Preface

A language is not worth knowing unless it teaches you to think differently.

—Larry Wall (the creator of Perl)

Enthused by the success of my previous books on one of Dennis Ritchie's two creations and undeterred by the fierce competition, I decided to focus on Ritchie's other creation—the C language. A major course correction in the midst of my venture led me to include content that addressed the needs of beginners who lacked knowledge of computers. The consequent delay was perhaps inevitable, but the much awaited book is eventually here.

UNIX and C were developed simultaneously at Bell Labs four decades ago. Ritchie created the C language with an unusual objective—to rewrite the code for the UNIX operating system. That did happen eventually, but C was too well designed to remain stuck there. Today, C is a general-purpose language whose application areas extend from operating systems and device drivers to the "apps" on the iPhone. The question arises that in spite of the subsequent entry of numerous languages like Java, Python and Ruby, how has C been able to strongly hold on to its forte?

The answer lies in the features that make it difficult to classify C purely as a high-level language. C understands that resources can be scarce at times, so why waste them? Why use one byte when one bit will serve the purpose? Using *all* the symbols available on the computer keyboard, C provides a vast arsenal of 40 operators that can perform amazing operations on data. C programs are compact, fast but also tricky—and often cryptic. This book is all about revealing these tricks and demystifying one feature that routinely confuses people—pointers.

How This Book is Different

With ample books on C already available in the market, the obvious question must be asked: Why another book now? The plain truth is that none of the current books has really worked for me. I believe that a text book must identify potential areas of confusion and anticipate the questions a reader is likely to come up with. As done previously, I put myself in the student's shoes, posed these questions to myself and answered them as well as I could. This book is primarily meant for self-study even though some guidance at times may be helpful.

I strongly disapprove of filling up a book with an endless stream of programs and tables simply for the "sake of completeness." Instead, this book focuses on the essential concepts supported by well-annotated and properly indented programs. *Every program in this book has been adequately explained*. Many of these programs are based on real-life situations that may often seem familiar if not interesting. Wherever possible, a program has been progressively enhanced with the exposition of a new feature of the language.

The strong feature of the book is its pedagogical aids. Apart from the standard highlights (like *Note*, *Tip* and *Caution*), there is also a *Takeaway* feature which summarizes key information. Special asides entitled *How It Works* provide additional information for serious programmers. Rather than providing a list of terms at the end of each chapter, a detailed *Glossary* has been included for quick lookups. Extensive cross-referencing has been made available by embedding parenthesized section numbers in the text. I have no doubt that you'll often follow these references either for refreshing your memory or for knowing what lies ahead.

This book was originally meant to be a text on C programming, but changed circumstances demanded the inclusion of three separate chapters on computer fundamentals. This background should prove useful, but excessive coverage on computer basics (which often is the case) can dull your interest in C. So some basics have been provided that should more than fulfill the readers' needs. However, it is possible for the reader to jump straight to C by ignoring the first three chapters.

All programs in this book have been tested on the GCC compiler that is shipped with every Linux distribution. Many of these programs have also been compiled and executed successfully on Microsoft Visual Studio. However, these programs might not run error-free on every system. Bugs in these programs are possible though, and I welcome the ones that you may hit upon.

Acknowledgments

The prime mover for this project is Vibha Mahajan, who has been associated with my activities for the last two decades. Hemant K Jha graciously withstood the punches without losing a firm grip on the project. I appreciate the content-enhancing efforts put in by Shalini Jha and Ranjana Chaube, and Salman Khurshid who provided valuable inputs related to pedagogy. Finally, Satinder Singh and Anuj Shriwastava handled all production-related issues with indisputable competence. Thank you, everyone including other team members who have not been named here.

Dennis Ritchie created C because he felt that "it was a good thing to do." When you step into the commercial world and explore other languages, the foundation provided by C would stand you in good stead for years to come. Every other language—be it C++, Java or Python—will be well within your reach. It won't be long before you realize that learning C represented a major milestone in your life. It was more than just "a good thing to do."

SUMITABHA DAS