科技英语丛书

C Programming Language

C语言专业英语

尹 东 方 毅
徐 骏 周远远 编著

中国科学技术大学出版社

科技英语丛书

C语言专业英语

C Programming Language

尹 东 方 毅
徐 骏 周远远 编著

中国科学技术大学出版社

内 容 简 介

本书简明地介绍了应用广泛的计算机编程语言——C 语言。全书包括 9 章和附录:第 1 章主要介绍计算机软、硬件知识及软件工程的重要性;第 2 章详细介绍算法的概念以及程序结构、编程环境等;第 3 章介绍 C 语言的基本知识,包括编程方法、数据类型、表达式;第 4 章介绍 3 种结构,包括顺序、选择和循环;第 5 章介绍数组概念及其应用;第 6 章介绍函数概念及其应用;第 7 章全面细致地介绍指针概念;第 8 章介绍结构体概念;第 9 章介绍文件概念及其使用;附录提供了多类库中的多个函数的原型申明。

本书可作为高等学校本科生的授课教材,也可为广大编程工作者提供参考服务。

图书在版编目(CIP)数据

C 语言专业英语/尹东,方毅,徐骏,周远远编著. —合肥:中国科学技术大学出版社,2014.7

ISBN 978-7-312-03436-7

Ⅰ.C… Ⅱ.①尹… ②方… ③徐… ④周… Ⅲ.C 语言—程序设计—英语 Ⅳ.H31

中国版本图书馆 CIP 数据核字(2014)第 110624 号

出版 中国科学技术大学出版社
安徽省合肥市金寨路 96 号,230026
http://press.ustc.edu.cn
印刷 安徽省瑞隆印务有限公司
发行 中国科学技术大学出版社
经销 全国新华书店
开本 710 mm×960 mm 1/16
印张 8.5
字数 166 千
版次 2014 年 7 月第 1 版
印次 2014 年 7 月第 1 次印刷
定价 16.00 元

Preface

Until now, C programming language is regarded as one of most common languages in the world. Its properties include simple expressions, smooth control flows and data structures, a rich set of operators, etc. Generally, languages are classified two categories: high-level language and low-level language. But C language is called "middle-level" language. It means that C language not only possess well structure in high-level language, but also has low layer driver in low-level language. It is less restricted, powerful, and therefore more convenient and effective.

C language was designed firstly by Dennis Ritchie in 1973. From then, it spreads all over the world from its original place—Bell Laboratory. It has been a public language for all of programmers in the world. It has disseminated into China for many years and translated into many kinds of Chinese edition. But because of differences between English and Chinese, many readers often feel confused and uncomfortable in understanding.

In order to catch up the international advanced level of education, the Ministry of Education proposed a policy called "bilingual teaching" ten years ago. It greatly facilitates and promotes the use of original materials. However, from our teaching experiences for many years, we find that there are some problems. For one example, the contents in original book are excessive so that it is no time for students to read completely. For another example, the profound concept and text description make students understanding difficultly. So, for sake of rapid increasing the students' standard, as "middleware", we write this book.

This book has nine chapters and an appendix. In Chapter 1, we simply introduce the concepts of computer hardware and software, especially the importance of software engineering. In Chapter 2, we tell the readers how to solve a problem using the computer soul—algorithm. The basic knowledge about program structure, data type, and expression is described in Chapter 3. The three basic programming structures including sequence, selection and circulation are explained in Chapter 4. A very important concept—array is narrated in Chapter 5. Function is the characteristic of C language, because C is a functional language. So, in Chapter 6,

we discuss the function concept for readers. Pointer is a major feature of C language. We interpret it thoroughly in Chapter 7. In Chapter 8, we explain the structure design for a program. The last chapter that is Chapter 9 will describe emphatically the concept about file. Of course, in order to study, grasp and apply C language for programmers, we write an appendix so that the programmer can rapidly search many functions in kinds of libraries.

This book is written for students in connection with bilingual teaching. During the writing, we have always obtained many more support and encouragement from teachers and teaching assistants in C course group including Jia Boqi, Gu Weibin, Yin Long, Zen Lingzi, etc. Here, we express our sincere thanks to them.

At last, without a doubt, there are some wrong in this book because of our knowledge level. We sincerely accept your criticism and correction of our readers.

Yin Dong, Fang Yi, Xu Jun, Zhou Yuanyuan
2014.02 in Hefei

前　言

迄今为止，C语言被认为是世界上最通用的计算机编程语言之一。它具有简洁的表达式、流畅的控制流和数据结构、丰富的运算符集等特点。通常，语言分为两类：高级语言和低级语言，而C语言可叫做“中级语言”，其意思是C语言不仅具有高级语言好的结构，而且具有低级语言的底层驱动功能。它限制少、功能强大，因此更加方便、有效。

C语言最初是由Dennis Ritchie在1973年设计的。从那时起，C语言从其发源地——贝尔实验室传播到全世界，成为世界上编程人员的公共语言。它传播到中国已有多年了，被翻译成多个中文版本。然而，由于中英文的差别，许多读者经常在理解上感到困惑和不解。

为了赶上世界先进教育水平，教育部10年前提出了“双语教学”政策，极大地促进和推动了原版教材的使用。然而，根据多年的教学经验，我们发现了一些问题。比如，原版书内容过多导致学生无法完整学习；深刻的概念和文字描述让学生们理解困难。因此，为了快速提升学生的水平，作为“中间件”，我们编写了此书。

本书包括9章和附录。第1章简单介绍了计算机软、硬件概念，特别提及软件工程的重要性。第2章讲述如何用计算机核心——算法来解决问题。第3章描述有关程序结构、数据类型、表达式等基本知识。第4章解释顺序、选择和循环3种基本编程结构。第5章讲述非常重要的概念——数组。第6章讨论函数，函数是C语言的特点，C语言是函数型语言。第7章全面讲解指针，指针是C语言的重要特征。第8章介绍结构体概念。第9章着重描述有关文件的概念。最后，为方便学习、掌握和应用C语言，附录给编程人员快速查找各类库中的诸多函数提供便利。

本书是面向双语教学的学生而编写的。在写作过程中得到了C语言课程组老师和助教们，如贾伯琪、顾卫兵、尹龙、曾凌子等的大力支持和鼓励，在此，对他们深表谢意。

毫无疑问，由于我们的知识水平有限，书中存在疏漏在所难免，恳请广大读者批评和指正。

尹东　方毅　徐骏　周远远

2014年2月于合肥

Contents

Chapter 1 Fundamentals

Objectives

- To understand computer hardware.
- To understand computer software.
- To appreciate the programming languages and integrated development.
- To appreciate the software engineering.

It is nearly 70 years from the first birth of computer to present. Computer brings us huge change in our society and life. It has become one necessary tool of our work and life. So the studying and understanding of computer science is very important for us.

1.1 A Simple Structure of Computer Hardware

In these years, the development of computer experiences several gaps. Personal computer (PC) as example, it has been 8086, 80286, 80386,..., PⅤ, dual-core, etc. PC develops very rapidly. However, it still composes of two parts: one main computer, another peripheral. Main computer includes Motherboard, Center Process Unit (CPU), Main Memory, Adapters, Hard Disk, DVD Player, etc. Peripheral includes Mouse, Keyboard, Monitor, Printer, Scanner, Digitizer, Plotter, etc. Fig. 1. 1 is a model of PC. Fig. 1. 2 is an instance of each part. Fig. 1. 3 is another kind of computer whose name is notebook computer.

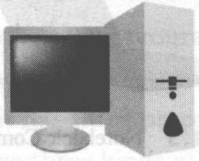
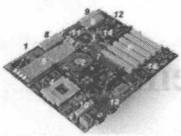
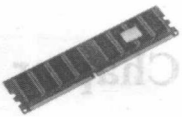
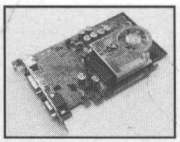


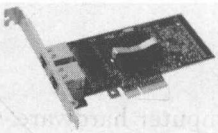
Fig. 1. 1 Model of PC


(a) Motherboard

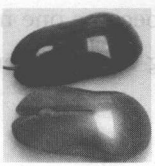

(b) CPU

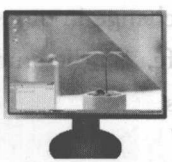

(c) Main memory




(d) Adapter (Video Card \\ Network Interface Card \\…)


(e) Hard Disk　　(f) DVD Player




(g) Mouse　　(h) Keyboard　　(i) Monitor

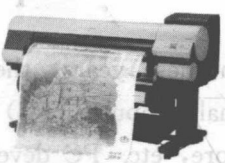

(j) Printer　　(k) Scanner　　(l) Digitizer　　(m) Plotter

Fig. 1.2　Instance for Each Part

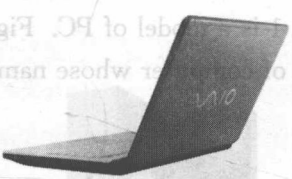


Fig. 1.3　Notebook Computer

1.2 The Computer Soul—Software

Software divides into two parts. One is called System Software such as Win 7, Windows XP, UNIX, and Linux. Another is called Application Software such as Microsoft Office, WinRAR, and C Programming Language. System Software is a system which controls and coordinates the computer and peripherals. It supports the development and operation of Application Software. Application Software is all kinds of programming languages used by user and its application program package programmed by language. Let us emphatically talk about programming software as follows.

Computer programming language includes low-level language and high-level language. Low-level language like machine language, Macro Assembler Language can directly operate those hardwares. But these languages are rarely used in recent years because of inconvenient program writing. High-level language including FORTRAN, BASIC, Pascal, and Lisp is current main programming tool. In this text, you will learn how to use the C programming language.

In fact, you can select any programming language to program an application software production. It all depends on your preferences. Because each of programming language has its own characteristics. There is no good or bad language, but which is more appropriate. Because of having low-level and high-level programming language characteristics, C programming language has been a crowd favorite. So C language is also called middle-level language. Please note that the word "middle-level" means that it has not only an ability to control and operate those low-level devices and also has good software structure. Good structure is an important aspect of a programming language.

1.3 C Language and Its Integrated Development

In general, the program in most of programming languages must first be translated into the low-level machine language appropriate to the computer on which the program will run. So it has three procedures—compiling, linking, executing. It is corresponding to have three kinds of files—source file, object file and executable file. A file containing program text is called source file. The program

text will be entered into source file by editor. The general process of entering or changing the contents of a source file is called editing that file. Once you have a source file, you can use the compiler to translate the source file into a format the computer can understand directly. The compiler translates the source file into the second file—object file. This object file contains the actual instructions appropriate for that computer system. Next, the object file is combined together with other object files to produce an executable file that can run on the system. These other object files typically include predefined object files—libraries which contain the machine language instructions for various operations commonly required by programs. The process of combing all the individual object files into an executable file is called linking. Once you have an executable file, you can run it on your computer. The entire process is illustrated by the diagram shown in Fig. 1. 4.

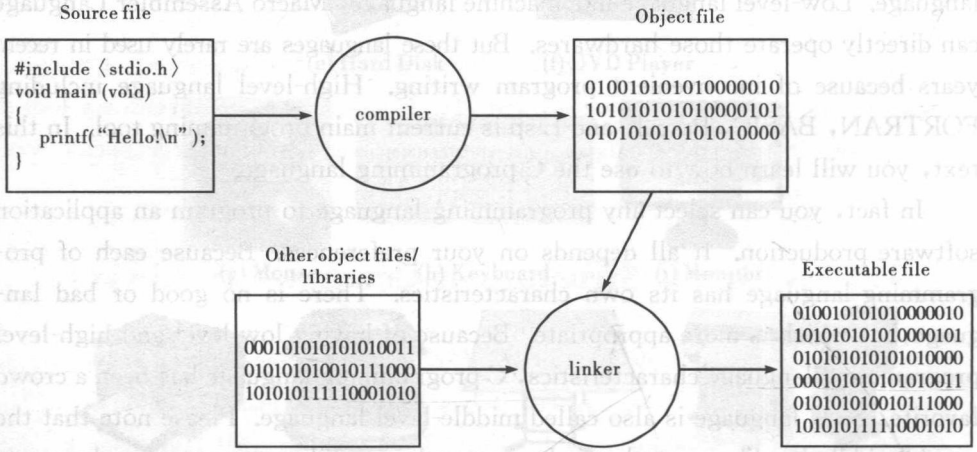


Fig. 1. 4 The Diagram of Entire Process

As Fig. 1. 4, in any case, the only file that contains something we can read is the source file. The other files contain information intended solely for the machine. As a programmer, all your work takes place in the context of the source. You edit it and use compiler to translate it.

Of course, programming languages have their own vocabulary and their own set of grammatical rules like human languages. These rules make it possible to determine that certain statements are properly constructed and the others are not. They are called syntax rules which determine whether a statement is legally constructed. When you compile a program, the compiler first checks whether

your program is syntactically correct. If you don't comply with the syntactic rules, the compiler will display an error message. Errors that result from breaking these rules are called syntax errors including warning and fatal error. Any error isn't allowed to happen. You must re-edit the program to correct the error until your program is completely right.

But syntax errors can be frustrating, particularly for new programmers. Although you don't violate the syntactic rules, your legal program somehow comes up with incorrect answers or fails to produce answers at all. Why? That's because you have made a mistake in the logic of the program which is the type of mistake called a bug. The process of finding and correcting such mistakes is called debugging. It is an important part of the programming process. Almost certainly, your legal program has a bug in it because your program is just suit for some data. As a programmer, you must constantly find bugs and fix them. Once that is done, you should find the next bug and fix it. You should always be skeptical of your own programs and test them as thoroughly as you can.

1.4 Software Engineering

One of the most surprising aspects of software development is that programs require maintenance. For most programs, the paying to programmers for maintaining the software after it has been released constitutes above 80 percent of the total cost. Software requires maintenance for two principal reasons. First, even after considerable testing, bugs can still survive in the original code. Then, when some unusual situation arises or a previously unanticipated load occurs, the bug makes the program fail. Thus, debugging is an essential part of program maintenance.

A reason that makes program maintenance so difficult is that most programmers don't write their programs for the long haul. To them it seems sufficient to get the program working and then they move on to something else. The discipline of writing programs that they can be understood and maintained by others is called software engineering. You are encouraged to write programs that demonstrate good engineering style.

It is very important that as you write your program, try to imagine how someone else might feel if called upon to read them two years later. A series of questions come up. Would your program make sense? Would the program itself

indicate to the new reader what you were trying to do? Would it be easy to change? Or would it seem obscure and convoluted? If you put yourself in the place of the future maintainer, it will help you to appreciate why good style is critical.

Many novice programmers are disturbed to learn that there is no precise set of rules you can follow to ensure good programming style. Good software engineering is not a cookbook sort of process. Instead, it is a skill blended with more than a little bit of artistry. Practice is critical. One learns to write good programs by writing them, and by reading others, much as one learns to be a novelist. Good programming requires discipline—the discipline not to cut corners or to forget about that future maintainer in the rush to complete a project. And good programming style requires developing an aesthetic sense—a sense of what it means for a program to be readable and well presented.

Summary

The purpose of this chapter is to set the stage for learning about computer science and programming. You have focused on what the programming process involves and how it relates to the domain of computer science.

The important points introduced in this chapter include:

(1) The physical components of a computer system—the parts you can see and touch—constitute hardware. Before computer hardware is useful, however, you must specify a sequence of instructions, or program, that tells the hardware what to do. Such programs are called software.

(2) Computer science is not so much the science of computers as it is the science of solving problems using computers.

(3) Programs are typically written using a high-level language that is then translated by a compiler into the low-level machine language of a specific computer system.

(4) To run a program, you must first create a source file containing the text of the program. The compiler translates the source file into an object file, which is then linked with other object files to create the executable program.

(5) The most serious type of programming error is one that is syntactically correct but that nonetheless causes the program to produce incorrect results or no results at all. This type of error, in which your program does not correctly solve

a problem because of a mistake in your logic, is called a bug. The process of finding and fixing bugs is called debugging.

(6) Most programs must be updated periodically to correct bugs or to respond to changes in the demands of the application. This process is called software maintenance. Designing a program so that it is easy to maintain is an essential part of software engineering.

Exercises

1. What is the difference between hardware and software?
2. What is meant by the term higher-level language?
3. What is the relationship between a source file and an object file? As a programmer, which of these files do you work with directly?
4. What is the difference between a syntax error and a bug?
5. What is meant by the term software maintenance?
6. Why is it important to apply good software engineering principles when you write your programs?

Chapter 2 Problem Solving Method

Objectives

- To understand why to use algorithm in programming.
- To program your first program.
- To keep away from error and run well.

You have learned about the basic principle in Chapter 1. From now on, you need to know something about algorithm, idioms, paradigms included, which are the programmer has to learn by heart.

2.1 Algorithms

Programming is the science of solving problems by computers. Before writing a program, a programmer must clearly understand the desired result and how the proposed program will produce it. In this regard, it is useful to realize that a computer program describes a computational procedure called an algorithm. An algorithm is a step-by-step sequence of instructions that describes how to perform a computation.

Firstly, we talk about something familiar to us, something like Gauss-as-a-child's story. When the teacher asked children how to add all the consecutive whole numbers from 1 to 100, only Gauss gave the following explanation: imagine that wrote the sum twice, once to the right and over backwards on top of each other, as follows:

$$1+2+3+\cdots+98+99+100$$
$$100+99+98+\cdots+3+2+1$$

If we add one column at all given the same (Fig. 2.1): $1+100=101$, $2+99=101$, $3+98=101$, etc. So the answer is 100 times 101 divided by 2 since we