

DEVELOPING

C++

SOFTWARE

Second Edition

Russel Winder

University College London, UK

JOHN WILEY & SONS

Chichester • New York • Brisbane • Toronto • Singapore

Contents

1. Introduction	1
1.1. Systems Development and Programming	1
1.2. What is Software?	2
1.3. Documentation in Programming	3
1.4. Perhaps the Simplest Complete C++ Program	4
1.5. Code Layout and Commenting	5
1.6. Producing and Running an Executable	7
2. Abstraction in Programming	9
2.1. Modularity and Abstraction	9
2.2. Specification and Implementation	10
2.3. Types and Support for Abstraction	10
2.4. An Example of Functional Abstraction	11
2.5. The Process of Design and Implementation	12
2.6. Libraries and Input-Output	14
3. The Primitive Data Types	17
3.1. Literals and Variables	17
3.1.1. Naming Variables	17
3.1.2. Initializing Variables	18
3.2. The Primitive Types: Values and Definitions	19
3.2.1. Integer Numbers	19
3.2.2. Characters	22
3.2.3. Real Numbers	23
3.3. Input-Output of Primitive Data Types	25
4. First Steps in Construction: Expressions	27
4.1. Evaluating Expressions	27
4.2. Operations and Expressions	28
4.3. Arithmetic Operators	28
4.3.1. Unary Operators	28
4.3.2. Binary Operators	28
4.3.3. An Example Program	29
4.4. Bit Manipulation Operators	30
4.4.1. Unary Operators	30

4.4.2.	Binary Operators	31
4.4.3.	An Example Program	31
4.5.	Relational and Boolean Operators	32
4.6.	Assignment Operators	33
4.6.1.	Simple Assignment	34
4.6.2.	Compound Assignment	35
4.6.3.	Increment and Decrement Operators	36
4.7.	Expression Evaluation: Priority and Associativity	37
4.8.	Expressions and Type Conversion	38
4.8.1.	Explicit Type Conversion	38
4.8.2.	Implicit Type Conversion	38
4.9.	Constant Variables (Named Constants)	40
5.	Control Flow: Sequence, Decision Making and Iteration	43
5.1.	Sequence	43
5.1.1.	Declarations and Definitions	44
5.1.2.	Expression Statements	45
5.1.3.	The Null Statement	45
5.1.4.	Compound Statements	46
5.2.	Decision Making	46
5.2.1.	Binary Selection	46
5.2.2.	The Conditional Expression	50
5.2.3.	Multi-way Selection	51
5.3.	Iteration	56
5.3.1.	Test-first Iteration	57
5.3.2.	Test-last Iteration	63
5.3.3.	Test-in-the-middle Iteration	64
5.3.4.	Genuinely Infinite Loops	66
5.4.	Uncontrolled Flow	66
6.	Functional Abstraction	69
6.1.	Declaration and Definition of Functions	69
6.2.	Representing Functions Diagrammatically	70
6.3.	Return Types and Returning Values	70
6.3.1.	Assigning Return Values: The Return Statement	71
6.3.2.	Void Functions	73
6.4.	Passing Parameters	73
6.4.1.	Passing by Value	74
6.4.2.	Passing by Reference	75
6.4.3.	References as Return Values	78
6.5.	Default Parameter Values	79
6.6.	Variable Number of Parameters	80
6.7.	In-line Functions	81
6.8.	Developing Functions	82
6.9.	Scope, Visibility and Lifetime	89
6.9.1.	Global and Local Things	90

6.9.2.	Scope	90
6.9.3.	Visibility	91
6.9.4.	Lifetimes	93
6.9.5.	Consequences for Programming	94
6.10.	Control Flow, an Addition: Recursion	95
6.11.	Orders of Growth of Resources	100
7.	A Primitive Data Structure: Arrays	103
7.1.	One-dimensional Arrays	103
7.2.	Initialization of Arrays	106
7.3.	Array Bounds	107
7.4.	Arrays as Function Parameters	108
7.5.	An Example using Arrays: Sorting the Contents of an Array	109
7.6.	Arrays of Characters: Strings	115
7.6.1.	String Literals	115
7.6.2.	Initialization of String Variables	116
7.6.3.	Input-Output of Strings	117
7.7.	Multi-dimensional Arrays	120
8.	User-defined Data Types: Classes	123
8.1.	Class Definitions	123
8.2.	Class Declarations	124
8.3.	Supporting Abstraction: Public and Private Information	125
8.4.	Breaking the Protection System: Friends	126
8.5.	Accessing Members of Classes	126
8.6.	Implementing Member Functions	128
8.7.	Constructors and Destructors	129
8.7.1.	Constructors	130
8.7.2.	Destructors	136
8.8.	Values and Literals	136
8.9.	Initializing Class Arrays	137
8.10.	In-line Member Functions	138
8.11.	Named Constants and Constant Member Functions	139
8.12.	Referring to Self	140
8.13.	Enumerations Revisited	140
9.	A Form of Polymorphism: Overloading	145
9.1.	Overloading Functions	146
9.2.	Using Member Functions to Achieve Polymorphism	147
9.3.	Overloading Operators	147
9.3.1.	Restrictions on Operator Overloading	148
9.3.2.	Overloading and Coercion	150
9.3.3.	Styles of Operator Overloading	152
9.3.4.	Overloading the Indexing Operator	154
9.3.5.	Overloading the () Operator	155
9.3.6.	Overloading Assignment	156

9.3.7.	Overloading Increment and Decrement	157
9.4.	The IOSTREAM Library and Overloading	158
9.5.	A Complex Number Class	165
9.6.	Sub-range Types	169
10.	Becoming More Dynamic: Pointer Types	173
10.1.	What is a Pointer?	173
10.2.	Declaration and Definition of Pointers	174
10.3.	Type Synonyms	176
10.4.	Pointers to Constants	177
10.5.	Constant Pointers	178
10.6.	Pointer Operations, Pointer Arithmetic and Arrays	179
10.7.	Pointers and Arrays as Parameters	183
10.8.	Pointers as Return Values	184
10.9.	Strings Revisited	186
10.9.1.	Initialization of String Variables	186
10.9.2.	Input-Output of Strings, Revisited	188
10.9.3.	Command Line Parameters	188
10.10.	Pointers to User-defined Types	192
10.11.	Pointers to Global Functions	194
10.12.	Pointers to Members of a Class	200
10.13.	Pointer Literals	206
11.	Dynamic Data Types	207
11.1.	The <code>new</code> Operator	207
11.2.	The <code>delete</code> Operator	208
11.3.	Running Out of Memory	211
11.4.	Dynamic Arrays	214
11.5.	Linked Lists	216
11.6.	Using Linked Lists to Build Abstractions	221
11.6.1.	Sorted Lists	221
11.6.2.	Stacks	236
11.6.3.	Queues	242
11.7.	An Example of Binary Trees: A Lexicon	246
11.7.1.	The ADT Approach	249
11.7.2.	A More Object-oriented Approach	259
11.8.	Class-specific Dynamic Storage Allocation	263
11.8.1.	Static Class Members	263
11.8.2.	Keeping Free-lists	265
12.	Becoming Polymorphic: Generic Types	269
12.1.	Template Functions	269
12.2.	Template Classes	276
12.2.1.	Pointers as Data Items	281
12.2.2.	User-defined Pointer Types	283
12.2.3.	Non-class Formals in Templates	291

12.3. Heterogeneous Collections	292
12.3.1. Unions	293
12.3.2. Anonymous Unions	294
12.3.3. Using Unions	295
13. Extendible Abstractions: Inheritance	299
13.1. The Jargon of Inheritance	300
13.2. Defining Sub-classes	300
13.3. Impending Polymorphism	304
13.4. Controlling Inheritance	305
13.4.1. Inheritance and Scope	305
13.4.2. Types of Inheritance	306
13.4.3. Inheritance and Friends	308
13.5. Multiple Inheritance	308
13.6. Use of :: for Accessing Variables and Functions	311
13.7. Pointers and References to Types involving Inheritance	313
13.8. Virtual Functions	317
13.9. Abstract Virtual Functions and Abstract Classes	320
13.10. Inheritance as Access Restriction	325
13.11. Inheritance and Code Reuse	327
13.12. Inheritance and Discriminated Types	332
13.13. Object-oriented Programming	341
14. Files: Generally Useful Things	347
14.1. Program Structure Revisited: Files and Compilation Control	347
14.1.1. The Compilation Process	347
14.1.2. Compilation Control	348
14.1.3. Scope and Files	355
14.1.4. Libraries	356
14.1.5. Automating Compilation Control	357
14.2. Files as Data Repositories	358
14.2.1. Basic Concepts in File Handling	358
14.2.2. IOSTREAM Input-Output	359
14.2.3. Processing Files	361
14.2.4. Filters	369
15. Bits and Pieces, Odds and Ends	371
15.1. Bit Manipulation	371
15.1.1. Saving Space	371
15.1.2. Handling Hardware	372
15.1.3. Bit-fields	373
15.1.4. Volatile	374
15.2. Optimization	375
15.2.1. Code Tracing	376
15.2.2. Register Variables	377
15.2.3. Use of Assembler Code	378

15.3. An Ending	378
Appendix A. Bibliography	379
A.1. C++ as a Language	379
A.2. Object-oriented Programming	379
A.3. Programming in Scheme (Lisp)	380
A.4. Algorithms	380
A.5. Software Engineering	381
A.6. Human-Computer Interaction	381
A.7. Specification of Systems	382
A.8. Mathematics	382
A.9. Computer Hardware	382
Appendix B. Answers to the Exercises	383
Exercise 3.1. – Page 26	383
Exercise 3.2. – Page 26	384
Exercise 4.1. – Page 30	385
Exercise 4.2. – Page 37	385
Exercise 4.3. – Page 37	385
Exercise 5.1. – Page 49	386
Exercise 5.2. – Page 68	387
Exercise 5.3. – Page 68	389
Exercise 6.1. – Page 76	390
Exercise 6.2. – Page 76	391
Exercise 6.3. – Page 89	391
Exercise 6.4. – Page 96	392
Exercise 6.5. – Page 102	393
Exercise 6.6. – Page 102	394
Exercise 7.1. – Page 114	394
Exercise 7.2. – Page 115	395
Exercise 8.1. – Page 134	396
Exercise 8.2. – Page 140	397
Exercise 8.3. – Page 144	398
Exercise 9.1. – Page 155	398
Exercise 9.2. – Page 164	400
Exercise 9.3. – Page 169	402
Exercise 9.4. – Page 172	409
Exercise 9.5. – Page 172	412
Exercise 10.1. – Page 176	417
Exercise 10.2. – Page 179	418
Exercise 10.3. – Page 179	418
Exercise 10.4. – Page 182	418
Exercise 10.5. – Page 190	419
Exercise 10.6. – Page 206	419
Exercise 11.1. – Page 216	424
Exercise 11.2. – Page 232	427

Exercise 11.3. – Page 236	429
Exercise 11.4. – Page 236	434
Exercise 11.5. – Page 240	438
Exercise 11.6. – Page 241	440
Exercise 11.7. – Page 242	440
Exercise 11.8. – Page 246	442
Exercise 11.9. – Page 252	446
Exercise 11.10. – Page 258	451
Exercise 11.11. – Page 262	454
Exercise 11.12. – Page 263	458
Exercise 11.13. – Page 263	458
Exercise 11.14. – Page 264	458
Exercise 12.1. – Page 281	459
Exercise 12.2. – Page 289	464
Exercise 12.3. – Page 291	464
Exercise 13.1. – Page 332	473
Exercise 13.2. – Page 346	474
Exercise 14.1. – Page 368	478

Index	485
-------------	-----