

DETAILED CONTENTS

PREFACE

WELCOME – AND THANK YOU!	xlvi
TARGET AUDIENCE	xlvi
Approach	xlvi
ARRANGEMENT	xlvi
PART I: THE C++ STUDENT SURVIVAL GUIDE	xlvi
CHAPTER 1: AN APPROACH TO THE ART OF PROGRAMMING	xlvi
CHAPTER 2: SMALL VICTORIES: CREATING PROJECTS WITH IDEs	xlvi
CHAPTER 3: PROJECT WALKTHROUGH: AN EXTENDED EXAMPLE	xlvi
CHAPTER 4: COMPUTERS, PROGRAMS, AND ALGORITHMS	xlvi
PART II: C++ LANGUAGE FUNDAMENTALS	xlvi
CHAPTER 5: SIMPLE PROGRAMS	xlvi
CHAPTER 6: CONTROLLING THE FLOW OF PROGRAM EXECUTION	xlvi
CHAPTER 7: POINTERS AND REFERENCES	xlvi
CHAPTER 8: ARRAYS	xlvi
CHAPTER 9: FUNCTIONS	xlvi
CHAPTER 10: TOWARD PROBLEM ABSTRACTION: CREATING NEW DATA TYPES	xlvi
CHAPTER 11: DISSECTING CLASSES	xlvi
CHAPTER 12: COMPOSITIONAL DESIGN	xlvi
CHAPTER 13: EXTENDING CLASS FUNCTIONALITY THROUGH INHERITANCE	xlvi
PART III: IMPLEMENTING POLYMORPHIC BEHAVIOR	l
CHAPTER 14: AD HOC POLYMORPHISM: OPERATOR OVERLOADING	l
CHAPTER 15: STATIC POLYMORPHISM: TEMPLATES	l
CHAPTER 16: DYNAMIC POLYMORPHISM: OBJECT-ORIENTED PROGRAMMING	l
PART IV: INTERMEDIATE CONCEPTS	li
CHAPTER 17: WELL-BEHAVED OBJECTS: THE ORTHODOX CANONICAL CLASS FORM	li
CHAPTER 18: MIXED LANGUAGE PROGRAMMING	li
CHAPTER 19: THREE DESIGN PRINCIPLES	li
CHAPTER 20: USING A UML MODELING TOOL	li
How To READ C++ FOR ARTISTS	lii
Pedagogy	liii
CHAPTER LAYOUT	liii
LEARNING OBJECTIVES	liii
INTRODUCTION	liii
CONTENT	liii
QUICK REVIEWS	liii
SUMMARY	liii
SKILL-BUILDING EXERCISES	liii
SUGGESTED PROJECTS	liii
SELF-TEST QUESTIONS	liii
REFERENCES	liii
NOTES	liii
CD-ROM	liv
SUPPORTSITE™ WEBSITE	liv
PROBLEM REPORTING	liv
ACKNOWLEDGEMENTS	lv

PART I: THE C++ STUDENT SURVIVAL GUIDE

1 AN APPROACH TO THE ART OF PROGRAMMING

INTRODUCTION	4
---------------------------	----------

<i>The Difficulties You Will Encounter Learning C++</i>	4
Required Skills	4
The Planets Will Come Into Alignment	4
How This Chapter Will Help You	5
PROJECT MANAGEMENT	5
<i>Three Software Development Roles</i>	5
Analyst	5
Architect	5
Programmer	6
<i>A Project Approach Strategy</i>	6
You Have Been Handed A Project – Now What?	6
Strategy Areas of Concern	6
Think Abstractly	7
<i>The Strategy In A Nutshell</i>	7
<i>Applicability To The Real World</i>	7
THE ART OF PROGRAMMING	8
<i>Don't Start At The Computer</i>	8
<i>Inspiration Strikes At The Weirdest Time</i>	8
<i>Own Your Own Computer</i>	8
You Either Have Time and No Money, or Money and No Time	8
The Family Computer Is Not Going To Cut It	9
<i>Set The Mood</i>	9
Location, Location, Location	9
<i>Concept Of The Flow</i>	9
The Stages of Flow	9
<i>Be Extreme</i>	10
The Programming Cycle	10
The Programming Cycle Summarized	11
<i>A Helpful Trick: Stubbing</i>	11
<i>Fix The First Compiler Error First</i>	11
MANAGING PROJECT COMPLEXITY	11
<i>Split Even Simple Projects Into Multiple Source Code Files</i>	12
Separating A Class's Interface From Its Implementation	12
Benefits of Separating Interface From Implementation	12
Helpful Preprocessor Directives	13
The Final Word on Preprocessor Directive Behavior	14
<i>Project File Format</i>	14
Header File	14
Implementation File	15
Main File	15
<i>Commenting</i>	16
C-Style Comments	16
C++-style Comments	17
Write Self-Commenting Code: Give Identifiers Meaningful Names	17
Adopt A Convention And Stick With It	19
Restrict The Number of Global Variables	19
Minimize Coupling, Maximize Cohesion	19
TEXTBOOKS, REFERENCE BOOKS, AND QUICK REFERENCE GUIDES	19
SUMMARY	20
SKILL BUILDING EXERCISES	20
SUGGESTED PROJECTS	21
SELF TEST QUESTIONS	22
REFERENCES	22
NOTES	23

2 SMALL VICTORIES: CREATING PROJECTS WITH IDE'S

INTRODUCTION	26
THE PROGRAM CREATION PROCESS	26
INTEGRATED DEVELOPMENT ENVIRONMENTS	27
<i>METROWERKS CODEWARRIOR</i>	28
<i>MICROSOFT VISUAL C++</i>	32
<i>INTERMISSION</i>	35
<i>TENON INTERSYSTEMS MACHTEN CODEBUILDER™</i>	36

ATTENTION LINUX USERS	36
ORGANIZING PROJECT FILES	37
CREATING SOURCE FILES	37
CREATING MAKEFILE	37
SUMMARY	39
SKILL BUILDING EXERCISES	39
SUGGESTED PROJECTS	40
SELF TEST QUESTIONS	40
REFERENCES	40
NOTES	41

3 PROJECT WALKTHROUGH: AN EXTENDED EXAMPLE

INTRODUCTION	44
THE PROJECT APPROACH STRATEGY	44
THE DEVELOPMENT CYCLE	45
THE PROJECT SPECIFICATION	46
ANALYZING THE PROJECT SPECIFICATION	47
REQUIREMENTS.....	47
PROBLEM DOMAIN	48
LANGUAGE FEATURES.....	50
DESIGN (FIRST ITERATION)	51
IMPLEMENTATION (FIRST ITERATION)	53
TESTING (FIRST ITERATION)	55
INTEGRATION (FIRST ITERATION)	55
DESIGN (SECOND ITERATION)	56
FUNCTION STUBBING	56
OTHER CONSIDERATIONS	56
IMPLEMENTATION (SECOND ITERATION)	57
TESTING (SECOND ITERATION)	59
INTEGRATION (SECOND ITERATION)	60
DESIGN (THIRD ITERATION)	60
IMPLEMENTATION (THIRD ITERATION)	60
TESTING (THIRD ITERATION)	62
INTEGRATION (THIRD ITERATION)	64
DESIGN (FOURTH ITERATION)	64
IMPLEMENTATION (FOURTH ITERATION)	66
TESTING (FOURTH ITERATION)	68
INTEGRATION (FOURTH ITERATION)	68
DESIGN (FIFTH ITERATION)	68
IMPLEMENTATION (FIFTH ITERATION)	70
TESTING (FIFTH ITERATION)	70
INTEGRATION (FIFTH ITERATION)	71
WRAPPING UP THE PROJECT	72
COMPLETE ROBOT RAT SOURCE CODE LISTING	72
SUMMARY	76
SKILL BUILDING EXERCISES	76
SUGGESTED PROJECTS	76
SELF TEST QUESTIONS	76
REFERENCES	77
NOTES	77

4 COMPUTERS, PROGRAMS, & ALGORITHMS

INTRODUCTION	80
WHAT IS A COMPUTER?	80
COMPUTER VS. COMPUTER SYSTEM	80

COMPUTER SYSTEM	80
PROCESSOR	82
THREE ASPECTS OF COMPUTER ARCHITECTURE	83
FEATURE SET	83
FEATURE SET IMPLEMENTATION	83
FEATURE SET ACCESSIBILITY	83
WHAT IS A PROGRAM?	83
TWO VIEWS OF A PROGRAM	84
CONCEPT OF OBSERVABLE BEHAVIOR	84
THE C++ PROGRAM TRANSFORMATION PROCESS	85
Phase 1	85
Phase 2	85
Phase 3	85
Phase 4	86
Phase 5	86
Phase 6	86
Phase 7	86
Phase 8	87
Phase 9	87
THE PROCESSING CYCLE	87
FETCH	88
DECODE	88
EXECUTE	88
STORE	88
MEMORY ORGANIZATION	88
MEMORY BASICS	88
MEMORY HIERARCHY	89
BITS, BYTES, WORDS	89
ALIGNMENT AND ADDRESSABILITY	90
ALGORITHMS	91
GOOD vs. BAD ALGORITHMS	92
DON'T REINVENT THE WHEEL!	94
SUMMARY	94
SKILL BUILDING EXERCISES	94
SUGGESTED PROJECTS	94
SELF TEST QUESTIONS	95
REFERENCES	95
NOTES	95

PART II: C++ LANGUAGE FUNDAMENTALS

5 Simple Programs

INTRODUCTION	100
A MINIMAL C++ PROGRAM	100
DISASSEMBLY IS A GREAT LEARNING TOOL	101
ANOTHER C++ PROGRAM	102
PARTS OF THE PROGRAM	103
COMMENTS	103
PREPROCESSOR DIRECTIVE	103
LIBRARIES	103
USING DIRECTIVE	103
main() FUNCTION	103
CONSTANTS	103
VARIABLES	103
STATEMENTS AND EXPRESSIONS	103
Keywords	104
FUNDAMENTAL TYPES	104
DETERMINING YOUR DATA TYPE RANGES	105
DETERMINING DATA TYPE SIZE WITH THE sizeof OPERATOR	106

LITERALS	106
INTEGER LITERALS	107
Decimal	107
Octal	107
Hexadecimal	107
A WORD OF CAUTION	107
CHARACTER LITERALS	108
Single CHARACTER LITERALS.....	108
Multiple CHARACTER LITERALS.....	108
ESCAPE SEQUENCES.....	109
FLOATING POINT LITERALS	110
STRING LITERALS	111
BOOLEAN LITERALS	111
EXPRESSIONS	111
OPERATORS	113
OPERATOR PRECEDENCE	114
USE PARENTHESES	115
Multiplicative OPERATORS	116
Multiplication OPERATOR.....	116
Division OPERATOR.....	116
Modulus OPERATOR.....	117
Additive OPERATORS	117
Addition OPERATOR.....	117
Subtraction OPERATOR	118
Shift OPERATORS	118
Left Shift OPERATOR	118
Right Shift OPERATOR	119
RELATIONAL OPERATORS	120
Less Than OPERATOR.....	120
Greater Than OPERATOR.....	120
Less Than OR Equal To OPERATOR.....	121
Greater Than OR Equal To OPERATOR.....	121
Equality OPERATORS	121
Equal To OPERATOR	121
Not Equal To OPERATOR.....	121
Bitwise AND OPERATOR - &	121
Bitwise Exclusive OR OPERATOR - ^	122
Bitwise Inclusive OR OPERATOR - 	122
Logical AND OPERATOR - &&	123
Logical OR OPERATOR - 	123
Conditional OPERATOR - ? :	123
ASSIGNMENT OPERATORS	124
lVALUE vs. rVALUE	124
Compound Assignment OPERATORS.....	125
COMMA OPERATOR - ,	125
INCREMENT AND DECREMENT OPERATORS (++, -)	125
IDENTIFIERS	126
Identifier NAMING CONVENTIONS	126
Hungarian Notation	126
CONSTANTS	128
VARIABLES	128
DECLARING	128
SCOPE	128
Local Scope	129
Function Scope.....	130
File Scope	130
Multifile VARIABLE USAGE	131
SHARING FILE SCOPE VARIABLES ACROSS MULTIPLE FILES	131
LIMITING FILE SCOPE VARIABLE VISIBILITY TO ONE FILE	131
THE main() FUNCTION	132
THE PURPOSE OF THE main() FUNCTION	132
TWO FORMS OF main()	132
EXITING main()	133
CALLING FUNCTIONS UPON EXITING main()	133

Simple Input and Output	133
<i>cin</i>	134
<i>Trapping Bad Input</i>	134
<i>COUT</i>	135
<i>LEARNING MORE ABOUT COUT AND CIN</i>	135
SUMMARY	135
Skill Building Exercises	136
SUGGESTED PROJECTS	137
Self Test Questions	138
REFERENCES	138
NOTES	139

6 CONTROLLING THE FLOW OF PROGRAM EXECUTION

INTRODUCTION	142
STATEMENTS, NULL STATEMENTS, AND COMPOUND STATEMENTS	142
<i>STATEMENTS</i>	142
<i>NULL STATEMENTS</i>	142
<i>COMPOUND STATEMENTS</i>	143
SELECTION STATEMENTS	143
<i>if STATEMENT</i>	143
<i>if STATEMENTS AND COMPOUND STATEMENTS</i>	144
<i>if-else STATEMENT</i>	145
<i>NESTING if-else STATEMENTS</i>	146
<i>switch STATEMENT</i>	147
ITERATION STATEMENTS	150
<i>while STATEMENT</i>	150
<i>CONTROLLING while STATEMENTS WITH SENTINEL VALUES</i>	151
<i>NESTING while STATEMENTS</i>	152
<i>DOING SOMETHING FOREVER</i>	152
<i>EXITING while LOOPS WITH THE break STATEMENT</i>	152
<i>do STATEMENT</i>	155
<i>NESTING do STATEMENTS</i>	155
<i>for STATEMENT</i>	155
<i>NESTING for STATEMENTS</i>	157
<i>break</i>	157
<i>DOING SOMETHING FOREVER WITH A for STATEMENT</i>	157
<i>CONTINUE</i>	158
<i>AVOIDING break AND CONTINUE</i>	158
WRITING ELEGANT CODE	158
Labeled STATEMENTS	159
<i>goto STATEMENT</i>	159
<i>Advice on Using Goto</i>	159
CONTROL STATEMENT USAGE GUIDE	160
SUMMARY	160
Skill Building Exercises	161
SUGGESTED PROJECTS	162
Self Test Questions	163
REFERENCES	164
NOTES	164

7 POINTERS AND REFERENCES

INTRODUCTION	166
BUT FIRST, A SHORT STORY	166
<i>WHAT IS AN OBJECT?</i>	167
<i>WHAT IS A MEMORY ADDRESS?</i>	168
<i>HOW DO YOU DETERMINE AN OBJECT'S MEMORY ADDRESS?</i>	169

What is a pointer?	171
How do you declare a pointer?	172
How do you access the object a pointer points to?	173
How do you dynamically create and delete objects?	174
The new operator	175
A neat trick: Calling object constructors	177
What's the difference between a pointer and a reference?	177
How do you declare and use references?	178
SUMMARY	178
Skill Building Exercises	179
SUGGESTED PROJECTS	179
SELF TEST	180
REFERENCES	180
NOTES	181

8 ARRAYS

INTRODUCTION	184
WHAT IS AN ARRAY?	184
Locating array elements	185
DECLARING & DEFINING STATICALLY ALLOCATED ARRAYS	185
Single-dimensional arrays	185
Accessing array elements	186
Subscript method	186
Pointer arithmetic method	187
Beware the uninitialized array!	187
Combining array definition with array declaration	187
Arrays of pointers	188
Multi-dimensional arrays	189
Arrays of two dimensions	189
Arrays of three or more dimensions	191
Automatic initialization of multi-dimensional arrays	193
DECLARING AND DEFINING DYNAMIC ARRAYS	196
Dynamically allocated single dimensional arrays	196
Dynamically allocated multi-dimensional arrays	197
STRINGS	200
SUMMARY	200
Skill Building Exercises	201
SUGGESTED PROJECTS	202
SELF TEST QUESTIONS	203
REFERENCES	203
NOTES	204

9 FUNCTIONS

INTRODUCTION	206
WHAT IS A FUNCTION?	206
Interface vs. implementation	207
Put function interface declarations in header files	207
#include...#define...#endif	207
Put function definitions in implementation files	207
Characteristics of a well-written function	208
DECLARING AND DEFINING FUNCTIONS	208
Naming functions	208
Function declaration	209
Function definition	209
Function calling	209

A Complete Example	210
Quick Review	211
Local Function Variable Scoping	212
Declaring Local Variables	212
Hiding Global Variables with Local Variables	212
Using Scoping Blocks in Functions	212
Static Function Variables	213
Scope of Function Parameters	214
Quick Review	215
Passing Arguments to Functions	215
Function Calling	215
Responsibilities of the Calling Function	216
Responsibilities of the Called Function	216
Passing Arguments by Value	216
Another Example	218
Passing Arguments by Reference	219
Continuing The Story... ..	220
Passing Pointers	220
Passing References	221
Passing Arrays to Functions	222
Passing Multi-Dimensional Arrays To Functions	223
Another Example	224
Using Function Return Values	225
Returning Objects	226
The Return Keyword: Mantra on Proper Usage	226
Another Example	227
Returning Pointers	227
How Not To Return a Pointer From a Function: Avoiding the Dangling Reference	229
Returning References	229
Quick Review	230
Function Overloading	231
Calling Functions Recursively	232
Another Example	233
Function Pointers	234
Declaring Function Pointers	235
Assigning The Address of a Function to a Function Pointer	235
Calling the Function via the Function Pointer	236
Arrays of Function Pointers	236
Implementing Callback Functions with Function Pointers	237
Creating a Function Library	239
Steps to Creating a Library	239
Create Empty Project	239
Add Implementation File	240
Set Library Target Settings	240
Name Library and Set Project Type	241
Build the Project	241
Use the Library	241
Summary	242
Skill Building Exercises	242
Suggested Projects	243
EISCS MKI Language Set	244
Sample Program	245
Basic Operation of the EISCS MKI	245
Memory	245
Instruction Decoding	245
Additional EISCS Specifications	246
Self Test Questions	247
References	247
Notes	248

10 TOWARDS PROBLEM ABSTRACTION: CREATING NEW DATA TYPES

INTRODUCTION	250
TOWARD DATA ABSTRACTION: typedef	250
CREATING TYPE SYNONYMS	250
CREATING ENUMERATED DATA TYPES WITH ENUM	252
ENUMS AND SWITCH STATEMENTS	252
CHANGING AN ENUM'S DEFAULT STATE VALUES	252
ENUM STATE NAME CONFLICTS	253
The Utility of NAME SPACES.....	253
Quick Summary	254
STRUCTURES: C-Style	254
ACCESSING STRUCTURAL ELEMENTS	255
ACCESSING STRUCTURAL DATA MEMBERS VIA THE DOT "." OPERATOR	255
ACCESSING STRUCTURAL ELEMENTS VIA THE SHORTHAND MEMBER ACCESS ">" OPERATOR.....	256
Quick Summary	258
STRUCTURES: C++-Style	259
PERSON STRUCTURE Redesign	259
Public Interface Functions and the Public Access Specifier	259
Private Data Members and the Private Access Specifier	260
The Deep Secret: The this Pointer.....	262
Quick Summary	262
CLASSES: A GENTLE INTRODUCTION	263
Quick Summary	265
THE DIFFERENCES BETWEEN STRUCTURES & CLASSES	265
Quick Summary	265
OBJECT-ORIENTED THINKING	266
Object Speak: A New Vocabulary	266
SUMMARY	267
SKILL BUILDING EXERCISES	268
SUGGESTED PROJECTS	269
SELF TEST QUESTIONS	270
REFERENCES	270
NOTES	271

11 DISSECTING CLASSES

INTRODUCTION	274
THE CLASS CONSTRUCT	274
PARTS OF A CLASS DECLARATION	274
A MINIMUM CLASS DECLARATION	275
PLACE CLASS DECLARATIONS IN SEPARATE HEADER FILES	276
THE UML CLASS DIAGRAM	276
THE CONCEPTS OF STATE AND BEHAVIOR	276
Object State	276
Object Behavior.....	276
CLASS MEMBER FUNCTIONS	276
CLASS MEMBER FUNCTION ACCESS TO CLASS ATTRIBUTES	278
OBTAINING ACCESS TO CLASS ATTRIBUTES FROM A MEMBER FUNCTION.....	278
OBTAINING ACCESS TO INSTANCE ATTRIBUTES FROM A MEMBER FUNCTION.....	278
SPECIAL MEMBER FUNCTIONS	278
CONSTRUCTOR.....	278
TESTCLASS EXAMPLE	279
COPY CONSTRUCTOR	281
TESTCLASS EXAMPLE EXTENDED	281
COPY ASSIGNMENT OPERATOR	282
TESTCLASS EXAMPLE EXTENDED	283
DESTRUCTOR	283
TESTCLASS EXAMPLE EXTENDED	285
BEHAVIOR OF DEFAULT SPECIAL FUNCTIONS	286
Quick Summary	287

ACCESSOR AND MUTATOR FUNCTIONS	287
ACCESSOR FUNCTIONS	288
MUTATOR FUNCTIONS	288
Quick SUMMARY	288
Using Access Specifiers To Control Horizontal Member Access	288
The CONCEPT of HORIZONTAL ACCESS	289
DATA ENCAPSULATION	289
ACCESS SPECIFIERS	289
The PUBLIC ACCESS SPECIFIER.....	289
The PROTECTED ACCESS SPECIFIER.....	289
The PRIVATE ACCESS SPECIFIER.....	289
OVERLOADING CLASS MEMBER FUNCTIONS	289
FUNCTION SIGNATURES	290
Why would you WANT TO OVERLOAD MEMBER FUNCTIONS?	290
SEPARATING A CLASS'S INTERFACE FROM ITS IMPLEMENTATION	290
MANAGE Physical Complexity	291
Allow the CREATION of Code LIBRARIES	293
A COMPLETE EXAMPLE: CLASS PERSON	293
SUMMARY	296
Skill BUILDING EXERCISES	297
SUGGESTED PROJECTS	298
SELF TEST QUESTIONS	298
REFERENCES	299
NOTES	299

12 Compositional Design

INTRODUCTION	302
MANAGING Physical Complexity	302
AGGREGATION	302
Simple vs. Composite Aggregation	302
The Relationship BETWEEN Aggregation AND Object Lifetime	302
Aggregation EXAMPLE CODE	303
Composite Aggregation Example.....	303
Another Composite Aggregation Example.....	304
Simple Aggregation EXAMPLE	305
EXTENDING THE CLASS DIAGRAM	307
SEQUENCE DIAGRAMS	308
Quick SUMMARY	308
THE AIRCRAFT ENGINE SIMULATION: AN EXTENDED AGGREGATION EXAMPLE	309
The PURPOSE of the ENGINE CLASS	309
AN ENGINE AND its PARTS	309
The ENGINE CLASS	311
THE ENTIRE AIRCRAFT ENGINE SIMULATION PROJECT	314
aircraftutils.h	314
fuelpump.h	314
oilpump.h	315
TEMPERATURESENSOR.h	315
OXYGENSENSOR.h	315
COMPRESSOR.h	316
ENGINE.h	316
fuelpump.cpp	317
oilpump.cpp	317
TEMPERATURESENSOR.cpp	318
OXYGENSENSOR.cpp	318
COMPRESSOR.cpp	319
ENGINE.cpp	320
main.cpp	321

SUMMARY	322
Skill Building EXERCISES	322
SUGGESTED PROJECTS	323
SELF TEST QUESTIONS	324
REFERENCES	324
NOTES	325

13 EXTENDING CLASS FUNCTIONALITY THROUGH INHERITANCE

INTRODUCTION	328
PURPOSE AND USE OF INHERITANCE	328
EXPRESSING INHERITANCE WITH A UML CLASS DIAGRAM	328
IMPLEMENTING BASECLASS AND DERIVEDCLASSONE	329
<i>Quick Review</i>	331
ACCESS SPECIFIERS AND VERTICAL ACCESS	332
<i>Public Inheritance</i>	333
<i>Protected Inheritance</i>	333
<i>Private Inheritance</i>	333
<i>Quick Review</i>	334
CALLING BASE CLASS CONSTRUCTORS	335
<i>Quick Review</i>	337
FUNCTION NAME HIDING: THIS IS NOT FUNCTION OVERRIDING!	337
<i>Function Hiding vs. Function Overloading</i>	337
<i>Quick Review</i>	340
WHAT THEN IS FUNCTION OVERRIDING?	340
CREATING VIRTUAL FUNCTIONS: THE VIRTUAL KEYWORD	340
<i>Purpose of Virtual Functions</i>	340
<i>Declaring and Using Virtual Functions</i>	340
<i>Virtual Destructors</i>	341
<i>Quick Review</i>	342
PURE VIRTUAL FUNCTIONS	342
<i>Declaring Pure Virtual Functions</i>	342
ABSTRACT CLASSES	343
FLEET SIMULATION SOURCE CODE	346
<i>ciws.h</i>	346
<i>live_inch.h</i>	347
<i>torpedo.h</i>	347
<i>gasturbine.h</i>	347
<i>nuke_plant.h</i>	348
<i>steam_plant.h</i>	348
<i>submarine.h</i>	348
<i>surface_ship.h</i>	349
<i>ciws.cpp</i>	349
<i>live_inch.cpp</i>	349
<i>gasturbine_plant.cpp</i>	350
<i>nuke_plant.cpp</i>	350
<i>steam_plant.cpp</i>	351
<i>submarine.cpp</i>	351
<i>surface_ship.cpp</i>	352
<i>torpedo.cpp</i>	352
<i>vessel.cpp</i>	353
Multiple Inheritance	353
VIRTUAL BASE CLASSES: VIRTUAL INHERITANCE	357
GETTING INHERITANCE RIGHT: SOME POINTS TO CONSIDER	360
<i>Two Different Uses of Inheritance</i>	361
<i>Reasoning About Object-Oriented Application Design</i>	361

<i>INCREMENTAL Code Evolution</i>	361
<i>PROTECT YOURSELF IN YOUR DESIGN</i>	362
SUMMARY	362
SKILL BUILDING EXERCISES	362
SUGGESTED PROJECTS	364
SELF TEST QUESTIONS	366
REFERENCES	366
NOTES	367

PART III: IMPLEMENTING POLYMORPHIC BEHAVIOR

14 Ad Hoc Polymorphism: Operator Overloading

INTRODUCTION	372
Ad Hoc Polymorphism: FUNCTION OVERLOADING	372
THE GOAL OF OPERATOR OVERLOADING	372
OVERLOADABLE OPERATORS	372
OVERLOADING OPERATORS	373
OVERLOADING IOSTREAM INSERTION AND EXTRACTION OPERATORS: <<, >>	374
OVERLOADING THE ASSIGNMENT OPERATOR: =	378
<i>Shallow Copy vs. Deep Copy</i>	379
OVERLOADING RELATIONAL OPERATORS: <, >, <=, >=	380
OVERLOADING EQUALITY OPERATORS: ==, !=	381
OVERLOADING ARITHMETIC OPERATORS: +, -, *, /, %	383
<i>A FEW WORDS ABOUT ERROR CHECKING</i>	384
OVERLOADING THE SUBSCRIPT OPERATOR: []	384
OVERLOADING COMPOUND ASSIGNMENT OPERATORS: +=, -=, *=, ETC.	386
OVERLOADING INCREMENT & DECREMENT OPERATORS: ++, --	387
OVERLOADING VARIOUS OTHER OPERATORS: (), +, -, <<, >>, ETC.	389
<i>THE FUNCTION OPERATOR: OPERATOR()</i>	392
<i>THE MEMBER OPERATOR: OPERATOR->()</i>	392
<i>THE COMMA OPERATOR: OPERATOR,() - A.K.A. THE SEQUENCING OPERATOR</i>	392
VIRTUAL OVERLOADED OPERATORS	392
SUMMARY	394
SKILL BUILDING EXERCISES	394
SUGGESTED PROJECTS	395
SELF TEST QUESTIONS	395
REFERENCES	396
NOTES	396

15 Static Polymorphism: Templates

INTRODUCTION	398
DEFINITION OF TEMPLATE	398
<i>FUNCTION TEMPLATES</i>	398
<i>CLASS TEMPLATES</i>	398
<i>STRUCTURE TEMPLATES</i>	398
HOW TEMPLATES WORK: AN ANALOGY	398
DECLARING AND USING FUNCTION TEMPLATES	399
<i>SEPARATING DECLARATION FROM IMPLEMENTATION: SOME BACKGROUND</i>	399
<i>WHEN IN DOUBT REFER TO YOUR COMPILER DOCUMENTATION</i>	400

<i>Example 15.1</i> CONTINUED	400
Using Multiple Placeholders	400
Quick Review	402
DECLARING AND USING CLASS TEMPLATES	405
<i>A MORE COMPLEX CLASS TEMPLATE EXAMPLE</i>	404
Quick Review	405
OVERVIEW OF THE STANDARD TEMPLATE LIBRARY (STL)	405
CONTAINERS AND CONTAINER ADAPTERS	406
ITERATORS.....	407
Algorithms	408
Quick Review	408
USING STANDARD TEMPLATE LIBRARY COMPONENTS	408
Using VECTOR	408
Using list	411
Quick Review	411
SUMMARY	412
SKILL BUILDING EXERCISES	412
SUGGESTED PROJECTS	413
SELF TEST QUESTIONS	413
REFERENCES	414
NOTES	414

16 DYNAMIC POLYMORPHISM: OBJECT-ORIENTED PROGRAMMING

INTRODUCTION	416
ABSTRACTION: AMPLIFY THE ESSENTIAL—ELIMINATE THE IRRELEVANT	416
OBJECT-ORIENTED PROGRAMMING DEFINED	417
DYNAMIC POLYMORPHISM DEFINED	417
LANGUAGE FEATURES THAT SUPPORT OBJECT-ORIENTED PROGRAMMING	417
AN EXAMPLE: CLASS INTERFACE	419
Quick Review.....	420
EXTENDED EXAMPLE: ENGINE COMPONENTS REVISITED	422
<i>A BASIS FOR COMPARISON</i>	422
<i>POLYMORPHIC ENGINE COMPONENT CODE</i>	424
icomponent.h.....	424
component.h	424
component.cpp	425
pump.h	425
pump.cpp	426
sensor.h	426
sensor.cpp	427
waterpump.h	427
waterpump.cpp	428
oilpump.h	428
oilpump.cpp	428
fuelpump.h	428
fuelpump.cpp	429
airflowsensor.h	429
airflowsensor.cpp	429
oxysensor.h	429
oxysensor.cpp	430
temperaturesensor.h	430
temperaturesensor.cpp	430
engine.h	431
engine.cpp	431
smallengine.h	432
smallengine.cpp	432
engineutils.h	433
main.cpp	433
<i>DISCUSSION OF THE POLYMORPHIC ENGINE COMPONENT CODE</i>	434
ICOMPONENT AND DERIVED CLASSES	434
WHAT IS MEANT BY A PURE VIRTUAL vs. A VIRTUAL MEMBER FUNCTION DECLARATION.....	434
ENGINE AND SMALLENGINE.....	434
<i>RUNNING THE POLYMORPHIC ENGINE COMPONENT PROGRAM</i>	434
A SHORT STORY	435

<i>Taming the Complexity of the C++ Language</i>	435
SUMMARY	436
SKILL BUILDING EXERCISES	436
SUGGESTED PROJECTS	439
SELF TEST QUESTIONS	440
REFERENCES	441
NOTES	441

PART IV: INTERMEDIATE CONCEPTS

17 Well Behaved Objects: The Orthodox Canonical Class Form

INTRODUCTION	446
WHAT IS A WELL-BEHAVED OBJECT?	446
<i>OBJECT USAGE CONTEXTS</i>	446
<i>Object Creation</i>	446
<i>Object Copying</i>	447
<i>Object Assignment</i>	447
<i>Object Destruction</i>	447
<i>Other Contexts By Design</i>	447
THE ORTHODOX CANONICAL CLASS FORM (OCCF)	448
<i>FOUR REQUIRED FUNCTIONS</i>	448
<i>Default Constructor</i>	449
<i>Destructor</i>	449
<i>Copy Constructor</i>	449
<i>Copy Assignment Operator</i>	449
<i>IMPLEMENTING Foo CLASS OCCF FUNCTIONS</i>	449
<i>CONSIDER FUTURE DESIRED BEHAVIOR</i>	449
<i>EXTENDING Foo TO PARTICIPATE IN OTHER CONTEXTS: OVERLOADING MORE OPERATORS</i>	451
<i>Quick Review</i>	452
SUMMARY	453
SKILL BUILDING EXERCISES	453
SUGGESTED PROJECTS	454
SELF TEST QUESTIONS	455
REFERENCES	455
NOTES	455

18 Mixed Language Programming

INTRODUCTION	458
C++ AND C	458
<i>How C++ Allows Overloaded Functions: Name Mangling</i>	458
<i>EXTERN KEYWORD</i>	458
<i>Building a C Library: The SQUARE() Function</i>	458
<i>Deciphering C Standard Library Files</i>	464
<i>Quick Review</i>	464
C++ AND ASSEMBLY	465
<i>SOME THINGS TO THINK ABOUT BEFORE USING ASSEMBLY</i>	465
<i>KNOW THY IMPLEMENTATION DEPENDENCIES</i>	465
<i>INLINE ASSEMBLY LANGUAGE IN A C++ FUNCTION</i>	465
<i>LINKING AN OBJECT FILE CREATED FROM ASSEMBLY LANGUAGE</i>	467
<i>PROCESS STEPS</i>	467
<i>Using Inline Assembly in the Macintosh Environment</i>	469
<i>Quick Review</i>	470
C++ AND JAVA: THE JAVA NATIVE INTERFACE (JNI)	470

STEPS TO CREATE A JNI C++ PROGRAM	470
Win32 JNI Example	471
STEP 1: CREATE JAVA SOURCE FILE.....	471
STEP 2: COMPILe JAVA SOURCE FILE.....	472
STEP 3: CREATE HEADER FILE.....	472
STEP 4: CREATE C++ SOURCE FILE.....	474
STEP 5: COMPILe C++ SOURCE FILE TO CREATE DYNAMIC LINK LIBRARY.....	474
STEP 6: RUN JAVA PROGRAM.....	474
Macintosh OSX JNI Example	475
STEP 1: CREATE JAVA SOURCE FILE.....	476
STEP 2: COMPILe JAVA SOURCE FILE.....	476
STEP 3: CREATE HEADER FILE.....	476
STEP 4: CREATE C++ SOURCE FILE.....	476
STEP 5: COMPILe C++ SOURCE FILE TO CREATE DYNAMIC LINK LIBRARY.....	476
STEP 6: RUN JAVA PROGRAM.....	477
WHEN TO USE JNI.....	477
Quick Review	477
SUMMARY	478
Skill Building Exercises	478
SUGGESTED PROJECTS	478
SELF TEST QUESTIONS	479
REFERENCES	479
NOTES	480

19 THREE DESIGN PRINCIPLES

INTRODUCTION	482
THE PREFERRED CHARACTERISTICS OF AN OBJECT-ORIENTED ARCHITECTURE	482
EASY TO UNDERSTAND – (HOW DOES THIS THING WORK?)	482
EASY TO REASON ABOUT – (WHAT ARE THE EFFECTS OF CHANGE?)	482
EASY TO EXTEND – (WHERE DO I ADD FUNCTIONALITY?)	482
THE LISKOV SUBSTITUTION PRINCIPLE & DESIGN BY CONTRACT	483
REASONING ABOUT THE BEHAVIOR OF SUPERTYPES AND SUBTYPES	483
RELATIONSHIP BETWEEN THE LSP AND DbC.....	483
THE COMMON GOAL OF THE LSP AND DbC.....	483
C++ SUPPORT FOR THE LSP AND DbC.....	483
DESIGNING WITH THE LSP/DbC IN MIND	483
THE POWER AND DANGER OF C++.....	484
CLASS DECLARATIONS VIEWED AS BEHAVIOR SPECIFICATIONS	484
PRECONDITIONS, POSTCONDITIONS, AND CLASS INVARIANTS	484
CLASS INVARIANT.....	484
PRECONDITION.....	484
POSTCONDITION.....	484
AN EXAMPLE.....	485
USING INCREMENTER AS A BASE CLASS	486
CHANGING THE PRECONDITIONS OF DERIVED CLASS FUNCTIONS	488
ADOPTING THE SAME PRECONDITIONS	489
WEAKENING PRECONDITIONS	489
STRENGTHENING PRECONDITIONS.....	491
Quick Review	493
CHANGING THE POSTCONDITIONS OF DERIVED CLASS FUNCTIONS	493
SPECIAL CASES OF PRECONDITIONS AND POSTCONDITIONS	494
FUNCTION ARGUMENT TYPES	494
FUNCTION RETURN TYPES.....	496
FUNCTION ACCESS RIGHTS.....	496
Quick Review	497
THREE RULES OF THE SUBSTITUTION PRINCIPLE	497
SIGNATURE RULE.....	497
METHODS RULE.....	497
PROPERTIES RULE.....	497
THE OPEN-CLOSED PRINCIPLE	497
ACHIEVING THE OPEN-CLOSED PRINCIPLE	498
AN OCP EXAMPLE	498
ADDITIONAL OCP CONVENTIONS.....	498
RELATIONSHIP BETWEEN THE OCP AND THE LSP/DbC	498

<i>Quick Review</i>	498
The Dependency Inversion Principle	500
<i>CHARACTERISTICS OF BAD SOFTWARE ARCHITECTURE</i>	500
<i>CHARACTERISTICS OF GOOD SOFTWARE ARCHITECTURE</i>	501
<i>SELECTING THE RIGHT ABSTRACTIONS TAKES EXPERIENCE</i>	501
<i>Quick Review</i>	501
SUMMARY	501
TERMS AND DEFINITIONS	502
SKILL BUILDING EXERCISES	502
SUGGESTED PROJECTS	503
SELF TEST QUESTIONS	503
REFERENCES	503
NOTES	504

20 Using A UML Modeling Tool

INTRODUCTION	506
THE PURPOSE OF A UML MODELING TOOL	506
INTRODUCING EMBARCADERO TECHNOLOGIES' DESCRIBE™	507
<i>PRIMARY FEATURES</i>	507
THE PROJECT SPECIFICATION: ROBOT RAT	508
CREATING USE CASE DIAGRAMS	509
<i>Adding DOCUMENTATION TO DIAGRAM ELEMENTS</i>	511
<i>PROGRAMMER PERSPECTIVE USE CASES</i>	512
PAUSING TO CONSIDER DESIGN ISSUES	513
CREATING CLASS DIAGRAMS	515
<i>CREATING AN OVERALL PACKAGE ARCHITECTURE DIAGRAM</i>	515
<i>MOVING BEYOND THE PACKAGE DIAGRAM</i>	516
<i>Adding OPERATIONS AND ATTRIBUTES TO CLASSES</i>	517
ITERATING THROUGH THE DESIGN PROCESS	519
CREATING SEQUENCE DIAGRAMS	522
<i>PROPER USE OF SEQUENCE DIAGRAMS</i>	522
<i>Adding OBJECTS TO SEQUENCE DIAGRAMS</i>	522
<i>Adding MESSAGES TO SEQUENCE DIAGRAMS</i>	523
GENERATING SOURCE CODE	525
REVERSE ENGINEERING	527
<i>MERGING SYSTEMS</i>	528
LINKING DIAGRAM OBJECTS TO DIAGRAMS	529
GENERATING WEB PROJECT REPORTS	530
SUMMARY	531
ROBOTRAT SOURCE CODE	532
<i>abstraction.h</i>	532
<i>abstracmarker.h</i>	532
<i>abstractcontrolledobject.h</i>	532
<i>position.h</i>	533
<i>marker.h</i>	533
<i>remotecontrolledobject.h</i>	534
<i>abstractcontrolledrodent.h</i>	534
<i>robotrat.h</i>	534
<i>rodentworld.h</i>	535
<i>userinterface.h</i>	535
<i>controller.h</i>	536
<i>position.cpp</i>	536
<i>marker.cpp</i>	538
<i>remotecontrolledobject.cpp</i>	538
<i>robotrat.cpp</i>	539

<i>RODENWORLD.CPP</i>	540
<i>USERINTERFACE.CPP</i>	542
<i>CONTROLLER.CPP</i>	543
<i>MAIN.CPP</i>	544
Skill Building Exercises	544
SUGGESTED PROJECTS	545
Self Test Questions	545
REFERENCES	546
NOTES	546

APPENDICES

Appendix A: Project Approach Strategy Checkoff List

PROJECT APPROACH STRATEGY CHECKOFF LIST	549
--	------------

Appendix B: ASCII Table

ASCII TABLE	551
--------------------------	------------

Appendix C: ANSWERS TO SELF-TEST QUESTIONS

CHAPTER 1	555
CHAPTER 2	556
CHAPTER 3	557
CHAPTER 4	558
CHAPTER 5	560
CHAPTER 6	561
CHAPTER 7	563
CHAPTER 8	564
CHAPTER 9	565
CHAPTER 10	567
CHAPTER 11	568
CHAPTER 12	569
CHAPTER 13	569
CHAPTER 14	571
CHAPTER 15	571
CHAPTER 16	572
CHAPTER 17	574
CHAPTER 18	575
CHAPTER 19	576
CHAPTER 20	578

