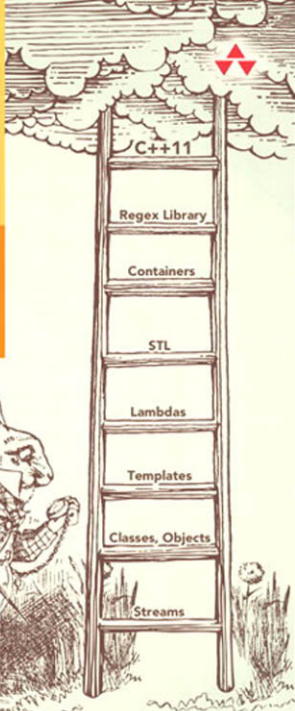


C++

FOR THE IMPATIENT



BRIAN OVERLAND

C++ for the Impatient

Brian Overland

◆◆ Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

C++ for the Impatient

Table of Contents

Contents

Preface

Acknowledgments

About the Author

1 C++ Fundamentals

1.1 Elements of a C++ Program

1.1.1 The #include Directive

1.1.2 The using Statement

1.1.3 The main Function

1.2 Dealing with Flashing Console

1.3 Working with Microsoft Visual Studio

1.4 Doing More with C++

1.5 Adding Simple Variable Declarations

1.6 Introduction to C++ Control Structures

1.6.1 Making Decisions with if

1.6.2 Looping with while

1.7 General Structure of a C++ Program

1.8 More about Namespaces

1.9 Some Comments about Comments

1.9.1 C++ Comments (Line Comments)

1.9.2 C-Language-Style Comments

1.10 Sample App: Adding Machine

Exercises

Table of Contents

1.11 Sample App: Calculating Phi

Exercises

2 Data

2.1 Declaring Simple Variables

2.2 Primitive Data Types

2.2.1 Integer Types: Guidelines

2.2.2 Floating-Point Types: Guidelines

2.3 Symbolic Names (Symbols)

2.4 Numeric Literals

2.5 Mixing Numeric Types

2.5.1 Integer versus Floating Point

2.5.2 bool versus Integer Types

2.5.3 Signed versus Unsigned Integers

2.6 String and Character Literals

2.6.1 Single-Quoted Characters

2.6.2 Double-Quoted Strings

2.6.3 Special Characters (Escape Sequences)

2.6.4 Wide-Character Strings

2.6.5 Raw String Literals (C++11)

2.7 Data Declarations: The Complete Syntax

2.8 Enumerated Types

2.9 Special Declarations (typedef, auto, decltype)

2.9.1 The typedef Keyword

2.9.2 The auto and decltype Keywords (C++11)

2.10 Sample App: Type Promotion

Exercises

3 Operators

3.1 Precedence, Associativity, and Lvalues

Table of Contents

3.1.1 Precedence

3.1.2 Associativity

3.1.3 Lvalues

3.2 Concise Summary of Operators

3.3 Operators in Detail

3.3.1 The Scope Operator (::)

3.3.2 Data-Access and Other High-Precedence Operators

3.3.3 Prefix Operators

3.3.4 Pointer-to-Member Operators

3.3.5 Multiply and Divide

3.3.6 Add and Subtract

3.3.7 Shift Operators

3.3.8 Less Than and Greater Than

3.3.9 Test for Equality

3.3.10 Bitwise and Logical Conjunctions

3.3.11 The Conditional Operator (?:)

3.3.12 Assignment Operators

3.3.13 The throw Operator

3.3.14 The Join Operator (,)

3.4 The Great Controversy: Postfix or Prefix?

3.5 Bitwise Operators in Detail

3.5.1 Bitwise AND, OR, and Exclusive OR

3.5.2 Operating on Signed and Unsigned

3.5.3 Bitwise Shifts

3.6 Cast Operators

3.6.1 The static_cast Operator

3.6.2 The reinterpret_cast Operator

3.6.3 The const_cast Operator

3.6.4 The dynamic_cast Operator

3.6.5 The C-Language Cast

Table of Contents

3.7 Sample App: Binary Printout

Exercises

4 Control Structures

4.1 Concise Summary of C++ Statements

4.2 Null Statements (;) and Expression Statements

4.3 Compound Statements

4.4 if and if-else Statements

4.5 while and do-while Statements

4.6 for Statements

4.7 Range-based for Statements (C++11)

4.8 switch Statements

4.9 Jump Statements (break, continue, goto)

4.10 Exception Handling (try, catch)

4.10.1 What Is an Exception?

4.10.2 try-catch Blocks: General Syntax

4.10.3 catch Blocks and Exception Objects

4.10.4 Throwing an Exception

4.11 Sample App: Guess-the-Number Game

Exercises

4.12 Sample App: Computer Guesses the Number

Exercises

5 Functions

5.1 Overview of Traditional (Named) Functions

5.1.1 Function Prototypes versus Definitions

5.1.2 Prototyping a Function (Simplified Syntax)

5.1.3 Defining a Function

5.1.4 Calling a Function

5.2 Local and Global Variables

Table of Contents

- 5.3 Complete Function Declaration Syntax
- 5.4 Function Overloading
- 5.5 Arguments with Default Values
- 5.6 Variable-Length Argument Lists
- 5.7 Lambda, or Anonymous, Functions (C++11)
 - 5.7.1 Basic Lambda Syntax
 - 5.7.2 Lambda Closure Syntax
 - 5.7.3 The mutable Keyword
 - 5.7.4 Using Lambdas with the STL
 - 5.7.5 Storing and Returning Lambdas
- 5.8 constexpr Functions (C++11)
- 5.9 Sample App: Odds at Dice
- Exercises

6 Pointers, Arrays, and References

- 6.1 References
 - 6.1.1 Reference Arguments
 - 6.1.2 Returning a Reference from a Function
 - 6.1.3 References Modified by const
- 6.2 Arrays
 - 6.2.1 Simple (One-Dimensional) Arrays
 - 6.2.2 Array Processing with Loops
 - 6.2.3 Passing Arrays to Functions
 - 6.2.4 Multidimensional Arrays
- 6.3 Pointers
 - 6.3.1 The Concept of Pointer
 - 6.3.2 Pointers as Arguments
 - 6.3.3 Pointers Used to Access Arrays
 - 6.3.4 Pointer Arithmetic
 - 6.3.5 Pointers versus Array Arguments

Table of Contents

6.3.6 Pointers and Memory Allocation

6.3.7 Pointers to const Types

6.3.8 Applying const to the Pointer Itself

6.3.9 Void Pointers (void*)

6.4 Complex Declarations Involving Pointers

6.5 Passing and Returning Function Pointers

6.6 Smart Pointers (C++11)

6.6.1 The shared_ptr Template

6.6.2 The unique_ptr Template

6.7 Sample App: Sieve of Eratosthenes

Exercises

7 Classes and Objects

7.1 Overview: Structures, Unions, and Classes

7.2 Basic Class Declaration Syntax

7.2.1 Declaring and Using a Class

7.2.2 Data-Member Access (Public, Private, Protected)

7.2.3 Defining Member Functions

7.2.4 Calling Member Functions

7.2.5 Private Member Functions

7.2.6 Classes Containing Classes

7.2.7 Static Members

7.3 Constructors

7.3.1 The Default Constructor

7.3.2 Overloaded Constructors and Conversion Functions

7.3.3 Copy Constructor

7.3.4 Constructor Initialization Lists

7.3.5 Delegated Constructors (C++11)

7.3.6 Default Member Initialization (C++11)

7.4 Destructors

Table of Contents

7.5 The Hidden this Pointer

7.6 Operator Functions (Op Overloading)

7.6.1 Operator Functions as Members

7.6.2 Operator Functions as Friends

7.6.3 Assignment Operator (=)

7.6.4 Function-Call Operator, ()

7.6.5 Subscript Operator, []

7.6.6 Increment and Decrement (++ , --)

7.6.7 Incoming and Outgoing Conversion Functions

7.7 Deriving Classes (Subclassing)

7.7.1 Subclass Syntax

7.7.2 Base-Class Access Specifiers

7.7.3 Inherited Constructors (C++11)

7.7.4 Up-casting: Child Objects and Base-Class Pointers

7.7.5 Virtual Functions and Overriding

7.7.6 Pure Virtual Functions

7.7.7 Override Keyword (C++11)

7.7.8 Resolving Name Conflicts within Hierarchies

7.8 Bit Fields

7.9 Unions

7.9.1 Named Unions

7.9.2 Anonymous Unions

7.10 Sample App: Packed Boolean

Exercises

8 Preprocessor Directives

8.1 General Syntax of Preprocessor Directives

8.2 Summary of Preprocessor Directives

8.3 Using Directives to Solve Specific Problems

8.3.1 Creating Meaningful Symbols with #define

Table of Contents

8.3.2 Creating Macros with `#define`

8.3.3 Conditional Compilation (`#if`, `#endif`, and so on)

8.4 Preprocessor Operators

8.5 Predefined Macros

8.6 Creating Project Header Files

9 Creating and Using Templates

9.1 Templates: Syntax and Overview

9.2 Function Templates

9.2.1 Function Templates with One Parameter

9.2.2 Dealing with Type Ambiguities

9.2.3 A Function Template with Two Parameters

9.3 Class Templates

9.3.1 Simple Class Templates

9.3.2 Instantiating Class Templates

9.4 Class Templates with Member Functions

9.4.1 Class Templates with Inline Member Functions

9.4.2 Class Templates with Separate Function Definitions

9.5 Using Integer Template Parameters

9.6 Template Specialization

9.7 Variadic Templates (C++11)

9.7.1 A More Sophisticated Variadic Template

9.7.2 Summary of Rules for Variadic Templates

9.7.3 Tuples

9.8 Sample App: Type Promotion, v 2

Exercises

10 C-String Library Functions

10.1 Overview of the C-String Format

10.2 Input and Output with C-Strings

Table of Contents

10.3 C-String Functions

10.4 String Tokenizing with strtok

10.5 Individual-Character Functions

10.6 Memory-Block Functions (memcpy, and so on)

10.7 Wide-Character Functions (wstrcpy, and so on)

11 C I/O Library Functions

11.1 Overview of C Library I/O

11.2 Console I/O Functions

11.3 Print/Scan Formats

11.3.1 printf Format Specifiers (%)

11.3.2 Advanced printf Format Syntax

11.3.3 Using Format Specifiers with scanf

11.4 Input and Output to Strings

11.5 File I/O

11.5.1 Opening a File

11.5.2 Closing a File

11.5.3 Reading and Writing Text Files

11.5.4 Reading and Writing Binary Files

11.5.5 Random-Access Functions

11.5.6 Other File-Management Functions

12 Math, Time, and Other Library Functions

12.1 Trigonometric Functions

12.2 Other Math Functions

12.3 The C Date and Time Library

12.3.1 Date and Time Functions

12.3.2 The TM (Time) Data Structure

12.3.3 Date/Time Format Specifiers (strftime)

12.4 String-to-Number Conversions

Table of Contents

12.5 Memory-Allocation Functions

12.6 Standard C Randomization Functions

12.7 Searching and Sorting Functions

12.7.1 The bsearch Function (Binary Search)

12.7.2 The qsort Function (Quick Sort)

12.7.3 Comparison Functions Using C-Strings

12.8 Other Standard C Library Functions

12.9 Sample App: Idiot Savant

Exercises

13 C++ I/O Stream Classes

13.1 The Basics of C++ I/O Streams

13.1.1 Writing Output with <<

13.1.2 Reading Input with >>

13.2 Reading a Line of Input with getline

13.3 The C++ Stream-Class Hierarchy

13.4 Stream Objects: Manipulators and Flags

13.4.1 Stream Manipulators

13.4.2 Stream Format Flags

13.5 Stream Member Functions (General Purpose)

13.5.1 Input Stream Functions

13.5.2 Output Stream Functions

13.5.3 Flag-Setting Stream Functions

13.6 File Stream Operations

13.6.1 Overview: Text versus Binary File I/O

13.6.2 Creating a File Object

13.6.3 File-Specific Member Functions

13.6.4 Reading and Writing in Binary Mode

13.6.5 Random-Access Operations

Table of Contents

13.7 Reading and Writing String Streams

13.8 Overloading Shift Operators for Your Classes

13.9 Sample App: Text File Reader

Exercises

14 The C++ STL String Class

14.1 Overview of the String Class

14.2 String Class Constructors

14.3 String Class Operators

14.3.1 Assigning Strings (=)

14.3.2 Joining Strings (+)

14.3.3 Comparing Strings

14.3.4 Indexing Strings ([])

14.4 Concise Summary of Member Functions

14.5 Member Functions in Detail

14.6 String Class Iterators

14.6.1 Standard (Forward) Iterators

14.6.2 Reverse Iterators

14.6.3 Iterator Arithmetic

14.6.4 Member Functions Using Iterators

14.7 Wide-Character String Class (basic_string)

15 Introduction to STL (vector, deque)

15.1 A Tour of the Container Templates

15.2 Introduction to Iterators

15.3 The vector Template

15.3.1 Vector Iterators

15.3.2 Vector Constructors

15.3.3 List Initialization (C++11)

15.3.4 Concise Summary of Vector Functions

Table of Contents

15.3.5 Vector Member Functions in Detail

15.4 The deque Template

15.4.1 Deque Iterators

15.4.2 Deque Constructors

15.4.3 Concise Summary of Deque Functions

15.4.4 Deque Functions in Detail

15.5 The bitset Template

15.5.1 bitset Constructors

15.5.2 bitset Member Functions

15.6 Sample App: Alpha File Organizer

Exercises

16 STL Sequence Containers (List)

16.1 Sorting Elements (Strict Weak Ordering)

16.2 The list Template

16.2.1 List Iterators

16.2.2 List Constructors

16.2.3 Concise Summary of list Functions

16.2.4 List Member Functions in Detail

16.3 The stack Template

16.4 The queue Template

16.5 The priority_queue Template

16.5.1 Priority Queue Constructors

16.5.2 Priority Queue Member Functions

16.6 Sample App: Find the Median

Exercises

17 STL Associated Containers (map, set)

17.1 The pair Template

17.2 The map Template

Table of Contents

- 17.2.1 Populating a Map
- 17.2.2 Finding an Existing Map Element
- 17.2.3 A More Sophisticated Record Type
- 17.2.4 Iterating through a Map
- 17.2.5 Map Implementation: Binary Trees
- 17.2.6 Map Constructors
- 17.2.7 Concise Summary of Map Functions
- 17.2.8 Map Member Functions in Detail

17.3 The set Template

- 17.3.1 Populating a Set
- 17.3.2 Finding an Element in a Set
- 17.3.3 Set Constructors
- 17.3.4 Concise Summary of Set Functions
- 17.3.5 Set Member Functions in Detail

17.4 The multimap Template

17.5 The multiset Template

17.6 Unordered Containers (C++11)

- 17.6.1 Unordered Containers: Basic Concepts
- 17.6.2 Fine-Tuning Hash-Table Performance
- 17.6.3 Writing Your Own Hash and Equals Functions

17.7 Sample App: Guess-the-Word Game

Exercises

18 STL Algorithms

- 18.1 STL Algorithms: General Concepts
- 18.2 Using Lambda Functions (C++11)
- 18.3 Algorithms and Iterators
- 18.4 Insert Iterators
- 18.5 Sample App: Finding the Median

Table of Contents

18.6 Concise Summaries of Algorithms

18.6.1 Read-Only Algorithms

18.6.2 Modifying Algorithms

18.6.3 Sorting and Reordering Algorithms

18.6.4 Heap Algorithms

18.6.5 Numeric Algorithms

18.6.6 Predefined Function Objects

18.7 Detailed Descriptions of Algorithms

19 C++11 Randomization Library

19.1 Issues in Randomization

19.1.1 The Problem of Biased Distribution

19.1.2 The Problem of Pseudorandom Sequences

19.1.3 The Problem of Getting a Seed

19.2 A Better Randomization Scheme

19.3 Common Engines

19.4 Common Distributions

19.5 Operations on Engines

19.6 Operations on Distributions

19.7 Sample App: Dice Game

Exercises

20 C++11 Regular-Expression Library

20.1 Overview of C++11 Regular Expressions

20.2 Dealing with Escape Sequences (\)

20.3 Constructing a RegEx String

20.3.1 Matching Characters

20.3.2 Pattern Modifiers

20.3.3 Recurring Groups

20.3.4 Character Classes

Table of Contents

20.4 Matching and Searching Functions

20.5 Find All, or Iterative, Searches

20.6 Replacing Text

20.7 String Tokenizing

20.8 Catching RegEx Exceptions

20.9 Sample App: RPN Calculator

Exercises

A: A Painless Introduction to Rvalue References (C++11)

A.1 The Trouble with Copying

A.2 Move Semantics: C++11 to the Rescue!

A.3 Rvalue Refs in a Users String Class

A.4 Verifying Runtime-Performance Improvement

A.5 Rvalues and Contained Objects

A.6 References Reconsidered: Rvalues and Lvalues

B: Summary of New Features in C++11

B.1 Improvements in Object Construction

B.2 Other Core-Language Enhancements

B.3 Other New Keywords

B.4 Extensions to the Standard Library

C: ASCII Codes

Index