.6.5	Problems with Automatic Type Conversions	229
4.6.6	Other Uses of the <code>friend</code> Keyword	231
4.6.7	<code>friend</code> versus Object-Oriented Programming	231
4.6.8	Summary	232
4.7	Exception Handling for Classes	234
4.7.1	Motivation for Exception Handling in the <code>Fraction</code> Class	234
4.7.2	Exception Handling for the <code>Fraction</code> Class	235
4.7.3	Exception Classes	243
4.7.4	Rethrowing Exceptions	244
4.7.5	Exceptions in Destructors	245
4.7.6	Exceptions in Interface Declarations	245
4.7.7	Hierarchies of Exception Classes	246
4.7.8	Design of Exception Classes	250
4.7.9	Throwing Standard Exceptions	252
4.7.10	Exception Safety	252
4.7.11	Summary	253
5	Inheritance and Polymorphism	255
5.1	Single Inheritance	258
5.1.1	The <code>Fraction</code> Class as a Base Class	258
5.1.2	The Derived Class <code>RFraction</code>	261
5.1.3	Declaration of the Derived Class <code>RFraction</code>	262
5.1.4	Inheritance and Constructors	265
5.1.5	Implementation of Derived Classes	268
5.1.6	Application of Derived Classes	271
5.1.7	Constructors for Base-Class Objects	273
5.1.8	Summary	274
5.2	Virtual Functions	276
5.2.1	Problems with Overriding Functions	276
5.2.2	Static and Dynamic Binding of Functions	279
5.2.3	Overloading versus Overriding	284
5.2.4	Access to Parameters of the Base Class	285

5.2.5	Virtual Destructors	286
5.2.6	Using Inheritance Correctly	287
5.2.7	Additional Pitfalls when Overriding Functions	293
5.2.8	Private Inheritance and Pure Access Declarations	295
5.2.9	Summary	299
5.3	Polymorphism	300
5.3.1	What is Polymorphism?	300
5.3.2	Polymorphism in C++	301
5.3.3	An Example of Polymorphism in C++	302
5.3.4	The Abstract Base Class <code>GeoObj</code>	306
5.3.5	Application of Polymorphism Inside Classes	315
5.3.6	Polymorphism is not a Selection	321
5.3.7	Reconversion of an Object into its Actual Class	322
5.3.8	Design by Contract	326
5.3.9	Summary	327
5.4	Multiple Inheritance	329
5.4.1	An Example of Multiple Inheritance	329
5.4.2	Virtual Base Classes	335
5.4.3	Identity and Addresses	339
5.4.4	Multiple Derivation of the Same Base Class	342
5.4.5	Summary	343
5.5	Design Pitfalls of Inheritance	344
5.5.1	Inheritance versus Containment	344
5.5.2	Design Error: Limiting Inheritance	344
5.5.3	Design Error: Value-Changing Inheritance	346
5.5.4	Design Error: Value-Interpreting Inheritance	347
5.5.5	'Avoid Inheritance!'	348
5.5.6	Summary	349
6	Dynamic and Static Members	351
6.1	Dynamic Members	352
6.1.1	Implementing a <code>String</code> Class	352
6.1.2	Constructors and Dynamic Members	358
6.1.3	Implementing a Copy Constructor	360
6.1.4	Destructors	361
6.1.5	Implementing the Assignment Operator	362
6.1.6	Other Operators	363

6.1.7	Reading a String	366
6.1.8	Commercial Implementations of String Classes	368
6.1.9	Other Uses of Dynamic Members	370
6.1.10	Summary	372
6.2	Other Aspects of Dynamic Members	373
6.2.1	Dynamic Members with Constant Objects	373
6.2.2	Conversion Functions for Dynamic Members	376
6.2.3	Conversion Functions for Conditions	378
6.2.4	Constants Become Variables	381
6.2.5	Preventing Predefined Functions	383
6.2.6	Proxy Classes	384
6.2.7	Exception Handling Using Parameters	387
6.2.8	Summary	392
6.3	Inheritance of Classes with Dynamic Members	393
6.3.1	The <code>CPPBook::String</code> Class as a Base Class	393
6.3.2	The Derived <code>ColString</code> Class	396
6.3.3	Deriving Friend Functions	399
6.3.4	Dot-C File of the Derived <code>ColString</code> Class	402
6.3.5	Application of the <code>ColString</code> Class	403
6.3.6	Deriving the Special Functions for Dynamic Members	404
6.3.7	Summary	405
6.4	Classes Containing Classes	406
6.4.1	Objects as Members of Other Classes	406
6.4.2	Implementing the <code>Person</code> Class	406
6.4.3	Summary	413
6.5	Static Members and Auxiliary Types	414
6.5.1	Static Class Members	414
6.5.2	Type Declarations Within Classes	420
6.5.3	Enumeration Types as Static Class Constants	423
6.5.4	Nested and Local Classes	424
6.5.5	Summary	425
7	Templates	427
7.1	Why Templates?	428
7.1.1	Terminology	428
7.2	Function Templates	429
7.2.1	Defining Function Templates	429

7.2.2	Calling Function Templates	430
7.2.3	Practical Hints for Working with Templates	431
7.2.4	Automatic Type Conversions with Templates	431
7.2.5	Overloading Templates	432
7.2.6	Local Variables	435
7.2.7	Summary	435
7.3	Class Templates	436
7.3.1	Implementation of the Class Template Stack	436
7.3.2	Application of the Class Template Stack	440
7.3.3	Specialization of Class Templates	441
7.3.4	Default Template Parameters	444
7.3.5	Summary	447
7.4	Non-Type Template Parameters	448
7.4.1	Example of Using Non-Type Template Parameters	448
7.4.2	Limitations of Non-Type Template Parameters	451
7.4.3	Summary	452
7.5	Additional Aspects of Templates	453
7.5.1	The typename Keyword	453
7.5.2	Members as Templates	454
7.5.3	Static Polymorphism with Templates	457
7.5.4	Summary	461
7.6	Templates in Practice	462
7.6.1	Compiling Template Code	462
7.6.2	Error Handling	468
7.6.3	Summary	469
8	The Standard I/O Library in Detail	471
8.1	The Standard Stream Classes	472
8.1.1	Stream Classes and Stream Objects	472
8.1.2	Handling the Stream Status	474
8.1.3	Standard Operators	477
8.1.4	Standard Functions	478
8.1.5	Manipulators	481
8.1.6	Format Definitions	483
8.1.7	Internationalization	494
8.1.8	Summary	496

8.2	File Access	498
8.2.1	Stream Classes for Files	498
8.2.2	Application of the Stream Classes for Files	499
8.2.3	File Flags	501
8.2.4	Explicitly Opening and Closing Files	502
8.2.5	Random File Access	504
8.2.6	Redirecting the Standard Channels into Files	506
8.2.7	Summary	508
8.3	Stream Classes for Strings	509
8.3.1	String Stream Classes	509
8.3.2	Lexical Cast Operator	512
8.3.3	char* Stream Classes	514
8.3.4	Summary	516
9	Other Language Features and Details	517
9.1	Additional Details of the Standard Library	518
9.1.1	Vector Operations	518
9.1.2	Common Operations of all STL Containers	524
9.1.3	List of all STL Algorithms	526
9.1.4	Numeric Limits	531
9.1.5	Summary	535
9.2	Defining Special Operators	536
9.2.1	Smart Pointers	536
9.2.2	Function Objects	540
9.2.3	Summary	545
9.3	Additional Aspects of new and delete	546
9.3.1	No-Throw Versions of new and delete	546
9.3.2	Placement New	546
9.3.3	New Handlers	547
9.3.4	Overloading new and delete	552
9.3.5	The new Operator with Additional Parameters	556
9.3.6	Summary	556
9.4	Function Pointers and Member Pointers	557
9.4.1	Function Pointers	557
9.4.2	Pointers to Members	558
9.4.3	Pointers to Members for External Interfaces	561
9.4.4	Summary	563

9.5	Combining C++ Code with C Code	564
9.5.1	External Linkage	564
9.5.2	Header Files for C and C++	565
9.5.3	Compiling main()	565
9.5.4	Summary	565
9.6	Additional Keywords	566
9.6.1	Unions	566
9.6.2	Enumeration Types with enum	566
9.6.3	The <code>volatile</code> Keyword	568
9.6.4	Summary	568
10	Summary	569
10.1	Hierarchy of C++ Operators	569
10.2	Class-Specific Properties of Operations	572
10.3	Rules for Automatic Type Conversion	573
10.4	Useful Programming Guidelines and Conventions	574
	Bibliography	577
	Glossary	581
	Index	587