

Exploring C++

Alice E. Fischer

University of New Haven

January 2009

(Revised to February 27, 2016)

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	3
1.2.4	Indentation	3
1.2.5	Function Definitions	4
1.2.6	Types, type definitions and struct	5
1.2.7	Projects	5
1.2.8	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	11
3	C++ I/O for the C Programmer	15
3.1	Familiar Things in a New Language	15
3.2	Include Files	15
3.3	Streams and Files	15
3.4	Input	18
3.5	Output	19
3.6	Generic Insertion Sort	21
3.6.1	Main Program	22
3.6.2	The MemList Header File	23
3.6.3	The MemList Functions	23
3.7	End of File and Error Handling	25
3.7.1	Using the command line.	25
3.7.2	Reading Lines of Text	26
3.7.3	EOF and error handling with numeric input.	29
3.8	Assorted short notes.	30
3.9	I/O and File Handling Demonstration	31
4	An Introduction to Classes	37
4.1	What is OO?	37
4.2	Code Files and Header Files	38
4.3	Class Basics	38
4.3.1	Data members.	39
4.3.2	Functions	40

4.3.3	Typical Class Methods	42
4.4	Inline Functions	42
4.5	Declaration, Implementation, and Application of a Stack Class	43
4.5.1	The Input and Output (banners have been deleted).	43
4.5.2	The main function, from file main.cpp	44
4.5.3	The Brackets class.	45
4.5.4	The Token Class	48
4.5.5	The Stack Class	50
4.6	A Stack Template	53
5	Functions and Parameter Passing	57
5.1	Function Calls	57
5.2	Parameter Passing	58
5.2.1	Three Odd Functions	58
5.2.2	Calling The Odd Functions	58
5.2.3	Parameter Passing Mechanisms	59
5.3	Function Returns	61
5.3.1	L-values and r-values.	61
5.3.2	Using L-values.	62
5.3.3	Using Return Values in Assignment Statements	62
6	Objects, Allocation, and Disasters	65
6.1	Objects: Static, Automatic, Dynamic, and Mixed	65
6.1.1	Storage Class	65
6.1.2	Assignment and Copying	65
6.2	Static Class Members	66
6.3	Common Kinds of Failure	67
6.4	Causes and Cures	68
6.4.1	A shallow copy disaster.	68
6.4.2	Dangling pointers.	69
6.4.3	Storing things in limbo.	69
6.4.4	The wrong ways to delete.	70
6.4.5	Walking on memory.	71
6.4.6	Null pointers.	72
7	C++ Bells and Whistles	75
7.1	Optional Parameters	75
7.1.1	A Short Example	75
7.2	Const Qualifiers	76
7.3	Operator Extensions	77
7.3.1	Global Binary Operator Extensions	77
7.3.2	Binary Operator Extensions in a Class.	78
7.3.3	Unary operators.	79
7.4	Static Operators in a Class	80
8	Interacting Classes	83
8.1	The Roles of a Class	83
8.1.1	Finding the Classes	83
8.1.2	Diagramming One Class	84
8.2	Class Relationships	84
8.2.1	Composition.	84
8.2.2	Aggregation.	84
8.2.3	Association.	84
8.2.4	Derivation.	85
8.2.5	Friendship.	85

8.2.6	An example.	85
8.2.7	Elementary Design Principles	85
8.3	The Program: Making a Bar Graph	87
8.3.1	Specification	87
8.3.2	The Main Program and Output	88
8.3.3	The Data Structure and UML Diagrams	90
8.3.4	Class Item	91
8.3.5	Class Graph	92
8.3.6	Classes Row and Cell	94
8.4	An Event Trace	97
9	Array Data Structures	101
9.1	Allocation and Deallocation of Arrays.	102
9.2	The Flexible Array Data Structure	104
9.2.1	Implementation in C++	104
9.2.2	Implementation in C	107
9.3	Ragged Arrays	107
9.3.1	Dynamic Ragged Arrays	108
9.4	The StringStore Data Structure	109
9.4.1	The StringStore and Pool Classes.	109
9.5	The StringArray	111
9.6	Hashing	115
9.6.1	The Hash Table Array	115
9.6.2	Hash Functions.	116
9.7	Example: Combining Several Data Structures	117
9.7.1	The Main Program	118
9.7.2	The Dictionary Class	119
9.7.3	The FlexArray and StringStore Classes	122
9.7.4	A Better Way	124
10	Construction and Destruction	125
10.1	New C++ Concepts	125
10.1.1	Talking About Yourself	125
10.1.2	Constructor Initializers	125
10.2	Dynamic Allocation Incurs Overhead.	126
10.2.1	Allocation Example: a Van Containing Boxes	127
10.2.2	How Allocation and Deallocation Work	130
10.3	Construction of C++ Objects	132
10.3.1	A Class Object is Constructed in Layers	132
10.3.2	Layering Demo Program	132
10.4	Constructors and Construction	134
10.4.1	A Variety of Constructors	135
10.5	Destruction Problems	138
11	Modules and Makefiles	139
11.1	Modular Organization and makefiles.	139
11.1.1	History	139
11.1.2	General Concepts	140
11.1.3	Enumerations and External Objects	141
11.1.4	Rules	142
11.2	A Makefile defines the project.	142
11.2.1	Making the Bargraph Application	143

12	Derived Classes	145
12.1	How Derivation is Used	145
12.1.1	Resolving Ambiguous Names	146
12.1.2	Ctor Initializers	146
12.2	Visibility and Protection Level	147
12.2.1	Inherited Functions	148
12.3	Class Derivation Demo	149
12.3.1	Inherited Data Members	149
13	Templates	153
13.1	Basic Template Syntax	153
13.2	A Template for FlexArrays	154
13.3	Adapting a Template	155
13.4	A Precedence Parser: Instantiation of a Template	157
13.4.1	The Operator Class	161
13.4.2	The Main Program	161
13.4.3	UML for Templates	163
13.5	The Standard Template Library	164
13.6	Containers	164
13.7	Iterators	166
13.8	Using Simple STL Classes	166
13.8.1	String	166
13.8.2	Vector	167
13.8.3	Map	169
14	Derivation and Inheritance	173
14.1	Playing	173
14.1.1	The Hangman Application	173
14.1.2	Hangman: The Main Program	174
14.1.3	Call Graphs	175
14.1.4	UML Diagrams: A View of the Class Relationships	176
14.1.5	The Hangman Data Structures	178
14.2	Hangman: The Game and Board Classes	178
14.2.1	The Game Class	179
14.2.2	The Board Class	180
14.3	The Word Classes	183
14.3.1	The BaseWord Declaration	183
14.3.2	The Derived Word Classes	184
14.4	RandString Adapts a Reusable Data Structure	186
14.4.1	The RandString Declaration	186
15	Polymorphism and Virtual Functions	189
15.1	Basic Concepts	189
15.1.1	Definitions	189
15.1.2	Virtual functions.	189
15.2	Polymorphic Classes	190
15.3	Creating a Polymorphic Container	191
15.3.1	Container: An Abstract Class	192
15.3.2	Linear: A Polymorphic Class	192
15.3.3	Cell: The Helper Class	195
15.3.4	Exam: The Actual Data Class	195
15.3.5	Class diagram.	196
15.4	Stack: Fully Specific	196
15.5	Queue: Fully Specific	198

15.6	A Main Program and its Output	200
16	Abstract Classes and Multiple Inheritance	203
16.1	An Abstract Class Defines Expectations	203
16.2	Abstraction Example: an Ordered Type	203
16.3	Multiple Inheritance	204
16.3.1	Item: The Data Class	204
16.4	Linear Containers You Can Search	206
16.4.1	PQueue: a Sorted Linear Container	206
16.4.2	List: An Unordered Container	207
16.4.3	The Main Program	208
16.5	C++ Has Four Kinds of Casts	209
16.5.1	Static Casts	209
16.5.2	Reinterpret Casts	210
16.5.3	Const Casts	210
16.5.4	Dynamic Casts	211
16.6	Virtual Inheritance and Dynamic Casts	211
16.6.1	Virtual Inheritance	212
16.6.2	Dynamic Casts on the Donut	214
17	Exceptions	217
17.1	Handling Errors in a Program	217
17.1.1	What an Exception Handler Can Do	218
17.2	Defining Exceptions in C++	218
17.2.1	Defining an Exception Type	218
17.2.2	Exception Specifications	219
17.2.3	Playing card demo.	220
17.2.4	Throwing an Exception	221
17.2.5	Catching an Exception	222
17.2.6	Built-in Exceptions	224
17.2.7	Summary	224
18	Design Patterns	225
18.1	Definitions and General OO Principles	225
18.1.1	Definitions	225
18.2	General OO Principles	225
18.3	Patterns	226
18.3.1	GRASP: General Responsibility Assignment Software Patterns	226
18.4	More Complex Design Patterns	226
19	The Template Library	231
19.1	The Standard Template Library	231
19.2	Iterators	231
19.3	Containers	232
19.3.1	String	233
19.3.2	Vector	234
19.3.3	Map	236

