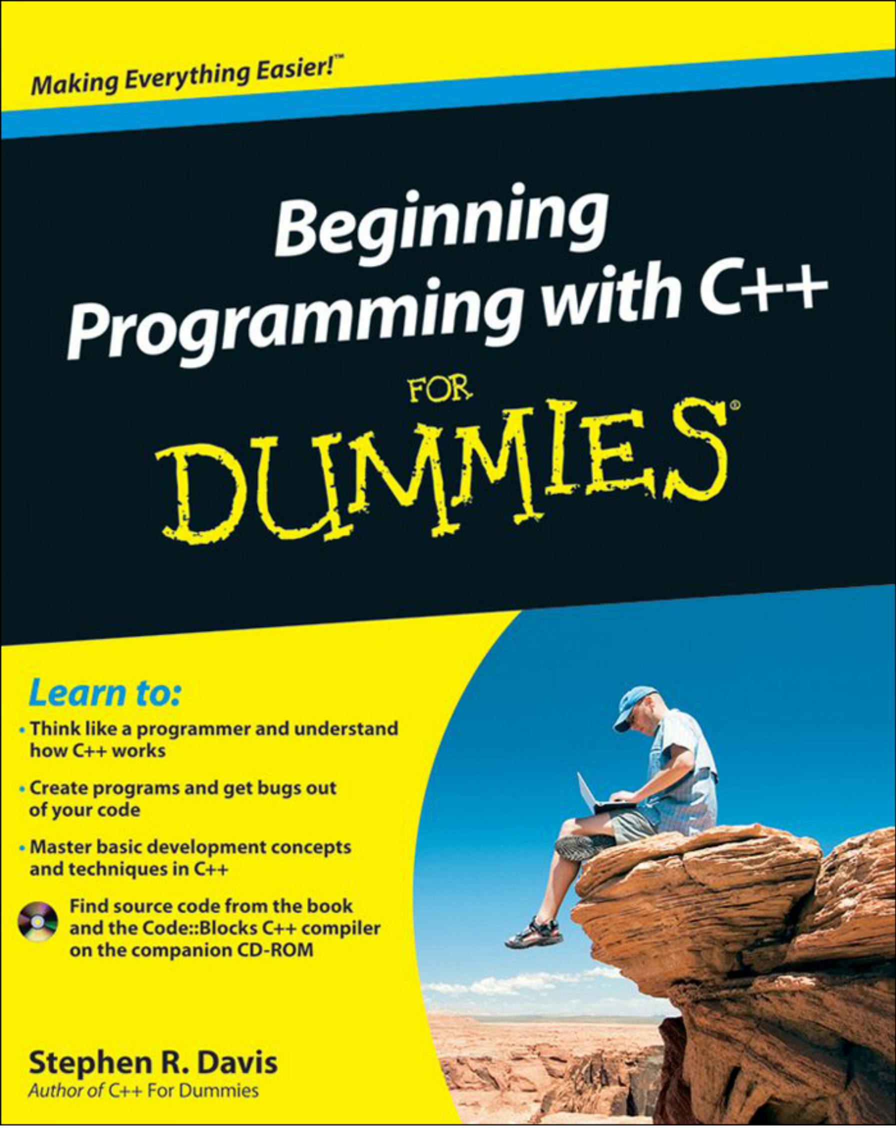

Making Everything Easier!™
Beginning Programming with C++
FOR DUMMIES®
Learn to:
• Think like a programmer and understand how C++ works
• Create programs and get bugs out of your code
• Master basic development concepts and techniques in C++
Find source code from the book and the Code::Blocks C++ compiler on the companion CD-ROM
Stephen R. Davis
Author of C++ For Dummies

Beginning Programming with C++ FOR DUMMIES®

by Stephen R. Davis

WILEY
Wiley Publishing, Inc.

Disclaimer: This eBook does not include ancillary media that was packaged with the printed version of the book.

Beginning Programming with C++ For Dummies®

Published by
Wiley Publishing, Inc.
111 River Street
Hoboken, NJ 07030-5774

www.wiley.com

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002.

For technical support, please visit www.wiley.com/techsupport.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Control Number: 2010930969

ISBN: 978-0-470-61797-7

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

WILEY

Contents at a Glance

Table of Contents

Introduction

Welcome to *Beginning Programming with C++ For Dummies.* This book is intended for the reader who wants to learn to program.

Somehow over the years, programming has become associated with mathematics and logic calculus and other complicated things. I never quite understood that. Programming is a skill like writing advertising or drawing or photography. It does require the ability to think a problem through, but I've known some really good programmers who had zero math skills. Some people are naturally good at it and pick it up quickly, others not so good and not so quick. Nevertheless, anyone with enough patience and "stick-to-itiveness" can learn to program a computer. Even me.

About Beginning Programming with C++ For Dummies

Learning to program necessarily means learning a programming language. This book is based upon the C++ programming language. A Windows version of the suggested compiler is included on the CD-ROM accompanying this book. Macintosh and Linux versions are available for download at `www.codeblocks.org`. (Don't worry: I include step-by-step instructions for how to install the package and build your first program in the book.)

The goal of this book is to teach you the basics of programming in C++, not to inundate you with every detail of the C++ programming language. At the end of this book, you will be able to write a reasonably sophisticated program in C++. You will also be in a position to quickly grasp a number of other similar languages, such as Java and C#.NET.

In this book, you will discover what a program is, how it works, plus how to do the following:

- Install the CodeBlocks C++ compiler and use it to build a program
- Create and evaluate expressions
- Direct the flow of control through your program

- Create data structures that better model the real world
- Define and use C++ pointers
- Manipulate character strings to generate the output the way you want to see it
- Write to and read from files

Foolish Assumptions

I try to make very few assumptions in this book about the reader, but I do assume the following:

- **You have a computer.** Most readers will have computers that run Windows; however, the programs in this book run equally well on Windows, Macintosh, Linux, and Unix. In fact, since C++ is a standardized language, these programs should run on any computer that has a C++ compiler.
- **You know the basics of how to use your computer.** For example, I assume that you know how to run a program, copy a file, create a folder, and so on.
- **You know how to navigate through menus.** I include lots of instructions like "Click on File and then Open." If you can follow that instruction, then you're good to go.
- **You are new to programming.** I don't assume that you know anything about programming. Heck, I don't even assume that you know what programming is.

Conventions Used in This Book

To help you navigate this book as efficiently as possible, I use a few conventions:

- C++ terms are in monofont typeface, `like this`.
- New terms are emphasized with *italics* (and defined).
- Numbered steps that you need to follow and characters you need to type are set in **bold**.

What You Don't Have to Read

I encourage you to read one part of the book; then put the book away and play for a while before moving to the next part. The book is organized so that by the end of each part, you have mastered enough new material to go out and write programs.

I'd like to add the following advice:

- If you already know what programming is but nothing about C++, you can skip Chapter 1.
- I recommend that you use the CodeBlocks compiler that comes with the book, even if you want to use a different C++ compiler after you finish the book. However, if you insist and don't want to use CodeBlocks, you can skip Chapter 2.
- Skim through Chapter 3 if you've already done a little computer programming.
- Start concentrating at Chapter 4, even if you have experience with other languages such as BASIC.
- You can stop reading after Chapter 20 if you're starting to feel saturated. Chapter 21 opens up the new topic of object-oriented programming — you don't want to take that on until you feel really comfortable with what you've learned so far.
- You can skip any of the TechnicalStuff icons.

How This Book Is Organized

Beginning Programming with C++ For Dummies is split into seven parts. You don't have to read it sequentially, and you don't even have to read all the sections in any particular chapter. You can use the Table of Contents and the Index to find the information you need and quickly get your answer. In this section, I briefly describe what you'll find in each part.

Part I: Let's Get Started

This part describes what programs are and how they work. Using a fictitious tire-changing computer, I take you through several algorithms for removing a tire from a car to give you a feel for how programs work. You'll also get CodeBlocks up and running on your computer before leaving this part.

Part II: Writing a Program: Decisions, Decisions

This part introduces you to the basics of programming with C++. You will find out how to declare integer variables and how to write simple expressions. You'll even discover how to make decisions within a program, but you won't be much of an expert by the time you finish this part.

Part III: Becoming a Functional Programmer

Here you learn how to direct the flow of control within your programs. You'll find out how to loop, how to break your code into modules (and why), and how to build these separate modules back into a single program. At the end of this part, you'll be able to write real programs that actually solve problems.

Part IV: Data Structures

This part expands your knowledge of data types. Earlier sections of the book are limited to integers; in this part, you work with characters, decimals, and arrays; and you even get to define your own types. Finally, this is the part where you master the most dreaded topic, the C++ pointer.

Part V: Object-Oriented Programming

This is where you expand your knowledge into object-oriented techniques, the stuff that differentiates C++ from its predecessors, most notably C. (Don't worry if you don't know what object-oriented programming is — you aren't supposed to yet.) You'll want to be comfortable with the material in Parts I through IV before jumping into this part, but you'll be a much stronger programmer by the time you finish it.

Part VI: Advanced Strokes

This is a collection of topics that are important but that didn't fit in the earlier parts. For example, here's where I discuss how to create, read to, and write from files.

Part VII: The Part of Tens

This part includes lists of what to do (and what not to do) when programming to avoid creating bugs needlessly. In addition, this part includes some advice about what topics to study next, should you decide to expand your knowledge of C++.

The CD-ROM Appendix

This part describes what's on the enclosed CD-ROM and how to install it.

Icons Used in This Book

What's a *Dummies* book without icons pointing you in the direction of really great information that's sure to help you along your way? In this section, I briefly describe each icon I use in this book.

TIP

The Tip icon points out helpful information that is likely to make your job easier.

REMEMBER

This icon marks a generally interesting and useful fact — something that you might want to remember for later use. I also use this icon to remind you of some fact that you may have skipped over in an earlier chapter.

WARNING!

The Warning icon highlights lurking danger. With this icon, I'm telling you to pay attention and proceed with caution.

TECHNICAL STUFF

When you see this icon, you know that there's techie stuff nearby. If you're not feeling very techie, you can skip this info.

ON THE CD

This icon denotes the programs that are included on this book's CD-ROM.

Where to Go from Here

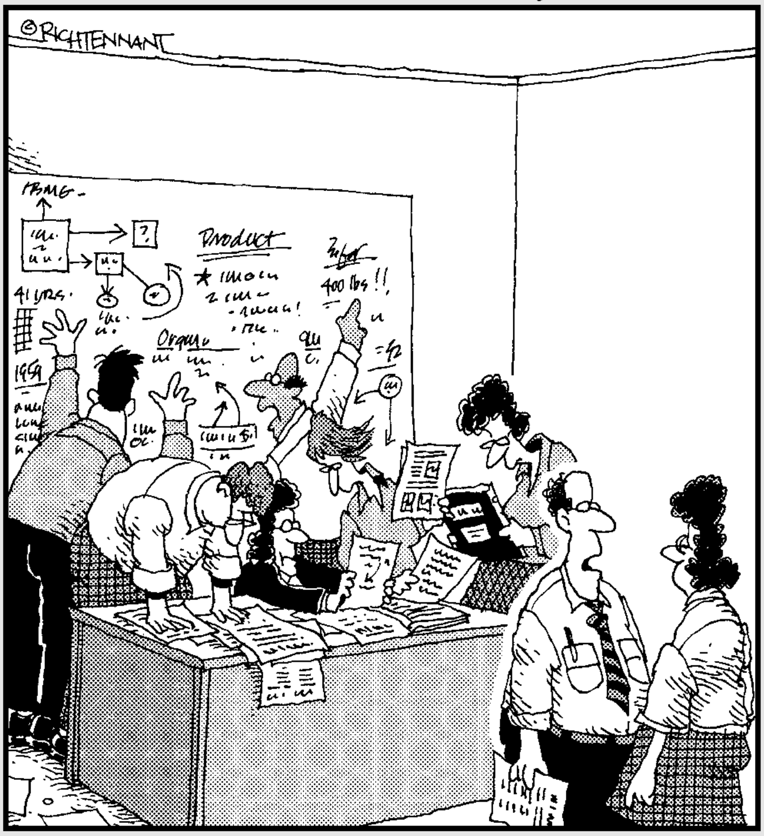
You can find a set of errata and Frequently Asked Questions for this and all my books at www.stephendavis.com. You will also find a link to my e-mail address there. Feel free to send me your questions and comments (that's how I learn). It's through reader input that these books can improve.

Now you've stalled long enough, it's time to turn to Chapter 1 and start discovering how to program!

Part I

Let's Get Started

The 5th Wave By Rich Tennant



"We're outsourcing everything but our core competency. Once we find out what that is, we'll begin the outsourcing process."

In this part . . .

You will learn what it means to program a computer. You will also get your first taste of programming — I take you through the steps to enter, build, and execute your first program. It will all be a bit mysterious in this part, but things will clear up soon, I promise.

Chapter 1

What Is a Program?

In This Chapter

- Understanding programs
- Writing your first "program"
- Looking at computer languages

In this chapter, you will learn what a program is and what it means to write a program. You'll practice on a Human Computer. You'll then see some program snippets written for a real computer. Finally, you'll see your first code snippet written in C++.

Up until now all of the programs running on your computer were written by someone else. Very soon now, that won't be true anymore. You will be joining the ranks of the few, the proud: the programmers.

How Does My Son Differ from a Computer?

A computer is an amazingly fast but incredibly stupid machine. A computer can do anything you tell it (within reason), but it does *exactly* what it's told — nothing more and nothing less.

In this respect, a computer is almost the exact opposite of a human: humans respond intuitively. When I was learning a second language, I learned that it wasn't enough to understand what was being said — it's just as important and considerably more difficult to understand what was left unsaid. This is information that the speaker shares with the listener through common experience or education — things that don't need to be said.

For example, I say things to my son like, "Wash the dishes" (for all the good it does me). This seems like clear enough instructions, but the vast majority of the information contained in that sentence is implied and unspoken.

Let's assume that my son knows what dishes are and that dirty dishes are normally in the sink. But what about knives and forks? After all, I only said dishes, I didn't say anything about eating utensils, and don't even get me started on glassware. And did I mean wash them manually, or is it okay to load them up into the dishwasher to be washed, rinsed, and dried automatically?

But the fact is, "Wash the dishes" is sufficient instruction for my son. He can decompose that sentence and combine it with information that we both share, including an extensive working knowledge of dirty dishes, to come up with a meaningful understanding of what I want him to do — whether he does it or not is a different story. I would guess that he can perform all the mental gymnastics necessary to understand that sentence in about the same amount of time that it takes me to say it — about 1 to 2 seconds.

A computer can't make heads or tails out of something as vague as "wash the dishes." You have to tell the computer exactly what to do with each different type of dish, how to wash a fork, versus a spoon, versus a cup. When does the program stop washing a dish (that is, how does it know when a dish is clean)? When does it stop washing (that is, how does it know when it's finished)?

My son has gobs of memory — it isn't clear exactly how much memory a normal human has, but it's boat loads. Unfortunately, human memory is fuzzy. For example, witnesses to crimes are notoriously bad at recalling details even a short time after the event. Two witnesses to the same event often disagree radically on what transpired.

Computers also have gobs of memory, and that's very good. Once stored, a computer can retrieve a fact as often as you like without change. As expensive as memory was back in the early 1980s, the original IBM PC had only 16K (that's 16 thousand bytes). This could be expanded to a whopping 64K. Compare this with the 1GB to 3GB of main storage available in most computers today (1GB is *one billion bytes*).

As expensive as memory was, however, the IBM PC included extra memory chips and decoding hardware to detect a memory failure. If a memory chip went bad, this circuitry was sure to find it and report it before the program went haywire. This so-called Parity Memory was no longer offered after only a few years, and as far as I know, it is unavailable today except in specific applications where extreme reliability is required — because the memory boards almost never fail.

On the other hand, humans are very good at certain types of processing that computers do poorly, if at all. For example, humans are very good at pulling the meaning out of a sentence garbled by large amounts of background noise. By contrast, digital cell phones have the infuriating habit of just going silent whenever the noise level gets above a built-in threshold.

Programming a "Human Computer"

Before I dive into showing you how to write programs for computer consumption, I start by showing you a program to guide human behavior so that you can better see what you're up against. Writing a program to guide a human is much easier than writing programs for computer hardware because we have a lot of familiarity with and understanding of humans and how they work (I assume). We also share a common human language to start with. But to make things fair, assume that the human computer has been instructed to be particularly literal — so the program will have to be very specific. Our guinea pig computer intends to take each instruction quite literally.

The problem I have chosen is to instruct our human computer in the changing of a flat tire.

The algorithm

The instructions for changing a flat tire are straightforward and go something like the following:

1. Raise the car.
2. Remove the lug nuts that affix the faulty tire to the car.
3. Remove the tire.
4. Mount the new tire.
5. Install the lug nuts.
6. Lower the car.

(I know that technically the lug nuts hold the wheel onto the car and not the tire, but that distinction isn't important here. I use the terms "wheel" and "tire" synonymously in this discussion.)