Introduction

Every day, more and more people learn and use the C++ programming language. I have taught C to thousands of students in my life. I see many of those students now moving to C++ in their school work or career. The C++ language is becoming an industry-accepted standard programming language, using the solid foundation of C to gain a foothold. C++ is simply a better C than C.

C++ By Example is one of several books in Que's new line of *By Example* series. The philosophy of these books is simple: The best way to teach computer programming concepts is with multiple examples. Command descriptions, format syntax, and language references are not enough to teach a newcomer a programming language. Only by looking at numerous examples and by running sample programs can programming students get more than just a "feel" for the language.

Who Should Use This Book

This book teaches at three levels: beginning, intermediate, and advanced. Text and numerous examples are aimed at each level. If you are new to C++, and even if you are new to computers, this book attempts to put you at ease and gradually build your C++ programming skills. If you are an expert at C++, this book provides a few extras for you along the way.

The Book's Philosophy

This book focuses on programming *correctly* in C++ by teaching structured programming techniques and proper program design. Emphasis is always placed on a program's readability rather than "tricks of the trade" code examples. In this changing world, programs should be clear, properly structured, and well-documented, and this book does not waver from the importance of this philosophy.

This book teaches you C++ using a holistic approach. In addition to learning the mechanics of the language, you learn tips and warnings, how to use C++ for different types of applications, and a little of the history and interesting asides about the computing industry.

Many other books build single applications, adding to them a little at a time with each chapter. The chapters of this book are standalone chapters, and show you complete programs that fully demonstrate the commands discussed in the chapter. There is a program for every level of reader, from beginning to advanced.

This book contains almost 200 sample program listings. These programs show ways that you can use C++ for personal finance, school and business record keeping, math and science, and general-purpose applications that almost everybody with a computer can use. This wide variety of programs show you that C++ is a very powerful language that is easy to learn and use.

Appendix F, "The Mailing List Application," is a complete application—much longer than any of the other programs in the book—that brings together your entire working knowledge of C++. The application is a computerized mailing-list manager. Throughout the chapters that come before the program, you learn how each command in the program works. You can modify the program to better suit your own needs. (The comments in the program suggest changes you can make.)

Overview of This Book

This book is divided into eight parts. Part I introduces you to the C++ environment, as well as introductory programming concepts. Starting with Part II, the book presents the C++ programming

language commands and built-in functions. After mastering the language, you can use the book as a handy reference. When you need help with a specific C++ programming problem, turn to the appropriate area that describes that part of the language to see numerous examples of code.

To get an idea of the book's layout, read the following description of each section of the book:

Part I: Introduction to C++

This section explains what C++ is by describing a brief history of the C++ programming language and presenting an overview of C++'s advantages over other languages. This part describes your computer's hardware, how you develop your C++ programs, and the steps you follow to enter and run programs. You begin to write C++ programs in Chapter 3.

Part II: Using C++ Operators

This section teaches the entire set of C++ operators. The rich assortment of operators (more than any other programming language except APL) makes up for the fact that the C++ programming language is very small. The operators and their order of precedence are more important to C++ than most programming languages.

Part III: C++ Constructs

C++ data processing is most powerful due to the looping, comparison, and selection constructs that C++ offers. This part shows you how to write programs flowing with control computations that produce accurate and readable code.

Part IV: Variable Scope and Modular Programming

To support true structured programming techniques, C++ must allow for local and global variables, as well as offer several

ways to pass and return variables between functions. C++ is a very strong structured language that attempts, if the programmer is willing to "listen to the language," to protect local variables by making them visible only to the parts of the program that need them.

Part V: Character Input/Output and String Functions

C++ contains no commands that perform input or output. To make up for this apparent oversight, C++ compiler writers supply several useful input and output functions. By separating input and output functions from the language, C++ achieves better portability between computers; if your program runs on one computer, it will work on any other.

This part also describes several of the other built-in math, character, and string functions available with C++. These functions keep you from having to write your own routines to perform common tasks.

Part VI: Arrays and Pointers

C++ offers single and multidimensional arrays that hold multiple occurrences of repeating data, but that do not require much effort on your part to process.

Unlike many other programming languages, C++ also uses pointer variables a great deal. Pointer variables and arrays work together to give you flexible data storage that allow for easy sorting and searching of data.

Part VII: Structures and File Input/Output

Variables, arrays, and pointers are not enough to hold the types of data that your programs require. Structures allow for more powerful grouping of many different kinds of data into manageable units.

Your computer would be too limiting if you could not store data to the disk and retrieve that data back in your programs. Disk

files are required by most "real world" applications. This section describes how C++ processes sequential and random-access files and teaches the fundamental principles needed to effectively save data to the disk. The last chapter in this section introduces object-oriented programming and its use of *classes*.

Part VIII: References

This final section of the book includes a reference guide to the ASCII table, the C++ precedence table, and to keywords and functions in C++. Also in this section are the mailing list application and the answers to the review questions.

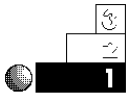
Conventions Used in This Book

The following typographic conventions are used in this book:

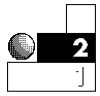
- Code lines, variables, and any text you see on-screen are in `monospace`.
- Placeholders on format lines are in *`italic monospace`*.
- Filenames are in regular text, all uppercase (CCDOUB.CPP).
- Optional parameters on format lines are enclosed in flat brackets ([ ]). You do **not** type the brackets when you include these parameters.
- New terms, which are also found in the glossary, are in *italic*.

Index to the Icons

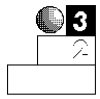
The following icons appear throughout this book:



Level 1 difficulty



Level 2 difficulty



Level 3 difficulty

Tip

Note

Caution

Pseudocode

The pseudocode icon appears beside pseudocode, which is typeset in *italic* immediately before the program. The pseudocode consists of one or more sentences indicating what the program instructions are doing, in English. Pseudocode appears before selected programs.

Margin Graphics (Book Diagrams)

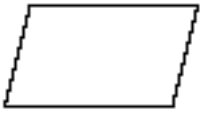
To help your understanding of C++ further, this book includes numerous *margin graphics*. These margin graphics are similar to *flowcharts* you have seen before. Both use standard symbols to represent program logic. If you have heard of the adage "A picture is worth a thousand words," you will understand why it is easier to look at the margin graphics and grasp the overall logic before dissecting programs line-by-line.

Throughout this book, these margin graphics are used in two places. Some graphics appear when a new command is introduced, to explain how the command operates. Others appear when new commands appear in sample programs for the first time.

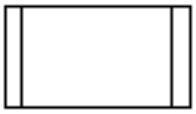
The margin graphics do not provide complete, detailed explanations of every statement in each program. They are simple instructions and provide an overview of the new statements in question. The symbols used in the margin graphics, along with descriptions of them, follow:

Terminal symbol
({,},Return...)

Assignment staement (`total = total + newvalue; ctr = ctr = 1;...`)



Input/output
(scanf , print f...)

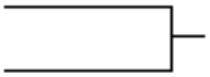


Calling a function

Small circle; loop begin

Large dot; begining and end of `IF-THEN`, `IF-THEN-ELSE`, and `Switch`

Input/output of arrays; assumes implied `FOR` loop(s) needed to deal with array I/O



Comment bracket; used for added info, such as name of a function

The margin graphics, the program listings, the program comments, and the program descriptions in the book provide many vehicles for learning the C++ language!

Companion Disk Offer

If you'd like to save yourself hours of tedious typing, use the order form in the back of this book to order the companion disk for *C++ By Example*. This disk contains the source code for all complete programs and sample code in this book, as well as the mailing-list application that appears in Appendix F. Additionally, the answers to many of the review exercises are included on the disk.