21世纪高等学校计算机公共课程“十二五”规划教材

C语言程序设计与实践

夏 耘 主编
臧劲松 黄小瑜 副主编

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

十二五 21世纪高等学校计算机公共课程“十二五”规划教材

C语言程序设计与实践

C YUYAN CHENGXU SHEJI YU SHIJIAN

夏 耘 主编
臧劲松 黄小瑜 副主编

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书是根据教育部对计算机小公共课程——“程序设计及应用”的教学要求编写而成，吸取国内、外先进教材的经验，融入计算思维，力求兼有国内、外教材的优点，增加了可供学生应用的与本学科有关的题目，让学生学有所用，从而激发其学习兴趣，达到理论和实践相结合的目的，使学生获得尽可能好的学习效果。本书贯彻启发思辨原则，以设计创新的启发式教学内容为纲，将启发式教学方法变成可操作的教学方法，通过任务驱动、项目引领实施可操作的启发式教学，实现“教”与“学”互动，充分调动学生学习的主动性和创造性，达到创新能力的培养和提高教学效果的目标。

全书从实用角度出发，在每一章中都设计了课堂练习、课后实验和课外练习，为每个知识点设计有趣实用的情节，让学生动手、动脑，反复练习，从而达到巩固程序设计中所涉及知识点的目的。

本书以 Code block 为编程环境，对程序设计基本步骤、基本知识、语法、编程方法和常用算法进行了较为系统、详细的介绍，实例丰富、有趣，阅读轻松，操作容易。

本书适合作为高等院校非计算机专业的教材，也可作为计算机成人教育各类进修班与培训班，以及广大工程技术人员和管理人员学习计算机编程知识的教材。

图书在版编目（CIP）数据

C 语言程序设计与实践 / 夏耘主编. —北京：中国铁道出版社，2013.1

21 世纪高等学校计算机公共课程“十二五”规划教材

ISBN 978-7-113-15678-7

Ⅰ. ①C… Ⅱ. ①夏… 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（2012）第 319626 号

书　　名：C 语言程序设计与实践

作　　者：夏耘　主编

策　　划：吴宏伟　孟　欣　　**读者热线：400-668-0820**

责任编辑：孟　欣　徐盼欣

封面设计：付　巍

封面制作：白　雪

责任印制：李　佳

出版发行：中国铁道出版社（100054，北京市西城区右安门西街 8 号）

网　　址：http://www.51eds.com

印　　刷：化学工业出版社印刷厂

版　　次：2013 年 1 月第 1 版　　2013 年 1 月第 1 次印刷

开　　本：787mm×1092mm　1/16　**印张：18.25**　**字数：441 千**

印　　数：1～3 000 册

书　　号：ISBN 978-7-113-15678-7

定　　价：40.00 元

前言

FOREWORD

Richard M. Karp 提出的“计算透镜”（Computational Lens）理念被认为是未来 20 年计算机科学可能的发展方向之一。其核心理念是将计算作为一种通用的思维方式，通过这种广义的计算（涉及信息处理、执行算法、关注复杂度）来描述各类自然过程和社会过程，从而解决各个学科的问题。这一理念试图将计算机科学由最初的数值计算工具、仿真与可视化技术以及后来基于网络、面向多学科的 e-Science 平台，变成普遍适用于自然和社会领域的通用思维模式。

计算思维的切入点就是程序设计，程序设计离不开编程语言。C 语言以其小巧、灵活、高效等特点成为当今软件开发的主流。C 语言作为大学生学习程序设计的入门语言，教育部考试中心及大部分省市也将 C 语言程序设计纳入计算机等级考试的科目。

本书将实际问题作为切入点，将计算思维融入程序设计中，旨在倡导学生通过程序设计提升发现问题、解决问题与技术创新的能力。读者通过本书设置的循序渐进的教程，从体验程序、调试程序、编写部分程序到独立编写完整的程序；在学习中会遇到不少问题，本书提供了资料包，为每章教学中可能出现的问题进行归纳、总结，倡导在学习中发现问题—解决问题—归纳总结的教学法，解决了学生长期以来学习 C 语言程序设计上课能听、下课不能解题，谈编程色变的问题。

本书注重基本概念的系统化，叙述简明扼要，书中对教学重点逐一进行了点拨。本书内容精练，结构合理，重点突出，对读者可能遇到的难点做了十分清楚和详细的阐述。

本书整理了课堂的教案，注重训练环节，体现了在理论指导下让学生动手、动脑的基本思想，提出理性思维和理性实践的观点。按照建构主义的学习理论，学生作为学习的主体，在与客观环境（指所学内容）的交互过程中构建自己的知识结构。本书引导学生在解题编程中探索其中带规律性的认识，进而将感性认识升华到理性高度，这样学生就能举一反三。本书可供各层面学生、教师、自学应试者阅读。

本书第 1～5 章为基础篇，每章设置了课堂练习、课后实验和课外练习，将该章应掌握的基本算法融入其中，基础篇中主要将学生引进门（程序设计入门）；第 6 章为实践篇，介绍项目的开发流程，在项目设计中分别应用指针、链表、文件以及数据库技术，通过实践篇的学习读者能独立编写完整的程序，可以结合实际生活创作属于自己（项目开发组）的作品。

本书由夏耘任主编，由臧劲松、黄小瑜任副主编。第 1～3 章由夏耘执笔，第 4 章由黄小瑜执笔，第 5 章由臧劲松执笔，第 6 章由夏耘和臧劲松共同执笔。

本书由上海理工大学光电与计算机与工程学院计算机基础教学的一线教师共同编写，在编写过程中，组织了集体统稿、定稿，并得到了清华大学、交通大学、复旦大学、华东师范大学、华东理工大学、上海理工大学、上海大学等高校计算中心各位老师的帮助。在此一并致谢。

由于时间仓促和水平有限，本书中难免还存在一些不足之处，请广大读者批评指正。

编者

2012 年 11 月

CONTENTS 目录

第 1 篇 基础篇

第 2 篇 实践篇

第1篇

基础篇

第1章 构建程序

构建桥梁的是建筑师，他们具有桥梁专业知识，按桥梁设计的规范为城市构建一座座美丽的大桥；构建程序的自然是程序员，从一个普通人到程序员需要经历怎样的磨练？不同的人用相同的素材却能构建不同的效果，有精致的也有粗糙的，究其原因是程序员的素质。为此，构建精致程序应从提升程序员的修养入手，程序员的修养就是为人之道，此道是信息社会走向成功之道，希望读者能通过本书的学习修得此道，走向成功。

1.1 程序员的修养

1.1.1 程序员的为人之道

当你读本书时，请把自己当成一名程序员，了解程序员的修养是第1课，它将带你步入程序世界。在程序设计的过程中，可能会遇到不计其数的困难和问题，可能有极多的挫折和失败，而成功只有一次。因此，一个程序员必须具有永不放弃的精神，要有自信，每时每刻都要鼓励自己："你能解决所出现的问题，你会成功"，这将是你的座右铭。只有这样才能坚持下去。

程序员为解决一个问题，可能连续十几甚至几十小时坐在计算机前不停地工作。一个问题解决了，可能又有其他的问题出现。如果此时坚持不了，可能前功尽弃。轻易言败的人是不可能成功的。执着是程序员打开成功之门的第1把钥匙。

当你坐在计算机前玩十几甚至几十小时的游戏时会感觉十分愉快，而坐在计算机前调试程序不足30分钟就想砸计算机，或气冲冲地离开。为什么会出现这个现象呢？因为你始终站在自己的角度考虑问题，完全忽视了计算机的想法。调试程序是人通过程序掌控计算机，要掌控计算机必须了解计算机，"知彼知己者百战不殆"。当你了解计算机的想法后，就会找到计算机不受掌控的原因了，也就能顺利解决问题。

你的学长在刚开始学程序设计时遇到的问题是程序不执行，他想砸计算机，但他没有这样做，而是选择了求助。其实，他也可以通过换位思考自己解决所遇问题，但求助也是个不错的方法。在后续的学习中，他学会了如何与计算机交流。

计算机"不给力"不执行的程序代码为：

```
void f1()                          /*自定义函数 f1()*/
{    printf("周一: 1: 9: 00-10: 00 商务会晤\n,2: 10: 30-12: 30 例会\n, 3: 14: 00-17:
     00 公司计划讨论会\n");        /*在屏幕上显示双引号中的文字，显示结束后光标定位在文字
                                   的下一行行首*/
}
void ShowMenu()                    /*自定义函数 ShowMenu()*/
{    char *str[5]={"请选择:","1:输出周一工作计划","2: 输出周二工作计划","3: 输出
     周三工作计划","0:退出"}; /*定义字符指针数组 str 并初始化，即为每个元素赋值*/
```

```
    int i;                          /*定义整型变量 i*/
    for(i=0;i<5;i++)                /*循环语句，循环次数 5 次*/
      printf("%s\n",str[i]);        /*循环次数 5 次，每次在屏幕上显示字符指针数组
                                    str 所指向的字符串*/
  }
void f2()                           /*自定义函数 f2()*/
{   printf("周二: 13: 00-17: 00 财务会议\n");
    /*在屏幕上显示双引号中的文字，显示结束后光标定位在文字的下一行行首*/
}
void f3()                           /*自定义函数 f3()*/
{   printf("周三: 1: 9: 00-10: 30 秘书会议 2: 12: 00-14: 30 汇报会\n") ;
    /*在屏幕上显示双引号中的文字，显示结束后光标定位在文字的下一行行首*/
}
void main()                         /*主函数*/
{   char ch;                        /*定义字符变量 ch*/
    for(;;)                         /*循环语句，循环无数次*/
    {   system("cls");              /*清除屏幕上本操作之前所显示的所有信息*/
        ShowMenu();                 /*执行自定义函数 ShowMenu()*/
        ch=getch();                 /*键盘输入一个字符存放到字符变量 ch 中*/
        switch(ch)                  /*根据字符变量 ch 的值执行不同的自定义函数*/
        {
          case '1':
            /*键盘输入字符变量 ch 为字符常量 1 时执行下列语句*/
            f1();                   /*执行自定义函数 f1()*/
            system("pause");
            /*系统等待键盘输入任意值后显示后续的信息*/
            break;                  /*退出此开关语句*/
          case '2':
            /*键盘输入字符变量 ch 为字符常量 2 时执行下列语句*/
            f2();                   /*执行自定义函数 f2()*/
            system("pause");        /*执行函数 system()*/
            break;                  /*退出此开关语句*/
          case '3':
            /*键盘输入字符变量 ch 为字符常量 3 时执行下列语句*/
            f3();                   /*执行自定义函数 f3()*/
            system("pause");
            /*系统等待键盘输入任意值后显示后续的信息*/
            break;                  /*退出此开关语句*/
          case '0':
            /*键盘输入字符变量 ch 为字符常量 0 时执行下列语句*/
            return;                 /*退出此主函数*/
            default:
            /*键盘输入字符变量 ch 为其他字符时执行下列语句*/
            break;                  /*退出此开关语句*/
        }
    }
}
```

在上述代码中，每一条语句都正确，只是其中的 void f3()函数定义时漏了函数结束符号“}”。计算机是个十分严谨的工作者，当它收到全部指令后才开始执行指令，而上述程序由于指令不完整，因此计算机无法执行。

如果你认为坐在计算机前几十小时调试程序中的一个问题代表程序员有耐心，也是错误的，在处理问题时需要事先估算解决问题所需要的时间，如果在这期间解决不了问题，就应去寻求帮助，而不是钻牛角尖。

整个程序设计的过程就是一个研究学习—应用—再研究学习—再应用的过程。一名优秀的程序员认为自己所掌握的技术不足，需要再提高。而自满自足的人则往往是刚入门就感觉自己很伟大，这样的人会很快落后以致落伍。

追求新的、时尚的技术没有错，但在程序设计时，适应项目（或问题）需求永远优先。

1.1.2　了解编程

计算机编程语言的种类繁多，按照计算机语言的发展过程，大致可将计算机语言分成3种类型。

1．机器语言

机器语言具有如下特点：

（1）机器语言是任何计算机能够直接识别和执行的语言。

（2）因为不存在翻译过程，所以机器语言的执行速度快。

（3）设计人员必须熟知机器指令的所有二进制符号。

（4）用机器语言编写的程序在不同种类的计算机上不能通用。

2．汇编语言

在汇编语言中，枯燥单调的二进制代码被一些表示指令功能的英文缩写词（助记符）所替代，显然，MOV可以表示传送操作，ADD可以表示加法操作，而LJMP则可以表示程序的跳转（Long Jump），这确实给编写程序带来了很大的方便。

汇编语言具有如下特点：

（1）汇编语言虽然相对于机器语言而言使用简单，但它同机器语言一样，也属于“低级语言”，它是“面向机器”的，也就是说，不同种类的计算机有着不同的汇编语言。

（2）用汇编语言编写的程序并不能直接被机器识别和运行，它必须通过“编译”和“连接”，翻译成机器语言后才能被计算机执行。

3．高级语言

高级语言的发展也经历了从早期语言到结构化程序设计语言（Programming Language），从面向过程到非过程化程序设计语言的过程。相应地，软件的开发也由最初的个体手工作坊式的封闭式生产，发展为产业化、流水线式的工业化生产。

20世纪60年代中后期，软件越来越多，规模越来越大，而软件的生产基本上是各自为战，缺乏科学规范的系统规划与测试、评估标准，其恶果是大批耗费巨资建立起来的软件系统由于含有错误而无法使用，甚至带来巨大损失，软件给人的感觉是越来越不可靠，以致几乎没有不出错的软件。这一切，极大地震动了计算机界，史称“软件危机”。人们认识到：大型程序的编制不同于小程序，它应该是一项新的技术，应该像处理工程一样处理软件研制的全过程。程序的设计应易于保证正确性，也便于验证正确性。于是人们提出了结构化程序设计方法，第一个结构化程序设计语言——Pascal语言出现，标志着结构化程序设计时期的开始。

20世纪80年代初开始，在软件设计思想上又产生了一次革命，其成果就是面向对象的程序设计。在此之前的高级语言几乎都是面向过程的，程序的执行是流水线似的，在一个模块被

执行完成前，不能做别的事情，也无法动态地改变程序的执行方向。这和人们日常处理事务的方式是不一致的，对人而言是希望发生一件事就处理一件事，也就是说，不能面向过程，而应是面向具体的应用功能，也就是对象（Object）。其方法就是软件的集成化，如同硬件的集成电路一样，生产一些通用的、封装紧密的功能模块，称之为软件集成块，它与具体应用无关，但能相互组合，完成具体的应用功能，同时又能重复使用。对使用者来说，只关心它的接口（输入量、输出量）及能实现的功能，至于如何实现，使用者完全不用关心。C++、Visual Basic、Delphi 就是面向对象的程序设计语言的典型代表。

高级语言所编制的程序是不能直接被计算机识别的，必须经过转换才能被执行。按转换方式，高级语言可分为两类：

（1）解释类：执行方式类似于日常生活中的“同声翻译”，应用程序源代码一边由相应语言的解释器“翻译”成目标代码（机器语言），一边执行，因此效率比较低，而且不能生成可独立执行的可执行文件，应用程序不能脱离其解释器。但这种方式比较灵活，可以动态地调整、修改应用程序。

（2）编译类：编译是指在源程序执行之前，就将程序源代码“翻译”成目标代码（机器语言），因此其目标程序可以脱离其语言环境独立执行，使用比较方便，效率较高。但应用程序一旦需要修改，必须先修改源代码，再重新编译生成新的目标文件（*.obj）后才能执行。

高级语言的下一个发展目标是面向应用，也就是说：只需要告诉程序要干什么，程序就能自动生成算法，自动进行处理，这就是非过程化的程序设计语言。

程序设计语言是用于编写计算机程序的语言。语言的基础是一组记号和一组规则。根据规则由记号构成的记号串的总体就是语言。在程序设计语言中，这些记号串就是程序。程序设计语言包含 3 方面，即语法、语义和语用。语法表示程序的结构或形式，即表示构成程序的各个记号之间的组合规则，但不涉及这些记号的特定含义，也不涉及使用者。语义表示程序的含义，即表示按照各种方法所表示的各个记号的特定含义，但也不涉及使用者。语用表示程序与使用的关系。

程序设计语言的基本成分有：

（1）数据成分，用于描述程序所涉及的数据。

（2）运算成分，用于描述程序中所包含的运算。

（3）控制成分，用于描述程序中所包含的控制。

（4）传输成分，用于表达程序中数据的传输。

程序设计语言按照语言级别可以分为低级语言和高级语言。低级语言有机器语言和汇编语言。

按照用户的要求有过程式语言和非过程式语言之分。过程式语言的主要特征是：用户可以指明一列可顺序执行的运算，以表示相应的计算过程，如 FORTRAN、COBOL、Pascal 等。

按照应用范围，有通用语言与专用语言之分。如 FORTRAN、COLBAL、Pascal、C 等都是通用语言。目标单一的语言称为专用语言，如 APT 等。

按照使用方式，有交互式语言和非交互式语言之分。具有反映人机交互作用的语言成分的语言称为交互式语言，如 BASIC 等。不反映人机交互作用的语言称为非交互式语言，如 FORTRAN、COBOL、ALGOL 60、Pascal、C 等都是非交互式语言。

按照成分性质，有顺序语言、并发语言和分布语言之分。只含顺序成分的语言称为顺序语言，如 FORTRAN、C 等。含有并发成分的语言称为并发语言，如 Pascal、Modula 和 Ada 等。

程序设计语言是软件的重要方面，其发展趋势是模块化、简明化、形式化、并行化和可视化。

如果按语种分，可以分为英文符号语言和汉语符号语言两类。易语言是一门计算机编程程序语言。以“易”著称，以中文作为程序代码表达的语言形式。易语言的创始人是吴涛。早期版本的名字为E语言。易语言最早版本的发布可追溯至2000年9月11日。可以说，创造易语言的初衷是进行用中文来编写程序的实践。从2000年至今，易语言已经发展到一定的规模，功能上、用户数量上都十分可观。

易语言功能强大实用，现已具有数十个各种应用范围支持库、上百个数据类型和界面组件、近万条支持命令，支持现今所有数据库，功能丝毫不比其他同类产品差。模块化开发支持大型软件项目的分工协作。易语言中的模块称为易模块，通过使用易模块，用户可以将常用的代码封装起来重复使用到其他程序，或提供给第三方使用，或用作开发大型软件项目中的某个部分，然后在软件项目的封装阶段将所有这些模块组织编译成为一个完整程序。

易语言系统全部自行设计开发。自有编译器。所编译目标程序运行速度快，且没有安全隐患；自带小型数据库，减少开发项目投入成本，且容易学习；支持跨操作系统平台编程，支持Windows和Linux程序开发；中文本地化支持，支持中文格式日期和时间处理、汉字发音处理、全半角字符处理、人民币金额处理、农历日期转换等。

易语言自带即时帮助系统，在易语言使用者有问题时，轻轻一点，就可以得到与当前主题相关的详细帮助。易语言的帮助文档众多，内有大量知识库及开发资料。易语言的例程众多，可以在资源网、大赛展区、论坛上搜索到。易语言爱好者交流论坛已有数万名注册用户。易语言使用者可以将自己在使用过程中所遇到的问题提出，专家会以最快的速度答复。通过论坛搜索功能，也可得到需要的答案。通过易语言的图书、教学片、多媒体教学光盘等，也可以得到相关的帮助信息。

当理解了一种语言的原理后，就会发现各种语言之间的相似之处。在解决某一问题时选择语言的原则是：能够用来写出有效而且简洁的代码的语言环境即可。应该让语言去适应项目。踏实、勤奋是程序员打开成功之门的第2把钥匙。

创造不是指你要发明别人不知道的技术或方法，而是说不能仅仅知道怎么做，还要知道为什么这样做，之后你才能创造。其实程序设计的整个过程就是创造的过程。求知欲和创造欲是原动力。有求知欲才能不停地学习，有创造欲才能不停地超越自己。

1.1.3　编程习惯

养成良好的习惯对人们自身是很有好处的，但是这并不是一件轻而易举的事，因为人并不是单纯地受理性支配，还要受自己思维和行为惯性的制约。所以，要养成良好的习惯，首先需要克服这些惯性作用。在编程时有人会追求自己独树一帜的风格，而降低程序的可读性，这是不可取的。编程时程序代码简单易懂是我们的目标，因为程序易读有利于团队合作。程序代码应符合该编程语言的风格。

1．缩进格式

缩进格式是为了清楚地定义一个块的开始和结束，使人们更容易理解程序代码。

程序的缩进大大增加了程序可读性。使用缩进格式使得程序嵌套层数太多时出现警告，此警告提醒编程者检查程序中所使用的嵌套层是否合理（一般嵌套层数不宜超过3层）。

2．括号配对

在C语言编程中，花括号十分重要，它是一段代码或一个结构体开始和结束的标志，花括

号不配对，计算机拒绝执行。如何才能确保初学者正确使用花括号呢？下列输入方法可以帮助避免花括号不配对的问题。

（1）当编写 main()函数时，先输入

```
void main()
{}
```

再在一对花括号中输入其他语句。

（2）当使用 if...else 语句时，则输入

```
if()
{}
else {}
```

再在第一对花括号中输入 if 条件表达式值为真时所执行的语句，在第二对花括号中输入 if 条件表达式值为假时所执行的语句。

（3）当使用 switch 语句时，则输入

```
switch()
{   case 0:
}
```

再在花括号中输入其他 case 语句。

（4）当使用 for 语句时，则输入

```
for()
{}
```

再在花括号中输入循环体语句。

（5）当使用 while 语句时，则输入

```
while()
{}
```

再在花括号中输入循环体语句。

（6）当定义函数时，则输入

```
void function()
{}
```

再在花括号中输入函数体语句。

（7）当定义结构体时，则输入

```
struct structname
{};
```

再在花括号中输入结构体成员。

3. 命名风格

（1）函数名加注释，说明该函数功能，以及函数返回值有哪些，分别是什么意思，函数的参数是什么类型，需要什么值。例如：

```
/*计算两个整数中的最大值*/
/*参数: 整数 1, 整数 2*/
/*返回值: 两个整数中的最大值*/
int max(int x,int y)
{   return x>y?x:y;          /*return 后面的值是一个表达式*/
}
```

（2）定义变量时注意见名知意。例如：

```
#include <stdio.h>
```

```
void main()
{   float r,area;              /*声明实型变量用来存放圆半径和圆面积*/
    r=3;                       /*为变量 r 赋初值*/
    area=3.1415926*r*r;        /*根据面积公式计算圆面积*/
    printf("%f\n ",area);
}
```

4．构建函数

用户构建的函数应该短小，一个函数只完成一项任务。如果需要写一个很复杂的函数，则应使用具有描述性的名字和注释帮助人们理解该函数。在函数中局部变量一般不宜超过 10 个，局部变量过多可能导致函数出错，当发现函数中局部变量过多时，应该将函数体分割成更小的函数。

5．注释

注释分为功能性注释和序言性注释。

（1）功能性注释：功能性注释出现在源程序中，用于描述其后的语句或程序段的功能。书写功能性注释要注意以下几点：第一，描述一段程序，而不是每一条语句；第二，利用缩进和空行，使程序与注释容易区别；第三，注释要准确无误。

（2）序言性注释：序言性注释通常位于每个程序模块的开头部分，它给出程序的整体说明，对于理解程序具有引导作用。有些软件开发部门对序言性注释做了明确而严格的规定，要求程序编制者逐项列出。其内容包括：程序标题；有关该模块功能和目的的说明；主要算法；接口说明，包括调用形式、参数描述、子程序清单；有关数据描述；模块位置（在哪一个源文件中，或隶属于哪一个软件包）；开发简历：模块设计者、复审者、复审日期。

为代码添加注释可以增加程序的可读性，但过多的注释可能会影响对代码本身的设计，因为编写注释花费了大量的时间，而代码的优化时间自然减少了。在 C 语言中注释的方法有两种，即“/*根据面积公式计算圆面积*/”和“//根据面积公式计算圆面积”。

6．备份代码

很多人都有过因磁盘故障而丢失大量数据的经历。数据备份是一个良好的习惯，每次修改完程序都需要做一次备份，为了安全一般需要做多个备份。

7．简化算法

在用计算机求解问题之前，必须先将所制定的解题方案“告诉”计算机，确定交付计算机执行的解题方案中的每个详细步骤，并且将此过程完整地描述出来，使计算机按照人们规定的计算顺序去自动执行，这种解题方案的描述称为算法。现代意义上的算法，通常是指可以用计算机来解决的某一类问题的程序或步骤。而一个算法应该具有以下几个重要特征：

（1）有穷性：一个算法必须保证执行有限步之后结束。

（2）确切性：算法的每一步骤必须有确切的定义。

（3）输入：一个算法有 0 个或多个输入，以刻画运算对象的初始情况。所谓 0 个输入，是指算法本身设定了初始条件。

（4）输出：一个算法有 1 个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是无意义的。

（5）可行性：算法原则上能够精确地运行，而且人们用笔和纸做有限次运算后即可完成。

在编程前先分析所选用的解题方案是否还有简化的空间，如果存在简化的空间则应先简化再编程实现。

8. 编写文档

编码的目的是产生程序，为了提高程序的可维护性，源代码是需要实现文档化的。源程序文档化包括选择标识符（变量和标号）的名字、安排注释以及标准的书写格式等。

用标准的书写格式书写源程序代码应注意以下几点：

（1）每行只写一条语句。

（2）用分层缩进的写法显示嵌套结构层次，这样可使程序的逻辑结构更加清晰，层次更加分明。

（3）书写表达式时适当使用空格或圆括号作为隔离符。

（4）可在注释段与程序段，以及不同的程序段之间插入空行。

9. 数据说明

在编写程序时，要注意数据说明的风格。规范数据说明有利于程序的测试、排错和维护。一般的说明顺序为：常量说明、简单变量类型说明、数组说明、数据块说明、所有的文件说明。类型说明顺序为：整型量说明、实型量说明、字符量说明。当用一个语句说明多个变量名时，应当对这些变量按字母顺序排列。对于复杂数据结构，应利用注释说明实现这个数据结构的特点。

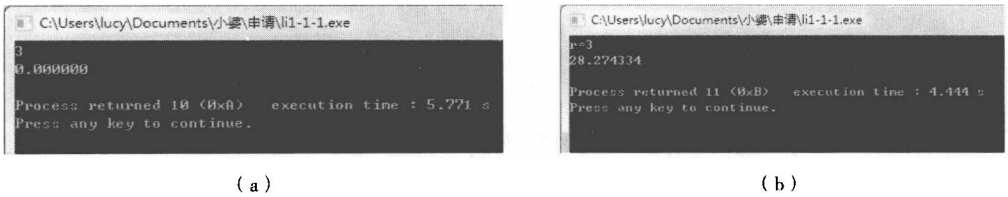
10. 输入/输出方法

输入/输出的方式和格式应当尽量避免因设计不当给用户带来的麻烦。源程序的输入/输出风格必须满足用户需求。在设计程序时，应考虑：输入数据时，要使输入的步骤和操作尽可能简单；应允许使用自由格式输入；应允许缺省值；对输入的数据要进行检验，以保证每个数据的有效性。输出信息正确，输出界面美观。

【例 1-1】计算圆面积，源程序代码 1.1.1 的输入复杂，在输入时因用户不了解输入格式，输入数据不正确导致计算结果不正确（见图 1-1）；而源程序代码 1.1.2 的输入/输出设计合理，计算结果正确（见图 1-2）。

【源程序代码 1.1.1】

```
#include <stdio.h>
void main()
{   float r,area;                    /*声明实型变量用来存放圆半径和圆面积*/
    scanf("r=%f",&r);                /*输入变量 r 的值*/
    area=3.1415926*r*r;              /*根据面积公式计算圆面积*/
    printf("%f\n ",area);
}
```



（a）　　（b）

图 1-1　源程序代码 1.1.1 的运行结果

【分析】在图 1-1（a）中，由于输入值不正确导致计算结果错，而在图 1-1（b）中数据输入正确才计算出正确的结果。同样的程序却有不一样的计算结果，其原因是输入格式设计中的问题。程序中 scanf("r=%f",&r);要求在输入时先输入字符“r=”后输入半径的值 3，再回车结束本次输入。

【源程序代码 1.1.2】

```
#include <stdio.h>
void main()
{    float r,area ;                    /*声明实型变量用来存放圆半径和圆面积*/
     printf("输入半径 r 的值");
     scanf("%f",&r);                   /*输入变量 r 的值*/
     area=3.1415926*r*r;               /*根据面积公式计算圆面积*/
     printf ("\n 半径%f 的圆面积为%f\n ",r,area);
}
```

【分析】运行源程序代码 1.1.2 时，用户看到提示信息“输入半径 r 的值”，自然输入一个数值，程序则以该数值为半径计算圆面积并输出。源程序代码 1.1.2 的输入/输出格式的设计合理、友好。

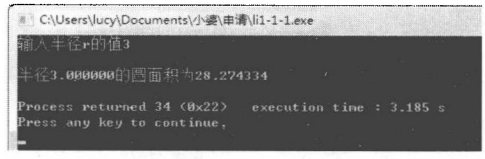


图 1-2　源程序代码 1.1.2 的运行结果

11．程序调试

在程序设计的过程中，不可避免地会发生错误。程序调试就是对程序进行测试，查找程序中隐藏的错误，并将这些错误修正或排除。一般应经过以下几个步骤：

（1）进行静态检查。一个程序编写结束后，不要匆忙用编译器编译，应对源程序进行人工检查。这一步是十分重要的，能够发现程序设计人员由于疏忽而造成的大多数错误。为了减少编程错误，在编写程序中应力求做到以下几点：

① 应当采用结构化程序方法编程，以增加可读性。

② 应尽可能多加注释，以帮助理解每段程序的作用。

③ 在编写复杂的程序时，不要将全部的语句都写在 main()函数中，而要多利用函数。用一个函数来实现单独的功能，既易于阅读也便于调试。各函数之间除了用参数传递数据这一渠道外，能够不用其他的渠道就尽量不用，数据间应尽量减少耦合的关系。

（2）上机动态检查调试。根据编译器提示的语法错误，找出编译器提示的全部错误（error）并一一改正，直到通过编译，生成下载文件或调试文件，还应该仔细检查编译器的警告（warning）信息，确认所有的警告信息并不会影响编译结果的正确性。有时，编译器的错误提示并非正确，而且出错的情况繁多且各种错误相互关联，因此要善于分析，找出真正的错误。可以借助编译器的工具对错误进行分析。

1.2　初识程序

1.2.1　编程环境

由以上论述可知，用高级语言编写的程序需要经过编译器编译才能转换成计算机能够直接执行的机器指令（二进制代码）。编译器不仅要完成编译工作，还要允许程序员反复地修改程序、调试程序，编译器必须具备程序编辑、程序调试的功能。调试器并不能解决程序中出现的问题，它仅仅是一种辅助编程的工具。首先应该分析程序，搞清楚到底怎么回事，一旦确定错误大致出在什么位置，便可以用调试器调试程序中特定变量的值。通过观察这些代码，可以了解到程序执行的全过程。其实编译器就是一个集成编程环境，编程环境的选择直接影响编程的效率。