Chapter 1: The Study of Data Structures

The study of data structures has long been considered the cornerstone and starting point for the systematic examination of computer science as a discipline. There are many reasons for this. There is the practical realization that almost every program of more than trivial complexity will need to manage collections of information, and so will require the use of one or more data structures. Learning the tools and techniques that have proven over a long period of time to be useful for software development will help the student become a more productive programmer. But that is only the first, and most practical, of many reasons.

Data structures are one of the easiest ideas to visualize *abstractly*, as most data structures can be matched with a metaphor that the reader will already be familiar with from everyday life. A *stack*, for example, is a collection that is managed in much the same fashion as a stack of dishes; namely, only the topmost item is readily available, and the top must be removed before the item underneath can be accessed. The ability to deal with abstract ideas, and the associated concept of *information hiding*, are the primary tools that computer scientists (or, for that matter, all scientists) use to manage and manipulate complex systems. Learning to program with abstractions, rather than entirely with concrete representations, is a sign of programming maturity.

The analysis of data structures involves a variety of mathematical and other analytical techniques. These help reinforce the idea that computer science is a *science*, and that there is much more to the field than simple programming.

It is in the examination of data structures that the student will likely first encounter the problems involved in the management of large applications. Modern software development typically involves teams or programmers, often dozens or more, working on separate components of a larger system. The difficulties in such development are typically not algorithmic in nature, but deal more with management of information. For example, if the software developed by programmer A must interact with the software developed by programmer B, what is the minimal amount of information that must be communicated between A and B in order to ensure that their components will correctly interact with each other? This is not a problem that the student will likely have seen in their beginning programming courses. Once more, abstraction and information hiding are keys to success. Each of these topics will be explored in more detail in the chapters that follow.

The book is divided into three sections. In the first section you will learn about tools and techniques used in the analysis of data structures. In the second part you will learn about the abstractions that are considered the classic core of the study of data structures. The third section consists of worksheets.

Active Learning

It is in the third section that this book distinguishes itself most clearly from other examinations of the same topic. The third section is a series of worksheets, each of which is tied to the material presented in earlier chapters. Simply stated, this book asks you to *do* more, and to *read* (or, if you are in a traditional classroom setting, *listen to*) less. By becoming a more *active* participant in the educational process, the knowledge you gain will become more familiar to you, and you will hopefully be more comfortable with your abilities. This approach has been used in a variety of different forms for many years, with great success.

While the worksheets can be used in an individual situation, it is the author's opinion (supported by experience), that they work best in a group setting. When students work in a group, they can help each other learn the basic material. Additionally, group work helps students develop their communication skills, as well as their programming skills. In the authors use, a typical class consists of a short fifteen to twenty minute lecture, followed by one or two worksheets completed in class.

At the end of each chapter are study questions that you can use to measure your understanding of the material. These questions are not intended to be difficult, and if you have successfully mastered the topic of the chapter you should be able to immediately write short answers for each of these questions. These study questions are followed by more traditional short answer questions, analysis questions that require more complex understanding, and programming projects. Finally, in recent years there has been an explosion of information available on the web, much of it actually accurate, truthful and useful. Each chapter will end with references to where the student can learn more about the topic at hand.

A Note on Languages

Most textbooks are closely tied to a specific programming language. In recent years it has also become commonplace to emphasize a particular programming paradigm, such as the use of object-oriented programming. The author himself is guilty of writing more than a few books of this sort. However, part of the beauty of the study of data structures is that the knowledge transcends any particular programming language, or indeed most programming paradigms. While the student will of necessity need to write in a particular programming language if they are to produce a working application, the presentation given in this book is purposely designed to emphasize the more abstract details, allowing the student to more easily appreciate the essential features, and not the unimportant aspects imposed by any one language. An appendix at the end of the book is devoted to describing a few language-specific features.