Neil's Guide to C++

Legalities – Please Read!

Preface

Introduction

I've written this book as a strong starting point for programming using the C++ language. You may reread key sections now and then to refresh your memory or even use it as a handy reference, but the primary goal is to bust open your brain and stuff the seeds of C++ into your consciousness.

It may take you a week to read this, if you finish it, or a year. Mastery of the C++ language does not come from any single source, be it book or tutor. It comes from experience, experimentation, and mingling with your programming brothers and sisters. Know ye that I thought to forsake this timeless advice and bound ahead with great strides. But lo! After starting to learn and use C++ in the fall of 1996 I am still discovering things. Do not be discouraged; expect neither too little nor too much, but be pleasantly surprised by both.

References in this genre of programming are plentiful and generally complete, but the city of building block books is sporadically filled with useful, useless, and flawed explanations. Authors have sown their work not only with knowledge, but with their self-same biases, prejudices, hyped enthusiasm, or lack thereof, and coding habits for good or ill. The blood and brains of scholars is fired and filled by the quirks of people they have never met in person.

Newbies, I have seen or talked to, wander lost among the maze of boundless and circular information. Prizes of cheese escape them beyond barriers unbroken, unexplained. I write this book to bring **you** to the cheese and send you home happy!

Why Choose C++?

It seems as if every book and tutorial I've come across on C++ has its whole history and a complete section dedicated to explaining it's superiority, inferiority, and/or it's purpose. Evangelists naturally love to talk extensively about subjects dear to their hearts. I only have one answer to why you should choose C++: your own motivations. Some possibilities are if you think knowing C++ will help you expand your marketability as a professional, pass your Computer Science course, write small hobby applications or tools, or create killer applications. There is no reason for me to provide you with reasons.

About this Book

This book is concentrated on those who wish to learn the C++ language where the target is first-comers to computers and/or technology, computer operators (PC or

otherwise) with no programming experience, and those with limited experience looking to expand their horizons. If you are struggling with, intimidated by, or wary of this subject then I suggest you give this book a go. I do not expect (self-proclaimed) experts to read this without cringing at the simplistic, ye almost *human*, explanations of their closely guarded secrets. I believe in the quality of technical writing as it relates to the ability to communicate effectively to the reader. That said I try not to assume that because I think a concept is simple it deserves less of an explanation.

This book is intended as a complete guide to the C++ *language*, but not a reference to its standard library, STL, or other common API's. All of the provided information has been rigorously researched, reviewed by others, and checked against the latest C++ standard (ISO/IEC 14882, First Edition, 1998-09-01).

This book is written for any decent C++ compiler, even those around before the C++ standard was published. It was a difficult choice to completely favor the current standard or a broader range of compilers that didn't necessarily follow the entire standard for various reasons. Truly all C++ compilers will eventually support the standard, but where does that leave free, unattended compilers like TC-Lite which were built before then? I've attempted to strike a bargain between the two in that my examples use the old-style headers, but I also mention the new-style, the differences, and how to modify my examples to strictly conform to the standard.[1]

Although the language guide is complete, not all situations and complexities may be covered because of the near limitless possibilities. The answer to a question involving a variant situation, however, may be a simple extension of things written about in this book. If you have a specific question that doesn't seem to be answered here visit http://www.neilstuff.com/ for assistance in finding the answer.

I do not provide inline notes for differences between C++ and other languages. If you are coming from a different programming language, you should recognize some things and be unfamiliar with others. But regardless you will see the differences yourselves. I do, however, use the English language as part of many analogies. I basic grasp of that will help you understand them, but isn't required.

My opinions, when I choose to disclose them, are clearly marked throughout the book. I've tried to keep them separate from the body text. Every one has their own opinions and I don't want my personal ones interfering with your learning. You can choose to skip them if you wish. Concepts that I find silly or disgusting *may or may not* be marked. Please do not assume that everything I teach you in this book I particularly like *or* dislike. This book is for learning the language, not my personal preferences.

Part one of this book introduces the reader to computers, programming, and related concepts. Each chapter in this part reveals some more detail about a specific thing,

---

[1] Personally, I think changing the names of all the headers just to move all the standard functionality into the 'std' namespace was inane and just created confusion. Why *shouldn't* the standard functionality exist globally?

except for the first which is a broad overview. This should be a starting point for beginners and a helpful read for other novices. If you consider yourself a god of computers then you may skip this section with my blessing.

The second part of the book is a guide to the language of C++ separated into chapters. Each chapter contains a group of related concepts and following chapters will typically build on the previous topics covered. Ideally you will be able to refer to chapters afterwards as refreshers and tutorials to questions on the concepts covered.

Lastly, the book is capped with an extensive glossary of terms and advanced concepts, which is meant to be read by individual topic. If you have an idea of where you want to and what you want to learn after this language, this should help you get started. It clears up the basic questions misconceptions of common fields of study such as multimedia (graphics, audio, etc.), game programming, and other popular API's (STL, Win32, etc.).

Rules to Reading

1. **Be Patient**

   C++ is a massive concept. Many people will never wrap their mind around the entire language and for many more it will take years. Do not let this discourage you! A house can be built of brick, board, or even bones … but it does not have to use all of them! Such is the case with this language. You should expect results based on your desire to conquer this topic and how deep into it you want to go. If you only want to dabble a bit and only succeed in finishing the first few chapters, consider it a success rather than a failure.

2. **Accept Text**

   If you are looking to be whizzed immediately off your eye balls and into a world drowned in high performance multimedia please come down off your cloud. My way of teaching C++ is from the beginning, so you'll start with numbers and text in a plain monochrome format. Even the most extraordinary things have roots in simplicity.

3. **Question Everything**

   I whole-heartedly welcome questions, comments, and criticisms on this work. It is an accumulation of all that I have learned and wish to share with my fellow people. If you have something to add, then by all means let me know so I might expand my knowledge further.

Notes

- My approach to teaching the C++ language is skewed from most authors and verily, even teachers. I've always felt that beginners are thrown head long into the strangest mess of technical bramble with the way things usually flow. If you like the way most C++ books are laid out and it makes sense to you, this book may be a bit of a warped introduction. However, if you find that you can't seem to learn C++ regardless of the book you pick up, this one may be uniquely tasteful. At first I tried reading other C++ books to get an idea of how to write my own. But now that I am actually writing it, I'm finding I only use other books as references for specific concepts rather than structure.

- I left 'easy' out of the title because it all depends on how well my writing reaches you. A lot of publications (free or otherwise) make the claim that their content is "easy" to digest; I've never seen any content that was easy for everyone. Likewise, this book isn't necessarily for *beginners* either; some people might consider themselves as adept and still have trouble consuming this topic.

- The content of chapters will build upon the *knowledge* of key points in previous chapters. However, I've tried to organize this so you can enter in at any chapter without having to read the previous ones. That said, there will be no examples that must be incrementally built through a standard progression of the chapters. I always hated those programs that just kept getting bigger and bigger each new chapter and here I am trying to *learn* the language. I will try to save "real world" examples for specific times and they may be utilized or skimmed over. What's worse than incremental examples? Incremental examples on boring topics (no more bank programs!).

Table of Contents

This is not intended as a map of things that are already written, but a path of things to work on or that I already have. It's a huge task to tackle a book and there's a lot of stuff I want to cover. I even keep coming upon things I forgot to mention in earlier chapters so first I make a change here before going back (just in case I forget). In other words, this is to keep my in line so I don't ramble off into complete gibberish or start writing about things out of there chapter context.

    f. Storage
    g. Logic
    h. Hardware
2. Logic and Languages
    a. Software
    b. Operating Systems
    c. Languages
    d. Machine Language
    e. Assembly Language
    f. High-Level Languages
    g. Compiled Languages
    h. Interpreted Languages
    i. Methods of Programming
        i. Procedural
        ii. Object-Oriented
3. Numbering Systems
    a. Introduction
    b. The Decimal Numbering System
    c. Bases Loaded
    d. The Imaginary Numbering System (Fivemal)
    e. Distinguishing Numbering Systems
    f. Converting Between Numbering Systems
    g. The Binary Numbering System (base 2)
    h. The Hexadecimal Numbering System (base 16)
    i. The Dead Numbering System (base 8)
4. Storage
    a. Computers Use Binary
    b. N-Bit CPU
    c. Hexadecimal Saves Us
    d. Multiple Byte Data Storage
5. The C++ Language ← Do I need this chapter? I want to remove it
    a. Choosing a Language
    b. History of C++
6. Preparation
    a. What You Need
    b. Text Editor
    c. C++ Software
    d. Command Console

Part 2: Programming in C++

1. Syntax and Expressions
    a. Breaking into the Circle
    b. The 'Hello World' Program
    c. Syntax
    d. Lexical Conventions
    e. Expressions

    f. My Dear Aunt Sally (operator precedence)
    g. Outputting Numbers
    h. Integers
    i. Floating Point
    j. Limits
    k. White Space
    l. Comments
2. Numbers and Values
    a. What is a Variable
    b. Data Types
    c. Declaring a Variable
    d. Declaring Multiple Variables
    e. Identifiers
    f. C++ Keywords
    g. Literals
    h. Constants
    i. Assignment
    j. Basic Arithmetic
    k. Casting
    l. Output
    m. Input
    n. Increment and Decrement
    o. Land Plot Analogy
3. Blocks and Flow
    a. Statement Blocks
    b. Scope and Nesting
    c. Locals
    d. Globals
    e. Namespaces
    f. Flow Control
    g. If
    h. Relational and Equality
    i. Logical Operators
    j. Complex Conditions
    k. Else
    l. Looping
    m. While
    n. Do-While
    o. For
    p. Breaking and Continuing
4. References and Pointers
    a. Introduction
    b. Reference Variables
    c. Memory Address
    d. Pointers
    e. Dereference

    f. Pointer Types
    g. Void Pointers
    h. Null Pointers
5. Structures
    a. Introduction
    b. Structures
    c. Struct
    d. Creating Struct Variables
    e. Initialize Struct Variables
    f. Using Struct Variables
    g. References and Structures
    h. Pointers and Structures
    i. Bit-Fields
    j. Unions
    k. Structure Padding
6. Functions
    a. Introduction
    b. Modularization
    c. Prototyping and Definition
    d. First Example
    e. Return
    f. Void
    g. Correct Main
    h. Parameters
    i. Passing by Value
    j. Passing by Reference
    k. Constant Parameters
    l. Scope
    m. Call Stack
7. Arrays
    a. Introduction
    b. Array Declaration
    c. Looping Subscripts
    d. Initializing Arrays
    e. Declaring Arrays without Size
    f. Copying an Array
    g. Arrays in Memory
    h. Pointers to Arrays
    i. Pointer Arithmetic
    j. References to Arrays
    k. Constant Arrays
    l. Array Function Parameters
    m. Passing Arrays to Functions
    n. SizeOf Array
8. Strings
    a. Text is Character Strings

    b. Character Literals
    c. ASCII and Assumptions
    d. Char
    e. Character Functions
    f. String Literals and Constants
    g. Variable Strings
    h. Initializing Strings
    i. Pointers to Strings
    j. Cin and Strings
    k. String Assignment
    l. SizeOf String
    m. Concatenation
    n. Comparing Strings
    o. Converting Strings to Numbers
    p. Changing String Case
    q. Unicode Strings
9. Pre-Processing
    a. The “Other” Compiler
    b. #include
    c. Custom Headers
    d. Macros
    e. Destroying Macros
    f. Flow Control
    g. Existence Checking
    h. Parenthesis Unnecessary
    i. Raising Errors
    j. Concise Types
10. Classes and Objects
    a. Blobs with Class
    b. Simple Declaration and Usage
    c. Member Functions
    d. Constructors and Destructors
    e. Custom Constructors
    f. Initializing Member Variables
    g. Inheritance
    h. Overloading
    i. Class Namespaces
    j. Static Members
    k. Access Modifiers
    l. Friends
    m. This
    n. Structures are Classes
11. Files and I/O
    a. Console I/O (**cout**/**cin**)
    b. Escape Characters
    c. File I/O (**ofstream**/**ifstream**)

d. Binary Files
e. Reading in a Structure

Part 3: Glossary of Terms and Advanced Concepts

Concept Coverage Plan by Chapter

After writing eleven chapters in part 2 I decided the book was heading into a hole of sorts. I hadn't covered my behind enough in the previous chapters. Now that I've had experience in writing this thing, I know better how to do it, but I also realize that I need a better outline to follow. I keep learning more about concepts that I either didn't cover enough or didn't cover at all. Thus, below is an outline of all the concepts and which chapters will cover them.

Notes

- The standard does not speak of 'iostream.h' and other C++ headers that *used* to end with '.h' and now do not.

Concepts needing to be researched:

- mutable and explicit keywords
- the use of 'asm' among popular compiles. the standard stays this keyword is implementation specific (yikes!)

The beginning of each chapter should list the following:

- Requirements
- Objectives
- Concepts covered in detail including keywords, operators, and standard functions or classes.

Part 1: Introduction to Computer Programming

Chapter 1: Overview of Systems and Concepts

-

Chapter 2: Numbering Systems

-

Chapter 3: Digital Storage

-

Chapter 4: Arithmetic

-

Chapter 5: Software Programming

- code libraries
- object code
- source code

Chapter 6: Command Consoles

- Unices console commands
- DOS/Windows console commands
- Macintosh console commands

Chapter 7: C++ Software
- Text Editors:
    - TextPad
    - UltraEdit
    - EmEdit
- Syntax Coloring
- Basic operations: open/save
- Compilers:
    - TC-Lite
    - C++Builder 5.5
    - Visual C++
    - DJGPP
    - MinGW / Dev C++
    - Cygwin
    - Digital Mars
    - OpenWatcom
- Blarg.

Part 2: The C++ Language

Chapter 1: Overview
- Hello World
- Strict C++
- Lexical Conventions
- Syntax
- Operands, Operators, Operations, and Expressions
- Operation notations: infix, prefix, and postfix.
- Operator Precedence (My Dear Aunt Sally)
- Operator Associatively
- LValues and RValues
- Keywords Listing (intro)
- Operators Listing (intro)
- Statements: Semi-Colons
- White Space
- Comments
- Arithmetic

Chapter 2: Identified Numbers

- Variables: Lifetime (Storage Class), Scope, Address, Name. Only the name is covered in this chapter. Lifetime and Scope are covered in the next chapter and 'Address' in the pointers chapter.
- Memory in C++ (organized by bytes which may or may not be 8 bits, but usually are)
- Implicit and explicit conversions in expressions (a long + an int for example will yield a long value).
- Consts, Enums
- Casting
- Primitive Data Types
- Declaration, Initialization
- Identifiers
- Literals
- Assignment
- Increment/Decrement
- Output/Input (with cout/cin)
- sizeof()
- register, volatile, and auto storage classes
- scalar types … found this exerpt: "A scalar type is an arithmetic type (i.e. a built-in integer or floating point type), an enumeration type, a pointer, a pointer to member, or a const- or volatile-qualified version of one of these types."

Chapter 3: Flow Control

- Statement Blocks
- Boolean Logic
- Boolean Constants 'true' and 'false'
- Scope
- Nesting
- Lifetime (Storage Class)
- Locals and Globals
- Namespaces
- If Statements
- Relational and Equality Operators
- Logical Operators
- Else
- ElseIf with Else and If (nested if)
- Switch
- While Statement
- Do-While Statement
- For Statement
- Break and Continue
- Goto
- Conditional (Ternary) Operator
- Comma Operator

Chapter 4: Indirect Variables

- References
- Pointers
- Memory Layout and Addresses
- Address Assignment
- Dereferencing
- Pointer Types (char* versus int*, etc.)
- Void Pointers
- NULL

Chapter 5: Aliases and Compounds

- Typedef
- Structures
- Type Scope
- Member Access
- Definition vs Declaration
- References to Structures
- Pointers to Structures
- Structure Padding and Sizeof with Structures
- Bit Fields
- Bit Manipulation: masking, shifting, outside of structs, etc.
- Unions

Chapter 6: Method Delegation

- Functions: Declaration (Prototype) and Definition
- Return
- Using Functions in Expressions / Call Stack
- Void Functions
- Correct Main() and Main() Alterations
- Parameters
- Void Parameter List
- Passing By Value
- Passing By Reference
- Passing By Address
- Constant Parameters
- Const Functions
- Default Parameter Values
- Structs as Parameters
- Function Scope
- Static for Function Locals
- Standard Functions (Standard Library Intro)
- Function Pointers
- Variable Parameter List (Variable Arguments)
- Ambiguities between declaratory and expressions when a statement begins with ‘T(‘. See [stmt.ambig] in C++ Standard for details.

Chapter 7: Arrays

- Arrays: Declaration and Definition
- Subscripts and Indices
- Pointer Arithmetic
- Initialization
- Constant Arrays
- Looping by Index
- Looping by Pointer Arithmetic
- Auto-Size Arrays (array[] = values)
- Copying Arrays
- Arrays in Memory and Pointers to Arrays
- Array Parameters
- Pointer Parameters as Arrays
- References to Arrays (illegal, period)
- SizeOf with Array

Chapter 8: Strings

- Text is Strings
- Character Literals
- Char Variables
- Character Functions
- String Literals and Constants
- Variable Strings
- Initializing Strings
- Pointers to Strings
- Inputting Strings (cin)
- Outputting Strings
- String Assignment
- SizeOf / String Length
- Concatenation
- Comparing Strings
- Strings and Numbers
- Compound Strings
- String Manipulation
- Unicode Strings (wchar_t and ‘C’L)
- Typical String Manipulations: Sub-String, Replace

Chapter 9: Preprocessing

- Preprocessor
- Header Files
- #include
- Custom Headers
- Multiple Source Files
- Macros
- Destroying Macros
- Selection (#if/#ifdef)

- Existence Checking
- Parenthesis Unnecessary
- Raising Errors
- Macro Constants (__LINE__, etc.)
- Concise Types (type aliases using macros)
- Pragma Directive
- Macro Functions: ##macro and #macro
- extern keyword, extern “C”

Chapter 10: Classes

- Classes
- Declaration
- Instances: Objects
- Member Functions
- Constructors and Destructors
- Fundamental Type Constructors (int(), char(), etc.)
- Copy Constructor
- Member Types (nested types)
- Member Scope
- Custom Namespaces and Class Namespaces
- Initializing Member Data in Constructor
- Inheritance
- Overloading
- Static Members
- Access Modifiers: Public, Protected, Private
- Friends
- This Pointer
- Structures Are Classes

Chapter 11: Streaming I/O

- Files
- Opening/Closing
- Reading/Writing Bytes
- Reading Values with ‘>>’
- Writing Values with ‘<<’
- Binary Files/Saving Structs and other native types
- I/O Manipulation

Chapter 12: Memory Management

- Multidimensional Arrays?
- Pointer Pointers
- Arrays of Strings
- new, placement new, nothrow/throw new
- array vs “plain” allocation (non-array)
- placement allocation ~ explicit destructor calls
- delete

- Heap vs Stack
- Memory Leaks
- Creating new Object (primitive type, user-type)
- Creating new Array
- Creating new Array of Arrays
- Clever Casting

Chapter 13: Templates and Overloading
- Template Classes
- Template Functions
- Operator Overloading
- Overloading Casting Operators (to make a string class act like char* for example)

Chapter 14: Inheritance
- Const, volatile, and const volatile member functions.
- Member Function Pointers
- Multiple inheritance … casting a pointer to a base class from a multiply-derived one
- Static arrays of member function pointers (this will be good!)
- Virtual Functions
- Virtual Functions don't work in Constructors/Destructors
- Abstract Classes
- Multiple Inheritance
- Casting Base Pointers to Derived Objects with Single and Multiple inheritance

Chapter 15: Exceptions
- try/catch
- terminate(), unexpected(), uncaught_exception()
- Microsoft Visual C++ specifics: __try, __catch, __finally

Chapter 16: Type Information
- limits.h
- C++ and C ways of type information
- typeof
- dynamic_cast

Chapter 17: Miscellaneous Compiler Specifics
- asm keyword
- __cdecl
- __stdcall

Appendices

Appendix A: Standard Library Outline
-

Appendix B: Glossary of Terms and Concepts

- Localization and Internationalization
- Linked Lists, Stacks, Queues, Binary (B) Trees
- Recursion
- Code Libraries, Libraries, Import Libraries, Static Libraries, Dynamic Libraries
- Callback Functions
- Multi-Threading
- Standard String Type
- Color Text
- Multimedia / Games Programming: Graphics, Sound, Controllers
- COM
- Binary Compatibility (between C programs, etc.)