Exploring C++

Alice E. Fischer

University of New Haven

January 2009
(Revised to September 1, 2010)

Copyright ©2009

by Alice E. Fischer

All rights reserved. No part of this manuscript may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the authors.

Contents

1 Preamble **1**

1.1 Commandments . . . 1
1.2 Style . . . 1
1.2.1 General Rules . . . 1
1.2.2 Naming . . . 2
1.2.3 Usage . . . 2
1.2.4 Indentation . . . 3
1.2.5 Function Definitions . . . 3
1.2.6 Types, type definitions and struct . . . 4
1.2.7 Using the Tools Library . . . 5

2 Issues and Overview **7**

2.1 Why did C need a ++? . . . 7
2.1.1 Design Goals for C . . . 7
2.1.2 C++ Extends C. . . . 7
2.1.3 Modeling. . . . 8
2.2 Object Oriented Principles. . . . 9
2.3 Important Differences . . . 10
2.4 Generic Insertion Sort . . . 12
2.4.1 Main Program . . . 13
2.4.2 The DataPack Header File . . . 13
2.4.3 The Data Pack Functions . . . 14

3 C++ I/O for the C Programmer **19**

3.1 Familiar Things in a New Language . . . 19
3.2 Include Files . . . 19
3.3 Streams and Files . . . 19
3.3.1 Common Stream Errors, Misconceptions, and Problems . . . 21
3.4 Input . . . 21
3.5 Output . . . 23
3.6 I/O and File Handling Demonstration . . . 24
3.7 End of File and Error Handling . . . 29
3.7.1 Using the command line. . . . 29
3.7.2 Reading Lines of Text . . . 30
3.7.3 EOF and error handling with numeric input. . . . 32
3.8 Assorted short notes. . . . 33

4 An Introduction to Classes **35**

4.1 Class Basics . . . 35
4.1.1 Data members. . . . 35
4.1.2 Functions . . . 37
4.2 Inline Functions . . . 38
4.2.1 Code Files and Header Files . . . 39

4.3 Declaration, Implementation, and Application of a Stack Class 40
4.3.1 The Input and Output (banners have been deleted). 40
4.3.2 The main function: main.c and main.cpp 41
4.3.3 The Brackets class. 43
4.3.4 Class Declaration: token.h and token.hpp 46
4.3.5 The Stack Class 48

5 Functions and Parameter Passing 53
5.1 Function Calls 53
5.2 Parameter Passing 54
5.2.1 Three Odd Functions 54
5.2.2 Calling The Odd Functions 54
5.2.3 Parameter Passing Mechanisms 55
5.3 Function Returns 57
5.3.1 L-values and r-values. 57
5.3.2 Using L-values. 58
5.3.3 Using Return Values in Assignment Statements 58

6 Objects, Allocation, and Disasters 61
6.1 Objects: Static, Automatic, Dynamic, and Mixed 61
6.1.1 Storage Class 61
6.1.2 Assignment and Copying 61
6.2 Static Class Members 62
6.3 Common Kinds of Failure 63
6.4 Causes and Cures 64
6.4.1 A shallow copy disaster. 64
6.4.2 Dangling pointers. 65
6.4.3 Storing things in limbo. 65
6.4.4 The wrong ways to delete. 66
6.4.5 Walking on memory. 67
6.4.6 NULL pointers. 68

7 C++ Bells and Whistles 71
7.1 Optional Parameters 71
7.1.1 A Short Example 71
7.2 Const Qualifiers 72
7.3 Operator Extensions 74
7.3.1 Global Binary Operator Extensions 74
7.3.2 Binary Operator Extensions in a Class. 74
7.3.3 Unary operators. 76

8 Interacting Classes 77
8.1 The Roles of a Class 77
8.1.1 Finding the Classes 77
8.1.2 Diagramming One Class 78
8.2 Class Relationships 78
8.2.1 Composition. 78
8.2.2 Aggregation. 78
8.2.3 Association. 78
8.2.4 Derivation. 79
8.2.5 Friendship. 79
8.2.6 An example. 79
8.2.7 Elementary Design Principles 79
8.3 The Program: Making a Bar Graph 81
8.3.1 Specification 81

8.3.2 The Main Program and Output . . . 82
8.3.3 The Data Structure and UML Diagrams . . . 83
8.3.4 Class Item . . . 84
8.3.5 Class Graph . . . 85
8.3.6 Classes Row and Cell . . . 87
8.4 An Event Trace . . . 90

9 Array Data Structures 93
9.1 Allocation and Deallocation of Arrays. . . . 94
9.2 The Flexible Array Data Structure . . . 96
9.2.1 Implementation in C++ . . . 96
9.2.2 Implementation in C . . . 99
9.3 Ragged Arrays . . . 99
9.3.1 Dynamic Ragged Arrays . . . 100
9.4 The StringStore Data Structure . . . 101
9.4.1 The StringStore and Pool Classes. . . . 101
9.5 The StringArray . . . 103
9.6 Hashing . . . 107
9.6.1 The Hash Table Array . . . 107
9.6.2 Hash Functions. . . . 108
9.7 Example: Combining Several Data Structures . . . 109
9.7.1 The Main Program . . . 110
9.7.2 The Dictionary Class . . . 111
9.7.3 The FlexArray and StringStore Classes . . . 114
9.7.4 A Better Way . . . 116

10 Construction and Destruction 117
10.1 New C++ Concepts . . . 117
10.1.1 Talking About Yourself . . . 117
10.1.2 Constructor Initializers . . . 117
10.2 Dynamic Allocation Incurs Overhead. . . . 118
10.2.1 Allocation Example: a Van Containing Boxes . . . 119
10.2.2 How Allocation and Deallocation Work . . . 122
10.3 Construction of C++ Objects . . . 124
10.3.1 A Class Object is Constructed in Layers . . . 124
10.3.2 Layering Demo Program . . . 124
10.4 Constructors and Construction . . . 126
10.4.1 A Variety of Constructors . . . 127
10.5 Destruction Problems . . . 130

11 Modules and Makefiles 131
11.1 Modular Organization and makefiles. . . . 131
11.1.1 History . . . 131
11.1.2 General Concepts . . . 132
11.1.3 Enumerations and External Objects . . . 133
11.1.4 Rules . . . 134
11.2 A Makefile defines the project. . . . 134
11.2.1 Making the Bargraph Application . . . 135

12 Derived Classes 137
12.1 How Derivation is Used . . . 137
12.1.1 Resolving Ambiguous Names . . . 138
12.1.2 Ctor Initializers . . . 138
12.2 Visibility and Protection Level . . . 139
12.2.1 Inherited Functions . . . 140

12.3 Class Derivation Demo . . . 141
12.3.1 Inherited Data Members . . . 141

13 Templates 145
13.1 Basic Template Syntax . . . 145
13.2 A Template for FlexArrays . . . 146
13.3 Adapting a Template . . . 147
13.4 A Precedence Parser: Instantiation of a Template . . . 149
13.4.1 The Operator Class . . . 153
13.4.2 The Main Program . . . 153
13.4.3 UML for Templates . . . 155
13.5 The Standard Template Library . . . 156
13.6 Containers . . . 156
13.7 Iterators . . . 158
13.8 Using Simple STL Classes . . . 158
13.8.1 String . . . 158
13.8.2 Vector . . . 159
13.8.3 Map . . . 161

14 Derivation and Inheritance 165
14.1 Playing . . . 165
14.1.1 The Hangman Application . . . 165
14.1.2 Hangman: The Main Program . . . 166
14.1.3 Call Graphs . . . 167
14.1.4 UML Diagrams: A View of the Class Relationships . . . 168
14.1.5 The Hangman Data Structures . . . 170
14.2 Hangman: The Game and Board Classes . . . 170
14.2.1 The Game Class . . . 171
14.2.2 The Board Class . . . 172
14.3 The Word Classes . . . 175
14.3.1 The BaseWord Declaration . . . 175
14.3.2 The Derived Word Classes . . . 176
14.4 RandString Adapts a Reusable Data Structure . . . 178
14.4.1 The RandString Declaration . . . 178

15 Polymorphism and Virtual Functions 181
15.1 Basic Concepts . . . 181
15.1.1 Definitions . . . 181
15.1.2 Virtual functions. . . . 181
15.2 Polymorphic Classes . . . 182
15.3 Creating a Polymorphic Container . . . 183
15.3.1 Container: An Abstract Class . . . 184
15.3.2 Linear: A Polymorphic Class . . . 184
15.3.3 Cell: The Helper Class . . . 187
15.3.4 Exam: The Actual Data Class . . . 187
15.3.5 Class diagram. . . . 188
15.4 Stack: Fully Specific . . . 188
15.5 Queue: Fully Specific . . . 190
15.6 A Main Program and its Output . . . 192

16 Abstract Classes and Multiple Inheritance 195
16.1 An Abstract Class Defines Expectations . . . 195
16.2 Abstraction Example: an Ordered Type . . . 195
16.3 Multiple Inheritance . . . 196
16.3.1 Item: The Data Class . . . 196
16.4 Linear Containers You Can Search . . . 198
16.4.1 PQueue: a Sorted Linear Container . . . 198
16.4.2 List: An Unordered Container . . . 199
16.4.3 The Main Program . . . 200
16.5 C++ Has Four Kinds of Casts . . . 201
16.5.1 Static Casts . . . 201
16.5.2 Reinterpret Casts . . . 202
16.5.3 Const Casts . . . 202
16.5.4 Dynamic Casts . . . 203
16.6 Virtual Inheritance and Dynamic Casts . . . 203
16.6.1 Virtual Inheritance . . . 204
16.6.2 Dynamic Casts on the Donut . . . 206

17 Exceptions 209
17.1 Handling Errors in a Program . . . 209
17.1.1 What an Exception Handler Can Do . . . 210
17.2 Defining Exceptions in C++ . . . 210
17.2.1 Defining an Exception Type . . . 210
17.2.2 Exception Specifications . . . 212
17.2.3 Playing card demo. . . . 212
17.2.4 Throwing an Exception . . . 212
17.2.5 Catching an Exception . . . 214
17.2.6 Built-in Exceptions . . . 215
17.2.7 Summary . . . 216

18 Design Patterns 217
18.1 Definitions and General OO Principles . . . 217
18.1.1 Definitions . . . 217
18.2 General OO Principles . . . 217
18.3 Patterns . . . 218
18.3.1 GRASP: General Responsibility Assignment Software Patterns . . . 218
18.4 More Complex Design Patterns . . . 218