1 Introduction

Since its creation in 1987 Perl has become one of the most widely used programming languages. One measure of this is the frequency with which various languages are mentioned in job adverts. The site www.indeed.com monitors trends: in 2010 it shows that the only languages receiving more mentions on job sites are C and its offshoots C++ and C#, Java, and JavaScript.

Perl is a general-purpose programming language, but it has outstanding strengths in processing text files: often one can easily achieve in a line or two of Perl code some text-processing task that might take half a page of C or Java. In consequence, Perl is heavily used for computer-centre system admin, and for Web development – Web pages are HTML text files.

Another factor in the popularity of Perl is simply that many programmers find it fun to work with. Compared with Perl, other leading languages can feel worthy but tedious.

Perl is a language in which it is easy to get started, but – because it offers handy ways to do very many different things – it takes a long time before anyone *finishes* learning Perl (if they do ever finish). One standard reference, Steven Holzner's *Perl Black Book* (second edn, Paraglyph Press, 2001) is about 1300 dense pages long. So, for the beginner, it is important to focus on the core of the language, and avoid being distracted by all the other features which are there, but are not essential in the early stages.

This book helps the reader to do that. It covers everything he or she needs to know in order to write successful Perl programs and grow in confidence with the language, while shielding him or her from confusing inessentials.[1] Later chapters contain pointers towards various topics which have deliberately been omitted here. When the core of the language has been thoroughly mastered, that will be soon enough to begin broadening one's knowledge. Many productive Perl programmers have gaps in their awareness of the full range of language features.

The book is intended for beginners: readers who are new to Perl, and probably new to computer programming. The book takes care to spell out concepts that would be very familiar to anyone who already has experience of programming in some other language. However, there will be readers who use this book to begin learning Perl, but who have worked with another language in the past. For the benefit of that group, I include occasional brief passages drawing attention to features of Perl that could be confusing to someone with a background in another language. Programming neophytes can skim over those passages.

The reader I had in mind as I was writing this book was a reader much like myself: someone who is not particularly interested in the fine points of programming languages for their own sake, but who wants to use a programming language because he has work he wants to get done, and programming is a necessary step towards doing it. As it happens, I am a linguist by training, and much of my own working life is spent studying patterns in the way the English language is used in everyday talk. For this I need to write software to analyse files of transcribed tape-recordings, and Perl is a very suitable language to use for this. Often I am well aware that the program I have written is not the most elegant possible solution to some task at hand, but so long as it works correctly I really don't care. If some geeky type offered to show me how I could eliminate several lines of code, or make my program run twice as fast, by exploiting some little-known feature of the language which would yield a program delivering exactly the same results, I would not be very interested.

Too many computing books are written by geeks who lose sight of the fact that, for the rest of us, computers are tools to get work done rather than ends in themselves. Making programs short is good if it makes them easier to grasp and hence easier to get right; but if brevity is achieved at the cost of obscurity, it is bad. As for speed: computer programs run so fast that, for most of us, speeding them up further would be pointless. (For every second of time my programs take to run, I probably spend a day thinking about the results they produce.)

That does not mean that, in writing this book, I would have been justified in focusing only on those particular elements of Perl which happen to be useful in my own work and ignoring the rest – certainly not. Readers will have their own tasks for which they want to write software, which will often be very different from my tasks and will sometimes make heavy use of aspects of Perl that I rarely exploit. I aim to cover those aspects, as well as the ones which I use frequently. But it does mean that the book is oriented towards Perl programming as a practical tool – rather than as a labyrinth of fascinating intellectual arcana.

If, after working through this book, you decide to make serious use of Perl, sooner or later you will need to consult some larger-scale Perl book – one organized more as a reference manual than a teaching introduction. This short book cannot pretend to cover the reference function, but there is a wide choice of books which do. (And of course there are plenty of online reference sources.) Many Perl users will not need to go all the way to Steven Holzner's 1300-pager quoted above. The manual which I use constantly is a shorter one by the same author, *Perl Core Language Little Black Book* (second edn, Paraglyph Press, 2004) – I find Holzner's approach particularly well suited to my own style of learning, but readers whose learning styles differ might find that other titles suit them better.

Because the present book deliberately limits the aspects of Perl which it covers, it is important that readers should not fall into the trap of thinking "Doesn't Perl have a such-and-such function, then? – that sounds like an awkward gap to have to work round". Whatever such-and-such may be, very likely Perl has got it, but it is one of the things which this book has chosen not to cover.

Having said all that, though, let me stress that what the present book does teach you is not so limited as to be unusable in practice. Far from it. Many, many real-life programming tasks can be very successfully achieved in Perl without venturing beyond the elements of the language covered here. The programming examples you will encounter in this book will all be short programs to carry out little “toy” tasks, to make them easy to learn from; but although programs to achieve real-life tasks will often be *longer* (because the tasks involve more complications), they will not need to be different in kind. This book offers everything you need to begin working as a Perl programmer. Good luck, and have fun!