

Exploring C++

Alice E. Fischer

University of New Haven

January 2009

(Revised to September 5, 2012)

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	Manipulators	23
3.6	Output	23
3.7	I/O and File Handling Demonstration	24
3.8	End of File and Error Handling	29
3.8.1	Using the command line.	29
3.8.2	Reading Lines of Text	31
3.8.3	EOF and error handling with numeric input.	33
3.9	Assorted short notes.	34
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
7.4	Static Operators in a Class	77
8	Interacting Classes	79
8.1	The Roles of a Class	79
8.1.1	Finding the Classes	79
8.1.2	Diagramming One Class	80
8.2	Class Relationships	80
8.2.1	Composition.	80
8.2.2	Aggregation.	80
8.2.3	Association.	80
8.2.4	Derivation.	81
8.2.5	Friendship.	81
8.2.6	An example.	81
8.2.7	Elementary Design Principles	81

8.3	The Program: Making a Bar Graph	83
8.3.1	Specification	83
8.3.2	The Main Program and Output	84
8.3.3	The Data Structure and UML Diagrams	86
8.3.4	Class Item	87
8.3.5	Class Graph	88
8.3.6	Classes Row and Cell	90
8.4	An Event Trace	93
9	Array Data Structures	97
9.1	Allocation and Deallocation of Arrays.	98
9.2	The Flexible Array Data Structure	100
9.2.1	Implementation in C++	100
9.2.2	Implementation in C	103
9.3	Ragged Arrays	103
9.3.1	Dynamic Ragged Arrays	104
9.4	The StringStore Data Structure	105
9.4.1	The StringStore and Pool Classes.	105
9.5	The StringArray	107
9.6	Hashing	111
9.6.1	The Hash Table Array	111
9.6.2	Hash Functions.	112
9.7	Example: Combining Several Data Structures	113
9.7.1	The Main Program	114
9.7.2	The Dictionary Class	115
9.7.3	The FlexArray and StringStore Classes	118
9.7.4	A Better Way	120
10	Construction and Destruction	121
10.1	New C++ Concepts	121
10.1.1	Talking About Yourself	121
10.1.2	Constructor Initializers	121
10.2	Dynamic Allocation Incurs Overhead.	122
10.2.1	Allocation Example: a Van Containing Boxes	123
10.2.2	How Allocation and Deallocation Work	126
10.3	Construction of C++ Objects	128
10.3.1	A Class Object is Constructed in Layers	128
10.3.2	Layering Demo Program	128
10.4	Constructors and Construction	130
10.4.1	A Variety of Constructors	131
10.5	Destruction Problems	134
11	Modules and Makefiles	135
11.1	Modular Organization and makefiles.	135
11.1.1	History	135
11.1.2	General Concepts	136
11.1.3	Enumerations and External Objects	137
11.1.4	Rules	138
11.2	A Makefile defines the project.	138
11.2.1	Making the Bargraph Application	139

12	Derived Classes	141
12.1	How Derivation is Used	141
12.1.1	Resolving Ambiguous Names	142
12.1.2	Ctor Initializers	142
12.2	Visibility and Protection Level	143
12.2.1	Inherited Functions	144
12.3	Class Derivation Demo	145
12.3.1	Inherited Data Members	145
13	Templates	149
13.1	Basic Template Syntax	149
13.2	A Template for FlexArrays	150
13.3	Adapting a Template	151
13.4	A Precedence Parser: Instantiation of a Template	153
13.4.1	The Operator Class	157
13.4.2	The Main Program	157
13.4.3	UML for Templates	159
13.5	The Standard Template Library	160
13.6	Containers	160
13.7	Iterators	162
13.8	Using Simple STL Classes	162
13.8.1	String	162
13.8.2	Vector	163
13.8.3	Map	165
14	Derivation and Inheritance	169
14.1	Playing	169
14.1.1	The Hangman Application	169
14.1.2	Hangman: The Main Program	170
14.1.3	Call Graphs	171
14.1.4	UML Diagrams: A View of the Class Relationships	172
14.1.5	The Hangman Data Structures	174
14.2	Hangman: The Game and Board Classes	174
14.2.1	The Game Class	175
14.2.2	The Board Class	176
14.3	The Word Classes	179
14.3.1	The BaseWord Declaration	179
14.3.2	The Derived Word Classes	180
14.4	RandString Adapts a Reusable Data Structure	182
14.4.1	The RandString Declaration	182
15	Polymorphism and Virtual Functions	185
15.1	Basic Concepts	185
15.1.1	Definitions	185
15.1.2	Virtual functions.	185
15.2	Polymorphic Classes	186
15.3	Creating a Polymorphic Container	187
15.3.1	Container: An Abstract Class	188
15.3.2	Linear: A Polymorphic Class	188
15.3.3	Cell: The Helper Class	191
15.3.4	Exam: The Actual Data Class	191
15.3.5	Class diagram.	192
15.4	Stack: Fully Specific	192
15.5	Queue: Fully Specific	194

15.6	A Main Program and its Output	196
16	Abstract Classes and Multiple Inheritance	199
16.1	An Abstract Class Defines Expectations	199
16.2	Abstraction Example: an Ordered Type	199
16.3	Multiple Inheritance	200
16.3.1	Item: The Data Class	200
16.4	Linear Containers You Can Search	202
16.4.1	PQueue: a Sorted Linear Container	202
16.4.2	List: An Unordered Container	203
16.4.3	The Main Program	204
16.5	C++ Has Four Kinds of Casts	205
16.5.1	Static Casts	205
16.5.2	Reinterpret Casts	206
16.5.3	Const Casts	206
16.5.4	Dynamic Casts	207
16.6	Virtual Inheritance and Dynamic Casts	207
16.6.1	Virtual Inheritance	208
16.6.2	Dynamic Casts on the Donut	210
17	Exceptions	213
17.1	Handling Errors in a Program	213
17.1.1	What an Exception Handler Can Do	214
17.2	Defining Exceptions in C++	214
17.2.1	Defining an Exception Type	214
17.2.2	Exception Specifications	216
17.2.3	Playing card demo.	216
17.2.4	Throwing an Exception	216
17.2.5	Catching an Exception	218
17.2.6	Built-in Exceptions	219
17.2.7	Summary	220
18	Design Patterns	221
18.1	Definitions and General OO Principles	221
18.1.1	Definitions	221
18.2	General OO Principles	221
18.3	Patterns	222
18.3.1	GRASP: General Responsibility Assignment Software Patterns	222
18.4	More Complex Design Patterns	222

