高等学校计算机程序设计课程系列教材

C语言程序设计教程

姜恒远 主编

陶 烨 张 莉 张 萍 黄达明 编

高等教育出版社
HIGHER EDUCATION PRESS

高等学校计算机程序设计课程系列教材

C语言程序设计教程

C Yuyan Chengxu Sheji Jiaocheng

姜恒远　主编

陶　烨　张　莉　张　萍　黄达明　编

高等教育出版社·北京
HIGHER EDUCATION PRESS　BEIJING

内容提要

本书是为普通高等学校非计算机专业学生编写的教材。全书共分为10章，按C语言程序设计教学大纲并结合C语言程序设计等级考试的大纲要求，系统介绍C程序设计语言及其程序设计的方法与技术。本书取材适当、结构合理、概念清晰、循序渐进、习题丰富。为便于教学，提供了配套的PPT讲稿、习题解析、上机实验题、实验指导与参考答案，以及相应的教学资源网站。

本书既可作为高等学校非计算机专业学生的“C语言程序设计”课程教材，也可作为计算机专业本科生程序设计课程的教材与参考书，对于参加C语言等级考试的读者也有一定的参考价值。

图书在版编目(CIP)数据

C语言程序设计教程/姜恒远主编．—北京：高等教育出版社，2010.8

ISBN 978-7-04-030276-9

Ⅰ.①C… Ⅱ.①姜… Ⅲ.①C语言-程序设计-高等学校-教材 Ⅳ.①TP312

中国版本图书馆CIP数据核字（2010）第098688号

策划编辑	张　龙	责任编辑	张海波	封面设计	于文燕	责任绘图	尹　莉
版式设计	余　杨	责任校对	殷　然	责任印制	尤　静		

出版发行	高等教育出版社	购书热线	010-58581118
社　　址	北京市西城区德外大街4号	咨询电话	400-810-0598
邮政编码	100120	网　　址	http://www.hep.edu.cn
			http://www.hep.com.cn
		网上订购	http://www.landraco.com
经　　销	蓝色畅想图书发行有限公司		http://www.landraco.com.cn
印　　刷	北京京科印刷有限公司	畅想教育	http://www.widedu.com
开　　本	787×1092　1/16	版　　次	2010年8月第1版
印　　张	23.5	印　　次	2010年8月第1次印刷
字　　数	570 000	定　　价	32.00元

物料号　30276-00

序

社会信息化乃人类社会发展之必然，语文、数学、计算机犹如天、地、气鼎足而三，人人不可须臾离也。

使用计算机须熟谙程序设计语言，用以描述解题算法，亦即书写程序。C 语言乃步 FORTRAN、ALGOL 60、Pascal 等语言后尘之一种颇具代表性之语言。衡量程序设计语言之品质，自来即有多种观点：语言设计者观点、语言实现者观点、语言使用者观点，等等，此诸观点，互有异同，征诸“设计语言之目的在于应用”，语言使用者观点诚执其牛耳。据此观点，C 语言不失为一种较佳语言。坊间讲述 C 语言程序设计之读物汗牛充栋，然多为读者考虑、真正易学易用者甚鲜。

姜君恒远教授耕耘计算机领域数十载，年少问学于余，天资颖悟，刻苦好学，主讲 C 语言程序设计多有年所，经验宏富，近偕同仁陶烨、张莉、张萍、黄达明诸君编著是书，披览之余，感其数善存焉。一曰内容精当，书中涵盖 C 语言主要内容，取其本，舍其末，由浅入深，循序渐进；二曰面向读者，篇章安排，符合读者思维规律，且对易于混淆内容，多有提醒；三曰文笔通畅，非形式讲解程序设计语言，可能出现歧义，姜君此书，既能力避歧义，而又叙述严谨，通畅易读，实属难能可贵。

凛凛岁暮，瑞雪迎春，夜阑削稿，良用怃然。

徐家福

己丑除夕于南京大学

前　言

计算机是人类20世纪发明创造的最先进的计算工具。计算机的出现使科学技术研究在传统的理论推导和科学实验手段之外又增加了一种新的研究手段——“计算”。这种研究手段能够突破传统的研究手段的限制而获得更令人满意的效果。

为了适应信息化社会发展的需求，使学生具备利用计算机解决问题的基本技能、胜任专业研究与应用的需要，多数高等学校都开设了计算机语言程序设计课程。作为程序设计的入门，虽然选择过于灵活的C语言作为教学内容是否合适目前还有争论，但编者从多年的C语言程序设计教学的实践中认识到，C语言是一种结构化、应用面广、代码效率高、实用易学的优秀过程性程序语言，它能很好地体现程序设计的基本思想、概念与技术。

一本有针对性的C语言程序设计入门教材将给学生一个正确的指导方向，是获得良好教学效果的前提。国内外关于C语言的书籍或教材较多，其中不乏经典著作，但适合国内教学实际的并不多。目前见到的一些中译本的C语言书籍可以说是国外的较好教材，虽然内容丰富，叙述详细、透彻，但面面俱到导致重点不突出；而国内作者编写的C语言书籍大多比较简练，但在教学过程中需要针对培养学生的程序设计能力与程序设计的基本素质等方面做一些改进，才能达到理想的教学效果。

为把教学改革落实到具体的课程教学环节中，满足教学需要，根据教育部高等学校计算机基础课程教学指导委员会制定的《高等学校计算机基础课程教学基本要求》中关于C语言的教学内容和要求，结合国家及省市普通高等学校非计算机专业学生计算机基础知识和应用能力等级考试的C语言考试大纲，编者在长期从事C语言程序设计课程教学实践的基础上编写了本书。本书在如下方面做了努力。

1. 体现先进性

尽管目前大多数C编译程序还没有完全实现对C99的修改，但本书仍按C99叙述，因为C99保持了C的本质特性。选择何种C编译程序并不重要，除本书介绍的C编译环境外，读者可选用自己熟悉的C编译环境。

2. 考虑适用性

本书涵盖了C语言的绝大多数内容，讲授全书内容约需60学时。但考虑不同学校的实际情况，书中带“*”的章节可以选讲或不讲，对教学没有影响。由于本书内容的全面性与可取舍的灵活性，所以既可作为高等学校非计算机专业学生的“C语言程序设计”课程教材，也可作为计算机专业本科生程序设计课程的教科书与参考书，亦特别适合于自学者使用，尤其对参加C语言等级考试的读者会有一定帮助。限于篇幅，书中没有涉及的内容，若课程需要且时间允许，讲授者完全可以自行补充讲授。本书除了

介绍必要的语言知识、程序设计的方法与技术外，还介绍大量常用算法的程序设计。书中的例子和程序都经过充分测试，可以实际应用。

3. 注重实用性

学习C语言的目的是应用，本书把培养学生解决实际应用问题的能力与程序设计思想和方法作为重点，而摒弃语言语法规则的“说明书”式的叙述，注重书本内容与实践的结合。与本书配套的学习指导书中的内容有较强的应用性与实用性，充分利用C编译程序这一“最好的老师”可以有效地帮助读者学好本课程。

4. 安排合理性

本书遵循程序设计语言课程的特点与教学规律，精心组织了内容。本书内容编排的总原则是：突出重点，分散难点，弱化不常用功能，回避语言副作用。内容由浅入深、循序渐进，对于初学者或没有程序设计语言经验的读者来说不致感到太多困难。

不可否认，程序设计需要灵感、天赋与兴趣，但仅有这些是不够的。程序设计是一门科学，需要具备一定的基础知识，经过必要的训练与充分的实践才能掌握其规律。本书旨在帮助读者掌握程序设计的基本规律，扮演一个将读者引入计算机世界的向导角色。

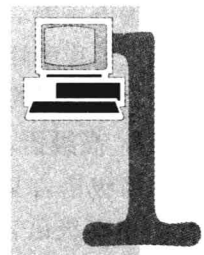
本书的第1、2、9、10章由姜恒远与黄达明编写，第3、4章由张莉编写，第5、6章由张萍编写，第7、8章由陶烨编写。本书在编写过程中得到了南京大学计算机科学与技术系的大力支持和帮助，很多专家、学者提出了不少宝贵建议和意见，特别是包亮、王远审阅和修改了有关章节，借此向他们表示感谢。

限于编者水平，书中欠妥之处恳请读者指正。编者邮件地址是 h. y. jiang@163. com。

编　者

2010年3月

目 录



C程序设计概述

第1章

1.1 程序设计语言、程序与程序设计

1.1.1 程序设计语言

程序设计语言是用以书写程序的语言。据统计，计算机程序设计语言有上百种，每种程序设计语言都有自身的特点和用途。按应用范围，有可用于多种不同目的的通用程序语言（如C、Pascal、FORTRAN等）与仅用于单一目的的专用程序设计语言之分；按使用者对处理事物的描述要求，有需要使用者指明处理过程、输入和预期输出的面向过程的程序设计语言与无须指明处理过程而得到所需结果的面向对象的程序设计语言之分；按使用方式，有具备人机交互功能的交互式程序设计语言（如BASIC）与不提供交互语言成分的非交互式语言（如FORTRAN、C等）之分；而如果从对计算机的依赖性级别，以及接近数学语言与自然语言的程度及程序设计语言的发展过程来看，通常将程序设计语言分为机器语言、汇编语言和高级语言。

1. 机器语言

计算机是人类20世纪发明、创造的最先进的计算工具。迄今为止，绝大多数计算机都是基于“存储程序与程序控制”原理工作的，即先把为求解问题而编排的计算机能直接执行的指令序列及处理的数据都以位的形式存储到计算机的主存储器中，由计算机自动按指定存储位置及预定的顺序逐条取出指令并根据指令规定的操作与操作数执行要求的运算。指令序列也称求解步骤，如图1-1(a)所示。

一台计算机的CPU能理解且能直接执行的指令集合即是该计算机的机器语言。用机器语言编写的求解问题的代码质量高，占用存储空间小，执行速度快，能由计算机直接执行；但是它依赖于具体计算机，难记忆、易出错、阅读理解困难，编程人员必须了解特定计算机的硬件及指令系统。

2. 汇编语言

汇编语言是一种符号化的机器语言，即用某种助记符号代替指令操作码、操作数。

用汇编语言表示的算法代码是符号化的指令序列，如图1-1(b)所示，比机器语言代码直观，但要用汇编程序翻译成等价的机器语言指令序列后才能由计算机执行。除此之外，汇编语言与机器语言有相同的特点。

3. 高级语言

高级语言是接近于自然语言和数学语言，在一定程度上与具体计算机无关的符号化语言。用高级语言描述的算法代码是一种符号化的语句序列，如图1-1(c)所示，也称其为源代码。

```
10110000
00001100
00101100
00000101
11110100
```
(a) 指令序列

```
MOV   AL,12D
SUB   AL,5D
HLT
```
(b) 符号化的指令序列

```
int main(void)
{
   printf("%d\n",12-5);
}
```
(c) 语句序列

图1-1　用机器语言、汇编语言、高级语言描述的计算“12-5”的代码

高级语言易学易用，代码易理解、调试、修改和移植，但大部分语言不支持对硬件的直接操作，其代码需要翻译成等价的指令序列后才能由计算机执行。处理方式分两种：编译方式与解释方式。

编译方式用相应的编译程序(compiler，或称为编译器)对高级语言描述的源代码进行若干次扫描(如语法分析、语义分析、生成各种对照表、生成中间代码、优化等)后生成目标代码。但目标代码并不能直接执行，还需要经过连接、装配成可执行指令序列代码后才能运行并获得结果。虽然编译方式实现复杂，但相对而言能产生效率较高的目标代码，编译一次，运行多次。这种翻译方式适合于结构复杂、要求运行效率高的应用开发。

解释方式用解释程序(interpreter)对源代码逐句扫描、处理、执行后直接获得结果。解释方式实现简单但效率低，同一代码每次运行都要进行解释。一般来说，交互式程序设计语言采用解释方式。

无论编译方式还是解释方式，通常高级语言的一条语句要对应多条机器指令，运行效率一般低于低级语言代码。综上所述，程序设计语言规定了一组记号和一组规则，用于描述或书写计算机程序。记号组合反映的内容之含义称为语义。

尽管计算机程序设计语言的差别很大，但无论哪种语言(即使是机器语言也是如此)，其基本语言成分都可归结为四大类：用以刻画程序所处理的数据对象的值、存储、类型的数据成分，用以规定程序设计中所能进行的运算(如算术运算、逻辑运算、集合运算等)成分，用以控制程序执行流程的控制成分，用以表达程序中数据输入输出的传输成分。

1.1.2　程序

按词典解释，“程序”是完成事情的先后次序。日常生活中程序概念比比皆是，例如学生报到注册过程、到图书馆查阅资料和借书过程、大会议程等。计算机中的程序概念与日常生活中的程序概念是类似的，即按特定的算法，用某种计算机语言描述完成指定任务的处理过程。由于计

算机是按人的意志工作的，要想达到人处理事情的灵活性，必须告诉计算机所有可能发生的情况，计算机才能“随机应变”。

从不同的角度观察计算机程序，其含义各异。从外部表示形式看，一个程序或者是位(bit)代码化的指令序列，或者是符号化的指令序列，或者是语句序列(如图 1-1 所示)。就程序本质而言，程序刻画了计算(数值计算与非数值计算，与处理同义)过程的处理对象(数据)与处理规则(算法)。因此从内容上看，一个程序包含算法描述和数据描述两方面的内容，是数据与算法的结合。从存储、功能与效果的角度看，程序是写在纸上或存储在外部存储介质上的一个静态实体，只有被装入主存启动执行后才起作用。从这个意义上说，计算机程序又是一个具有抽象性和动态性的逻辑实体，程序的作用与效果(可能是各种不同形式的输出)体现在程序的动态运行过程中。动态运行中的程序常被称为进程或任务，不同于静态实体的程序。

除上述程序的基本性质外，程序还有以下一些有趣特性。

① 程序具有易复制性，且复制的效果与原版完全相同。需要区别程序的复制、移植与仿真概念。复制程序指在同一媒体或另外一种媒体上重复产生程序副本。而移植程序是将一个程序转移到不同的硬件系统上运行，有时需要对程序代码作与硬件相关的修改，而不仅是简单的程序复制。仿真则是在另外一个系统中模拟程序功能。

② 因目前还无法从理论上证明一个程序能百分之百正确执行，程序的开发者或开发商只对程序运行错误导致的问题负有限责任。

③ 程序不同于任何工业产品，不会老化、磨损、失效和报废。

④ 由于程序本身的脆弱性，很难防止程序被恶意修改与破坏。

1.1.3 程序设计

苏黎世联邦工业大学的计算机科学家 N. Wirth 认为：程序 = 算法 + 数据结构。通俗地说，一个算法(algorithm)是对求解具体问题的一种方法的精确描述。严格地讲，算法是可接收零个或多个输入，至少产生一个输出且满足下列性质的操作序列。

① 有穷性：解题序列是一个有穷动作序列。

② 有限性：序列中的每个操作的执行次数和时间是有限的。

③ 确定性：序列中每个操作无歧义。

④ 唯一性：序列中仅有一个初始操作，且每个操作仅有一个后继操作。

⑤ 可行性：序列中的每个操作都能有效执行。

例如，假定有一张足够大的厚度确定的纸，不断将其对折，问多少次后可达到珠穆朗玛峰的高度(8844.43 米)？求解该问题的一种算法如下。

S1：0→COUNT

S2：输入纸张厚度→T

S3：如果 T≥8844.43，转 S6

S4：T×2→T

S5：COUNT + 1→COUNT，转 S3

S6：输出 COUNT

有了上述算法，只要用特定语言的合适语句按规定的语言规则描述这些操作，就形成了能够

完成特定功能的程序。用C语言实现该算法的程序见1.2.4节中的例1.4。

可见,程序的功能就是算法的具体实现,因此有“算法是程序的灵魂”的说法。本书的其余各章中将介绍常用的算法设计方法。

数据结构是数据的组织形式,涉及操作对象以及它们之间的关系和操作。

程序设计(programming)就像建筑师建造建筑物,算法就是建筑物的设计蓝图,数据就是各种建筑材料,语言就是建筑工具,程序员就是能工巧匠,他们利用建筑工具按设计图纸正确、有效地组织、使用各种建筑材料(数据结构)构造出所希望的建筑物(程序)。

总之,程序设计是指设计、编制、测试程序的方法和过程。不能把程序设计简单理解为编程,它还包含设计、调试、测试或验证程序的方法技术及程序设计风格等内涵。

N. Wirth的程序定律表明程序设计中算法是第一位的,是程序设计的重点,反映了以功能为中心的程序设计思想。但实际应用中,用户首先关心的是自己的数据能否被处理,然后才关心如何处理,用什么工具来处理等。例如撰写文章,文章内容是待处理的数据,而“记事本”或Word则是具体的编辑工具。从应用角度出发,很多系统已把程序设计的重心转移到数据上来。以数据为中心设计程序在Windows系统中已得到充分体现,例如,当用户把CD光盘(音乐数据)放入光盘驱动器中,音乐播放程序立即被启动并开始播放音乐;当用户用鼠标双击一个Word·文档时,Word程序立即启动。因此可以对N. Wirth的程序定律作一点修正:程序=数据结构+算法。其中“算法”与“数据结构”位置的简单对调虽然不影响程序的功能与含义,但却反映了不同的程序设计理念,体现了以数据为中心的程序设计思想。

无论以功能为中心的程序设计还是以数据为中心的程序设计,都是把功能与数据分割开来的传统程序设计方法。然而易变的功能与数据很难重用已有的程序。众所周知,现实世界是由客观对象组成的,如果把处理一类对象的算法(操作)及其抽象数据结构封装在一起,即对象=(算法+数据结构),则有程序=(对象+对象+…)。这就是所谓的面向对象的程序设计方法。本书仅介绍传统程序设计。

传统程序设计的本质是功能设计,一般采用结构化程序设计方法。结构化程序设计方法的基本原则是自顶向下(top-down)、逐步精化(stepwise refinement)、由抽象到具体的功能分解过程。例如,求二次方程 $ax^2+bx+c=0$ 的根的结构化程序设计过程如下。

S1:输入系数a、b、c

S2:求根计算

S3:输出根

进一步,将S1精化为:

S1.1:输入系数a、b、c

S1.2:判别输入的a是否为0,若为0,转S1.1重新输入系数a、b、c

将S2精化为:

S2.1:计算判别式 $b^2-4ac\rightarrow disc$

S2.2:计算 $\frac{-b}{2a}\rightarrow u$

S2.3:计算 $\frac{\sqrt{|disc|}}{2a}\rightarrow v$

将 S3 精化为：

S3.1：如果 disc = 0，计算并输出两个相同的根 x1 = x2 = u

S3.2：如果 disc > 0，计算并输出两个实根 x1 = u + v，x2 = u - v

S3.3：如果 disc < 0，计算并输出一对共轭复根 x1 = u + vi，x2 = u - vi

用 C 语言实现该算法的程序见 1.2.4 节中的例 1.3。

用结构化程序设计方法进行程序设计的优点在于程序具有层次性与结构性，每层相对独立，便于测试或验证，有利于分工实现。当然，结构化程序设计方法并不是最好的唯一程序设计方法。对结构化程序设计方法的主要批评归结为难以适应程序功能需求的变动，对数据结构及程序的易复用性未予以足够重视等。

排错、测试或验证程序的目的都是为了产生没有错误的程序。但这 3 个术语的含义不同。

排错(debugging)通常是指由程序设计者自己查找和改正程序中错误的过程，也称纠错目的是确定错误的性质、原因和位置且改正错误。

程序中的错误按开发程序的过程分为编译期错误、连接期错误与运行期错误。编译期错误是编译程序发现的语法错误；连接期错误是连接与装配程序在连接过程中发现的连接对象(调用的过程或函数、外部变量等)不存在的错误；运行期错误是指程序执行时才发现的逻辑性错误，如除数为 0、计算负数的平方根、结果不正确等。编译程序与连接程序能够发现前两种错误并会指出错误的原因与出处，很容易改正，而程序中存在的逻辑性错误则比较难于发现与更正。如果程序较小，可通过人工模拟程序执行来查找错误的原因并更正，但如果程序较大，则需要采用程序开发环境提供的调试程序的工具与手段，如单步执行程序(逐步跟踪程序的运行)，也可以设置“断点”(指示程序运行到某一处暂停)，观察变量的变化情况，找出问题的所在。事实上，在开发程序的过程中还可能存在“警告性”错误，例如编译程序发现程序中引用了某个未赋初值的变量，不要忽视这种错误信息，它们往往存在潜在的问题。

计算机程序与任何其他工业产品一样在交付使用前应加以严格测试。测试(testing)与调试的概念常常被混淆。测试的目的除了发现程序存在错误外，还需证实程序是否达到规定的功能，具有检验质量及性能评价等含义。程序测试直接关系到程序质量，在程序设计中占有相当大的比重。测试不完备的程序在重要应用中带来的损失是不可估量的，因此应予以足够的重视。例如在 1963 年美国发往火星的火箭中一个 FORTRAN 控制程序里，循环语句 DO 5 I = 1,3 误写成 DO 5 I = 1.3，仅一点之差致使火箭爆炸，造成数千万美元的损失。

验证(verification)程序正确性的理想方法是理论证明。尽管程序正确性证明(proof of program correctness)的理论与技术已取得显著的发展，但远未达到实用化程度，目前验证程序的方法主要还是依靠有局限性的测试技术。程序验证与测试的详细内容已超出了本书的范围，故不作说明。

程序设计风格指的是编码风格(coding style)，简单地说就是程序设计者表达程序的习惯方式。为什么程序设计要强调编码风格，甚至作为程序质量的一个方面来衡量，是因为良好的程序设计风格有助于提高程序的易读性、易测试性及易维护性。理由很简单，若一个投入运行的程序在运行过程中发现了错误或需要扩充功能，很难依赖原来的开发人员来解决(例如人员更替原因，即使是开发者本人也未必记得程序的细节)。如果程序容易读懂，就便于测试及排除错误，扩充功能，改善性能。因此，良好的程序设计风格是程序设计者应具备的基本素质。良好的程序

设计风格包括多方面的内容，下面是一些主要的基本指导原则。

① 规范化的程序书写格式。某些高级程序设计语言（如 FORTRAN）有规定的程序书写格式，但大多数程序设计语言的程序书写格式比较自由，例如 C 语言编译程序能够接受以如下形式书写的 C 语言程序。

```
#include <stdio.h>
#include <math.h>
int main(void){ double x; do{ scanf("%lf",&x); if(x>=0) printf("%lf\n",sqrt(x)); else printf
("agument error"); }while(x>=0);}
```

为了便于别人阅读和理解程序，对于书写格式要求不严格的程序设计语言，在书写程序时还是应该遵守约定俗成的书写格式。如在具有独立含义的语言元素（标识符、常量、运算符等）之间插入适量的空格，在不同的程序段之间用空行隔开，每行书写一条语句，用缩进对齐（indentation）的写法反映程序的结构层次等。可将以上程序段改写为以下形式。

```
#include <stdio.h>
#include <math.h>
int main(void)
{
    double x;

    do {
        scanf ( "%lf", &x );
        if ( x>=0 )
            printf ( "%lf\n" , sqrt (x) );
        else
            printf ( "argument error" );
    } while ( x>=0 );

}
```

② 注释程序。程序设计语言一般都允许使用注释行（comment）。适当注解程序对阅读理解程序有很大的帮助，也为日后维护程序提供明确的指导信息。注释分为功能性注释和序言性（prologue）注释。序言性注释通常位于每个程序模块的开头部分，对程序作整体说明，内容包括程序标题、程序的功能和目的、主要算法、数据描述、接口说明（包括调用形式、参数描述）等。功能性注释位于重要的程序段、特定的控制结构和不易理解的代码开始的地方，注解其后的语句、变量或程序段的作用及功能。

③ 标识符命名应“见名知义”。程序中如变量名、函数名、数组名、符号常量名等是由程序设计者按一定规则命名的可识别符号。好的标识符能体现符号所代表对象的含义，较易理解。如用 average 表示平均值、total 表示总量、count 表示计数等。标识符中要避免使用可能混淆的字母和数字，如字母 o、Q、l，数字 0 与 1。长标识符不利于程序移植。不主张使用由单个字母组成的标识符。不能使用已被程序设计语言赋予特殊意义的标识符（关键字）作为自定义标识符。也不宜使用语言中保留给系统本身使用的标识符（保留字，如 C 中__kcab 这样的以两个下画线开始的标识符）及系统的库函数名作为自定义标识符，避免引起名字混乱，产生含义覆盖。程序设

计中建议使用匈牙利命名法(由若干个英文单词组成,每个单词首字母大写,标识符前面的若干字符表示数据类型,如 IntCount、CharName)或使用有含义的英文单词与汉语拼音命名对象。

④ 用简明的方法表达算法。程序设计中程序的正确性、易读性、易维护性是很重要的,不提倡过分追求编程技巧。例如执行下面的 C 程序段将得到单位矩阵 a。

```
int n=5,i,j,a[5][5];
for(i=0;i<n;i++)
    for(j=0;j<n;j++)
        a[i][j]=((i+1)/(j+1))*((j+1)/(i+1));
```

上面程序段的功能显然不易理解,而下列写法却一目了然。

```
int n=5,i,j,v[5][5];
for(i=0;i<n;i++)
    for(j=0;j<5;j++){
        v[i][j]=0;
        v[i][i]=1;
    }
```

编写简明、清晰程序的另一方面,是对程序中的复杂计算与处理应尽量使用与之相关的系统库函数,既方便又能保证正确性。读者应注意了解、收集相关的信息与资料。

⑤ 提供友好的输入输出界面。程序通常是写给别人使用的,所以程序设计风格的一个原则是方便使用者,尽量避免因设计不当给用户带来麻烦。输入时,程序应预先给出诸如“请输入”字样的提示信息,必要时还应指出输入数据的形式(格式尽可能自由、简单,适应使用者的习惯与文化层次)、允许的范围;接收数据时应进行必要的校验并给出确认信息或纠正错误输入的建议信息等;输出的数据要有清晰、明了的格式;程序最好具备“在线”帮助功能;程序运行中要及时给出反映程序运行的状态信息。

1.2 C语言概述

目前广泛使用的 C 程序设计语言(简称 C 语言)是一种结构化、模块化、面向过程的、编译型的通用程序设计语言。这种程序设计语言既有一般高级程序设计语言的特性,又有低级程序设计语言的功能,程序的易读性、移植性好,特别适用于软件开发。因此,它一经问世就表现出很强的生命力,在短短几年中,便从最初的仅作为实验室内部开发软件用的工具语言迅速发展成为在诸多领域中普遍应用的、深受欢迎的主要程序设计语言之一。

1.2.1 C语言的发展

C 语言中的许多重要思想来自 Martin Richards 于 1967 年开发的一种能够直接对硬件操作的无类型语言(basic conbined programming language,BCPL)。而对 C 语言产生直接影响的是贝尔实验室的 Ken Thompson 于 1970 年为 PDP-7 计算机开发第一个 UNIX 操作系统在 BCPL 基础上改写的 B 语言。1973 年贝尔实验室在为新型 PDP-11 计算机配置 UNIX 操作系统的工作中,Dennis M. Ritchie 扩充、改进了 B 语言,形成了 C 语言。因此,C 语言是在 BCPL 和 B 语言的基础

上发展起来的,是与著名的 UNIX 操作系统的产生相辅相成的。最初 C 语言以 B. W. Kernighan 和 D. M. Ritchie 所著的 The C Programming Language 为标准,谓之 K&R C。

C 语言为日后在各种计算机系统上移植 UNIX 及各种系统软件发挥了极其重要的作用,同时 C 语言本身也得到了迅速推广和应用。但是在移植 C 语言和 UNIX 的过程中形成了多种 C 语言版本,各种版本在功能上及在函数库中设置的函数名称、种类和数量上存在着某些差异。为了改变这种局面,与 C 语言的发展相适应,美国国家标准学会(American National Standards Institute,ANSI)于 1983 年开始为 C 语言制定统一标准,该标准颁布于 1987 年,称之为 87 ANSI C。

1989 年,ISO 为 C 语言制定了国际标准 C89。最新的 C 语言国际标准是 1999 年颁布的 C99。本书按 C99 标准叙述,但由于很多 C 语言程序编译器遵循 C89 标准,本书将在 C99 标准与 C89 标准不一致时特别说明。

1.2.2 C 语言的特性

由于用 C 语言成功地书写了操作系统,所以人们一直把 C 语言看做"系统程序设计语言"。但读者必须明白,虽然 C 语言适用于系统程序设计,但它提供给用户的只是一种单线程控制流结构(single-thread control flow struction),即顺序、选择、循环之类的简单操作,并没有为用户提供任何专用于系统程序设计的设施,诸如并行、同步操作机构等。

虽然 C 语言起源于 BCPL 和 B 语言,且具有若干类似的特性,但从语言能处理的数据类型方面看它们是不同的。BCPL 和 B 是一种无类型语言(typeless language),处理的数据对象仅仅是机器字,数据处理通过特定的操作和函数调用来实现;而 C 语言中有相当丰富的数据类型,如字符数据、不同长度的整型数和浮点数以及指针、数组、结构、联合等具有结构特性的数据类型。

就表示能力与处理能力而言,C 语言是一种能把高级语言(如 BASIC、Pascal、FORTRAN 等)的表示能力与低级语言(如汇编语言)能直接处理和硬件有关的操作能力结合起来的语言。在面向用户方面,可以像使用高级语言那样利用 C 语言去方便、高效地书写、修改、移植、维护应用程序;而在面向机器方面,又可以像使用 Pascal、FORTRAN 等高级语言那样实现一般只能用汇编语言才能处理的功能,如直接进行位、字节、地址和寄存器操作等。这也是为什么 C 语言特别适合于系统软件设计且被广泛应用的一个重要原因。

C 语言是一种结构化程序设计语言。它为开发结构良好的程序(well-structured program)提供了先进的程序构造技术(程序的逻辑结构由顺序、选择和循环 3 种基本结构组成)、代码书写技术(程序代码以代数式形式书写)、数据构造技术(各种不同的数据类型定义,如数组、结构、联合、枚举等)。所有这些技术保证了用 C 语言能便于应用自顶向下、逐步精化的结构化程序设计方法,编写出高效的、易于阅读与维护的程序。

C 语言中的函数是构成一个 C 语言程序的基本构件。函数是具有特定功能的、独立的并具有清晰引用接口的代码块。C 语言的这种函数结构特性不仅使得大型复杂的问题可分解成若干功能单一的模块来独立编写,而且更重要的是它能使程序具有层次结构特性。因此,C 语言是一种便于模块化程序设计的语言。

相对一些进行严格类型检查的高级程序设计语言而言,C 语言不能算做一种强类型语言(strongly-typed language)。它在数据转换方面提供相对自动的转换操作,但不具数据转换的任