高等学校计算机教育“十二五”规划教材

C语言程序设计基础教程

周艳芳　主编

YUYAN CHENGXU SHEJI JICHU JIAOCHENG

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

C语言程序设计基础教程

高等学校计算机教育“十二五”规划教材

C语言程序设计基础教程

周艳芳　主　编

任化敏　王润华　刘晓辉
钟　铮　李彩玲　副主编
赵姝菊　冯淑杰　参　编

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书内容的组织采取“案例驱动”+“课后实训”的方式，一方面通过大量的实例和实例间的反复对比，使学生掌握C语言的基础知识、基本概念、程序设计的思想和编程技巧；另一方面通过实验，使学生逐步提高阅读程序、调试程序、分析问题和解决问题的能力。

本书共分10章：第1章 C语言概述；第2章 C程序的基础知识；第3章 顺序结构程序设计；第4章 选择结构程序设计；第5章 循环结构程序设计；第6章 函数与预处理命令；第7章 数组与字符串；第8章 指针；第9章 结构体与其他数据类型；第10章 文件。另外，附录中介绍了C语言关键字等内容。每章中除了具体内容的讲解和例题的详细解析外，还包括“小结”、“实验”和“习题”。

本书适合作为高等学校计算机相关专业的教材，也可作为计算机等级考试（二级）以及C语言程序设计爱好者的参考用书。

图书在版编目（CIP）数据

C语言程序设计基础教程/周艳芳主编. —北京：中国铁道出版社，2012.9
高等学校计算机教育“十二五”规划教材
ISBN 978-7-113-14905-5

Ⅰ. ①C… Ⅱ. ①周… 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191423号

书　　名：C语言程序设计基础教程
作　　者：周艳芳　主编

策　　划：秦绪好　　**读者热线：**400-668-0820
责任编辑：赵　鑫
编辑助理：赵　迎
封面设计：刘　颖
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：北京鑫正大印刷有限公司
版　　次：2012年9月第1版　　2012年9月第1次印刷
开　　本：787mm×1092mm　1/16　**印张：**20.25　**字数：**490千
印　　数：1～3 000册
书　　号：ISBN 978-7-113-14905-5
定　　价：38.00元

高等学校计算机教育“十二五”规划教材

编审委员会

主　任：陈　明

副主任：叶曲伟　严晓舟

委　员：（按姓氏笔画排序）

王希更　王智广　王锁柱　刘贵龙　李　环
李海生　李素朵　张晓明　陈志泊　陈晓云
赵裕民　郝　莹　侯耀军　姚　琳　秦绪好
袁东光　袁　薇　徐孝凯　徐　慧　郭成芳
曹永存　解　凯　管建和

序言

PREFACE

随着计算机科学与技术的飞速发展，现代计算机系统的功能越来越强大、应用也越来越广泛，尤其是快速发展的计算机网络。它不仅是连接计算机的桥梁，而且已成为扩展计算能力、提供公共计算服务的平台，计算机科学对人类社会的发展做出了卓越的贡献。

计算机科学与技术的广泛应用是推动计算机学科发展的原动力。计算机科学是一门应用科学。因此，计算机学科的优秀创新人才不仅应具有坚实的理论基础，还应具有将理论与实践相结合来解决实际问题的能力。培养计算机学科的创新人才是社会的需要，是国民经济发展的需要。

计算机学科的发展呈现出学科内涵宽泛化、分支相对独立化、社会需求多样化、专业规模巨大化和计算教育大众化等特点。一方面，使得计算机企业成为朝阳企业，软件公司、网络公司等 IT 企业的数量和规模越来越大，另一方面，对计算机人才的需求规格也发生了巨大变化。在大学中，单一计算机精英型教育培养的人才已不能满足实际需要，社会需要大量的具有职业特征的计算机应用型人才。

计算机应用型教育的培养目标可以利用知识、能力和素质三个基本要素来描述。知识是基础、载体和表现形式，从根本上影响着能力和素质。学习知识的目的是为了获得能力和不断地提升能力。能力和素质的培养必须通过知识传授来实现，能力和素质也必须通过知识来表现。能力是核心，是人才特征的最突出的表现。计算机学科人才应具备计算思维能力、算法设计与分析能力、程序设计能力和系统能力（系统的认知、设计、开发和应用）。计算机应用型教育对人才培养的能力要求主要包括应用能力和通用能力。应用能力主要是指用所学知识解决专业实际问题的能力；通用能力表现为跨职业能力，并不是具体的专业能力和职业技能，而是对不同职业的适应能力。计算机应用型教育培养的人才所应具备的三种通用能力是学习能力、工作能力、创新能力。基本素质是指具有良好的公民道德和职业道德，具有合格的政治思想素养，遵守计算机法规和法律，具有人文、科学素养和良好的职业素质等。计算机应用型人才素质主要是指工作的基本素质，且要求在从业中必须具备责任意识，能够对自己职责范围内的工作认真负责地完成。

计算机应用型教育课程类型分为通用课程、专业基础课程、专业核心课程、专业选修课程、应用课程、实验课程、实践课程。课程是载体，是实现培养目标的重要手段。教育理念的实现必须借助于课程来完成。本系列规划教材的特点是重点突出、理论够用、注重应用，内容先进、实用。

本系列教材的不足之处，敬请各位专家、老师和广大同学指正。

陈明

2012 年 3 月

前言

FOREWORD

C语言是国内外广泛流行的程序设计语言，它功能强大，数据类型丰富，使用灵活，兼具面向硬件编程的低级语言特性及通用性强、可移植性好等语言特性。但是对于高等职业技术学校的学生来说，大多数从未接触过程序设计语言，根据笔者多年的教学经验，学生以C语言作为入门语言有一定的难度；另外，C语言是“全国计算机等级考试（二级）”的科目之一，但是多数学生无法获得有针对性的辅导。鉴于这样的情况，本书采取“案例驱动”+“课后实训”的方式，一方面通过大量的实例和实例间的反复对比，使学生掌握C语言的基础知识、基本概念、程序设计的思想和编程技巧；另一方面通过实验部分，使学生逐步提高阅读程序、调试程序、提高分析问题和解决问题的能力。

本书具有如下特色：

（1）通过实例来讲解语法知识，使难懂的语法易于理解掌握。

（2）程序解析详细，既点拨了重点难点，又引导了程序设计的思路。

（3）实验部分包括程序改错、程序填空和程序设计，通过这三类实验达到提高阅读程序、调试程序和设计程序的能力。

（4）习题部分紧扣每章的重点内容，精选了历年“全国计算机等级考试二级”的试题，使学生在消化语法的同时也能进行实战。

全书共分10章：第1章 C语言概述；第2章 C程序的基础知识；第3章 顺序结构程序设计；第4章 选择结构程序设计；第5章 循环结构程序设计；第6章 函数与预处理命令；第7章 数组与字符串；第8章 指针；第9章 结构体与其他数据类型；第10章 文件。另外，附录中介绍了C语言关键字等内容。每章除了具体内容的讲解和例题的详细解析外，还包括“小结”、“实验”和“习题”。“小结”归纳了本章的要点和重点；“实验”给出了有针对性的“程序填空”、“程序改错”和“程序设计”题目，另外还配有“实验评价表”，可从不同方面对所学知识进行实践与检验；“习题”部分包括“选择题”和“填空题”，这些题目全部针对本章的重点和难点，精选自历年的“全国计算机等级考试（二级）”试题，通过“习题”的解答，既巩固和应用了所学的基本知识，又进行了二级的实战，可谓“一举多得”。

本书由周艳芳主编，任化敏、王润华、刘晓辉、钟铮、李彩玲任副主编，赵姝菊、

冯淑杰参与编写工作。本书在编写过程中得到了陈明教授的帮助和支持，并给予了指导和把关；另外，还得到了中国铁道出版社编辑的指导和支持，在此一并表示诚挚的谢意。

由于时间仓促，编者水平有限，书中难免有疏漏和不足之处，敬请广大读者批评指正。

编　者

2012 年 3 月

目录

CONTENTS

第1章 C语言概述

C语言是由贝尔实验室的Brian.W.Fernighan和Dennis.M.Ritchie于1972年推出的。它是一种通用的程序设计语言，具有丰富的运算符和表达式以及先进的控制结构和数据结构。它具有表达能力强、编译目标文件质量高、语言简单灵活、容易移植、容易实现等特点，是学习和掌握更高层语言的开发工具，是C++/C#、Visual C++和Java语言程序设计的基础。

1.1 C语言的发展和主要特点

1.1.1 C语言的起源

美国的贝尔实验室（Bell Laboratory）成立至今，成果极其丰硕且造就了不少人才，C语言即是在这个实验室里由Dennis Ritchie于1972年开发出来的。C语言的前身为B语言，最早是用来编写DEC PDP-11计算机的系统程序。这个系统程序与人们熟悉的UNIX操作系统有着密不可分的关系。原本C语言只能在大型计算机里执行，现在已成功地移植到个人计算机里，而且有不同的版本出现，其中，人们比较熟悉的有Turbo C、Microsoft C、Quick C与Lattice C等。

1.1.2 C语言的特点

任何一种计算机语言的发展均有其目的。在C语言诞生之前，已经有很多程序设计语言产生，例如BASIC语言，其主要的目的是要让计算机的初学者可以很容易地编写程序，其语法近似英文，而且浅显易懂；此外，应科学计算与商业用途的需要，FORTRAN与COBOL语言也应运而生；其他高级语言如Pascal等也有其特定的用途，但这些语言常因发展背景与语言本身的限制而无法兼顾实用与性能。C语言的诞生恰恰可以弥补上述的缺憾。一般而言，C具有如下列几个特点：

1. 高性能的编译式语言

一般来说，当源程序代码编写完成后，必须转换成机器所能理解的语言，才能正确地执行。所有的程序语言中都附有这种转换程序，而转换程序可以大致分为两种，即解释器（Interpreter）与编译器（Compiler）。

所谓解释器，就是当人们要执行程序时，它会逐行地检查程序的语法，如果没有错误，再直接执行该行程序（如果碰到错误就会立刻中断执行），直到程序完毕，如图1-1所示。利用这种

方式完成的程序语言，最著名的就属 BASIC。由于解释器只需要将程序逐行翻译并访问源程序即可，所以所占用的内存较少，但是每一行程序在执行前才被翻译，将导致翻译时间会延迟执行时间，因此执行的速度会变慢，效率也较低。

然而编译器则会将整个程序都检查完成，先产生一个目标文件（OBJ 文件），将其他要连接进来的程序连接后，再执行该程序（见图 1-2）。源程序每修改一次，就必须重新编译，才能保持其可执行文件为最新的状况，同时，在执行的过程中也不需要因为等待程序的编译而中断。经过编译器所编译出来的程序，在执行时不需要再翻译，因此，执行效率与速度远高于解释程序。但是，由于编译器会产生诸如目标文件等的相关文件，也较占用内存空间。常见的编译式程序语言有 C、COBOL、Pascal 等。其中，C 的执行效率与使用的普遍性远远超过其他的程序语言。

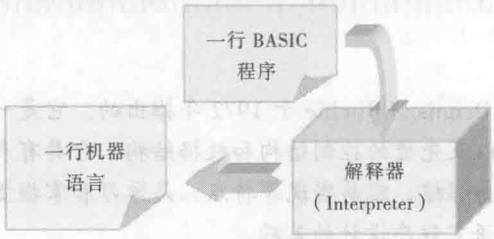


图 1-1 解释器会逐行检查程序的语法，再直接执行该行程序，直到程序完毕

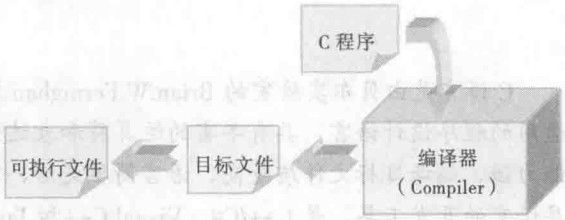


图 1-2 编译器先产生目标文件，再执行该程序

2. 介于高级语言与低级语言之间的一种语言

程序语言按其特点可大致分为两类：低级语言与高级语言。

低级语言在计算机里的执行效率相当高，而且对于硬件（如鼠标、键盘等）控制的程度相当好，但对用户而言，它却生涩难懂，不容易编写、阅读与维护。

高级语言为叙述性语言，它与人类所惯用的语法比较接近，所以容易编写、排错，但是相对的，它对硬件的控制能力却比较差，执行效率也远不及低级语言。常见的高级语言有 BASIC、FORTRAN、Pascal、COBOL 等。

C 语言不但具有低级语言的优点（对硬件的控制能力强），同时也兼顾了高级语言的特点（易于排错、编写），所以有人称之为“中级语言”。此外，C 语言还可以很容易地与汇编语言连接，利用低级语言的特点来提高程序代码的执行效率。

3. 灵活的控制流程与结构化的格式

C 语言是性能高、语法清晰的语言。它融合了计算机语言里流程控制的特点，使得程序员可以很容易地设计出具有结构化及模块化的程序语言，如图 1-3 所示。

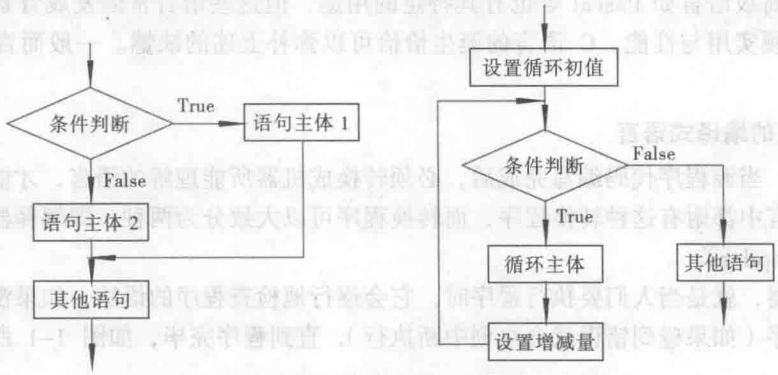


图 1-3 灵活的控制流程与结构化的格式

由于 C 语言的高性能与灵活性，许多操作系统（如 UNIX、MS-Windows 等）均由它编写。此外，许多高级语言的编译器（Compiler）或解释器（Interpreter）也是 C 语言的杰作。

4. 可移植性好

程序语言的可移植性（Portability）就像硬件的兼容性（Compatibility）一样。例如，一块声卡，如果在各家厂商的主板上都能顺利地安装，或者是只需要调整一下设置（如 Jumper、IRQ 等）即可安装，那么这块声卡的兼容性就好；但如果仅可在特定的主板上使用，那么这块声卡的兼容性就差。

同样，程序语言的可移植性好，意味着在某一操作系统编写的语言可以在少量修改或完全不修改的情况下即可在另一个操作系统里执行。C 语言可以说是一个可移植性极好的语言，当跨平台执行 C 语言时（如将 UNIX 里的 C 程序代码移植到 Windows 的环境里执行），通常只要修改极少部分的程序代码，再重新编译即可执行。此外，提供 C 编译器的系统近 50 种，从早期的 Apple II 到超级计算机 Cray，均可找到 C 编译器的踪迹。

5. 为程序员所设计的语言

C 语言可以说是专为程序员所设计的语言。它可以直接按照内存的地址来存取变量，以提高程序执行的效率。此外，C 语言也提供了丰富的运算符（Operator），使得 C 语言的语法更为简洁、有力。更方便的是，在大多数的 C 语言环境里都提供了已编写好的函数库（Library），包含了许多 C 语言函数，以供程序员使用而不需要重新编写程序代码。

任何事物都不是十全十美的，C 语言也有一定的缺陷，C 语言的语法严谨、简洁，相对的用户就必须花更多的心思在学习 C 语言的语法上，尤其是指针（Pointer）的应用，常常让初学者摸不着边际。但一旦熟悉了 C 语言的语法，它将十分便利、快速。

1.2　第一个 C 程序的规划、设计和运行

1.2.1　程序的规划与操作

一般来说，程序的设计分为自顶向下法（Top-down Approach）与自底向上法（Bottom-up Approach）。在程序设计的过程中，如果将问题分解成多个模块（Modules），再将这些模块分别分解成更小的模块，依此类推，直到分解成最容易编写的最小模块为止，这种程序设计的方式称为自顶向下法（Top-down Approach）。利用“自顶向下”方式编写的程序，其结构有层次，容易看懂和维护，同时可以降低开发的成本，但是在程序分解成模块的过程中可能因此占用较多的内存空间，造成执行时间过长。

如果在程序设计时，先将整个问题里最简单的部分编写出来，再一一结合各个部分以完成整个程序，这种设计的方式称为自底向上法（Bottom-up Approach）。利用“自底向上”方式编写的程序不太容易看懂和维护，造成程序设计者的负担，反而容易增加开发的成本。

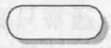
因此，在编写程序前的规划就显得相当重要。如果程序的内容很简单，可以容易地将程序写出来；但是当程序很大或很复杂时，规划的工作就很重要，它可以让程序设计有明确的方向，尤其是程序的逻辑不清楚时，有了事前的规划流程，就可以根据这个流程来一步一步设计出理想的程序。

除了容易实现外，还可以养成规划程序的习惯，会使程序简洁许多。这也意味着程序执行的速度将会更快、更有效率。程序规划很重要，下面来看程序设计的六大步骤。

1. 规划程序

首先，必须明确编写程序的目的、程序的用户对象及需求度，如计算员工每个月的工资、绘制图表、数据排序等，再根据这些数据及程序语言的特性选择一个合适的程序语言，达到设计程序的目的。设计者可以在纸上先绘制出简单的流程图，将程序的起始到结束的过程写出，这样，一方面便于理清程序的思路；另一方面可以根据这个流程图进行编写程序的工作。表 1-1 是绘制流程图时常会用到的流程图符号介绍。

表 1-1 常用的流程图符号介绍

符 号	介 绍	符 号	介 绍	符 号	介 绍
	开始/结束符号		判断		输入/输出
	程序前进的方向		预定函数的执行		连接点
	设置/过程		文件		

下面以一个日常生活中的例子“出门时如果下雨就带伞，否则戴太阳眼镜”，简单地说明如何绘制程序流程图。

在流程图 1-4 里，在选择方块中输入“下雨”，如果“下雨”这件事为真，即执行“带伞”的动作，否则执行“戴太阳眼镜”的动作。因此，在程序方块里分别输入“带伞”及“戴太阳眼镜”，不管执行哪一个动作，都必须“出门”，最后再根据程序的流向，用箭头表示清楚。

其实不管是程序设计还是日常生活的过程，都可以用流程图来表示，因此，学习绘制流程图是十分有意思的事。

2. 编写程序代码及注释

程序经过先前的规划之后，便可以根据所绘制的流程图来编写程序内容。这种方式会比边写边想下一步该怎么做要快得多。如果事先没有规划程序，在边写边想时，往往会写了改，改了又写。此外，笔者认为编写程序时应把注释加上，这样在长时间放置或者是设计者之外的人维护程序时，可以增加这个程序的可读性，相对也会增加程序维护的容易程度。

```
01  #include<stdio.h>
02  #include<stdlib.h>
03  int main(void)
04  {
05      int num=2;    /*定义整型变量 num，并赋值为 2*/
06      printf("I have %d cats.\n",num);  /*调用 printf()*/
07      return 0;
08  }
```

加上注释可增加程序的可读性

3. 编译程序代码

程序编写完毕，必须要将程序代码转换成计算机能够识别的语言。这样就需要转换程序，其实就是所谓的编译器（或编译程序）。通过编译程序的转换，只有在没有错误的时候，源程序才会变成可以执行的程序。若是编译器在转换的过程中碰到不认识的语法、未定义的变量等，则必须先把这些错误纠正过来，再重新编译完成，没有错误后，才可以执行所设计的程序。