



# Contents

---

<i>Preface</i> .....	<i>xvii</i>
<b>1 ■ The Story of C++</b> .....	<b>1</b>
The Origins of C++ .....	2
The Creation of C .....	2
Understanding the Need for C++ .....	4
C++ Is Born .....	5
The Evolution of C++ .....	6
What Is Object-Oriented Programming? .....	6
Encapsulation .....	7
Polymorphism .....	7
Inheritance .....	8
C++ Implements OOP .....	8
How C++ Relates to Java and C# .....	8
<b>2 ■ An Overview of C++</b> .....	<b>11</b>
Your First C++ Program .....	12
Entering the Program .....	12
Compiling the Program .....	13
Run the Program .....	14
A Line-by-Line Explanation .....	14

Handling Syntax Errors .....	16
A Second Simple Program .....	17
A More Practical Example .....	18
A New Data Type .....	19
A Quick Review .....	20
Functions .....	20
A Program with Two Functions .....	21
Function Arguments .....	22
Functions Returning Values .....	24
The main() Function .....	25
The General Form of C++ Functions .....	26
Some Output Options .....	26
Two Simple Commands .....	27
The if Statement .....	27
The for Loop .....	28
Blocks of Code .....	29
Semicolons and Positioning .....	30
Indentation Practices .....	31
C++ Keywords .....	31
Identifiers in C++ .....	32
The Standard C++ Library .....	32
<b>3 ■ The Basic Data Types .....</b>	<b>33</b>
Declaration of Variables .....	35
Local Variables .....	35
Formal Parameters .....	36
Global Variables .....	37
Some Type Modifiers .....	38
Literals .....	41
Hexadecimal and Octal Literals .....	43
String Literals .....	43
Character Escape Sequences .....	44
Variable Initializations .....	45
Operators .....	46
Arithmetic Operators .....	46
Increment and Decrement .....	48
How C++ Got Its Name .....	49
Relational and Logical Operators .....	50
Expressions .....	53
Type Conversion in Expressions .....	53
Converting to and from bool .....	53
Casts .....	54
Spacing and Parentheses .....	55

<b>4</b>	<b>Program Control Statements</b>	<b>57</b>
	The if Statement	58
	The Conditional Expression	59
	Nested ifs	60
	The if-else-if Ladder	61
	The for Loop	62
	Some Variations on the for Loop	64
	Missing Pieces	66
	The Infinite Loop	66
	Time Delay Loops	67
	The switch Statement	67
	Nested switch Statements	71
	The while Loop	71
	The do-while Loop	73
	Using continue	74
	Using break to Exit Loops	75
	Nested Loops	76
	Using the goto Statement	77
	Putting Together the Pieces	78
<b>5</b>	<b>Arrays and Strings</b>	<b>81</b>
	One-Dimensional Arrays	82
	No Bounds Checking	84
	Sorting an Array	85
	Strings	86
	Reading a String from the Keyboard	87
	Some String Library Functions	89
	strcpy	89
	strcat	89
	strcmp	90
	strlen	91
	Using the Null Terminator	93
	Two-Dimensional Arrays	94
	Multidimensional Arrays	96
	Array Initialization	96
	Unsigned Array Initializations	100
	Arrays of Strings	101
	An Example Using String Arrays	102
<b>6</b>	<b>Pointers</b>	<b>105</b>
	What Are Pointers?	106
	The Pointer Operators	107
	The Base Type Is Important	108
	Assigning Values Through a Pointer	110

Pointer Expressions . . . . .	110
Pointer Arithmetic . . . . .	111
Pointer Comparisons . . . . .	112
Pointers and Arrays . . . . .	112
Indexing a Pointer . . . . .	115
Are Pointers and Arrays Interchangeable? . . . . .	116
Pointers and String Literals . . . . .	117
A Comparison Example . . . . .	117
Arrays of Pointers . . . . .	118
The Null Pointer Convention . . . . .	121
Multiple Indirection . . . . .	122
Pointers and 16-bit Environments . . . . .	122
Problems with Pointers . . . . .	124
Uninitialized Pointers . . . . .	124
Invalid Pointer Comparisons . . . . .	124
Forgetting to Reset a Pointer . . . . .	125
<b>7 ■ Functions, Part One: The Fundamentals . . . . .</b>	<b>127</b>
Scope Rules of Functions . . . . .	128
Local Variables . . . . .	128
Formal Parameters . . . . .	134
Global Variables . . . . .	134
Passing Pointers and Arrays . . . . .	136
Calling Functions with Pointers . . . . .	136
Calling Functions with Arrays . . . . .	137
Passing Strings . . . . .	140
argc and argv: Arguments to main() . . . . .	141
Passing Numeric Command Line Arguments . . . . .	144
Converting Numeric Strings to Numbers . . . . .	145
The return Statement . . . . .	145
Returning from a Function . . . . .	146
Returning Values . . . . .	147
void Functions . . . . .	149
Functions That Return Pointers . . . . .	149
Function Prototypes . . . . .	151
Headers: A Closer Look . . . . .	152
Old-Style versus Modern Function Parameter Declarations . . . . .	153
Recursion . . . . .	153
<b>8 ■ Functions, Part Two: References, Overloading, and Default Arguments . . . . .</b>	<b>157</b>
Two Approaches to Argument Passing . . . . .	158
How C++ Passes Arguments . . . . .	158
Using a Pointer to Create a Call-by-Reference . . . . .	159

Reference Parameters .....	160
Declaring Reference Parameters .....	163
Returning References .....	164
Creating a Bounded Array .....	167
Independent References .....	168
A Few Restrictions When Using References .....	169
Function Overloading .....	170
The overload Anachronism .....	173
Default Function Arguments .....	173
Default Arguments versus Overloading .....	175
Using Default Arguments Correctly .....	177
Function Overloading and Ambiguity .....	177
<b>9 ■ More Data Types and Operators .....</b>	<b>181</b>
The const and volatile Qualifiers .....	182
const .....	182
volatile .....	184
Storage Class Specifiers .....	185
auto .....	185
extern .....	186
static Variables .....	187
Register Variables .....	191
The Origins of the register Modifier .....	192
Enumerations .....	193
typedef .....	197
More Operators .....	197
Bitwise Operators .....	197
AND, OR, XOR, and NOT .....	198
The Shift Operators .....	202
The ? Operator .....	203
Compound Assignment .....	205
The Comma Operator .....	205
Multiple Assignments .....	206
Using sizeof .....	206
Dynamic Allocation Using new and delete .....	207
Initializing Dynamically Allocated Memory .....	210
Allocating Arrays .....	210
C's Approach to Dynamic Allocation: malloc() and free() .....	211
Precedence Summary .....	213
<b>10 ■ Structures and Unions .....</b>	<b>215</b>
Structures .....	216
Accessing Structure Members .....	218
Arrays of Structures .....	219

	A Simple Inventory Example .....	219
	Passing Structures to Functions .....	226
	Assigning Structures .....	227
	Pointers to Structures and the Arrow Operator .....	228
	References to Structures .....	232
	Arrays and Structures Within Structures .....	233
	C Structure Versus C++ Structures .....	234
	Bit-Fields .....	235
	Unions .....	237
	Anonymous Unions .....	242
	Using sizeof to Ensure Portability .....	243
	Moving On to Object-Oriented Programming .....	243
<b>11</b>	<b>Introducing the Class .....</b>	<b>245</b>
	Class Fundamentals .....	246
	The General Form of a class .....	250
	A Closer Look at Class Member Access .....	250
	Constructors and Destructors .....	252
	Parameterized Constructors .....	255
	An Initialization Alternative .....	259
	Classes and Structures Are Related .....	260
	Structures versus Classes .....	262
	Unions and Classes Are Related .....	263
	Inline Functions .....	264
	Creating Inline Functions Inside a Class .....	265
	Arrays of Objects .....	267
	Initializing Object Arrays .....	268
	Pointers to Objects .....	270
	Object References .....	272
<b>12</b>	<b>A Closer Look at Classes .....</b>	<b>273</b>
	Friend Functions .....	274
	Overloading Constructors .....	278
	Dynamic Initialization .....	280
	Applying Dynamic Initialization to Constructors .....	280
	Assigning Objects .....	282
	Passing Objects to Functions .....	283
	Constructors, Destructors, and Passing Objects .....	284
	A Potential Problem When Passing Objects .....	285
	Returning Objects .....	288
	A Potential Problem When Returning Objects .....	289
	Creating and Using a Copy Constructor .....	291
	Copy Constructors and Parameters .....	292
	Copy Constructors and Initializations .....	294

	Using Copy Constructors When an Object Is Returned . . .	295
	Copy Constructors—Is There a Simpler Way? . . . . .	296
	The this Keyword . . . . .	297
<b>13</b>	<b>Operator Overloading . . . . .</b>	<b>299</b>
	Operator Overloading Using Member Functions . . . . .	300
	Using Member Functions to Overload Unary Operators . .	303
	Operator Overloading Tips and Restrictions . . . . .	308
	Nonmember Operator Functions . . . . .	309
	Order Matters . . . . .	309
	Using a Friend to Overload a Unary Operator . . . . .	313
	Overloading the Relational and Logical Operators . . . . .	316
	A Closer Look at the Assignment Operator . . . . .	317
	Overloading [] . . . . .	320
	Overloading () . . . . .	324
	Overloading Other Operators . . . . .	325
	Another Example of Operator Overloading . . . . .	325
<b>14</b>	<b>Inheritance . . . . .</b>	<b>331</b>
	Introducing Inheritance . . . . .	332
	Base Class Access Control . . . . .	335
	Using protected Members . . . . .	337
	Using protected for Inheritance of a Base Class . . . . .	340
	Reviewing public, protected, and private . . . . .	342
	Inheriting Multiple Base Classes . . . . .	342
	Constructors, Destructors, and Inheritance . . . . .	343
	When Constructors and Destructors Are Executed . . . . .	343
	Passing Parameters to Base Class Constructors . . . . .	346
	Granting Access . . . . .	350
	Reading C++ Inheritance Graphs . . . . .	352
	Virtual Base Classes . . . . .	352
<b>15</b>	<b>Virtual Functions and Polymorphism . . . . .</b>	<b>357</b>
	Pointers to Derived Types . . . . .	358
	References to Derived Types . . . . .	360
	Virtual Functions . . . . .	360
	Virtual Functions Are Inherited . . . . .	363
	Why Virtual Functions? . . . . .	365
	A Simple Application of Virtual Functions . . . . .	366
	Pure Virtual Functions and Abstract Classes . . . . .	370
	Early versus Late Binding . . . . .	372
	Polymorphism and the Purist . . . . .	373
<b>16</b>	<b>Templates . . . . .</b>	<b>375</b>
	Generic Functions . . . . .	376
	A Function with Two Generic Types . . . . .	378
	Explicitly Overloading a Generic Function . . . . .	379

	Overloading a Function Template .....	381
	Using Standard Parameters with Template Functions ...	382
	Generic Function Restrictions .....	383
	Creating a Generic abs() Function .....	383
	Generic Classes .....	384
	An Example with Two Generic Data Types .....	387
	Creating a Generic Array Class .....	388
	Using Non-Type Arguments with Generic Classes .....	389
	Using Default Arguments with Template Classes .....	391
	Explicit Class Specializations .....	393
<b>17</b>	<b>Exception Handling .....</b>	<b>395</b>
	Exception Handling Fundamentals .....	396
	exit() and abort() .....	398
	Catching Class Types .....	401
	Using Multiple catch Statements .....	402
	Options for Exception Handling .....	404
	Catching All Exceptions .....	404
	Restricting Exceptions Thrown by a Function .....	406
	Rethrowing an Exception .....	408
	Handling Exceptions Thrown by new .....	409
	The nothrow Alternative .....	410
	Overloading new and delete .....	411
	Overloading the nothrow Version of new .....	415
<b>18</b>	<b>The C++ I/O System .....</b>	<b>417</b>
	Old VS Modern C++ I/O .....	418
	C++ Streams .....	418
	The C++ Predefined Streams .....	419
	The C++ Stream Classes .....	419
	Overloading the I/O Operators .....	420
	Creating Inserters .....	421
	Using Friend Functions to Overload Inserters .....	423
	Overloading Extractors .....	424
	C I/O Versus C++ I/O .....	426
	Formatted I/O .....	426
	Formatting with the ios Member Functions .....	426
	Using I/O Manipulators .....	431
	Creating Your Own Manipulator Functions .....	433
	File I/O .....	435
	Opening and Closing a File .....	435
	Reading and Writing Text Files .....	438
	Unformatted Binary I/O .....	439
	Reading and Writing Blocks of Data .....	441
	Detecting EOF .....	442
	A File Comparison Example .....	443

More Binary I/O Functions .....	444
Random Access .....	446
Checking I/O Status .....	448
Customized I/O and Files .....	449
<b>19 ■ Run-Time Type ID and the Casting Operators .....</b>	<b>451</b>
Run-Time Type Identification (RTTI) .....	452
A Simple Application of Run-Time Type ID .....	456
typeid Can Be Applied to Template Classes .....	458
The Casting Operators .....	462
dynamic_cast .....	462
const_cast .....	467
static_cast .....	468
reinterpret_cast .....	469
The Traditional Cast Versus the Four Casting Operators ..	470
<b>20 ■ Namespaces and Other Advanced Topics .....</b>	<b>471</b>
Namespaces .....	472
Namespace Fundamentals .....	472
using .....	475
Unnamed Namespaces .....	477
The std Namespace .....	478
Pointers to Functions .....	480
Finding the Address of an Overloaded Function .....	483
Static Class Members .....	484
const Member Functions and mutable .....	486
Explicit Constructors .....	488
An Interesting Benefit from Implicit	
Constructor Conversion .....	490
The Member Initialization Syntax .....	490
Using the asm Keyword .....	493
Linkage Specification .....	493
The .* and ->* Pointer-to-Member Operators .....	495
Creating Conversion Functions .....	497
<b>21 ■ Introducing the Standard Template Library .....</b>	<b>499</b>
An Overview of the STL .....	500
The Container Classes .....	502
Vectors .....	504
Accessing a Vector Through an Iterator .....	508
Inserting and Deleting Elements in a Vector .....	509
Storing Class Objects in a Vector .....	510
The Power of Iterators .....	513
Lists .....	514
Sort a List .....	519
Merging One List with Another .....	520
Storing Class Objects in a List .....	521

Maps .....	523
Storing Class Objects in a Map .....	528
Algorithms .....	529
Counting .....	532
Removing and Replacing Elements .....	533
Reversing a Sequence .....	535
Transforming a Sequence .....	535
Exploring the Algorithms .....	537
The string Class .....	537
Some string Member Functions .....	541
Putting Strings into Other Containers .....	545
Final Thoughts on the STL .....	545
<b>22 ■ The C++ Preprocessor .....</b>	<b>547</b>
#define .....	548
Function-Like Macros .....	550
#error .....	552
#include .....	552
Conditional Compilation Directives .....	553
#if, #else, #elif, and #endif .....	553
#ifdef and #ifndef .....	555
#undef .....	556
Using defined .....	557
The Diminishing Role of the Preprocessor .....	557
#line .....	558
#pragma .....	559
The # and ## Preprocessor Operators .....	559
Predefined Macro Names .....	560
Final Thoughts .....	561
<b>A ■ C-Based I/O .....</b>	<b>563</b>
C I/O Uses Streams .....	564
Understanding printf() and scanf() .....	565
printf() .....	565
scanf() .....	567
The C File System .....	572
fopen() .....	573
fputc() .....	574
fgetc() .....	574
feof() .....	575
fclose() .....	575
Using fopen(), fgetc(), fputc(), and fclose() .....	575
ferror() and rewind() .....	576
fread() and fwrite() .....	577

fseek() and Random-Access I/O .....	578
fprintf() and fscanf() .....	579
Erasing Files .....	580
<b>B ■ Working with an Older C++ Compiler .....</b>	<b>581</b>
Two Simple Changes .....	583
<b>C ■ The .NET Managed Extensions to C++ .....</b>	<b>585</b>
The .NET Keyword Extensions .....	586
__abstract .....	586
__box .....	587
__delegate .....	587
__event .....	587
__finally .....	587
__gc .....	587
__identifier .....	587
__interface .....	587
__nogc .....	587
__pin .....	588
__property .....	588
__sealed .....	588
__try_cast .....	588
__typeof .....	588
__value .....	588
Preprocessor Extensions .....	588
The attribute Attribute .....	589
Compiling Managed C++ .....	589
<b>■ Index .....</b>	<b>591</b>