

Accelerated C++

Practical Programming by Example

**Andrew Koenig
Barbara E. Moo**

◆ **Addison-Wesley**

**Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City**

Contents

Preface		xi
Chapter 0	Getting started	1
0.1	Comments	1
0.2	#include	2
0.3	The main function	2
0.4	Curly braces	2
0.5	Using the standard library for output	3
0.6	The return statement	3
0.7	A slightly deeper look	4
0.8	Details	5
Chapter 1	Working with strings	9
1.1	Input	9
1.2	Framing a name	11
1.3	Details	14
Chapter 2	Looping and counting	17
2.1	The problem	17
2.2	Overall structure	18
2.3	Writing an unknown number of rows	18
2.4	Writing a row	22
2.5	The complete framing program	27
2.6	Counting	30
2.7	Details	31

Chapter 3	Working with batches of data	35
3.1	Computing student grades	35
3.2	Using medians instead of averages	41
3.3	Details	48
Chapter 4	Organizing programs and data	51
4.1	Organizing computations	51
4.2	Organizing data	61
4.3	Putting it all together	65
4.4	Partitioning the grading program	68
4.5	The revised grading program	70
4.6	Details	71
Chapter 5	Using sequential containers and analyzing strings	75
5.1	Separating students into categories	75
5.2	Iterators	79
5.3	Using iterators instead of indices	82
5.4	Rethinking our data structure for better performance	84
5.5	The list type	85
5.6	Taking strings apart	87
5.7	Testing our split function	90
5.8	Putting strings together	91
5.9	Details	96
Chapter 6	Using library algorithms	101
6.1	Analyzing strings	101
6.2	Comparing grading schemes	110
6.3	Classifying students, revisited	116
6.4	Algorithms, containers, and iterators	120
6.5	Details	121
Chapter 7	Using associative containers	123
7.1	Containers that support efficient look-up	123
7.2	Counting words	124
7.3	Generating a cross-reference table	126
7.4	Generating sentences	129
7.5	A note on performance	136
7.6	Details	137
Chapter 8	Writing generic functions	139
8.1	What is a generic function?	139
8.2	Data-structure independence	143
8.3	Input and output iterators	150
8.4	Using iterators for flexibility	152
8.5	Details	153

Chapter 9	Defining new types	155
9.1	Student_info revisited	155
9.2	Class types	156
9.3	Protection	160
9.4	The Student_info class	163
9.5	Constructors	164
9.6	Using the Student_info class	166
9.7	Details	167
Chapter 10	Managing memory and low-level data structures	169
10.1	Pointers and arrays	169
10.2	String literals revisited	176
10.3	Initializing arrays of character pointers	177
10.4	Arguments to main	179
10.5	Reading and writing files	180
10.6	Three kinds of memory management	182
10.7	Details	185
Chapter 11	Defining abstract data types	187
11.1	The Vec class	187
11.2	Implementing the Vec class	188
11.3	Copy control	195
11.4	Dynamic Vecs	202
11.5	Flexible memory management	203
11.6	Details	209
Chapter 12	Making class objects act like values	211
12.1	A simple string class	212
12.2	Automatic conversions	213
12.3	Str operations	214
12.4	Some conversions are hazardous	221
12.5	Conversion operators	222
12.6	Conversions and memory management	223
12.7	Details	225
Chapter 13	Using inheritance and dynamic binding	227
13.1	Inheritance	227
13.2	Polymorphism and virtual functions	232
13.3	Using inheritance to solve our problem	237
13.4	A simple handle class	243
13.5	Using the handle class	247
13.6	Subtleties	248
13.7	Details	250

Chapter 14	Managing memory (almost) automatically	253
14.1	Handles that copy their objects	254
14.2	Reference-counted handles	260
14.3	Handles that let you decide when to share data	263
14.4	An improvement on controllable handles	264
14.5	Details	268
Chapter 15	Revisiting character pictures	269
15.1	Design	269
15.2	Implementation	278
15.3	Details	288
Chapter 16	Where do we go from here?	291
16.1	Use the abstractions you have	291
16.2	Learn more	293
Appendix A	Language details	295
A.1	Declarations	295
A.2	Types	299
A.3	Expressions	305
A.4	Statements	308
Appendix B	Library summary	311
B.1	Input-output	311
B.2	Containers and iterators	314
B.3	Algorithms	322
Index		325