PUTONG GAODENG
YUANXIAO
YINGYONGXING
BENKE GUIHUA JIAOCAI

普通高等院校应用型本科规划教材

C语言程序设计基础（第三版）

主编 刘莹 王宁 杨雪梅

C YUYAN
CHENGXU SHEJI JICHU

重庆大学出版社

PUTONG GAODENG YUANXIAO YINGYONGXING BENKE GUIHUA JIAOCAI

普通高等院校应用型本科规划教材

C语言程序设计基础（第三版）

主编 刘莹 王宁 杨霞梅

C YUYAN
CHENGXU SHEJI JICHU

重庆大学出版社

内容提要

本书针对程序设计语言的初学者，以C语言为载体，以微软Visual C++6.0为环境，通过讨论C程序设计的一般过程和方法，重点介绍了结构化程序设计的基本思想和实现方法。本书坚持“理论够用，程序设计实际能力为重”的宗旨，通过数据组织、控制结构、文件处理等程序设计基础知识的讨论，向读者介绍使用C语言进行程序设计的基本方法。通过对指针与函数关系、指针与数组关系、指针数组、动态数组实现方法、构造数据类型使用方法等方面的讨论，向读者介绍C语言特有的一些重要知识，使读者能够循序渐进地掌握使用C语言开发各类常见应用程序的基本技能。

本书在附录中还提供了ASCII码表、C程序设计中常用的标准库函数、使用Visual C++6.0集成环境开发C程序的基本方法等重要学习资料。

本书覆盖了C语言的应用基础，内容深入浅出，语言流畅，例题丰富，适合作为第一门程序设计语言课程教材，对于程序设计爱好者也是较好的入门教材或参考书。

图书在版编目(CIP)数据

C语言程序设计基础/刘莹，王宁，杨雪梅主编.--3版.--重庆：重庆大学出版社，2017.8

ISBN 978-7-5689-0703-3

Ⅰ.①C… Ⅱ.①刘… ②王… ③杨… Ⅲ.①C语言—程序设计—高等学校—教材 Ⅳ.①TP312.8

中国版本图书馆CIP数据核字(2017)第176933号

C语言程序设计基础

(第三版)

主　编　刘　莹　王　宁　杨雪梅

责任编辑：章　可　　版式设计：章　可

责任校对：秦巴达　　责任印制：张　策

*

重庆大学出版社出版发行

出版人：易树平

社址：重庆市沙坪坝区大学城西路21号

邮编：401331

电话：(023) 88617190　88617185(中小学)

传真：(023) 88617186　88617166

网址：http://www.cqup.com.cn

邮箱：fxk@cqup.com.cn (营销中心)

全国新华书店经销

重庆市国丰印务有限责任公司印刷

*

开本：787mm×1092mm　1/16　印张：20.75　字数：483千

2015年8月第1版　2017年8月第3版　2017年8月第5次印刷

ISBN 978-7-5689-0703-3　定价：45.00元

前言

计算机程序设计能力是理工科大学生应具有的基本素养之一。培养学生的逻辑思维能力、抽象能力和基本的计算机程序设计能力，引导学生进入计算机程序设计的广阔空间是大学计算机程序设计课程的主要目标。

计算机程序设计的主要任务就是用合适的计算机程序设计语言对解决问题的方法进行编码处理，即编制程序。C 语言功能丰富、表达能力强、程序执行效率高、可移植性好；C 语言既有高级计算机程序设计语言的特点，同时又具有部分汇编语言的特点，因而 C 语言具有较强的系统处理能力；C 语言是一种结构化程序设计语言，支持自顶向下、逐步求精的程序设计技术，通过 C 语言函数结构可以方便地实现程序的模块化；在 C 语言的基础之上发展起来的面向对象程序设计语言如 C++、Java、C#等与 C 语言有许多的共同特征，掌握 C 语言对学习进而应用这些面向对象的程序设计语言有极大的帮助。

本书编撰的宗旨是“理论够用，程序设计实际能力为重”，从结构化程序设计技术出发，以 C 程序设计语言为载体，介绍了 C 语言的基本语法、语义并对学习 C 语言过程中可能遇到的各种常见典型问题进行了分析，通过对程序设计技术基础范畴内各种典型问题的求解方法的描述以及相应 C 语言代码的描述，展现了在程序设计过程中如何对问题进行分析，如何组织数据和如何描述解决问题的方法，展现了在计算机应用过程中如何将方法和编码相联系的具体程序设计过程，进而向读者介绍了计算机结构化程序设计的基本概念、基本技术和方法。

本书的主要内容分为相辅相成的两个部分，第一部分包括第 1 章—第 7 章，主要介绍计算机程序设计高级语言共性的基础知识，包含的主要内容有：基本数据类型的使用，运算符和表达式计算基础，标准库函数的使用方法和顺序程序设计，结构化程序设计，函数调用中的数值参数和地址值参数传递，变量的作用域和生存期，数组使用和字符串处理基础，函数调用中的数组参数传递，文件数据处理基础。对于需要了解和掌握结构化程序设计基本思想的读者，通过这部分的学习可较为全面地掌握结构化程序设计的基本思想。第二部分包括第 8 章—第 11 章，主要介绍 C 语言特有的一些重要知识，包含的主要内容有：返回指针值的函数，指向函数的指针以及指向函数指针变量作函数的形式参数，数组与指针的关系，指针数组，命令行参数，用指针实现动态数组的方法，结构体数据类型，联合体数据类型，用 typedef 关键字描述复杂数据类型，编译预处理基础，位运算与枚举类型。通过第二部分内容的学习，读者可以较为全面地掌握 C 语言的基础知识。

本书选用 Microsoft Visual C++ 6.0 作为教学环境，书中的所有教学示例都在 Microsoft Visual C++ 6.0 集成开发环境中通过。附录中还提供了 ASCII 码表、C 程序设计中常用的标准库函数、使用 Visual C++ 6.0 IDE 开发 C 程序的基本方法等重要学习资料。

本书适于高等院校本科各专业作为程序设计语言类课程教材，同时可作为计算机专业本科学生、计算机应用开发人员、程序设计爱好者、计算机等级考试应试者在学习程序设计语言和程序设计技术时的参考资料。

参加本书编撰的作者都是长期从事程序设计课程教学的一线教师，在教材内容选取、章节教学顺序安排上都尽可能考虑了 C 语言基础内容与初学程序设计难度上的平衡，通过丰富的例题、流畅的语言对教学内容进行了深入浅出的描述。参加本书编撰的教师有：刘莹、王宁、杨雪梅。各章节编写分工如下：刘莹（第 1—4 章），王宁（第 5—7 章），杨雪梅（第 8—11 章）。全书由刘莹进行内容调整、修改，统一定稿。郑先锋副教授、熊壮副教授负责了全书的审阅。感谢职宏雷高工对本书提出的宝贵意见和在编写过程中提供的帮助。

限于编者水平，书中存在错误和不妥之处在所难免，恳请读者不吝指教。如需要联系作者，可发邮件至 ly640828@163.com 或 1062622134@qq.com。

编　者

2017 年 6 月

授课内容和知识点分配建议

章节内容	基本内容		文科	理工	计算机
1.为什么要学 C 语言	C 语言的发展历程				
	学习 C 语言的原因				
	编程的基础知识,软件开发的基本过程				
2.C 程序设计初步	C 程序的结构和处理过程				
	基本数据类型				
	数据的输入与输出				
	基本的运算符和表达式				
	数据类型转换				
	C 语言标准库	C 标准库函数使用方法			
		常用的标准数学函数	⊘		
3.分支结构程序设计	生活中的问题求解方法				
	计算机问题求解的特点				
	算法的概念及其描述方法				
	关系运算和逻辑运算				
	单分支程序设计				
	复合语句在程序中的使用				
	双分支程序设计				
	多分支程序设计				
4.循环结构程序设计	while 循环控制结构				
	do… while 循环控制结构				
	for 循环控制结构				
	空语句及其在程序中的使用		⊘	⊘	
	循环的嵌套结构				
	流程的转移控制	goto	⊘	⊘	
		break			
		continue			
	基本控制结构的简单应用	穷举方法			
		迭代方法			
		一元高阶方程的迭代	⊘	⊘	

续表

章节内容	基本内容		文科	理工	计算机
5.函数	分而治之与信息隐藏				
	函数的定义和调用				
	函数调用中的指针参数传递				
	函数的嵌套调用和递归调用		⊘		
	变量的作用域				
	变量的存储类型		⊘		
6.数组和字符串	数组的定义及数组元素的引用	一维数组定义和引用			
		二维数组和多维数组	⊘		
	字符数组和字符串				
	函数调用中的数组参数传递	一维数组作函数参数			
		二维数组作函数参数	⊘		
	数组的简单应用	数组元素随机生成	⊘		
		基于数组的排序算法	⊘		
		基于数组的查找方法	⊘		
7.C程序文件处理基础	文件处理基础		⊘		
	文件处理中数据的读/写方法	单个字符数据读写	⊘		
		字符串数据读写	⊘	⊘	
		格式化数据读写	⊘		
		数据块的读写	⊘	⊘	
	随机存取文件处理基础		⊘	⊘	
8.指针	指针与函数	返回指针值的函数	⊘		
		指向函数的指针变量	⊘	⊘	
	指针与一维数组		⊘		
	指针与二维数组	多级指针定义和引用	⊘	⊘	
		指向二维数组元素的指针变量	⊘		
		指向二维数组的指针变量	⊘		
	指针数组与命令行参数	指针数组的定义和引用	⊘		
		命令行参数	⊘	⊘	
	使用指针构建动态数组		⊘	⊘	
	指针与字符串		⊘		
9.编译预处理基础	宏定义预处理命令及其简单应用				
	文件包含预处理命令及其简单应用				
	条件编译预处理命令及其简单应用		⊘	⊘	
10.结构体和联合体	结构体类型的定义和使用		⊘		
	结构体数组		⊘	⊘	
	结构体数据类型与指针的关系		⊘	⊘	
	联合体数据类型		⊘		
11.枚举类型和位运算	枚举类型及其简单应用		⊘	⊘	
	位运算及其应用		⊘		

目 录

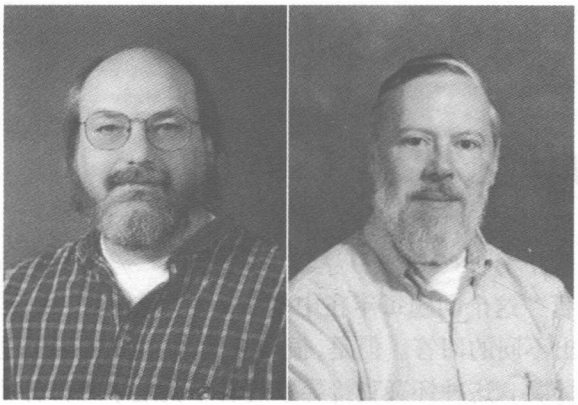

图 1.1　ken 和 dmr

和记忆，而人的自然语言机器又很难理解，于是人们开始琢磨怎么让计算机读懂自然语言。基本思路是设计一个翻译程序，直接把自然语言翻译成机器语言，这种翻译程序被命名为“编译器”。但是直接理解自然语言太难了，折中的办法是设计一种尽量接近自然语言，且还能被精确翻译为机器语言的语言，这种语言就是人们常说的编程语言。学编程的过程，其实就是学习怎样用编程语言说话并让编译器听懂的过程。第一种编程语言肯定是最接近机器而远离人类的，它就是汇编语言。虽然看上去有点像自然语言，如加法叫“ADD”，减法叫“SUB”，但它的语法完全是为机器服务的，每一行语句都和一条机器指令严格对应。不同计算机的机器指令是不一样的，所以针对一种计算机编写的汇编程序不能在另一种计算机上直接使用，必须重写（Space Travel 就被重写过很多次）。用专业术语来说，汇编语言缺少可移植性。

因为 Space Travel 的吸引力，使得很多人都希望他们的计算机上也能安装 UNIX。于是 ken 和 dmr 决定改用高级语言编写 UNIX，这样它就可以在更多类型的计算机上运行。

除了机器语言和汇编语言以外，几乎所有编程语言都被统称为高级语言。它的特点是更接近自然语言，而与机器语言联系不紧密。不同的高级语言编译器可以把同样的高级语言程序翻译成适应不同机器的指令，因而高级语言大多具有较好的可移植性。

在高级语言的选择上，ken 和 dmr 遇到了麻烦，虽然可供选择的高级语言有很多，如现在还在被使用的有 Basic 和 Fortran 等，但这些语言都是面向应用程序编写而设计的，层次太高，使机器能理解太困难，都假想其是在一个操作系统上运行，所以不适合用来开发操作系统。所以他们最后决定自己设计一种适合编写 UNIX 的高级语言。那一年是 1972 年，ken 继续完善 UNIX，dmr 设计新语言，两人一起开发编译器。因为该语言以 ken 早年设计的 B 语言为基础，所以就自然而然地被命名为 C 语言。

1983 年，因为 UNIX 和 C 语言的巨大成功，ken 和 dmr 共同获得了计算机界的最高奖——图灵奖。

UNIX 和 C 其实是可以为 ken 和 dmr 带来大量财富的。然而，他们从一开始就没有想去申请专利、商标、软件著作权等法律保护，而是把所有的一切，包括最宝贵的源代码，都全部公开发布。对他们来说，自己写的程序有人使用，就是最大的快乐，也是最大的财富。这恰好使得很多机构和个人都具有了自如地为 UNIX 和 C 添加代码、做各种贡献的条件，因

而又极大地促进了它们的发展。

1.2 C语言的流行趋势

最受欢迎的语言一定是被用得最多的。C语言现在用得多吗?

图1.2所示的是Tiobe在2017年4月公布的程序设计语言受欢迎程度的趋势图,横坐标代表统计时间(年份),纵坐标代表使用率(%)。从图中可以看到,C语言始终处于前两位。

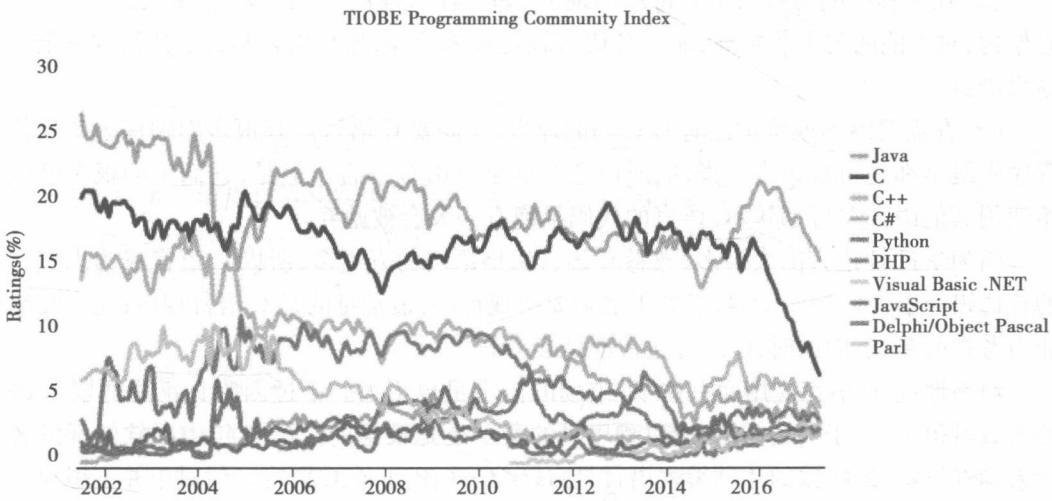

图1.2 2017年4月统计的10大流行编程语言的流行趋势图

dmr曾经说过一句话:"C诡异离奇,缺陷重重,并获得巨大成功。"因为诡异且有缺陷,所以会被尽量避免使用,取而代之的是弥补了这些缺陷的语言。因为C语言确实获得了巨大成功,所以它至今仍大受欢迎。但为什么却被尽量避免使用呢?那是因为C语言有自身的缺陷。

1.3 C语言的优与劣

C语言设计原则的第一条是:"信任程序员"。对程序设计语言了解不多的人,对这句话可能没有特别的感受。但对真正的程序员来说,凭这句话就足以一直钟爱C语言。

在C语言盛行的年代,计算机的价格相当昂贵,并且运算速度较慢,内存都是以千字节(KB)为单位。所以,那时候对程序最基本的要求就是运行效率。C语言完全满足人们对效率的苛求,精心设计的代码可以极大地节约计算机资源,又不像汇编语言那样难用,所以受到程序员的欢迎。后来,硬件价格越来越便宜,性能也越来越高,程序的运行效率已经不是追求的主要目标,安全性、稳定性和易于维护等变得重要起来,C语言的弊端便显现了。

C语言给程序员最大的发挥空间,让他们可以自由地编写代码,不去质疑这些代码是否合理,因为它"信任程序员",相信程序员的决定一定是正确的,即便有错误,也一定能自己修正。无限制的自由也成为了C语言最大的弱点,随着软件规模的膨胀,需要的程序越

来越多,C 语言的自由使得程序的错误率大幅度提高。

1.4 为什么学习 C 语言

现在,还需要使用 C 语言的地方大概只限于下面 3 个领域:

(1)C 语言仍然是编写操作系统的不二之选。它为操作系统而生,能更直接地与计算机底层打交道,精巧、灵活、高效。最重要的是操作系统的开发者都是最顶尖的程序员,他们有充足的能力和经验驾驭 C 语言。

(2)在对程序的运行效率有苛求的地方,如现在火爆的“云计算”领域,云平台作为基础架构,对性能的要求非常高,那么 C 语言就是首选了,因为 C 语言是目前执行效率最高的高级语言。

(3)在需要继承或维护已有 C 代码的地方,还需要 C 语言。有很多影响深远的软件和程序库最早都是用 C 语言开发的,所以还要继续应用 C 语言。但是,它们中的很多已经开始使用其他语言重写,那些 C 语言的代码早晚有一天会被抛弃。

因为学过 C 语言的人最多,熟悉 C 语言风格语法的人更多,所以 C 语言成为思想交流的首选媒介语言。例如,书籍里如果必须要出现程序,最常见的是 C 语言程序;在涉及编程能力考查的笔试、面试时,C 语言通常都是必考的。

坦率地说,C 语言应用面有些窄,市场的需求量也不大。不过因为真正能驾驭 C 语言的人数量很少,小于市场需求,所以程序员的薪水还是较高的。2017 年中国软件开发者薪资大调查显示,最赚钱的 4 种编程语言是:Java、C++、Python、C 语言。但对于并不想成为开发高手,或者兴趣不在底层开发的人来说,学它能有多大用处呢? 如果单纯从“用不上”这个角度得出“学 C 语言没有用”的结论,是有失公允的。即便对计算机及相关专业而言,C 语言的“用处”也不算大。学习 C 语言的意义并不在于使用它,而在于它可以让人们了解很多基本原理。

这里不妨根据未来的职业需求,把读者分为三类:

(1)不需要编程;

(2)需要编程,但不使用 C 语言;

(3)需要编程,且要使用 C 语言。

对于第三类读者,不仅要好好学 C 语言,而且要精深地学。

对于第二类读者,学习 C 语言最大的好处是可以更直接地体会计算机最基本的工作模式和方式。换而言之,就是能了解一些计算机底层的原理。这是在其他高级语言中很难体会到的。这些原理虽然也不常直接用到,但它们潜移默化的影响是惊人的,总是能在关键时刻发挥作用。另一个好处是 C 语言很适合作为入门级语言。这并不是 C 语言自身决定的,而是中国庞大的 C 语言教育体系决定的。关于 C 语言的书籍、资料、论坛、习题和教辅资料等是最多的,而且无一例外都是面向程序设计的初学者。相比之下,其他语言的很多教材是假定读者已经有了一定的编程经验,不介绍或只简单介绍那些通用的基本概念、理论与思维,直接跳到语言自身的特性。当然,像 Java、C++等语言也有很好的面向初学者的教材,直接学习它们可以了解更新的编程思想,距离实际应用更近,成效更显著。好在大多

数主流编程语言都是与C语言一脉相承的，使得从C语言入门后，再学其他语言，并不会感到困难。

C语言给予第一类读者的最大好处是为读者打开一扇了解计算机的窗口。在几乎做任何事情都离不开计算机的今天，越了解计算机也就意味着越能利用好计算机。

美国卡内基·梅隆大学计算机科学系前系主任周以真教授在2006年发表了一篇著名的文章——《计算思维(Computational Thinking)》。文中谈到“计算机科学的教授应当为大学新生开一门称为‘怎么像计算机科学家一样思维’的课，面向非专业的，而不仅仅是计算机科学专业的学生”，这是因为“机器学习已经改变了统计学。……计算生物学正在改变着生物学家的思考方式。类似地，计算博弈理论正改变着经济学家的思考方式，纳米计算改变着化学家的思考方式，量子计算改变着物理学家的思考方式”，所以“计算思维代表着一种普遍的认识和一类普适的技能，每一个人，不仅仅是计算机科学家，都应热心于它的学习和运用”。不过遗憾的是，我们现在还很少有学校开设这样的课程。所以程序设计课在某种程度上肩负了传播计算思维的责任。这也是对于不需要编程的学生而言，最大意义之所在。通过学习编程，了解什么是抽象、递归、复用、折中等计算思维，能在各行各业中更有效地利用计算机工具解决复杂问题。

1.5 什么是“编程”

已经确定要学习C语言，现在开始正式进入程序的世界。先了解一下什么是编程。

“编程”是“编写程序”的简称，术语称为“程序设计”。程序是计算机的主宰，控制着计算机该去做什么事。所有托付给计算机去做的事情都要被编成程序。假如没有程序，那么计算机什么事情都干不了。如果程序是“好”程序，那么计算机在它的指挥下可以又快又好地完成工作；如果程序有错误，那么计算机也会严格按照错误的指令去工作，造成的后果都不是我们所期望的。

如果想让计算机做一件事情，但是没有现成的程序可用，就需要编程。编程的第一步是“需求分析”，就是要弄清楚我们到底要让计算机做什么。这个过程貌似无甚复杂，也确实让不少人对它不屑一顾。但忽视它的结果就像考试时审题审得不对，后面的解题再漂亮，也拿不到分数，必须从头返工。所以有经验的程序员都会对需求分析相当谨慎。需求分析中最难的事情是开发者和用户之间的交流。用户不懂开发，开发者不懂用户的专业和业务，使双方都会有对牛弹琴的感觉，导致需求分析的过程可能要持续好几个月，甚至数年。如果开发者之前对专业就有所了解，或者用户懂一点开发，这件事就好办得多了。这也是非计算机专业的学生学习程序设计的好处。

编程的第二步是“设计”，就是搞明白计算机该怎么做这件事。设计的内容主要包括两方面：一方面是设计算法、数学建模，用数学方法对问题进行求解；另一方面是设计程序的代码结构，使程序更易于修正、扩充、维护等。数学部分往往属于非计算机专业范畴，程序设计部分则属于计算机专业范畴，两者的配合非常重要。并不是所有的数学模型都能用程序高效实现，而有些数学中难以处理的问题，却可以利用计算机的特点巧妙解决。计算思维就体现在这里。

编程的第三步才是真正“编写程序”，即把设计的结果变成一行行代码，输入程序编辑器中。虽然 Windows 内置的记事本也可用来编写程序，但一个顺手的编辑器可让编码的过程充满惬意。有经验的程序员喜欢使用 VIM 或 Emacs，如果有钻研精神，可以试试。新手一般会选择更容易入门的集成开发环境（IDE），如 Code::Blocks、Microsoft visual Studio 等。

编程的第四步是“调试程序”，就是将源代码编译，变成可执行的程序，然后运行之，看看能否满足第一步的要求。如果不满足，就要查找问题，修改代码，再重新编译、运行，直到满意为止。用到的主要工具是编译器和调试器，它们一般都已经内置在 IDE 中。如果不使用 IDE，只使用编辑器，则需要单独安装。推荐使用 gcc 编译器和 gdb 调试器。两者占据 UNIX/Linux 平台的主流，在 Windows 平台上亦可使用。

这个过程说起来简单，但每一个环节都有很多学问在里面。本书主要讲述的是第三步和第四步。前两步虽然也有涉及，但因为程序的规模较小，所以体现得并不多。但读者必须知道，待将来编写大规模的程序时，前两步的重要性是超过后两步的。

习题 1

简答题

1.查找资料，总结图 1.2 中列出的 10 种编程语言各自的特点和主要应用领域。（计算机专业）

2.查找资料，解释什么是图灵测试？（计算机专业）

3.程序和软件有何不同？（非计算机专业）

4.人与计算机之间用什么语言交流？如何实现更有效的人机交流？（非计算机专业）

5.程序开发的基本步骤是什么？（非计算机专业）