Preface

This book aims to be a *gentle* introduction to the main concepts of computer programming and the related subject of data structures and algorithms. Rather than focussing on particular programming languages that can appear alien and incomprehensible to beginners, it concentrates on the underlying concepts common to a whole range of programming languages. Whatever language you might be learning it should be of use if you are struggling to understand.

It is intended primarily for people with little background in the subject and for those for whom programming appears a little scary. The approach taken is that of understanding by analogy. The idea behind this approach was very clearly captured by Hideki Yukawa: the first Japanese winner of the Nobel Prize for Physics, here quoted from (Wilson 1999).

> *"Suppose there is something which a person cannot understand. He happens to notice the similarity of this something to some other thing which he understands quite well. By comparing them he may come to understand the thing which he could not understand up to that moment."*

He is discussing how scientists come to understand new areas at the frontiers of science. However, the words are just as applicable to those of us following behind and trying to understand things previously discovered by others.

Computer Science text books full of programming fragments can be hard to read. The details of particular languages can obscure the things that are common. It is the general concepts that matter most if a deep understanding of programming is to be obtained. Here I avoid discussing computer examples directly and instead explain the terminology and concepts using a variety of non-computing examples that should be familiar and understandable to all. By understanding how the concepts apply to everyday examples, I hope it will then be easier to follow the more technical details of a formal text book.

Of course analogy has to be treated with care. If pushed too far, the analogy breaks down and we can be left drawing wrong conclusions. By looking at each topic from a variety of different examples and looking at their commonality, I hope that this problem can be at least reduced.

People do not learn just by being told things or reading about them. The fact that I have read a booklet telling me how to juggle does not mean I can then pick up juggling balls and immediately juggle them without dropping them. I can only learn properly by lots of practice. We learn best by actually *doing.* This book also therefore contains lots of puzzles. If your aim in reading this book is to learn about programming you will help yourself achieve this if you actually try the puzzles rather than just reading them. If your aim is to learn how to program you will then need to actually go away and write programs. It is my hope that in reading this book before (or at the same time as) learning about programming more conventionally you will understand more deeply than otherwise.