Beginning C

Fifth Edition

Ivor Horton

Apress®

Beginning C, Fifth Edition

ISBN 978-1-4302-4881-1

ISBN 978-1-4302-4882-8 (eBook)

President and Publisher: Paul Manning
Lead Editor: Jonathan Gennick
Technical Reviewer: Marc Gregoire
Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Louise Corrigan, Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh
Coordinating Editor: Jill Balzano
Copy Editor: Mary Bearden
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `www.springeronline.com`. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail `rights@apress.com`, or visit `www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at `www.apress.com/bulk-sales`.

Any source code or other supplementary materials referenced by the author in this text is available to readers at `www.apress.com`. For detailed information about how to locate your book's source code, go to `www.apress.com/source-code/`.

For my daughter, Dany.

Contents at a Glance

Contents

About the Author

Ivor Horton graduated as a mathematician and was lured into information technology by promises of great rewards for very little work. In spite of the reality usually being a great deal of work for relatively modest rewards, he has continued to work with computers to the present day. He has been engaged at various times in programming, systems design, consultancy, and the management and implementation of projects of considerable complexity. Ivor has many years of experience in the design and implementation of computer systems applied to engineering design and manufacturing operations in a variety of industries. He has also spent a lot of time developing occasionally useful applications in a wide variety of programming languages and primarily teaching scientists and engineers to do likewise. He has been writing books on programming for many years, and his currently published works include tutorials on C, C++, and Java. At the present time, when he is not writing programming books or providing advice to others, he spends his time fishing, traveling, and enjoying life in general.

About the Technical Reviewer

Marc Gregoire is a software engineer from Belgium. He graduated from the Catholic University of Leuven, Belgium, with a degree in "Burgerlijk ingenieur in de computer wetenschappen" (equivalent to master of science in engineering in computer science). The year after, he received the cum laude degree of master in artificial intelligence at the same university. After his studies, Marc started working for a software consultancy company called Ordina Belgium. As a consultant, he worked for Siemens and Nokia Siemens Networks on critical 2G and 3G software running on Solaris for telecom operators. This required working in international teams spanning from South America and the United States to Europe, the Middle East, Africa, and Asia. Now, Marc is working for Nikon Metrology on 3D laser scanning software.

His main expertise is C/C++, specifically Microsoft VC++ and the MFC framework. Next to C/C++, Marc also likes C# and uses PHP for creating web pages. In addition to his main interest of Windows development, he also has experience in developing C++ programs running 24/7 on Linux platforms (e.g., EIB home automation software).

Since April 2007, he has received the yearly Microsoft MVP (Most Valuable Professional) award for his Visual C++ expertise.

Marc is the founder of the Belgian C++ Users Group (`www.becpp.org`) and an active member on the CodeGuru forum (as Marc G). He also creates freeware and shareware programs that are distributed through his web site at `www.nuonsoft.com`, and maintains a blog on `www.nuonsoft.com/blog/`.

Acknowledgments

The author is only one member of the large team of people necessary to get a book into print. I thank the entire Apress editorial and production teams for their help and support throughout. I would like to acknowledge the efforts of Jonathan Gennick, who initiated this new edition, and Jill Balzano, who has patiently dealt with my questions and problems in the editing process.

I would also like to thank my technical editor, Marc Gregoire, for doing such a fantastic job of reviewing the text and checking out all the code fragments and examples. He has an uncanny knack for finding my errors, and his many constructive comments and thoughtful suggestions have undoubtedly made the book a much better tutorial.

Introduction

Welcome to *Beginning C*: Fifth Edition. With this book you can become a competent C programmer using the latest version of the C language. In many ways, C is an ideal language with which to learn programming. It's very compact, so there isn't a lot of syntax to learn before you can write real applications. In spite of its conciseness, it's extremely powerful and is used by professionals in many different areas. The power of C is such that it can be applied at all levels, from developing device drivers and operating system components to creating large-scale applications. A relatively new area for C is in application development for mobile phones.

C compilers are available for virtually every kind of computer, so when you've learned C, you'll be equipped to program in just about any context. Once you know C, you have an excellent base from which you can build an understanding of the object-oriented C++.

My objective in this book is to minimize what I think are the three main hurdles the aspiring programmer must face: coming to grips with the jargon that pervades every programming language, understanding how to use the language elements (as opposed to merely knowing what they are), and appreciating how the language is applied in a practical context.

Jargon is an invaluable and virtually indispensable means of communication for the expert professional as well as the competent amateur, so it can't be avoided. My approach is to ensure that you understand the jargon and get comfortable using it in context. In this way, you'll be able to more effectively use the documentation that comes along with the typical programming product and also feel comfortable reading and learning from the literature that surrounds most programming languages.

Comprehending the syntax and effects of the language elements is obviously an essential part of learning C, but appreciating how the language features work and how they are used is equally important. Rather than just using code fragments, I provide you with practical working examples in each chapter that show how the language features can be applied to specific problems. These examples provide a basis for you to experiment and see the effects of changing the code.

Your understanding of programming in context needs to go beyond the mechanics of applying individual language elements. To help you gain this understanding, I conclude most chapters with a more complex program that applies what you've learned in the chapter. These programs will help you gain the competence and confidence to develop your own applications and provide you with insight into how you can apply language elements in combination and on a larger scale. Most important, they'll give you an idea of what's involved in designing real programs and managing real code.

It's important to realize a few things that are true for learning any programming language. First, there is quite a lot to learn, but this means you'll gain a greater sense of satisfaction when you've mastered it. Second, it's great fun, so you really will enjoy it. Third, you can only learn programming by doing it, and this book helps you along the way. Finally, it's certain you will make a lot of mistakes and get frustrated from time to time during the learning process. When you think you are completely stuck, you just need to be persistent. You will eventually experience that eureka moment and realize it wasn't as difficult as you thought.

How to Use This Book

Because I believe in the hands-on approach, you'll write your first programs almost immediately. Every chapter has several complete programs that put theory into practice, and these are key to the book. You should type in and run all the examples that appear in the text because the very act of typing them in is a tremendous memory aid. You should also attempt all the exercises that appear at the end of each chapter. When you get a program to work for the first time—particularly when you're trying to solve your own problems—you'll find that the great sense of accomplishment and progress makes it all worthwhile.

The pace is gentle at the start, but you'll gain momentum as you get further into the subject. Each chapter covers quite a lot of ground, so take your time and make sure you understand everything before moving on. Experimenting with the code and trying out your own ideas are important parts of the learning process. Try modifying the programs and see what else you can make them do—that's when it gets really interesting. And don't be afraid to try things out—if you don't understand how something works, just type in a few variations and see what happens. It doesn't matter if it's wrong. You'll find you often learn a lot from getting it wrong. A good approach is to read each chapter through, get an idea of its scope, and then go back and work through all the examples.

You might find some of the end-of-chapter programs quite difficult. Don't worry if it's not all completely clear on the first try. There are bound to be bits that you find hard to understand at first because they often apply what you've learned to rather complicated problems. If you really get stuck, you can skip the end-of-chapter exercises, move on to the next chapter, and come back to them later. You can even go through the entire book without worrying about them. However, if you can complete the exercises, it shows you are making real progress.

Who This Book Is For

Beginning C: Fifth Edition is designed to teach you how to write useful programs in C as quickly and easily as possible. By the end of *Beginning C*, you'll have a thorough grounding in programming the C language. This is a tutorial for those who've done a little bit of programming before, understand the concepts behind it, and want to further your knowledge by learning C. However, no previous programming knowledge on your part is assumed, so if you're a newcomer to programming, the book will still work for you.

What You Need to Use This Book

To use this book, you'll need a computer with a C compiler and library installed, so you can execute the examples, and a program text editor for preparing your source code files. The compiler you use should provide good support for the current International Standard for the C language, ISO/IEC 9899:2011, commonly referred to as C11. You'll also need an editor for creating and modifying your code. You can use any plain text editor such as Notepad or vi to create your source program files. However, you'll get along better if your editor is designed for editing C code.

I can suggest two sources for a suitable C compiler, both of which are freeware:

- The GNU C compiler, GCC, is available from `http://www.gnu.org` and supports a variety of operating system environments.
- The Pelles C compiler for Microsoft Windows is downloadable from `http://www.smorgasbordet.com/pellesc/` and includes an excellent IDE.

Conventions Used

I use a number of different styles of text and layout in the book to help differentiate between the different kinds of information. For the most part, their meanings will be obvious. Program code will appear like this:

```
int main(void)
{   printf("Beginning C\n");
   return 0;
}
```

When a code fragment is a modified version of a previous instance, I occasionally show the lines that have changed in bold type like this:

```
int main(void)
{
  printf("Beginning C by Ivor Horton\n");
   return 0;
}
```

When code appears in the text, it has a different typestyle that looks like this: `double`.

I'll use different types of "brackets" in the program code. They aren't interchangeable, and their differences are very important. I'll refer to the symbols ( ) as parentheses, the symbols { } as braces, and the symbols [ ] as square brackets.

Important new words in the text are shown in italic *like this*.