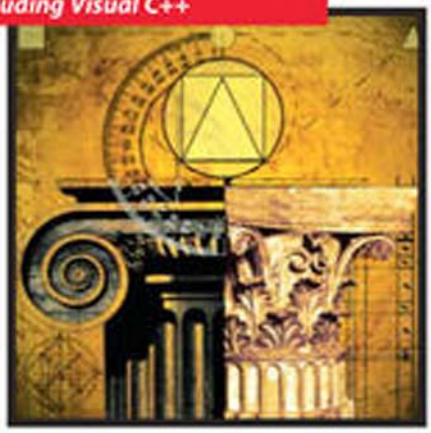
OSBORNE

Learn C++ from the Master

C++

from the GROUND UP

Works with all C++ compilers, including Visual C++



THIRD EDITION

Covers the International Standard for C++

Teaches the entire C++ language, from the basics to advanced features

Packed with insider tips and techniques, and hundreds of examples

Includes the Standard Template Library (STL)

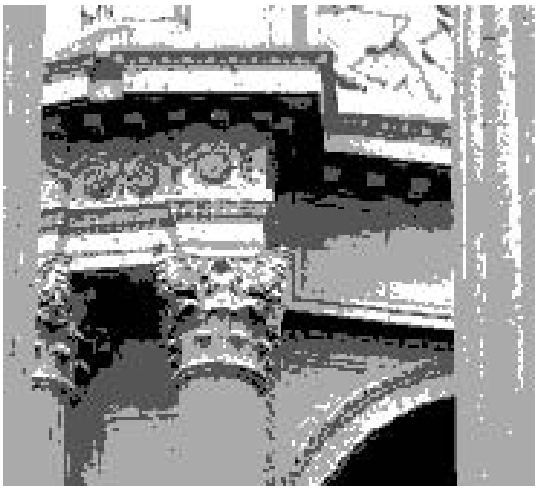
FREE CODE ONLINE

BONUS!

www.osborne.com

Herbert Schildt

Best-selling programming author with more than 3 million books sold!

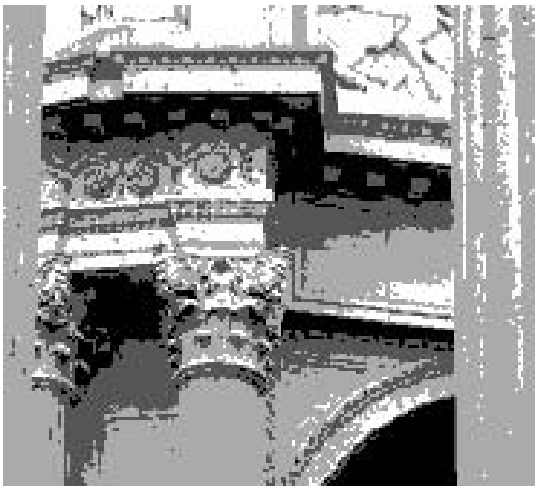


C++ from the Ground Up

Third Edition

About the Author

Herbert Schildt is the world's leading programming author. He is an authority on the C, C++, Java, and C# languages, and is a master Windows programmer. His programming books have sold more than 3 million copies worldwide and have been translated into all major foreign languages. He is the author of numerous bestsellers, including *C++: The Complete Reference, C#: The Complete Reference*, *Java 2: The Complete Reference*, *C: The Complete Reference*, *C++ From the Ground Up, C++: A Beginner's Guide, C#: A Beginner's Guide*, and *Java 2: A Beginner's Guide*. Schildt holds a master's degree in computer science from the University of Illinois. He can be reached at his consulting office at (217) 586-4683.



C++ from the Ground Up

Third Edition

Herbert Schildt

McGraw-Hill/Osborne

New York Chicago San Francisco
Lisbon London Madrid Mexico City
Milan New Delhi San Juan
Seoul Singapore Sydney Toronto

The McGraw-Hill Companies

McGraw-Hill/Osborne
2600 Tenth Street
Berkeley, California 94710
U.S.A.

To arrange bulk purchase discounts for sales promotions, premiums, or fund-raisers, please contact **McGraw-Hill/**Osborne at the above address. For information on translations or book distributors outside the U.S.A., please see the International Contact Information page immediately following the index of this book.

C++ from the Ground Up, Third Edition

1234567890 DOC DOC 019876543

ISBN 0-07-222897-0

Publisher
Brandon A. Nordin

Vice President & Associate Publisher
Scott Rogers

Acquisitions Editor
Lisa McClain

Project Editors
Jenn Tust, Elizabeth Seymour

Proofreader
Marian M. Selig

Indexer
Sheryl Schildt

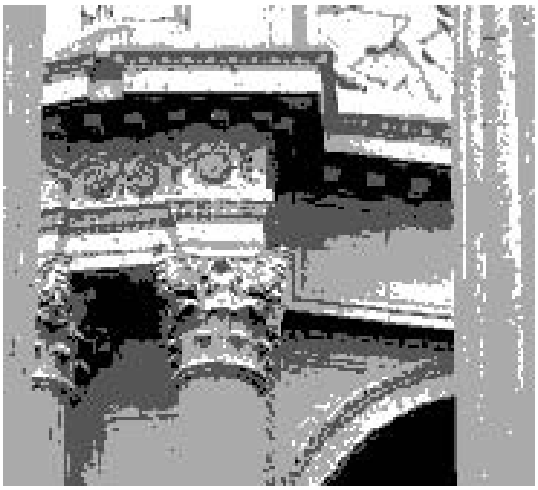
Computer Designers
Tabitha M. Cagan, Tara A. Davis, John Patrus, Lucie Ericksen

Illustrators
Michael Mueller, Lyssa Wald, Melinda Lytle

Cover Series Design
John Nedwidek, emdesign

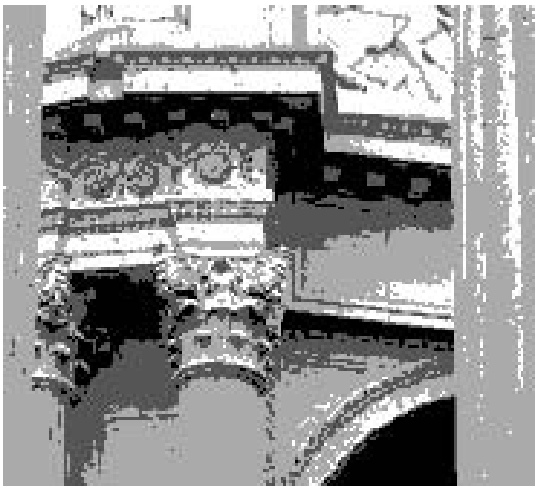
Cover Illustration
Lance Ravella

This book was composed with Corel VENTURA™ Publisher.



Contents

This page intentionally left blank



Preface

This book teaches you how to program in C++ — the most powerful computer language in use today. No previous programming experience is required. The book starts with the basics, covers the fundamentals, moves on to the core of the language, and concludes with its more advanced features. By the time you finish, you will be an accomplished C++ programmer.

C++ is your gateway to modern, object-oriented programming. It is the preeminent language for the development of high-performance software and is the choice of programmers worldwide. Simply put, to be a top-flight, professional programmer today implies competency in C++.

C++ is more than just a popular language. C++ provides the conceptual substrata that underlie the design of several other languages, and much of modern computing. It is no accident that two other important languages, Java and C#, are descended from C++. There is little in programming that has not been influenced by the syntax, style, and philosophy of C++.

Because C++ was designed for professional programming, C++ is not the easiest programming language to learn. It is, however, the *best* programming language to learn. Once you have mastered C++, you will be able to write professional-quality, high-performance programs. You will also be able to easily learn languages like Java or C# because they share the same basic syntax and design as C++.

What Is New in the Third Edition

In the time that has passed since the previous edition of this book, there have been no changes to the C++ language. There have, however, been big changes to the computing environment. For example, Java became the dominant language for Web programming, the .NET Framework was released, and C# was invented. Through all the changes of the past few years, one thing has remained constant: the staying

power of C++. C++ has been, is, and will remain the dominant language of "power programmers" well into the forseeable future.

The overall structure and organization of the third edition is similar to the second edition. Most of the changes involve updating and expanding the coverage throughout. In some cases, additional details were added. In other cases, the presentation of a topic was improved. In still other situations, descriptions were modernized to reflect the current programming environment. Several new sections were also added.

Two appendices were added. One describes the extended keywords defined by Microsoft that are used for creating managed code for the .NET Framework. The second explains how to adapt the code in this book for use with an older, non-standard C++ compiler.

Finally, all code examples were retested against the current crop of compilers, including Microsoft's Visual Studio .NET and Borland's C++ Builder.

What Version of C++

The material in this book describes Standard C++. This is the version of C++ defined by the ANSI/ISO Standard for C++, and it is the one that is currently supported by all major compilers. Therefore, using this book, you can be confident that what you learn today will also apply tomorrow.

How to Use This Book

The best way to learn any programming language, including C++, is by doing. Therefore, after you have read through a section, try the sample programs. Make sure that you understand why they do what they do before moving on. You should also experiment with the programs, changing one or two lines at a time and observing the results. The more you program, the better you become at programming.

If You're Using Windows

If your computer uses Windows and your goal is to write Windows-based programs, then you have chosen the right language to learn. C++ is completely at home with Windows programming. However, none of the programs in this book use the Windows graphical user interface (GUI). Instead, they are console-based programs that can be run under a Windows console session, such as that provided by the Command Prompt window. The reason for this is easy to understand: GUI-based Windows programs are, by their nature, large and complex. They also use many techniques not directly related to the C++ language. Thus, they are not well-suited for teaching a programming language. However, you can still use a Windows-based compiler to compile the programs in this book because the compiler will automatically create a console session in which to execute your program.

Once you have mastered C++, you will be able to apply your knowledge to Windows programming. In fact, Windows programming using C++ allows the use of class libraries such as MFC or the newer .NET Framework, which can greatly simplify the development of a Windows program.

Don't Forget: Code on the Web

Remember, the source code for all of the programs in this book is available free of charge on the Web at **http://www.osborne.com**. Downloading this code prevents you from having to type in the examples.

For Further Study

C++from the Ground Up is your gateway to the Herb Schildt series of programming books. Here are some others that you will find of interest.

To learn more about C++, try

C++: The Complete Reference

C++: A Beginner's Guide

Teach Yourself C++

STL Programming From the Ground Up

C++ Programmer's Reference

To learn about Java programming, we recommend the following:

Java 2: A Beginner's Guide

Java 2: The Complete Reference

Java 2 Programmer's Reference

To learn about C#, Herb offers these books:

C#: A Beginner's Guide

C#: The Complete Reference

To learn about Windows programming we suggest the following Schildt books:

Windows 98 Programming From the Ground Up

Windows 2000 Programming From the Ground Up

MFC Programming From the Ground Up

The Windows Programming Annotated Archives

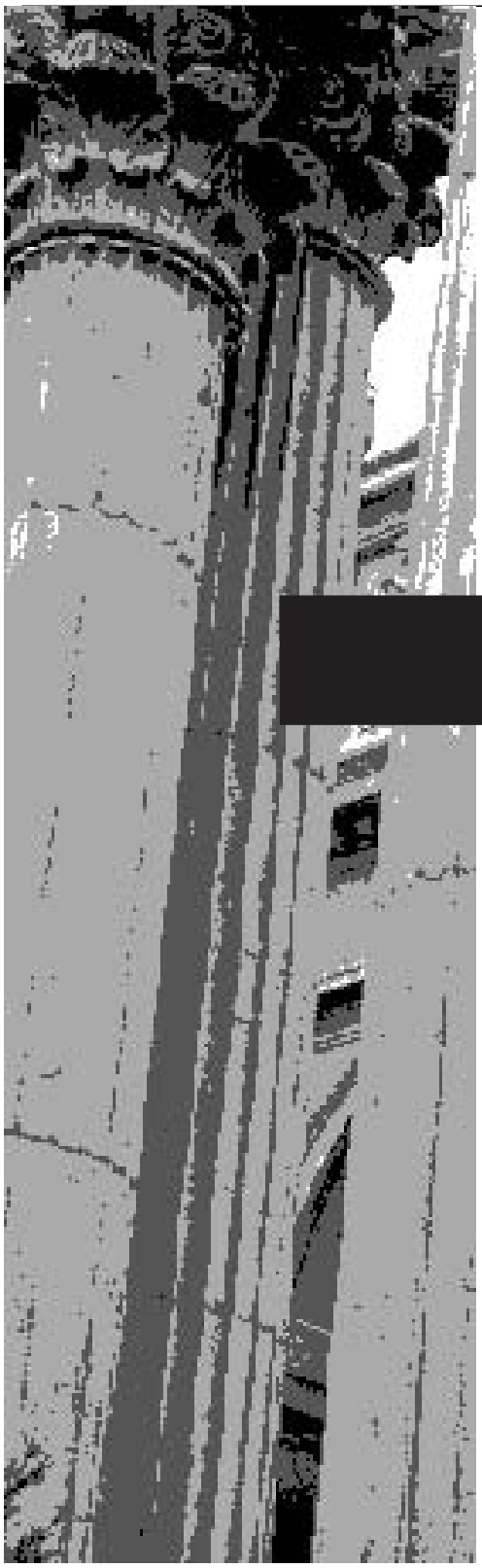
If you want to learn about the C language, which is the foundation of all modern programming, then the following titles will be of interest.

C: The Complete Reference

Teach Yourself C

When you need solid answers, fast, turn to Herbert Schildt, the recognized authority on programming.

This page intentionally left blank



CHAPTER 1

The Story of C++

C++ is the single most important language that any programmer can learn. This is a strong statement, but it is not an exaggeration. C++ is the center of gravity around which all of modern programming revolves. Its syntax and design philosophy define the essence of object-oriented programming. Moreover, C++ charts the course for future language development. For example, both Java and C# are directly descended from C++. C++ is also the universal language of programming; it is the language in which programmers share ideas with one another. To be a professional programmer today implies competency in C++. It is that fundamental and that important. C++ is the gateway to all of modern programming.

Before beginning your study of C++, it is important for you to know how C++ fits into the historical context of computer languages. Understanding the forces that drove its creation, the design philosophy it represents, and the legacy that it inherits makes it easier to appreciate the many innovative and unique features of C++. With this in mind, this chapter presents a brief history of the C++ programming language, its origins, its relationship to its predecessor (C), its uses, and the programming philosophies that it supports. It also puts C++ into perspective relative to other programming languages.

The Origins of C++

The story of C++ begins with C. The reason for this is simple: C++ is built upon the foundation of C. In fact, C++ is a superset of C. (Indeed, all C++ compilers can also be used to compile C programs!) Specifically, C++ is an expanded and enhanced version of C that embodies the philosophy of object-oriented programming (which is described later in this chapter). C++ also includes several other improvements to the C language, including an extended set of library routines. However, much of the spirit and flavor of C++ is inherited directly from C. To fully understand and appreciate C++, you need to understand the "how and why" behind C.

The Creation of C

The C language shook the computer world. Its impact should not be underestimated because it fundamentally changed the way programming was approached and thought about. C is considered by many to be the first modern "programmer's language." Prior to the invention of C, computer languages were generally designed either as academic exercises or by bureaucratic committees. C is different. C was designed, implemented, and developed by real, working programmers, and it reflected the way they approached the job of programming. Its features were honed, tested, thought about, and rethought by the people who actually used the language. The result of this process was a language that programmers liked to use. Indeed, C quickly attracted many followers who had a near-religious zeal for it, and it found wide and rapid acceptance in the programmer community. In short, C is a language designed by and for programmers.

C was invented and first implemented by Dennis Ritchie on a DEC PDP-11 using the UNIX operating system. C is the result of a development process that started with an older language called BCPL, which was developed by Martin Richards. BCPL influenced a language called B, invented by Ken Thompson, which led to the development of C in the 1970s.

For many years, the de facto standard for C was the one supplied with the Unix operating system and described in *The C Programming Language,* by Brian Kernighan and Dennis Ritchie (Prentice-Hall, 1978). However, because no formal standard existed, there were discrepancies between different implementations of C. To alter this situation, a committee was established in the beginning of the summer of 1983 to work on the creation of an ANSI (American National Standards Institute) standard that would define—once and for all—the C language. The final version of the standard was adopted in December 1989, the first copies of which became available in early 1990. This version of C is commonly referred to as *C89,* and it is the foundation upon which C++ is built.

NOTE: The C standard was updated in 1999 and this version of C is usually referred to as *C99.* This version contains some new features, including a few borrowed from C++, but, overall, it is compatible with the original C89 standard. At the time of this writing, no widely available compiler supports C99 and it is still C89 that defines what is commonly thought of as the C language. Furthermore, it is C89 that is the basis for C++. It is possible that a future standard for C++ will include the features added by C99, but they are not part of C++ at this time.

It may seem hard to understand at first, but C is often called a "middle-level" computer language. As it is applied to C, middle-level does not have a negative connotation; it does not mean that C is less powerful, harder to use, or less developed than a "high-level" language, or that it is as difficult to use as assembly language. (*Assembly language,* or *assembler,* as it is often called, is simply a symbolic representation of the actual machine code that a computer can execute.) C is thought of as a middle-level language because it combines elements of high-level languages, such as Pascal, Modula-2, or Visual Basic, with the functionality of assembler.

From a theoretical point of view, a *high-level language* attempts to give the programmer everything he or she could possibly want, already built into the language. A *low-level language* provides nothing other than access to the actual machine instructions. A *middle-level language* gives the programmer a concise set of tools and allows the programmer to develop higher-level constructs on his or her own. A middle-level language offers the programmer built-in power, coupled with flexibility.

Being a middle-level language, C allows you to manipulate bits, bytes, and addresses—the basic elements with which a computer functions. Thus, C does not attempt to buffer the hardware of the machine from your program to any significant extent. For example, the size of an integer in C is directly related to the word size of the CPU. In most high-level languages there are built-in statements for reading and writing disk files. In C, all of these procedures are performed by calls to library routines and not by keywords defined by the language. This approach increases C's flexibility.

C allows—indeed, needs—the programmer to define routines for performing high-level operations. These routines are called *functions,* and they are very important to the C language. In fact, functions are the building blocks of both C and C++. You can easily tailor a library of functions to perform various tasks that are used by your program. In this sense, you can personalize C to fit your needs.