Preface

We developed this book to address the difficulty beginning students often find reading computer language texts. We felt that if we could get students involved in a text, hold their interest, and get them thinking about the meaning and uses of C code, we could make the process of learning a first language easier and fun. To accomplish this, we use a question and answer (Q&A) style. The reader's thought processes are stimulated by the same questions about code that students themselves often ask. By answering them directly and clearly, we focus the reader's attention on the important issues of C programming.

We also observed that most computer language texts have very few figures. Because visual images are so useful in teaching, we have made a concerted effort to create figures that are accurate yet easy to follow. These figures provide students reinforcement and clarification of concepts illustrated by the actions of the programs. In particular, we have three-dimensional sketches of the actions of loops and if control structures. Students can look at these figures and quickly grasp the flow of these structures. We believe that these figures are an improvement over the standard flow charts. We recognise that one of the most difficult topics for beginning C programmers is pointers. The foundation for our pointer figures is established early in the text with a table of the names, types, addresses and values of variables. Throughout the text, we use this table with arrows indicating how information at one location in memory relates to that in another location.

Many texts have numerous pages of code that are not adequately explained, and most beginning students are not capable or willing to independently decipher even relatively simple programs without guidance. In this book, we guide students through the code so that they understand both the operations and the thought processes that created the code. The goal is to make students realise what aspects of programming require extra thought and care and the importance of getting the details correct.

The unique style has been enthusiastically accepted among the students who have used early drafts of the book. Faculty unknown and unrelated to the authors have presented this text and asked the students' opinions. When compared with other texts, students have overwhelmingly preferred ours. Given this acceptance, we believe that you will find this book to be a valuable teaching and learning tool.

Text Organisation

The first chapter is an introduction to computing that assumes students have no basic knowledge other than using a computer for simple word processing. It introduces the concept of programming languages, describes hardware, the way information is stored in memory, computer languages, compilers, and software engineering. The purpose of this chapter is to introduce students to the way that computers work and the concepts behind software design.

Chapters 2 through 4 cover the fundamental aspects of a procedural programming language, basic syntax and control structures. C library functions are described and their use illustrated in these chapters. In Chapter 5, user-defined functions are covered, emphasising the concepts of modularity and reusable code. Pointers are gently introduced, being integrated with the development of functions that use addresses as arguments in function calls. After this chapter, the effects of using C features with user-defined functions are described.

Chapter 6 focuses on numerical arrays.

Chapter 7 describes both strings and pointers. Because strings are most commonly manipulated using addresses, this chapter is well suited to describing working with pointers to modify memory. Chapter 8 covers structures in C and their uses in creating linked lists, stacks, queues and binary trees. Also, large program design is covered in this chapter. This is included because engineering programs can quickly become very large. The importance of using C features to handle large programs is considered fundamental to preparing students for employment with firms that develop commercial software products in engineering.

We have called Chapter 9 (available at **www.mheducation.asia/olc/cprogramming**) An Introduction to C++, but it is actually much more. Because of the thorough coverage we have given C we are able to describe many of the core issues of object oriented programming with C++. Classes, encapsulation and polymorphism are described in simple terms. This chapter is richly illustrated. The simple language and illustrations provide students the background to use many of the fundamental C++ features.

Most chapters are divided into two parts, the Lessons and the Application Programs. The Lessons teach syntax, form and basic constructs. The Application Programs illustrate how what is taught in the lessons can be used to solve real engineering and computer science problems. The Application Programs show the thought processes a developer goes through to create a program. The goal of the Application Programs is to give students the ability to follow a structured methodology in developing their own programs.

Features

1. The book uses a simple question and answer approach that students find more friendly and accessible than standard narrative. The value of this approach lies in the authors' ability to craft the questions based on what students often ask and tailoring the answers in a manner that is easily understood.
2. Each Lesson begins with a single sample program: the source code accompanied by guided observations. Students gain understanding of C because they are compelled to follow the details of this code. Next, the output is revealed and following that, in the Explanation portion of the Lesson, a series of questions and answers are used to explain what the program is doing.
3. The Application Programs given in the second part of some chapters illustrate the usefulness of the C language for solving engineering and computer science problems. They are comprehensively explained. The examples focus on program design, software engineering, modularity and creating reusable code.
4. There are numerous figures illustrating programming concepts. Many of the figures are unique and give students an ability to grasp concepts quickly and easily.
5. A structured four-step method (becoming five steps after introducing strings and more complex data structures) of program development is illustrated in describing the Application Programs. The method includes creating structure charts and data flow diagrams.
6. Numerical method examples, which can be used in courses that combine programming and numerical methods, are included in the Application Programs.
7. The Lessons have annotated code that assists students in understanding program details and flow. The annotations help students focus on the code and highlight the important points that the code is demonstrating.
8. We realise that students typically will not follow multiple pages of code on their own. Therefore, the Application Programs, each of which can cover two or three pages, are explained completely. There are no multiple pages of unexplained code.
9. We know that students struggle with the concept of pointers. We also have found that for students to understand pointers, the figures representing them should be rooted to something that the students can visualize. It is not enough to have a box with an arrow pointing to another box. Using tables and grid-like sketches of memory, we have taken much of the mystery out of pointers. We have found that after reading our text, students "get" the concept of pointers.

10. Modification exercises after the Application Programs can be used for courses that have a laboratory. Instructors can tell the students in advance of the laboratory session to read a particular Application Program. During the lab, students can be guided through some of the changes that need to be made. Further changes can be assigned as home exercises.
11. Beginning students struggle with debugging because the process is new and foreign to them. Students often become frustrated because they must debug their very first program. Recognising this, we have included a detailed example of debugging very early in the text (Chapter 1). Beginners also find debugging loops difficult. In this text, we focus on loops and illustrate how values change as loops are executed. Students learn to trace loops and find errors. In addition, common beginners' errors are noted at appropriate locations throughout the text.
12. The True-False questions at the end of each Lesson (with solutions) allow students to quickly assess their progress in grasping the basics.
13. The Application Exercises at the end of most chapters can be used by an instructor for home assignments.
14. All of the programs in the book are available at **www.mheducation.asia/olc/cprogramming**. Students can modify and execute these programs to gain insight into how they operate.
15. We have included the last chapter, An Introduction to C++, as an online chapter available at **www.mheducation.asia/olc/cprogramming**. It actually covers more than just the basics. After reading this chapter, students will be able to use many of the fundamental aspects of object oriented programming.
16. Many of the Application Programs give students an introduction to numerical methods.

How To Use This Book

Student

In Chapter 1, you should focus on understanding what can be stored in memory, how a compiler works, and the steps in software engineering, and most importantly, your first C program. The rest of the chapters cover programming in C. Each lesson in these chapters begins with a short introduction to the lesson's source code. Use the introduction to guide you through the important points of the code. Then read the code and the annotations in the boxes. You can even try executing the code and observe how the program behaves. After doing this, make sure that you begin to understand the major topics being covered in the lesson. Then read the Explanation and do the True-False and short answer exercises. If you do not do well on the exercises, re-visit the lesson to clear any queries.

When you feel comfortable with the lessons in a chapter, begin the Application Programs. The purpose of these is to illustrate the thought processes that you would typically go through when you write your programs and to show practical uses of C. You will find as you write your own programs that you will be addressing many of the same issues raised in the Application Programs. In these, focus on learning the methodology and understanding the logic of each program. Remember, the logical flow is very important in programming. A statement can be correct in terms of syntax but totally wrong logically. Grasping the why and how of each Application Program, gives you the confidence to write your own program. Do not just read, but try out the programs, make changes to experiment variations, these help to crystallise what you have gained from the lessons. You will be able to explain the various behaviours of a program. Use this knowledge in writing the programs assigned by your instructor.

Instructors

For a full semester course aiming to equip students for advance programming in later courses, for example, in C programming with an introduction to C++, we recommend that you cover the entire text in the order presented. However, it is possible to cover the material in an order different from that given. For instance, Lesson 3.2 (Single Character Data) can be covered immediately before the first lesson of Chapter 7. Also, Lessons 8.7 (Creating Header Files), 8.8 (Use of Multiple Source Code Files and Storage Classes) and Function-like Macros and Conditional Inclusion which are available on **www.mheducation.asia/olc/cprogramming**, can be covered with the material in Chapter 5 (Functions) if so desired.

It is also possible to postpone some of the last lessons in Chapters 7, 8 and 9 and cover them as time permits. For instance, Lesson 7.9 (Pointer Notation versus Array Notation) can be postponed until immediately before advanced topics on pointers (Additional materials for Chapter 8, Pointers to Functions and Functions Returning Pointers are available on **www.mheducation.asia/olc/cprogramming**).

For a full semester course aiming to build a foundation in programming, we recommend that you cover through Lesson 7.8 and Lessons 7.10, 8.1, 8.2, 8.3, 8.4, and 8.5.

For a short course aiming to give students a general hands-on programming experience, students will be able to write valuable and sophisticated C programs if you cover just the first six chapters.

The book is filled with exercises. After each lesson are True-False and short answer exercises. The students should do these on their own. Some simpler programs are also assigned at the end of some of the lessons. These can be assigned as homework. One week is probably a sufficient amount of time for students to complete one of these programs.

The Modification Exercises at the end of the Application Programs can be used for courses that have a laboratory. Students should prepare for the laboratory by studying the pertinent Application Program. During the lab, students can be guided to make the modifications required by the exercises. Some of these exercises are straightforward while others are difficult. The ones that are difficult can be assigned as home exercises.

At the end of most of the chapters are Application Exercises. These tend to be the most challenging exercises in the book, and therefore are most appropriate as home assignments. Two to four weeks is an appropriate amount of time for doing them depending on the difficulty level. McGraw-Hill will furnish a solutions manual to selected exercises.

In addition, the book can be used as a reference for much of ANSI C. Reference tables are distributed throughout the book and some are given in the appendices.

Instructor Supplements

Instructors can access the following supplements at **www.mheducation.asia/olc/cprogramming**.

- Solutions Manual
- Powerpoint slides
- Test Bank
- Additional exercises
- Additional reading material