Mc Graw Hill Education 大学计算机教育国外著名教材系列 影印版

Programming in ANSI C

Fourth Edition

标准C程序设计

（第4版）

E Balagurusamy 著

清华大学出版社

大学计算机教育国外著名教材系列（影印版）

Programming in ANSI C

Fourth Edition

标准 C 程序设计

（第 4 版）

E Balagurusamy

清华大学出版社
北京

E Balagurusamy
Programming in ANSI C, Fourth Edition
EISBN: 0-07-064822-0

图书在版编目(CIP)数据

标准 C 程序设计：第 4 版 = Programming in ANSI C, Fourth Edition: 英文 /（印） 巴拉古路萨米（Balagurusamy, E.）著. —影印本. —北京：清华大学出版社，2009.5
（大学计算机教育国外著名教材系列（影印版））
ISBN 978-7-302-19795-9

Ⅰ. 标… Ⅱ. 巴… Ⅲ. C 语言—程序设计—高等学校—教材—英文 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2009）第 045372 号

责任印制：孟凡玉

出版发行：清华大学出版社 **地 址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn **邮 编**：100084
社 总 机：010-62770175 **邮 购**：010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京鑫海金澳胶印有限公司
装 订 者：三河市金元印装有限公司
发 行 者：全国新华书店
开 本：185×230 **印张**：35.25
版 次：2009 年 5 月第 1 版 **印 次**：2009 年 5 月第 1 次印刷
印 数：1~3000
定 价：49.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话： 010-62770177 转 3103 产品编号：029501-01

出 版 说 明

进入21世纪，世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才，谁就能在竞争中取得优势。高等教育，作为培养高素质人才的事业，必然受到高度重视。目前我国高等教育的教材更新较慢，为了加快教材的更新频率，教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从1996年开始，与国外著名出版公司合作，影印出版了“大学计算机教育丛书（影印版）”等一系列引进图书，受到国内读者的欢迎和支持。跨入21世纪，我们本着为我国高等教育教材建设服务的初衷，在已有的基础上，进一步扩大选题内容，改变图书开本尺寸，一如既往地请有关专家挑选适用于我国高校本科及研究生计算机教育的国外经典教材或著名教材，组成本套“大学计算机教育国外著名教材系列（影印版）”，以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材，以利我们把“大学计算机教育国外著名教材系列（影印版）”做得更好，更适合高校师生的需要。

清华大学出版社

About the Author

E Balagurusamy, former Vice Chancellor, Anna University, Chennai, is currently Member, Union Public Service Commission, New Delhi. He is a teacher, trainer, and consultant in the fields of Information Technology and Management. He holds an ME (Hons) in Electrical Engineering and Ph.D. in Systems Engineering from the Indian Institute of Technology, Roorkee. His areas of interest include Object-Oriented Software Engineering, Electronic Business, Technology Management, Business Process Re-engineering, and Total Quality Management.

A prolific writer, he has authored a large number of research papers and several books. His best selling books, among others, include:

- Programming in C#
- Programming in Java, 3/e
- Object-Oriented Programming with C++, 3/e
- Programming in BASIC, 3/e
- Numerical Methods
- Reliability Engineering

A recipient of numerous honours and awards, he has been listed in the Directory of Who's Who of Intellectuals and in the Directory of Distinguished Leaders in Education.

Preface to the Fourth Edition

C is a powerful, flexible, portable and elegantly structured programming language. Since C combines the features of high-level language with the elements of the assembler, it is suitable for both systems and applications programming. It is undoubtedly the most widely used general-purpose language today.

Since its standardization in 1989, C has undergone a series of changes and improvements in order to enhance the usefulness of the language. The version that incorporates the new features is now referred to as C99. The fourth edition of ANSI C has been thoroughly revised and enlarged not only to incorporate the numerous suggestions received both from teachers and students across the country but also to highlight the enhancements and new features added by C99.

Organization of the book

The book starts with an overview of C, which talks about the history of C, basic structure of C programs and their execution. The second chapter discusses how to declare the constants, variables and data types. The third chapter describes the built-in operators and how to build expressions using them. The fourth chapter details the input and output operations. Decision making and branching is discussed in the fifth chapter, which talks about the if-else, switch and goto statements. Further, decision making and looping is discussed in Chapter six, which covers while, do and for loops. Arrays and ordered arrangement of data elements are important to any programming language and have been covered in chapters seven and eight. Strings are also covered in Chapter eight. Chapters nine and ten are on functions, structures and unions. Pointers, perhaps the most difficult part of C to understand, is covered in Chapter eleven in the most user-friendly manner. Chapters twelve and thirteen are on file management and dynamic memory allocation respectively. Chapter fourteen deals with the preprocessor, and finally Chapter 15 is on developing a C program, which provides an insight on how to proceed with development of a program. The above organization would help the students in understanding C better if followed appropriately.

New to the edition

The content has been revised keeping the updates which have taken placed in the field of C programming and the present day syllabus needs. As always, the concept of 'learning by example' has been stressed throughout the book. Each major feature of the language is treated in depth followed by a complete program example to illustrate its use. The sample programs are meant to be both simple and educational. Two new projects are added at the end of the book for students to go through and try on their own.

Each chapter includes a section at the beginning to introduce the topic in a proper perspective. It also provides a quick look into the features that are discussed in the chapter. Wherever necessary, pictorial descriptions of concepts are included to improve clarity and to facilitate better understanding. Language tips and other special considerations are highlighted as notes wherever essential. In order to make the book more user-friendly, we have incorporated the following key features.

- ***Codes with comments*** are provided throughout the book to illustrate how the various features of the language are put together to accomplish specified tasks.
- ***Supplementary information and notes*** that complement but stand apart from the general text have been included in boxes.
- ***Guidelines*** for developing efficient C programs are given in the last chapter, together with a ***list of some common mistakes*** that a less experienced C programmer could make.
- ***Case studies*** at the end of the chapters illustrate common ways C features are put together and also show real-life applications.
- The ***Just Remember*** section at the end of the chapters lists out helpful hints and possible problem areas.
- Numerous chapter-end ***questions*** and ***exercises*** provide ample opportunities to the readers to review the concepts learned and to practice their applications.
- ***Programming projects*** discussed in the appendix give insight on how to integrate the various features of C when handling large programs.

Supplementary Material

With this revision we have tried to enhance the online learning center too. The supplementary material would include the following:

For the Instructor

- Solutions to the debugging exercises

For the Student

- Exclusive project for implementation with code, step-by-step description and user manual
- Code for the two projects *(given in the book)*
- Two mini projects
- Reading material on C

This book is designed for all those who wish to be C programmers, regardless of their past knowledge and experience in programming. It explains in a simple and easy-to-understand style the what, why and how of programming with ANSI C.

E Balagurusamy

Contents

1

Overview of C

1.1 HISTORY OF C

'C' seems a strange name for a programming language. But this strange sounding language is one of the most popular computer languages today because it is a structured, high-level, machine independent language. It allows software developers to develop programs without worrying about the hardware platforms where they will be implemented.

The root of all modern languages is ALGOL, introduced in the early 1960s. ALGOL was the first computer language to use a block structure. Although it never became popular in USA, it was widely used in Europe. ALGOL gave the concept of structured programming to the computer science community. Computer scientists like Corrado Bohm, Guiseppe Jacopini and Edsger Dijkstra popularized this concept during 1960s. Subsequently, several languages were announced.

In 1967, Martin Richards developed a language called BCPL (Basic Combined Programming Language) primarily for writing system software. In 1970, Ken Thompson created a language using many features of BCPL and called it simply B. B was used to create early versions of UNIX operating system at Bell Laboratories. Both BCPL and B were "typeless" system programming languages.

C was evolved from ALGOL, BCPL and B by Dennis Ritchie at the Bell Laboratories in 1972. C uses many concepts from these languages and added the concept of data types and other powerful features. Since it was developed along with the UNIX operating system, it is strongly associated with UNIX. This operating system, which was also developed at Bell Laboratories, was coded almost entirely in C. UNIX is one of the most popular network operating systems in use today and the heart of the Internet data superhighway.

For many years, C was used mainly in academic environments, but eventually with the release of many C compilers for commercial use and the increasing popularity of UNIX, it began to gain widespread support among computer professionals. Today, C is running under a variety of operating system and hardware platforms.

During 1970s, C had evolved into what is now known as *"traditional C"*. The language became more popular after publication of the book *'The C Programming Language'* by Brian Kerningham and Dennis Ritchie in 1978. The book was so popular that the language came to be known as "K&R C" among the programming community. The rapid growth of C led to the development of different versions of the language that were similar but often incompatible. This posed a serious problem for system developers.

To assure that the C language remains standard, in 1983, American National Standards Institute (ANSI) appointed a technical committee to define a standard for C. The committee approved a version of C in December 1989 which is now known as ANSI C. It was then approved by the International Standards Organization (ISO) in 1990. This version of C is also referred to as C89.

During 1990's, C++, a language entirely based on C, underwent a number of improvements and changes and became an ANSI/ISO approved language in November 1977. C++ added several new features to C to make it not only a true object-oriented language but also a more versatile language. During the same period, Sun Microsystems of USA created a new language **Java** modelled on C and C++.

All popular computer languages are dynamic in nature. They continue to improve their power and scope by incorporating new features and C is no exception. Although C++ and Java were evolved out of C, the standardization committee of C felt that a few features of C++/Java, if added to C, would enhance the usefulness of the language. The result was the 1999 standard for C. This version is usually referred to as C99. The history and development of C is illustrated in Fig. 1.1.

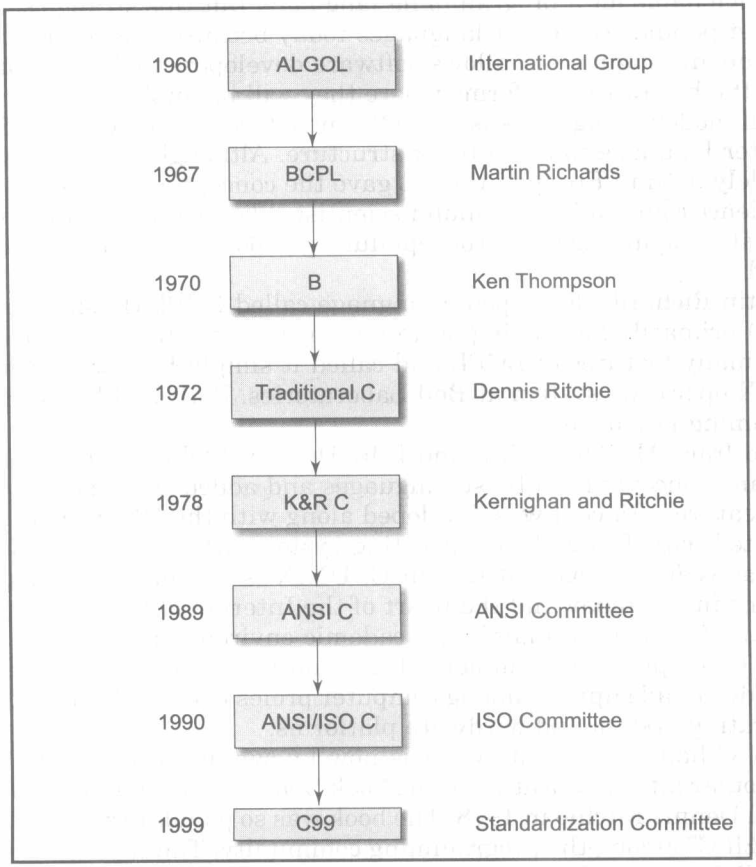

Fig. 1.1 *History of ANSI C*

Although C99 is an improved version, still many commonly available compilers do not support all of the new features incorporated in C99. We, therefore, discuss all the new features added by C99 in an appendix separately so that the readers who are interested can quickly refer to the new material and use them wherever possible.

1.2 IMPORTANCE OF C

The increasing popularity of C is probably due to its many desirable qualities. It is a robust language whose rich set of built-in functions and operators can be used to write any complex program. The C compiler combines the capabilities of an assembly language with the features of a high-level language and therefore it is well suited for writing both system software and business packages. In fact, many of the C compilers available in the market are written in C.

Programs written in C are efficient and fast. This is due to its variety of data types and powerful operators. It is many times faster than BASIC. For example, a program to increment a variable from 0 to 15000 takes about one second in C while it takes more than 50 seconds in an interpreter BASIC.

There are only 32 keywords in ANSI C and its strength lies in its built-in functions. Several standard functions are available which can be used for developing programs.

C is highly portable. This means that C programs written for one computer can be run on another with little or no modification. Portability is important if we plan to use a new computer with a different operating system.

C language is well suited for structured programming, thus requiring the user to think of a problem in terms of function modules or blocks. A proper collection of these modules would make a complete program. This modular structure makes program debugging, testing and maintenance easier.

Another important feature of C is its ability to extend itself. A C program is basically a collection of functions that are supported by the C library. We can continuously add our own functions to C library. With the availability of a large number of functions, the programming task becomes simple.

Before discussing specific features of C, we shall look at some sample C programs, and analyze and understand how they work.

1.3 SAMPLE PROGRAM I: PRINTING A MESSAGE

Consider a very simple program given in Fig. 1.2.

```
main( )
{
/*.....printing begins.....*/
    printf("I see, I remember");
/*.....printing ends.....*/
}
```

Fig. 1.2 *A program to print one line of text*

This program when executed will produce the following output:

```
I see, I remember
```

Let us have a close look at the program. The first line informs the system that the name of the program is **main** and the execution begins at this line. The **main( )** is a special function used by the C system to tell the computer where the program starts. Every program must have *exactly one main* function. If we use more than one **main** function, the compiler cannot understand which one marks the beginning of the program.

The empty pair of parentheses immediately following **main** indicates that the function **main** has no *arguments* (or parameters). The concept of arguments will be discussed in detail later when we discuss functions (in Chapter 9).

The opening brace "{ " in the second line marks the beginning of the function **main** and the closing brace "}" in the last line indicates the end of the function. In this case, the closing brace also marks the end of the program. All the statements between these two braces form the *function body*. The function body contains a set of instructions to perform the given task.

In this case, the function body contains three statements out of which only the **printf** line is an executable statement. The lines beginning with /* and ending with */ are known as *comment* lines. These are used in a program to enhance its readability and understanding. Comment lines are not executable statements and therefore anything between /* and */ is ignored by the compiler. In general, a comment can be inserted wherever blank spaces can occur—at the beginning, middle or end of a line—"but never in the middle of a word ".

Although comments can appear anywhere, they cannot be nested in C. That means, we cannot have comments inside comments. Once the compiler finds an opening token, it ignores everything until it finds a closing token. The comment line

```
/* = = = =/* = = = = = */ = = = = */
```

is not valid and therefore results in an error.

Since comments do not affect the execution speed and the size of a compiled program, we should use them liberally in our programs. They help the programmers and other users in understanding the various functions and operations of a program and serve as an aid to debugging and testing. We shall see the use of comment lines more in the examples that follow.

Let us now look at the **printf( )** function, the only executable statement of the program.

```
printf("I see, I remember");
```

printf is a predefined standard C function for printing output. *Predefined* means that it is a function that has already been written and compiled, and linked together with our program at the time of linking. The concepts of compilation and linking are explained later in this chapter. The **printf** function causes everything between the starting and the ending quotation marks to be printed out. In this case, the output will be:

```
I see, I remember
```

Note that the print line ends with a semicolon. *Every statement in C should end with a semicolon (;) mark.*

Suppose we want to print the above quotation in two lines as

```
I see,
I remember!
```

This can be achieved by adding another **printf** function as shown below: