

Contents

List of Programs xxvii

1 Overview 1

1.1 Why C++? 1

1.2 C++’s History 3

1.3 C++’s Future 4

1.4 There are Other OOP Languages 5

1.5 Programming Paradigms 6

1.6 Procedural and Modular Programming 6

1.7 Data Abstraction 8

1.8 Object-Oriented Programming 9

1.8.1 Abstraction 14

1.8.2 Encapsulation 15

1.8.3 Modularity 15

1.8.4 Inheritance 15

1.8.5 Polymorphism 15

1.8.6 Typing 16

1.8.7 Automatic Memory Management 17

1.8.8 Concurrency 18

1.8.9 Persistence 18

1.8.10 Operator Overloading 18

1.8.11 The Unified Modeling Language 18

1.9 Objects 19

1.10 Syntax, Semantics and Pragmatics 19

1.10.1 Syntax 19

1.10.2 Semantics 20

1.10.3 Pragmatics 20

1.11 Obtaining a C++ Compiler 20

1.12 Applications of C++ 20

1.13 References for C++ 20

1.14 References for Graphics 22

1.15 Notation 22

1.16 Summary 23

Exercises 23

2 The Development Environment 25

2.1 Hardware, Software and Setup Used 25

2.1.1 Hardware 25

2.1.2 Software 25

2.2 The Borland Integrated Development Environment 26

2.2.1 Editor 26

2.2.2 Text Highlighting 27

2.2.3 Project Manager 27

2.2.4 Messages 27

2.2.5 Browser 27

2.2.6	AppExpert	27
2.2.7	ClassExpert	28
2.2.8	Debugger	28
2.2.9	Resource Workshop	28
2.2.10	Help	28
2.2.11	Java Development Tools	28
2.3	Container Class and ObjectWindows Libraries	29
2.4	Summary	30
	Exercises	30
3	Getting Started	31
3.1	The Simplest of Programs	31
3.2	The Comment Line	32
3.3	Preprocessor Directives	34
3.4	What is IOSTREAM.H?	35
3.5	" " or <>	35
3.5.1	Other Header Files	36
3.6	Keywords	36
3.7	The <i>main()</i> Function	37
3.8	Program Statements	38
3.8.1	The <code>cout</code> Identifier/Object	39
3.8.2	The << Insertion Operator	39
3.8.3	String Constants	39
3.8.4	The <code>endl</code> Manipulator	40
3.9	The Standard Library Namespace	40
3.10	Compiling and Linking	41
3.10.1	Compiling	41
3.10.2	Linking	41
3.10.3	Note for GUI Compilers	41
3.11	Make-ing Executables	43
3.12	A Note on Programming Style and Notation	46
3.12.1	Be Consistent!	48
3.13	Summary	49
	Exercises	49
4	Fundamental Data Types, Declarations, Definitions and Expressions	51
4.1	Fundamental Data Types	51
4.1.1	The char Data Type	52
4.1.2	The int Data Type	57
4.1.3	The float Data Type	58
4.1.4	The double Data Type	59
4.1.5	The wchar_t Type	60
4.1.6	The enum Type	61
4.1.7	The bool Type	62
4.2	The short , long , unsigned , signed , const and volatile Data Type Modifiers	64
4.2.1	The short Modifier	65
4.2.2	The long Modifier	65
4.2.3	The unsigned Modifier	65
4.2.4	The signed Modifier	65
4.2.5	The const Modifier	65
4.2.6	The volatile Modifier	66

4.2.7	Choosing a Data Type	66
4.3	Constants	68
4.3.1	Integer Constants	68
4.3.2	Floating-Point Constants	69
4.3.3	Character and String Constants	69
4.3.4	Enumeration Constants	70
4.4	const or #define ?	70
4.5	The sizeof Operator	70
4.6	Keywords	71
4.7	Declaration or Definition?	71
4.8	Assignment	72
4.9	Initialisation	72
4.10	Expressions	73
4.11	Operators	74
4.11.1	Arithmetic Operators	74
4.11.2	Chaining Operators	74
4.11.3	Operator Shortcuts	75
4.11.4	The Increment (++) and Decrement (--) Operators	75
4.11.5	Operator Precedence, Associativity and Arity	76
4.12	Manipulators	77
4.12.1	The endl Manipulator	79
4.12.2	The flush Manipulator	79
4.12.3	The setw() Manipulator	79
4.12.4	The setprecision() Manipulator	80
4.12.5	The setiosflags() Manipulator	80
4.12.6	The resetiosflags() Manipulator	80
4.12.7	The dec , hex and oct Manipulators	80
4.13	Basic File Input and Output	81
4.14	Type Conversion and Casting	83
4.15	The Storage Class Specifiers auto , extern , register and static	84
4.15.1	The auto Specifier	84
4.15.2	The extern Specifier	85
4.15.3	The register Specifier	86
4.15.4	The static Specifier	87
4.16	The asm Declaration	88
4.17	Summary	89
	Exercises	89
5	Making Decisions and Repetition	91
5.1	Decisions	91
5.1.1	The if Statement	92
5.1.2	The if-else Statement	93
5.1.3	Nested ifs	98
5.1.4	Relational Operators	100
5.1.5	Logical Operators	101
5.1.6	The switch Statement	105
5.1.7	The Conditional Operator (?:)	108
5.2	Repetition	109
5.2.1	The for -Loop	109
5.2.2	The while -Loop	113
5.2.3	The do-while Loop	114
5.3	Forever Loops	115

5.4	Nested Loops	116
5.5	The break Statement	116
5.6	The continue Statement	119
5.7	The goto Statement	120
5.8	Loop Body Visibility	121
5.9	The Bitwise and Shift Operators	121
5.9.1	Bitwise AND Operator (&)	121
5.9.2	Bitwise OR Operator ()	123
5.9.3	Bitwise Exclusive OR Operator (^)	124
5.9.4	Bitwise One's Complement Operator (~)	125
5.9.5	The Shift Operators	126
5.9.6	The Bitwise and Shift Assignment Operators	127
5.10	Summary	127
	Exercises	128
6	Functions	131
6.1	A Look at Functions	132
6.2	The Body of a Function	132
6.2.1	Function Name	133
6.2.2	Function Body	133
6.2.3	Function Return Type	134
6.2.3.1	void Return Type	134
6.2.3.2	Default Return Type	135
6.3	The return Statement	135
6.4	Function Calling	136
6.5	Function Definition	137
6.6	Function Declaration	138
6.7	Function Scope	139
6.7.1	Local Scope	139
6.7.2	Global Scope	140
6.7.3	Precedence of Scope	141
6.7.4	Pros and Cons of Local and Global Scope	141
6.7.5	Scope Resolution Operator	142
6.8	Function Arguments	142
6.8.1	No Argument List	143
6.8.2	By Value or by Reference?	143
6.8.3	const Reference Arguments	146
6.8.4	const Return Value	146
6.9	Returning More than One Value	147
6.10	Return by Reference	148
6.11	Why Use Functions?	149
6.12	Overloaded Functions	156
6.13	Default Arguments	161
6.14	Local static Variables	162
6.15	inline Functions	163
6.16	The <i>main()</i> Function	165
6.16.1	Command Line Arguments	165
6.16.2	<i>WinMain()</i>	166
6.17	External Linkage	166
6.18	Library Functions	168
6.18.1	The ANSI C Standard Library	169
6.19	Header Files	169
6.19.1	ANSI C Header Files	169

6.19.2	C++ Header Files	173
6.19.3	Additional Header Files	174
6.20	Not in the Library	175
6.21	Intersection of Two Two-Dimensional Line Segments	175
6.21.1	Parametric Lines	176
6.21.2	Bounding Boxes and Parametric Intersection	179
6.21.3	Clockwise or Anticlockwise?	181
6.22	Summary	187
	Exercises	188
7	Arrays	191
7.1	An Array	191
7.2	The Size of an Array	193
7.3	The sizeof Operator	194
7.4	Array Indexing	195
7.4.1	First Element	195
7.4.2	Setting and Getting an Element	195
7.4.3	Last Element	196
7.5	Initialisation of One-Dimensional Arrays	197
7.6	Two-Dimensional Arrays	198
7.7	Initialisation of Two-Dimensional Arrays	200
7.8	How are Two-Dimensional Arrays Used?	201
7.9	Three- and Higher-Dimensional Arrays	202
7.10	Strings	203
7.10.1	Variable Strings	203
7.10.2	Variable Strings with Embedded Spaces	204
7.10.3	Constant Strings	206
7.11	Wasted Space	207
7.12	Arrays as Function Arguments	208
7.12.1	One-Dimensional Arrays	208
7.12.2	Two-Dimensional Arrays	210
7.13	Calling Functions with Arrays as Arguments	212
7.14	Strings as Function Arguments	212
7.15	Passing the Size of an Array as an Argument	214
7.16	Small, Medium, Compact, Large and Huge Memory Models	217
7.16.1	Small and Medium Memory Models	217
7.16.2	Compact, Large and Huge Memory Models	218
7.16.3	Libraries	218
7.16.4	Windows Programming	218
7.17	Summary	218
	Exercises	219
8	Structures, Unions, Enumerations and Typedefs	221
8.1	Structures	221
8.1.1	Declaring a Structure	223
8.1.2	Scope of a Structure	224
8.1.3	Structure Tag or Name	224
8.1.4	Defining a Structure Variable or Object	224
8.1.5	Accessing a Structure's Data Members	225
8.1.6	The Size of a Structure	225
8.1.7	Declaring and Defining a Structure in a Single Statement	226

8.1.8	Defining a Structure Variable or Object Without a Structure Name	227
8.1.9	Output of a Structure's Data Members	227
8.1.10	Assignment	227
8.1.11	Arrays of Structures	229
8.1.12	Structures as Function Arguments	229
8.1.13	Nested Structures	231
8.1.14	A Point Structure	233
8.1.15	Data and Functions	235
8.1.16	Private or Public?	237
8.1.17	Point, Line and Triangle Structures	237
8.1.18	Bit Fields	240
8.2	Unions	242
8.2.1	sizeof a union	243
8.3	Enumerations	244
8.4	typedefs	247
8.4.1	typedefs for Windows	248
8.4.2	typedefs Offer Little Type Safety	249
8.4.3	typedef Style	250
8.5	Summary	250
	Exercises	251
9	The C++ Class	253
9.1	A Point class	253
9.2	class Declaration Syntax	257
9.3	Objects and Instances	258
9.4	A public Point class	258
9.5	Member Functions	260
9.6	Naming of Member Functions	264
9.7	Calling Member Functions	264
9.8	Defining Member Functions	264
9.9	static Data Members and Member Functions	268
9.10	const and mutable Data Members	270
9.11	volatile Data Members	271
9.12	Bit Fields	272
9.13	Constructors	272
9.13.1	Constructors are Called Automatically	274
9.13.2	Default Initialisation	275
9.13.3	Overloaded Constructors	276
9.13.4	Default Constructors	277
9.13.5	Data Member Initialisation	278
9.13.6	Copy Constructor	280
9.13.7	Constructors, const and volatile	285
9.14	Destructors	285
9.14.1	Destructors, const and volatile	287
9.15	inline Member Functions	288
9.15.1	Automatic inline Member Functions	289
9.15.2	inline Constructors	290
9.16	const Member Functions	290
9.17	volatile Member Functions	292
9.18	Default Member Function Arguments	293
9.19	Functions with Object Arguments and Functions that Return Objects	295

9.19.1	Returning by Reference	298
9.20	Combining Constructor and Member Function Calls	299
9.21	Arrays of Objects	301
9.22	Local Classes	302
9.23	Nested Classes	303
9.24	Another class Declaration	305
9.25	struct and class	306
9.26	Intersection of Two Two-Dimensional Line Segments	306
9.27	Summary	309
	Exercises	311
10	Operators and Overloading	315
10.1	Overloading the Arithmetic Binary Operators	315
10.1.1	The operator Keyword	317
10.2	The Unary Increment (++) and Decrement (--) Operators	321
10.3	Physical Meaning of Overloaded Point Operators	323
10.4	The Relational and Logical Operators	324
10.5	The Assignment Operator	325
10.5.1	Default Assignment	325
10.5.2	Overloaded Assignment Operator	326
10.6	Non-Member Overloaded Operator Functions	327
10.7	The Array Subscript Operator	328
10.8	Composite Operators	330
10.9	Conversions	332
10.10	A String class	335
10.11	Overloading the C++ Input Extraction (>>) and Output Insertion (<<) Operators	338
10.12	Operators Which Cannot be Overloaded	343
10.13	Restrictions Attached to Operator Overloading	344
10.14	Choose Carefully Which Operators to Overload	344
10.15	Summary	345
	Exercises	345
11	Friends	347
11.1	friend Functions	347
11.1.1	So Why Use friend Functions?	349
11.1.2	friend Functions: Good, Bad or Ugly?	352
11.2	friend Classes	352
11.2.1	Vectors	356
11.3	Friends and Overloaded Operators	357
11.3.1	Complex Numbers	358
11.3.2	Complex class	359
11.3.3	The Use of Conversion Functions	361
11.4	Overloading the C++ Input Extraction (>>) and Output Insertion (<<) Operators	364
11.5	Summary	366
	Exercises	366
12	Pointers	369
12.1	A First Look at Pointers	369
12.2	The Address-of Operator	371
12.3	Multiple Pointer Definitions in a Single Statement	373
12.4	A Note on Pointer Definitions	374

12.5	Another Look at Pointers	374
12.6	Valid Pointers	377
12.7	The <code>NULL</code> Pointer	377
12.8	Don't Mix Types	378
12.9	<code>void</code> Pointers	379
12.10	<code>const</code> Pointers	382
12.11	Pointers and Function Arguments	383
12.12	Pointer Arithmetic	385
12.12.1	Pointer Addition	385
12.12.2	Pointer Subtraction	386
12.12.3	Increment and Decrement Operators and Pointers	387
12.13	Pointers and Relational Operators	389
12.14	Pointers are Quicker	389
12.15	Pointers and Arrays	390
12.15.1	One-Dimensional Arrays	390
12.15.2	Two-Dimensional Arrays	392
12.16	Pointers, Arrays and Function Arguments	393
12.17	Pointers to Pointers	395
12.18	Arrays of Pointers and Pointers to Arrays	397
12.19	Pointers to String Constants	400
12.20	Pointers to Functions	400
12.21	Returning a Pointer From a Function	406
12.22	Casting	407
12.23	Recap	410
12.24	Pointers to Objects	410
12.25	Pointers to Data Members and Member Functions	413
12.26	The <code>new</code> and <code>delete</code> Operators	414
12.26.1	Objects	414
12.26.2	Arrays	421
12.26.3	Objects, Arrays and <code>delete</code>	426
12.26.4	Data Member Initialisation Lists and <code>new</code>	426
12.26.5	Overloaded <code>new</code> and <code>delete</code> Operators	427
12.26.6	Placement	435
12.27	Returning Local Objects from Member Functions	436
12.28	Passing Objects to Functions	440
12.29	The <code>this</code> Pointer	442
12.29.1	Further Illustration of <code>this</code>	447
12.29.2	Restrictions on <code>this</code>	448
12.30	A <code>Matrix</code> <code>class</code>	448
12.30.1	A Subscript Operator for <code>class Matrix</code>	460
12.30.2	Gaussian Elimination of a System of Linear Equations	461
12.30.3	Return Value of <code>GaussElimination()</code>	465
12.31	Summary	466
	Exercises	467
13	Templates	469
13.1	Function Overloading	469
13.2	Template Functions	471
13.2.1	A Note on Style	473
13.2.2	Templates or Macros?	474
13.2.3	Template Function Arguments	474
13.2.4	Multiple Type Arguments	474

13.2.5	Template Functions do not Perform Implicit Casting	475
13.2.6	Overloading Template Functions	476
13.3	Template Classes	477
13.3.1	Declaring Template Classes in a Header File	483
13.3.2	friend Functions	484
13.3.3	Multiple Type and Non-Type Arguments	485
13.3.4	Specialisations	487
13.3.5	Combined Use of Template Functions and Classes	490
13.3.6	Nested Classes	491
13.3.7	The typename Keyword	492
13.4	Smart Pointers	493
13.5	A GlobalMemory Template class for Windows	495
13.6	Vector and Matrix Template Classes	497
13.6.1	Associative Arrays	500
13.7	Iterators	502
13.7.1	Iterator Functions	502
13.7.2	Iterator Classes	506
13.8	A LinkedList class	508
13.8.1	A LinkedListIterator class	517
13.9	A Polygon class	518
13.9.1	A class Representation of a Polygon	518
13.9.2	A Point Inside or Outside a Polygon	522
13.9.3	Convex and Concave Polygons, Sorting Vertices and Convex Hulls	524
13.10	Summary	533
	Exercises	534
14	Exception Handling	537
14.1	try , catch and throw	537
14.2	Multiple catch Statements	542
14.3	Handling all Exceptions	543
14.4	Functions and Exception Handling	544
14.4.1	Exception Specification	546
14.5	Re-Throwing an Exception	547
14.6	The EXCEPT.H Header File	549
14.6.1	The <i>terminate()</i> Function	549
14.6.2	The <i>set_terminate()</i> Function	549
14.6.3	The <i>unexpected()</i> Function	551
14.6.4	The <i>set_unexpected()</i> Function	551
14.6.5	The <i>xmsg</i> Class	552
14.7	The EXCEPTION.H and STDEXCEPT.H Header Files	553
14.8	Constructors, Copy Constructors and Destructors	554
14.9	Exception Handling and a Vector class	556
14.9.1	Exception Classes	560
14.9.2	Constructors	560
14.9.3	<i>Assert()</i> template Function	561
14.9.4	Declaring Exceptions	562
14.10	Passing Information with an Exception	563
14.10.1	Range and Index Classes	565
14.11	Exception Handling and the new Operator	570
14.12	Summary	574
	Exercises	574

15	Inheritance	577
15.1	Base and Derived Classes	578
15.2	Inheritance by Declaration	580
15.3	Access Specifiers	581
15.3.1	Default Specifier	582
15.3.2	Access Declaration	582
15.4	Properties of Base and Derived Classes	584
15.5	Deriving from a Derived class	585
15.6	protected Data Members	585
15.7	protected Access Specifier	587
15.8	Base and Derived class Constructors and Destructors	588
15.8.1	No Derived class Constructors or Destructor	588
15.8.2	Derived class Constructors and Destructor	589
15.9	Copy Constructors, Overloaded Assignment Operator and Inheritance	594
15.9.1	Copy Constructors	594
15.9.2	Overloaded Assignment Operator	596
15.10	new and delete Operators and Inheritance	601
15.11	Overriding class Member Functions	602
15.12	Overloaded Operator Functions and Inheritance	605
15.13	Friendship and Inheritance	605
15.14	Increased Functionality and Code Reusability	607
15.15	A Vector class for Three-dimensional Space	611
15.16	Containment Versus Inheritance	616
15.16.1	Containment	619
15.16.2	Inheritance	628
15.16.3	Containment or Inheritance?	629
15.17	Multiple Inheritance	630
15.17.1	Constructors, Destructors and Multiple Inheritance	632
15.18	virtual Base Classes	635
15.19	virtual Functions	637
15.19.1	Pure virtual Functions and Abstract Classes	644
15.19.2	An Application	648
15.20	Templates and Inheritance	664
15.20.1	A SortedVector class	668
15.20.2	Memory, LocalMemory and GlobalMemory Classes for Windows	672
15.20.3	The Matrix class Revisited	676
15.20.4	Stack and Queue Classes	679
15.20.5	VectorIterator and LinkedListIterator Classes	681
15.21	Exception Handling and Inheritance	684
15.21.1	Exception Specification	687
15.21.2	Re-Throwing an Exception	688
15.22	Summary	688
	Exercises	689
16	Run-Time Type Information and Casting	693
16.1	Recap	693
16.2	Use of virtual Functions	695
16.2.1	A Direct Method	695
16.2.2	The Double-Dispatch Method	697

16.3	The RTTI Approach	699
16.3.1	The <code>dynamic_cast<>()</code> Operator	699
16.3.2	The <code>typeid()</code> Operator	702
16.3.3	The <code>TYPEINFO.H</code> Header File and the <code>type_info</code> , <code>bad_cast</code> and <code>bad_typeid</code> Classes	703
16.4	Casting	706
16.4.1	The <code>static_cast<>()</code> Operator	706
16.4.2	The <code>reinterpret_cast<>()</code> Operator	708
16.4.3	The <code>const_cast<>()</code> Operator	709
16.5	Summary	710
	Exercises	710
17	Input and Output, Files and Streams	713
17.1	Why Wait Until Now?	714
17.2	Streams and Stream Classes	714
17.2.1	Character Traits	714
17.2.2	The <code>ios_base</code> Base class	716
17.2.3	The <code>basic_streambuf</code> Base class	716
17.2.4	Stream Classes Derived from <code>basic_ios</code> and <code>basic_streambuf</code>	716
17.2.5	Header Files	718
17.2.6	Predefined Streams	719
17.3	Standard Stream Output	720
17.4	Standard Stream Input	721
17.5	Overloading the Insertion and Extraction Operators	723
17.6	Formatted Input and Output	726
17.6.1	Formatting via Manipulators	726
17.6.2	Formatting via Member Functions of class <code>ios_base</code>	732
17.7	Locale	736
17.8	Working with Disk Files	737
17.8.1	File Input Using class <code>basic_ifstream</code>	737
17.8.2	File Output Using class <code>basic_ofstream</code>	741
17.8.3	File Input and Output Using class <code>basic_fstream</code>	743
17.8.4	Stream Opening mode and access	744
17.8.5	Stream Status Flags	745
17.8.6	The <code>open()</code> and <code>close()</code> Member Functions	747
17.8.7	The <code>read()</code> and <code>write()</code> Member Functions	750
17.8.8	Random File Access	752
17.8.9	Objects and Files	756
17.9	<code>ReadFile</code> , <code>WriteFile</code> and <code>ReadAndWriteFile</code> Classes	759
17.10	Reading a Collection of Objects from a Disk File	767
17.11	String Streams	783
17.12	Console Streams	785
17.13	Redirection	787
17.13.1	Redirecting Output from the Display Screen to a Disk File	788
17.13.2	Redirecting Input from the Keyboard to a Disk File	789
17.13.3	Redirecting both Input and Output	789
17.14	Command Line Arguments	789
17.15	The C Approach to Streams	793

17.15.1	The <code>STDIO.H</code> Header File	793
17.15.2	Predefined Streams	793
17.15.3	Output to the Display Screen Using the <code>printf()</code> Function	794
17.15.4	Input from the Keyboard Using the <code>scanf()</code> Function	798
17.15.5	The <code>vprintf()</code> , <code>vscanf()</code> , <code>vsprintf()</code> and <code>vsscanf()</code> Functions	805
17.15.6	File Input and Output	805
17.15.7	The <code>FILE</code> Structure	805
17.15.8	The <code>fopen()</code> Function	806
17.15.9	The <code>fclose()</code> Function	806
17.15.10	The <code>fgetc()</code> Function	807
17.15.11	Reading a Disk File	807
17.15.12	The <code>fgets()</code> Function	809
17.15.13	The <code>fputc()</code> Function	810
17.15.14	Writing to a Disk File	810
17.15.15	The <code>fputs()</code> Function	811
17.15.16	The <code>fread()</code> and <code>fwrite()</code> Functions	812
17.15.17	The <code>fprintf()</code> and <code>fscanf()</code> Functions	814
17.15.18	The <code>vfprintf()</code> and <code>vfscanf()</code> Functions	814
17.15.19	Random File Access	814
17.15.20	The <code>fseek()</code> , <code>ftell()</code> , <code>fgetpos()</code> and <code>fsetpos()</code> Functions	815
17.15.21	The <code>feof()</code> , <code>ferror()</code> , <code>fflush()</code> , <code>flushall()</code> , <code>freopen()</code> , <code>clearerr()</code> , <code>remove()</code> , <code>rename()</code> and <code>rewind()</code> Functions	817
17.16	A Device-Independent Bitmap class for Windows	818
17.16.1	The Windows Device-Independent Bitmap File Format	819
17.16.2	The <code>DIBitmap</code> class	821
17.17	Summary	838
	Exercises	839
18	The Preprocessor	841
18.1	The Preprocessor	841
18.2	Preprocessor Directives	841
18.2.1	The <code>#</code> Null Directive	842
18.2.2	The <code>#define</code> Directive	842
18.2.3	The <code>#error</code> Directive	845
18.2.4	The <code>#include</code> Directive	846
18.2.5	The <code>#line</code> Directive	846
18.2.6	The <code>#pragma</code> Directive	847
18.2.7	The <code>#undef</code> Directive	847
18.2.8	The <code>#elif</code> , <code>#else</code> , <code>#endif</code> , <code>#if</code> , <code>#ifdef</code> and <code>#ifndef</code> Conditional Directives	848
18.3	The <code>#</code> and <code>##</code> Operators	850
18.4	Predefined Macros	851
18.5	Summary	853
	Exercises	853
19	Namespaces	855
19.1	Namespaces – What are They and Why do we Need Them?	855

19.2	The namespace Declaration	856
19.3	The using Declaration	860
19.4	The using Directive	861
19.5	The Nameless namespace Declaration	863
19.6	Global Scope	863
19.7	Overloading Namespaces	864
19.8	Namespaces and Inheritance	865
19.9	The Standard Library Namespace	868
19.10	Java and Namespaces	869
19.11	Summary	869
	Exercises	870
20	The Standard Template Library	873
20.1	Overview of the Standard Template Library	874
20.2	The STL namespace	874
20.3	Global Functions	875
20.4	Iterators	875
20.4.1	Iterator Class Hierarchy	877
20.5	Function Objects	881
20.5.1	Predicates	882
20.5.2	Arithmetic Function Objects	883
20.5.3	Binders, Negaters and Member Function Adapters	883
20.6	Algorithms	886
20.6.1	Sequence Algorithms	886
20.6.2	Sorting and Searching Algorithms	898
20.6.3	Set Algorithms	903
20.6.4	Heap Algorithms	905
20.6.5	Permutation Algorithms	906
20.6.6	Numeric Algorithms	908
20.7	Containers	910
20.7.1	Sequence Containers	911
20.7.2	Associative Containers	913
20.7.3	Numeric Containers	917
20.8	Adaptors	924
20.8.1	Iterator Adaptors	924
20.8.2	Container Adaptors	928
20.9	Strings	931
20.9.1	Character Traits	931
20.9.2	Class <code>basic_string</code>	932
20.10	Allocators	939
20.11	Summary	940
	Exercises	940
	Conclusion	943
	Appendices	
A	C++ Keywords	945
B	ASCII Character Set	947
C	Operators: Precedence, Associativity and Arity	953
D	Glossary	955

<http://www.springer.com/978-1-85233-450-5>

An Introduction to Object-Oriented Programming in C++
with Applications in Computer Graphics

Seed, G.M.

2001, XL, 972 p. 20 illus., Softcover

ISBN: 978-1-85233-450-5