

# DETAILED CONTENTS

## PREFACE

<b>WELCOME – AND THANK YOU!</b> .....	<b>xlvi</b>
<b>TARGET AUDIENCE</b> .....	<b>xlvi</b>
<b>Approach</b> .....	<b>xlvi</b>
<b>ARRANGEMENT</b> .....	<b>xlvi</b>
<b>PART I: THE C++ STUDENT SURVIVAL GUIDE</b> .....	<b>xlvi</b>
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
<b>PART II: C++ LANGUAGE FUNDAMENTALS</b> .....	<b>xlvi</b>
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
<b>PART III: IMPLEMENTING POLYMORPHIC BEHAVIOR</b> .....	<b>l</b>
CHAPTER 14: AD HOC POLYMORPHISM: OPERATOR OVERLOADING .....	l
CHAPTER 15: STATIC POLYMORPHISM: TEMPLATES .....	l
CHAPTER 16: DYNAMIC POLYMORPHISM: OBJECT-ORIENTED PROGRAMMING .....	l
<b>PART IV: INTERMEDIATE CONCEPTS</b> .....	<b>li</b>
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
<b>How To READ C++ FOR ARTISTS</b> .....	<b>li</b>
<b>Pedagogy</b> .....	<b>lii</b>
CHAPTER LAYOUT .....	lii
LEARNING OBJECTIVES .....	lii
INTRODUCTION .....	lii
CONTENT .....	lii
QUICK REVIEWS .....	lii
SUMMARY .....	lii
SKILL-BUILDING EXERCISES .....	lii
SUGGESTED PROJECTS .....	lii
SELF-TEST QUESTIONS .....	lii
REFERENCES .....	lii
NOTES .....	lii
<b>CD-ROM</b> .....	<b>liv</b>
<b>SUPPORTSITE™ WEBSITE</b> .....	<b>liv</b>
<b>PROBLEM REPORTING</b> .....	<b>liv</b>
<b>ACKNOWLEDGEMENTS</b> .....	<b>lv</b>

## PART I: THE C++ STUDENT SURVIVAL GUIDE

### 1 AN APPROACH TO THE ART OF PROGRAMMING

<b>INTRODUCTION</b> .....	<b>4</b>
---------------------------	----------

<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
<b>PROJECT MANAGEMENT</b> .....	<b>5</b>
<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
<b>THE ART OF PROGRAMMING</b> .....	<b>8</b>
<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
<b>MANAGING PROJECT COMPLEXITY</b> .....	<b>11</b>
<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
<b>TEXTBOOKS, REFERENCE BOOKS, AND QUICK REFERENCE GUIDES</b> .....	<b>19</b>
<b>SUMMARY</b> .....	<b>20</b>
<b>SKILL BUILDING EXERCISES</b> .....	<b>20</b>
<b>SUGGESTED PROJECTS</b> .....	<b>21</b>
<b>SELF TEST QUESTIONS</b> .....	<b>22</b>
<b>REFERENCES</b> .....	<b>22</b>
<b>NOTES</b> .....	<b>23</b>

## 2 SMALL VICTORIES: CREATING PROJECTS WITH IDE'S

<b>INTRODUCTION</b> .....	<b>26</b>
<b>THE PROGRAM CREATION PROCESS</b> .....	<b>26</b>
<b>INTEGRATED DEVELOPMENT ENVIRONMENTS</b> .....	<b>27</b>
<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
<b>SUMMARY .....</b>	<b>39</b>
<b>SKILL BUILDING EXERCISES .....</b>	<b>39</b>
<b>SUGGESTED PROJECTS .....</b>	<b>40</b>
<b>SELF TEST QUESTIONS .....</b>	<b>40</b>
<b>REFERENCES .....</b>	<b>40</b>
<b>NOTES .....</b>	<b>41</b>

## 3 PROJECT WALKTHROUGH: AN EXTENDED EXAMPLE

<b>INTRODUCTION .....</b>	<b>44</b>
<b>THE PROJECT APPROACH STRATEGY .....</b>	<b>44</b>
<b>THE DEVELOPMENT CYCLE .....</b>	<b>45</b>
<b>THE PROJECT SPECIFICATION .....</b>	<b>46</b>
<i>Analyzing The Project Specification .....</i>	<i>47</i>
REQUIREMENTS.....	47
PROBLEM DOMAIN .....	48
LANGUAGE FEATURES.....	50
<i>Design (First Iteration) .....</i>	<i>51</i>
<i>Implementation (First Iteration) .....</i>	<i>53</i>
<i>Testing (First Iteration) .....</i>	<i>55</i>
<i>Integration (First Iteration) .....</i>	<i>55</i>
<i>Design (Second Iteration) .....</i>	<i>56</i>
FUNCTION STUBBING .....	56
OTHER CONSIDERATIONS .....	56
<i>Implementation (Second Iteration) .....</i>	<i>57</i>
<i>Testing (Second Iteration) .....</i>	<i>59</i>
<i>Integration (Second Iteration) .....</i>	<i>60</i>
<i>Design (Third Iteration) .....</i>	<i>60</i>
<i>Implementation (Third Iteration) .....</i>	<i>60</i>
<i>Testing (Third Iteration) .....</i>	<i>62</i>
<i>Integration (Third Iteration) .....</i>	<i>64</i>
<i>Design (Fourth Iteration) .....</i>	<i>64</i>
<i>Implementation (Fourth Iteration) .....</i>	<i>66</i>
<i>Testing (Fourth Iteration) .....</i>	<i>68</i>
<i>Integration (Fourth Iteration) .....</i>	<i>68</i>
<i>Design (Fifth Iteration) .....</i>	<i>68</i>
<i>Implementation (Fifth Iteration) .....</i>	<i>70</i>
<i>Testing (Fifth Iteration) .....</i>	<i>70</i>
<i>Integration (Fifth Iteration) .....</i>	<i>71</i>
<b>WRAPPING UP THE PROJECT .....</b>	<b>72</b>
<b>COMPLETE ROBOT RAT SOURCE CODE LISTING .....</b>	<b>72</b>
<b>SUMMARY .....</b>	<b>76</b>
<b>SKILL BUILDING EXERCISES .....</b>	<b>76</b>
<b>SUGGESTED PROJECTS .....</b>	<b>76</b>
<b>SELF TEST QUESTIONS .....</b>	<b>76</b>
<b>REFERENCES .....</b>	<b>77</b>
<b>NOTES .....</b>	<b>77</b>

## 4 COMPUTERS, PROGRAMS, & ALGORITHMS

<b>INTRODUCTION .....</b>	<b>80</b>
<b>WHAT IS A COMPUTER? .....</b>	<b>80</b>
COMPUTER VS. COMPUTER SYSTEM .....	80

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

## PART II: C++ LANGUAGE FUNDAMENTALS

### 5 Simple Programs

<b>INTRODUCTION .....</b>	<b>100</b>
<b>A MINIMAL C++ PROGRAM .....</b>	<b>100</b>
<i>Disassembly is a GREAT LEARNING TOOL</i> .....	101
<b>ANOTHER C++ PROGRAM .....</b>	<b>102</b>
<i>PARTS OF THE PROGRAM</i> .....	103
COMMENTS .....	103
PREPROCESSOR DIRECTIVE .....	103
LIBRARIES .....	103
Using Directive .....	103
main() FUNCTION .....	103
CONSTANTS .....	103
VARIABLES .....	103
STATEMENTS AND EXPRESSIONS .....	103
<b>Keywords .....</b>	<b>104</b>
<b>FUNDAMENTAL TYPES .....</b>	<b>104</b>
<i>DETERMINING YOUR DATA TYPE RANGES</i> .....	105
<i>DETERMINING DATA TYPE SIZE WITH THE sizeof OPERATOR</i> .....	106

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

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

## 6 CONTROLLING THE FLOW OF PROGRAM EXECUTION

<b>INTRODUCTION</b> .....	<b>142</b>
<b>STATEMENTS, NULL STATEMENTS, AND COMPOUND STATEMENTS</b> .....	<b>142</b>
<i>STATEMENTS</i> .....	142
<i>NULL STATEMENTS</i> .....	142
<i>COMPOUND STATEMENTS</i> .....	143
<b>SELECTION STATEMENTS</b> .....	<b>143</b>
<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
<b>ITERATION STATEMENTS</b> .....	<b>150</b>
<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
<b>WRITING ELEGANT CODE</b> .....	<b>158</b>
<b>Labeled STATEMENTS</b> .....	<b>159</b>
<i>goto STATEMENT</i> .....	159
<i>Advice on Using Goto</i> .....	159
<b>CONTROL STATEMENT USAGE GUIDE</b> .....	<b>160</b>
<b>SUMMARY</b> .....	<b>160</b>
<b>Skill Building Exercises</b> .....	<b>161</b>
<b>SUGGESTED PROJECTS</b> .....	<b>162</b>
<b>Self Test Questions</b> .....	<b>163</b>
<b>REFERENCES</b> .....	<b>164</b>
<b>NOTES</b> .....	<b>164</b>

## 7 POINTERS AND REFERENCES

<b>INTRODUCTION</b> .....	<b>166</b>
<b>BUT FIRST, A SHORT STORY</b> .....	<b>166</b>
<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
<b>SUMMARY</b> .....	<b>178</b>
<b>Skill Building Exercises</b> .....	<b>179</b>
<b>SUGGESTED PROJECTS</b> .....	<b>179</b>
<b>Self Test</b> .....	<b>180</b>
<b>REFERENCES</b> .....	<b>180</b>
<b>NOTES</b> .....	<b>181</b>

## 8 ARRAYS

<b>INTRODUCTION</b> .....	<b>184</b>
<b>WHAT IS AN ARRAY?</b> .....	<b>184</b>
Locating array elements .....	185
<b>DECLARING &amp; DEFINING STATICALLY ALLOCATED ARRAYS</b> .....	<b>185</b>
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
<b>DECLARING AND DEFINING DYNAMIC ARRAYS</b> .....	<b>196</b>
Dynamically allocated single dimensional arrays .....	196
Dynamically allocated multi-dimensional arrays .....	197
<b>STRINGS</b> .....	<b>200</b>
<b>SUMMARY</b> .....	<b>200</b>
<b>Skill Building Exercises</b> .....	<b>201</b>
<b>SUGGESTED PROJECTS</b> .....	<b>202</b>
<b>Self Test Questions</b> .....	<b>203</b>
<b>REFERENCES</b> .....	<b>203</b>
<b>NOTES</b> .....	<b>204</b>

## 9 FUNCTIONS

<b>INTRODUCTION</b> .....	<b>206</b>
<b>WHAT IS A FUNCTION?</b> .....	<b>206</b>
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
<b>DECLARING AND DEFINING FUNCTIONS</b> .....	<b>208</b>
Naming functions .....	208
Function declaration .....	209
Function definition .....	209
Function calling .....	209

A Complete Example .....	210
Quick Review .....	211
<b>Local Function Variable Scoping .....</b>	<b>212</b>
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
<b>Passing Arguments to Functions .....</b>	<b>215</b>
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
<b>Using Function Return Values .....</b>	<b>225</b>
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
<b>Function Overloading .....</b>	<b>231</b>
<b>Calling Functions Recursively .....</b>	<b>232</b>
Another Example .....	233
<b>Function Pointers .....</b>	<b>234</b>
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
<b>Creating a Function Library .....</b>	<b>239</b>
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
<b>Summary .....</b>	<b>242</b>
<b>Skill Building Exercises .....</b>	<b>242</b>
<b>Suggested Projects .....</b>	<b>243</b>
EISCS MKI Language Set .....	244
Sample Program .....	245
Basic Operation of the EISCS MKI .....	245
Memory .....	245
Instruction Decoding.....	245
Additional EISCS Specifications .....	246
<b>Self Test Questions .....</b>	<b>247</b>
<b>References .....</b>	<b>247</b>
<b>Notes .....</b>	<b>248</b>



## 10 TOWARDS PROBLEM ABSTRACTION: CREATING NEW DATA TYPES

<b>INTRODUCTION</b> .....	<b>250</b>
<b>TOWARD DATA ABSTRACTION: typedef</b> .....	<b>250</b>
CREATING TYPE SYNONYMS .....	250
<b>CREATING ENUMERATED DATA TYPES WITH ENUM</b> .....	<b>252</b>
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
<b>STRUCTURES: C-Style</b> .....	<b>254</b>
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
<b>STRUCTURES: C++-Style</b> .....	<b>259</b>
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
<b>CLASSES: A GENTLE INTRODUCTION</b> .....	<b>263</b>
Quick SUMMARY .....	265
<b>THE DIFFERENCES BETWEEN STRUCTURES &amp; CLASSES</b> .....	<b>265</b>
Quick SUMMARY .....	265
<b>OBJECT-ORIENTED THINKING</b> .....	<b>266</b>
OBJECT SPEAK: A NEW VOCABULARY .....	266
<b>SUMMARY</b> .....	<b>267</b>
<b>SKILL BUILDING EXERCISES</b> .....	<b>268</b>
<b>SUGGESTED PROJECTS</b> .....	<b>269</b>
<b>SELF TEST QUESTIONS</b> .....	<b>270</b>
<b>REFERENCES</b> .....	<b>270</b>
<b>NOTES</b> .....	<b>271</b>

## 11 DISSECTING CLASSES

<b>INTRODUCTION</b> .....	<b>274</b>
<b>THE CLASS CONSTRUCT</b> .....	<b>274</b>
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
<b>CLASS MEMBER FUNCTIONS</b> .....	<b>276</b>
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
<b>Using Access Specifiers To Control Horizontal Member Access .....</b>	<b>288</b>
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
<b>OVERLOADING CLASS MEMBER FUNCTIONS .....</b>	<b>289</b>
FUNCTION SIGNATURES .....	290
Why would you WANT TO OVERLOAD MEMBER FUNCTIONS? .....	290
<b>SEPARATING A CLASS'S INTERFACE FROM ITS IMPLEMENTATION .....</b>	<b>290</b>
MANAGE Physical Complexity .....	291
Allow the CREATION of Code LIBRARIES .....	293
<b>A COMPLETE EXAMPLE: CLASS PERSON .....</b>	<b>293</b>
SUMMARY .....	296
Skill BUILDING EXERCISES .....	297
SUGGESTED PROJECTS .....	298
SELF TEST QUESTIONS .....	298
REFERENCES .....	299
NOTES .....	299

## 12 Compositional Design

<b>INTRODUCTION .....</b>	<b>302</b>
<b>MANAGING Physical Complexity .....</b>	<b>302</b>
<b>AGGREGATION .....</b>	<b>302</b>
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
<b>EXTENDING THE CLASS DIAGRAM .....</b>	<b>307</b>
<b>SEQUENCE DIAGRAMS .....</b>	<b>308</b>
Quick SUMMARY .....	308
<b>THE AIRCRAFT ENGINE SIMULATION: AN EXTENDED AGGREGATION EXAMPLE .....</b>	<b>309</b>
The PURPOSE of the ENGINE CLASS .....	309
AN ENGINE AND its PARTS .....	309
The ENGINE CLASS .....	311
<b>THE ENTIRE AIRCRAFT ENGINE SIMULATION PROJECT .....</b>	<b>314</b>
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

<b>SUMMARY</b> .....	<b>322</b>
<b>Skill Building EXERCISES</b> .....	<b>322</b>
<b>SUGGESTED PROJECTS</b> .....	<b>323</b>
<b>SELF TEST QUESTIONS</b> .....	<b>324</b>
<b>REFERENCES</b> .....	<b>324</b>
<b>NOTES</b> .....	<b>325</b>

## 13 EXTENDING CLASS FUNCTIONALITY THROUGH INHERITANCE

<b>INTRODUCTION</b> .....	<b>328</b>
<b>PURPOSE AND USE OF INHERITANCE</b> .....	<b>328</b>
<b>EXPRESSING INHERITANCE WITH A UML CLASS DIAGRAM</b> .....	<b>328</b>
<b>IMPLEMENTING BASECLASS AND DERIVEDCLASSONE</b> .....	<b>329</b>
<i>Quick Review</i> .....	331
<b>ACCESS SPECIFIERS AND VERTICAL ACCESS</b> .....	<b>332</b>
<i>PUBLIC INHERITANCE</i> .....	333
<i>PROTECTED INHERITANCE</i> .....	333
<i>PRIVATE INHERITANCE</i> .....	333
<i>Quick Review</i> .....	334
<b>CALLING BASE CLASS CONSTRUCTORS</b> .....	<b>335</b>
<i>Quick Review</i> .....	337
<b>FUNCTION NAME HIDING: THIS IS NOT FUNCTION OVERRIDING!</b> .....	<b>337</b>
<i>FUNCTION HIDING VS. FUNCTION OVERLOADING</i> .....	337
<i>Quick Review</i> .....	340
<b>WHAT THEN IS FUNCTION OVERRIDING?</b> .....	<b>340</b>
<b>CREATING VIRTUAL FUNCTIONS: THE VIRTUAL KEYWORD</b> .....	<b>340</b>
<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
<b>PURE VIRTUAL FUNCTIONS</b> .....	<b>342</b>
<i>DECLARING PURE VIRTUAL FUNCTIONS</i> .....	342
<b>ABSTRACT CLASSES</b> .....	<b>343</b>
<b>FLEET SIMULATION SOURCE CODE</b> .....	<b>346</b>
<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
<b>MULTIPLE INHERITANCE</b> .....	<b>353</b>
<b>VIRTUAL BASE CLASSES: VIRTUAL INHERITANCE</b> .....	<b>357</b>
<b>GETTING INHERITANCE RIGHT: SOME POINTS TO CONSIDER</b> .....	<b>360</b>
<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
<b>SUMMARY</b> .....	<b>362</b>
<b>SKILL BUILDING EXERCISES</b> .....	<b>362</b>
<b>SUGGESTED PROJECTS</b> .....	<b>364</b>
<b>SELF TEST QUESTIONS</b> .....	<b>366</b>
<b>REFERENCES</b> .....	<b>366</b>
<b>NOTES</b> .....	<b>367</b>

## PART III: IMPLEMENTING POLYMORPHIC BEHAVIOR

### 14 Ad Hoc Polymorphism: Operator Overloading

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

### 15 Static Polymorphism: Templates

<b>INTRODUCTION</b> .....	<b>398</b>
<b>DEFINITION OF TEMPLATE</b> .....	<b>398</b>
<i>FUNCTION TEMPLATES</i> .....	398
<i>CLASS TEMPLATES</i> .....	398
<i>STRUCTURE TEMPLATES</i> .....	398
<b>HOW TEMPLATES WORK: AN ANALOGY</b> .....	<b>398</b>
<b>DECLARING AND USING FUNCTION TEMPLATES</b> .....	<b>399</b>
<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
<b>DECLARING AND USING CLASS TEMPLATES .....</b>	<b>405</b>
<i>A MORE COMPLEX CLASS TEMPLATE EXAMPLE</i> .....	404
Quick Review .....	405
<b>OVERVIEW OF THE STANDARD TEMPLATE LIBRARY (STL) .....</b>	<b>405</b>
CONTAINERS AND CONTAINER ADAPTERS .....	406
ITERATORS.....	407
Algorithms .....	408
Quick Review .....	408
<b>USING STANDARD TEMPLATE LIBRARY COMPONENTS .....</b>	<b>408</b>
Using VECTOR .....	408
Using list .....	411
Quick Review .....	411
<b>SUMMARY .....</b>	<b>412</b>
<b>SKILL BUILDING EXERCISES .....</b>	<b>412</b>
<b>SUGGESTED PROJECTS .....</b>	<b>413</b>
<b>SELF TEST QUESTIONS .....</b>	<b>413</b>
<b>REFERENCES .....</b>	<b>414</b>
<b>NOTES .....</b>	<b>414</b>

## 16 DYNAMIC POLYMORPHISM: OBJECT-ORIENTED PROGRAMMING

<b>INTRODUCTION .....</b>	<b>416</b>
<b>ABSTRACTION: AMPLIFY THE ESSENTIAL—ELIMINATE THE IRRELEVANT .....</b>	<b>416</b>
<b>OBJECT-ORIENTED PROGRAMMING DEFINED .....</b>	<b>417</b>
<b>DYNAMIC POLYMORPHISM DEFINED .....</b>	<b>417</b>
<b>LANGUAGE FEATURES THAT SUPPORT OBJECT-ORIENTED PROGRAMMING .....</b>	<b>417</b>
<b>AN EXAMPLE: CLASS INTERFACE .....</b>	<b>419</b>
Quick Review.....	420
<b>EXTENDED EXAMPLE: ENGINE COMPONENTS REVISITED .....</b>	<b>422</b>
<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
<b>A SHORT STORY .....</b>	<b>435</b>

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

## PART IV: INTERMEDIATE CONCEPTS

### 17 Well Behaved Objects: The Orthodox Canonical Class Form

<b>INTRODUCTION</b> .....	<b>446</b>
<b>WHAT IS A WELL-BEHAVED OBJECT?</b> .....	<b>446</b>
<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
<b>THE ORTHODOX CANONICAL CLASS FORM (OCCF)</b> .....	<b>448</b>
<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
<b>SUMMARY</b> .....	<b>453</b>
<b>SKILL BUILDING EXERCISES</b> .....	<b>453</b>
<b>SUGGESTED PROJECTS</b> .....	<b>454</b>
<b>SELF TEST QUESTIONS</b> .....	<b>455</b>
<b>REFERENCES</b> .....	<b>455</b>
<b>NOTES</b> .....	<b>455</b>

### 18 Mixed Language Programming

<b>INTRODUCTION</b> .....	<b>458</b>
<b>C++ AND C</b> .....	<b>458</b>
<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
<b>C++ AND ASSEMBLY</b> .....	<b>465</b>
<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
<b>C++ AND JAVA: THE JAVA NATIVE INTERFACE (JNI)</b> .....	<b>470</b>

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
<b>SUMMARY .....</b>	<b>478</b>
<b>Skill Building Exercises .....</b>	<b>478</b>
<b>SUGGESTED PROJECTS .....</b>	<b>478</b>
<b>SELF TEST QUESTIONS .....</b>	<b>479</b>
<b>REFERENCES .....</b>	<b>479</b>
<b>NOTES .....</b>	<b>480</b>

## 19 THREE DESIGN PRINCIPLES

<b>INTRODUCTION .....</b>	<b>482</b>
<b>THE PREFERRED CHARACTERISTICS OF AN OBJECT-ORIENTED ARCHITECTURE .....</b>	<b>482</b>
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
<b>THE LISKOV SUBSTITUTION PRINCIPLE &amp; DESIGN BY CONTRACT .....</b>	<b>483</b>
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
<b>THE OPEN-CLOSED PRINCIPLE .....</b>	<b>497</b>
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
<b>The Dependency Inversion Principle</b> .....	<b>500</b>
<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
<b>SUMMARY</b> .....	<b>501</b>
<b>TERMS AND DEFINITIONS</b> .....	<b>502</b>
<b>Skill Building Exercises</b> .....	<b>502</b>
<b>SUGGESTED PROJECTS</b> .....	<b>503</b>
<b>Self Test Questions</b> .....	<b>503</b>
<b>REFERENCES</b> .....	<b>503</b>
<b>NOTES</b> .....	<b>504</b>

## 20 Using A UML Modeling Tool

<b>INTRODUCTION</b> .....	<b>506</b>
<b>THE PURPOSE OF A UML Modeling Tool</b> .....	<b>506</b>
<b>INTRODUCING EMBARCADERO TECHNOLOGIES' DESCRIBE™</b> .....	<b>507</b>
<i>PRIMARY FEATURES</i> .....	507
<b>THE PROJECT SPECIFICATION: ROBOT RAT</b> .....	<b>508</b>
<b>CREATING USE CASE DIAGRAMS</b> .....	<b>509</b>
<i>Adding DOCUMENTATION TO DIAGRAM ELEMENTS</i> .....	511
<i>PROGRAMMER PERSPECTIVE USE CASES</i> .....	512
<b>PAUSING TO CONSIDER DESIGN ISSUES</b> .....	<b>513</b>
<b>CREATING CLASS DIAGRAMS</b> .....	<b>515</b>
<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
<b>ITERATING THROUGH THE DESIGN PROCESS</b> .....	<b>519</b>
<b>CREATING SEQUENCE DIAGRAMS</b> .....	<b>522</b>
<i>PROPER USE OF SEQUENCE DIAGRAMS</i> .....	522
<i>Adding OBJECTS TO SEQUENCE DIAGRAMS</i> .....	522
<i>Adding MESSAGES TO SEQUENCE DIAGRAMS</i> .....	523
<b>GENERATING SOURCE CODE</b> .....	<b>525</b>
<b>REVERSE ENGINEERING</b> .....	<b>527</b>
<i>MERGING SYSTEMS</i> .....	528
<b>LINKING DIAGRAM OBJECTS TO DIAGRAMS</b> .....	<b>529</b>
<b>GENERATING Web PROJECT REPORTS</b> .....	<b>530</b>
<b>SUMMARY</b> .....	<b>531</b>
<b>RobotRat Source Code</b> .....	<b>532</b>
<i>abstraction.h</i> .....	532
<i>abstracmarker.h</i> .....	532
<i>abstraccontrolledobject.h</i> .....	532
<i>position.h</i> .....	533
<i>marker.h</i> .....	533
<i>remotecontrolledobject.h</i> .....	534
<i>abstraccontrolledrodent.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
<b>Skill Building Exercises</b> .....	<b>544</b>
<b>SUGGESTED PROJECTS</b> .....	<b>545</b>
<b>Self Test Questions</b> .....	<b>545</b>
<b>REFERENCES</b> .....	<b>546</b>
<b>NOTES</b> .....	<b>546</b>

## APPENDICES

### Appendix A: Project Approach Strategy Checkoff List

<b>PROJECT APPROACH STRATEGY CHECKOFF LIST</b> .....	<b>549</b>
------------------------------------------------------	------------

### Appendix B: ASCII Table

<b>ASCII TABLE</b> .....	<b>551</b>
--------------------------	------------

### Appendix C: ANSWERS TO SELF-TEST QUESTIONS

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

