华章程序员书库

Apress®

HZ BOOKS 华章IT

C编程技巧

117个问题解决方案示例

C Recipes

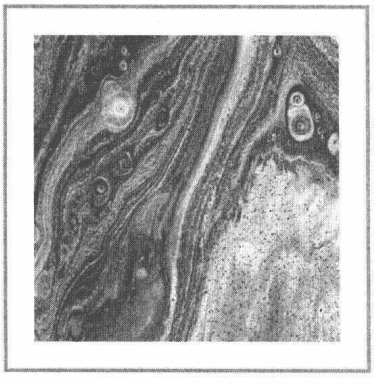
A Problem-Solution Approach

[印] 希里什·查万（Shirish Chavan）著

卢涛 译

机械工业出版社
China Machine Press

华章程序员书库



C Recipes
A Problem-Solution Approach

C编程技巧
117个问题解决方案示例

[印] 希里什·查万（Shirish Chavan）著

卢涛 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

C 编程技巧：117 个问题解决方案示例 /（印）希里什·查万（Shirish Chavan）著；卢涛译．
—北京：机械工业出版社，2019.4
（华章程序员书库）
书名原文：C Recipes: A Problem-Solution Approach

ISBN 978-7-111-62249-9

I. C… II. ① 希… ② 卢… III. C 语言 - 程序设计 IV. TP312.8

中国版本图书馆 CIP 数据核字（2019）第 049237 号

本书版权登记号：图字 01-2018-8098

C 编程技巧：117 个问题解决方案示例

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：卢 璐 责任校对：殷 虹
印 刷：北京诚信伟业印刷有限公司 版 次：2019 年 4 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：22.5
书 号：ISBN 978-7-111-62249-9 定 价：99.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 读者信箱：hzit@hzbook.com

Preface 前　言

本书包含了适合从初级到高级的各种读者的大量C语言技巧。本书按照“问题－解决方案”的体例编写，以便你可以快速找到所需问题的解决方案。本书每个解决方案都附带适当的代码和对该代码的简要讨论，力求在C的理论和实践之间取得完美的平衡。

C语言于1972年首次亮相。对于高级计算机语言而言，它现在处于退休年龄。但是，尽管C语言已有40多年的历史，它仍然很强大。C是十种最受欢迎的计算机语言之一，至少在接下来的20年内仍将如此。因此，你在C中获得的任何专业知识都不会很快过时，并且会使你在未来几年内继续保持高效。本书将帮助你解决C语言中的问题，并使你成为C语言的专家。

本书适用对象

本书主要面向在职专业人士。但是，它也适用于学生、教师、研究人员、代码测试人员和程序员。期望你具备C语言和编程的实际知识。

本书组织结构

本书由11章组成。第1章总览C语言。第2章涉及控制语句。第3～5章涉及函数、数组、指针和结构。在这些章节中，你将找到程序员在实际工作中面临的问题。

第6章处理数据文件，包含大量涉及保存文件到磁盘和从所保存的文件中获取数据的技巧。第7～9章涉及数据结构的广泛主题，这些章节涵盖了具有实用性的数据结构。第10章介绍了各种密码系统。C和密码学的组合是一个非常强大且有趣的组合。在本章中，你将体验到这种组合的强大功能。

第11章是本书的最后一章，讨论数值方法。计算机是作为数值计算机器被发明的，但随着时间的推移，它们已经成为了数据处理机器。然而，即使在今天，数值计算仍是计算机执行的最重要的工作之一。本章为你提供了许多用于数值计算实用程序的技巧。

我真诚地希望本书对广大读者有用。

致　谢 *Acknowledgments*

感谢 Apress 促成本书问世的每位工作人员。特别感谢编辑 Celestin Suresh John 先生和 Prachi Mehta 女士的耐心和指导。

很多技术朋友在本书的技术问题上帮助了我。其中包括：位于 Nagpur 的 Cryptex Technologies 公司（www.cryptextechnologies.com）的首席执行官 Ajay Dhande 先生；位于 Satara 的 Harsh 计算机研究所的 Shivajirao Salunkhe 教授和 Manisha Salunkhe 教授；位于 Satara 的 Arvind Gavali 工程学院（www.agce.sets.edu.in）的校长 Vilas Pharande 博士；位于 Satara Wai 的 Kalasagar Academy 公司（www.kalasagaracademy.in）的 Sachin Pratapure 教授和 Vishal Khade 教授；位于 Satara 的 Yashoda 工程学院的 Anant Bodas 教授和 Vikas Dhane 教授；位于 Amravati 的 Shrikant 计算机培训中心（https://www.sctcamravati.com）的 Sanjay Adhau 教授；位于 Amravati 的 PRMIT&R（mitra.ac.in）的校长 Mir Sadique Ali 博士；位于 Pune 的 Aphron Infotech 公司（www.aphroninfotech.in）的首席执行官 Nikhil Kumbhar 先生。我还要感谢运营 Coding Alpha 网站（www.codingalpha.com）的 Tushar Soni 先生和 Ajay Sawant 先生，以及运营 The Crazy Programmer 网站（www.thecrazyprogrammer.com）的 Neeraj Mishra 先生。他们都为本书的创作提供了宝贵的帮助。

Vijay Bhatkar 博士是著名的计算机科学家，也是印度超级计算机 PARAM 10000 之父，他一直是我的灵感源泉。很感谢他鼓舞我。

最后但同样重要的是，感谢 Jarron 先生和 John Borges 先生及其在 Pune 的技术图书服务团队。

谢谢你们一起让这本书成为可能。最后说明一下：Pune、Nagpur 和 Satara 都是印度马哈拉施特拉邦的城市。

Contents 目　录

前言

致谢

第 1 章　欢迎学习 C 语言 …… 1

1.1　程序、软件和操作系统 …… 2
1.2　机器语言和汇编语言 …… 2
1.3　过程式语言 …… 3
1.4　面向对象的语言 …… 3
1.5　计算机术语 …… 4
1.6　编译和解释语言 …… 4
1.7　第一个 C 程序 …… 5
1.8　C 的突出特点 …… 6
1.9　隐式类型转换 …… 7
1.10　显式类型转换 …… 9

第 2 章　控制语句 …… 10

2.1　求 1 到 *N* 的整数的总和 …… 10
2.2　计算数字的阶乘 …… 12
2.3　生成斐波那契数列 …… 14
2.4　确定给定数字是否为质数 …… 17
2.5　计算正弦函数 …… 20
2.6　计算余弦函数 …… 21
2.7　计算二次方程的根 …… 23
2.8　计算整数的反转数 …… 25
2.9　使用嵌套循环打印几何图案 …… 26
2.10　生成终值利息系数表 …… 28

第 3 章　函数和数组 …… 31

3.1　确定圆周率 π 的值 …… 32
3.2　从数字列表中选择质数 …… 34
3.3　使用递归进行数字求和 …… 37
3.4　使用递归计算斐波那契数列 …… 39
3.5　使用递归计算数字的阶乘 …… 40
3.6　搜索整数数组中的最大元素 …… 42
3.7　解决经典的汉诺塔问题 …… 43
3.8　解决八皇后问题 …… 46
3.9　计算给定对象集的排列和组合 …… 48
3.10　对两个矩阵求和 …… 50
3.11　计算矩阵的转置 …… 53
3.12　计算矩阵的乘积 …… 55

第 4 章　指针和数组 …… 59

4.1　从包含 int 类型数据的数组中获取数据 …… 59
4.2　使用数组名称从数组中获取数据 …… 61

4.3 从包含 char 和 double 类型数据的数组中获取数据 …… 62
4.4 访问越界数组元素 …… 64
4.5 存储字符串 …… 66
4.6 存储字符串而不进行初始化 …… 68
4.7 在交互式会话中存储字符串 …… 70
4.8 获取二维数组中元素的地址 …… 71
4.9 获取二维数组中行的基址 …… 73
4.10 从二维数组中获取数据 …… 74
4.11 使用数组名称从二维数组中获取数据 …… 76
4.12 使用指针数组从数组中获取数据 …… 78
4.13 物理交换字符串 …… 80
4.14 逻辑交换字符串 …… 82
4.15 以交互方式存储字符串 …… 85
4.16 将命令行参数传递给程序 …… 87
4.17 使用指向指针的指针获取存储的字符串 …… 90

第 5 章 利用指针使用函数和结构 …… 94

5.1 通过引用传递函数参数 …… 94
5.2 显示嵌套结构中存储的数据 …… 96
5.3 使用函数构建结构 …… 102
5.4 通过将结构传递给函数来修改结构中的数据 …… 103
5.5 通过将指向结构的指针传递给函数来修改结构中的数据 …… 105
5.6 使用结构数组存储和获取数据 …… 107
5.7 在交互模式下使用结构数组存储和获取数据 …… 110
5.8 使用函数指针调用函数 …… 113
5.9 实现基于文本的菜单系统 …… 115

第 6 章 数据文件 …… 118

6.1 逐个字符地读取文本文件 …… 118
6.2 文件打开失败时处理错误 …… 122
6.3 以批处理模式写入文本文件 …… 125
6.4 以交互模式写入文本文件 …… 127
6.5 逐个字符串地读取文本文件 …… 130
6.6 逐个字符地写入文本文件 …… 132
6.7 将整数写入文本文件 …… 134
6.8 将结构写入文本文件 …… 136
6.9 读取存储在文本文件中的整数 …… 139
6.10 读取存储在文本文件中的结构 …… 141
6.11 将整数写入二进制文件 …… 143
6.12 将结构写入二进制文件 …… 145
6.13 读取写入二进制文件的整数 …… 147
6.14 读取写入二进制文件的结构 …… 149
6.15 重命名文件 …… 151
6.16 删除文件 …… 152
6.17 复制文本文件 …… 153
6.18 复制二进制文件 …… 155
6.19 写入文件并读取该文件 …… 157
6.20 将文本文件定位到所需字符 …… 159
6.21 从键盘设备文件中读取 …… 165
6.22 将文本写入显示器设备文件 …… 167
6.23 从键盘设备文件读取文本并将其写入显示器设备文件 …… 169

第 7 章　自引用结构 ····· 171

7.1　以交互方式生成数字列表 ····· 171
7.2　使用匿名变量创建链表 ····· 173
7.3　从链表中删除组件 ····· 177
7.4　将组件插入链表 ····· 181
7.5　在交互式会话中创建链表 ····· 187
7.6　处理线性链表 ····· 191
7.7　创建具备前向和后向遍历功能的线性链表 ····· 200

第 8 章　栈和队列 ····· 203

8.1　将栈实现为数组 ····· 204
8.2　将栈实现为链表 ····· 207
8.3　将中缀表达式转换为后缀表达式 ····· 212
8.4　将中缀表达式转换为前缀表达式 ····· 215
8.5　将循环队列实现为数组 ····· 218

第 9 章　搜索和排序 ····· 223

9.1　使用线性搜索查找数据元素 ····· 224
9.2　使用二分搜索查找数据元素 ····· 226
9.3　使用冒泡排序对给定的数字列表进行排序 ····· 228
9.4　使用插入排序对给定的数字列表进行排序 ····· 231
9.5　使用选择排序对给定的数字列表进行排序 ····· 233
9.6　使用归并排序对给定的数字列表进行排序 ····· 235
9.7　使用希尔排序对给定的数字列表进行排序 ····· 238
9.8　使用快速排序对给定的数字列表进行排序 ····· 240

第 10 章　密码系统 ····· 243

10.1　使用反向密码方法 ····· 245
10.2　使用恺撒密码方法 ····· 248
10.3　使用转置密码方法 ····· 251
10.4　使用乘法密码方法 ····· 255
10.5　使用仿射密码方法 ····· 259
10.6　使用简单替换密码方法 ····· 263
10.7　使用 Vigenère 密码方法 ····· 268
10.8　使用一次性密钥密码方法 ····· 273
10.9　使用 RSA 密码方法 ····· 277

第 11 章　数值方法 ····· 283

11.1　用对分法求方程的根 ····· 284
11.2　用试位法求方程的根 ····· 286
11.3　用穆勒法求方程的根 ····· 289
11.4　用牛顿拉夫森迭代法求方程的根 ····· 292
11.5　用牛顿前向插值法构造新的数据点 ····· 294
11.6　用牛顿后向插值法构造新的数据点 ····· 296
11.7　用高斯前向插值法构造新的数据点 ····· 299
11.8　用高斯后向插值法构造新的数据点 ····· 301

11.9 用斯特林插值法构造新的数据点……304
11.10 用贝塞尔插值法构造新的数据点……306
11.11 用拉普拉斯－埃弗雷特插值法构造新的数据点……309
11.12 用拉格朗日插值法构造新的数据点……312
11.13 用梯形数值积分法计算积分值…314
11.14 用辛普森的 3/8 数值积分法计算积分值……316
11.15 用辛普森的 1/3 数值积分法计算积分值……318
11.16 用修正的欧拉方法求解微分方程……320
11.17 用龙格－库塔方法求解微分方程……322

附录 A 参考表…… 325
附录 B 库函数…… 334
附录 C C 习惯用法…… 338
附录 D 术语表…… 347

第 1 章 Chapter 1

欢迎学习 C 语言

C 是一门过程式编程语言。C 的早期历史与 UNIX 非常接近。这是因为 C 是专门为编写 UNIX 操作系统而开发的，UNIX 操作系统由贝尔实验室于 1969 年推出，用来取代 PDP-7 计算机的 Multics 操作系统。UNIX 的原始版本是用汇编语言编写的，但用汇编语言编写的程序比用高级语言编写的程序可移植性差。因此，AT&T 的人们决定用高级语言重写此操作系统。做出这个决定之后，他们开始寻找合适的语言，但是当时没有合适的允许位级编程的高级语言。

在同一时期（1970 年），Kenneth Thompson 开发了一种系统编程语言，按照其母语言 BCPL（由 Martin Richards 于 1967 年开发）命名为 B 语言。1972 年，C 语言作为 B 语言的改进版本首次亮相。C 语言由 Dennis Ritchie 开发，其名字来自 B（即字母表中，字母 C 跟着字母 B，并且在 BCPL 的名字中，字母 C 也跟着字母 B）。

Ritchie 和贝尔实验室的一组研究人员一起为 C 语言创建了一个编译器。与 B 语言不同，C 语言配备了大量标准类型。1973 年，新版本的 UNIX 发布了，其中 90% 以上的 UNIX 源代码都是用 C 语言重写的，这增强了它的可移植性。随着这个新版本 UNIX 的到来，计算社区意识到了 C 语言的强大功能。随着 Brian Kernighan 和 Dennis Ritchie 在 1978 年的《C 程序设计语言》[⊖]一书的出版，C 语言一举成名。

1983 年，美国国家标准协会（ANSI）成立了一个名为 X3J11 的委员会，以创建 C 语言的标准规格说明。1989 年，该标准被批准为 ANSI X3.159—1989 “Programming Language C”。这个版本的 C 语言通常称为 ANSI C、标准 C 或 C89。1990 年，国际标准化组织（ISO）采纳 ANSI C 标准（稍作修改），把它作为 ISO/IEC 8999:1990 发布。这个

⊖ 本书中文版、影印版已由机械工业出版社引进出版。——编辑注

版本通常称为C90。1995年，X3J11委员会修改了C89，并增加了一个国际字符集。1999年，它被进一步修改并发布为ISO 9899:1999。该标准通常称为C99。2000年，它被采纳为ANSI标准。

1.1 程序、软件和操作系统

在继续之前，先来解释**计算机程序**一词的含义（以下简称**程序**）。程序只不过是要送到计算机上的一组指令，这样计算机就可以完成一些人们需要它完成的工作。程序和软件之间的关系可以表示如下：

程序＋可移植性＋文档＋维护＝软件

可移植性是指程序在不同平台（例如Windows平台、UNIX平台等）上运行的能力。文档表示用户手册和插入程序中的注释。维护意味着根据用户的请求调试和修改程序。

Microsoft Windows是一种**操作系统**。它包含一个图形用户界面（GUI）。**图形**意味着图像，**界面**意味着中间人，因此GUI是用户和帮助用户的计算机的内部机器（意味着计算机用户）之间的图像中间人。在酒店，服务员接受你的订单，走进厨房，收集你点的菜肴，并为你服务。同样，操作系统接受你的命令，接近计算机的内部机器，然后为你服务。

1.2 机器语言和汇编语言

微处理器可以恰当地描述为个人计算机的大脑。微处理器只不过是一个芯片。有各种微处理器可供选择。微处理器和中央处理单元（CPU）是同义词。微处理器包含一个称为**算术逻辑单元**（ALU）的重要组件，它执行所有计算。ALU的一个显著特征是它只能理解机器语言，而机器语言又只包含两个字母，即0和1（相比之下英语由26个字母组成）。这是典型的机器语言指令：

```
10111100010110
```

几十年前，程序员确实使用机器语言来编写程序。键盘只包含两个键，标着0和1。编写一个机器语言程序，然后在计算机中键入它是一项费力而乏味的工作。之后出现了汇编语言，它减轻了程序员的负担。汇编语言是低级语言。以下是典型的汇编语言语句（执行两个数字的乘法运算），这肯定比前面给出的机器语言指令更具可读性：

```
MUL X, Y
```

如果机器语言程序包含50个语句，那么相应的汇编语言程序也将包含大约50个语句。由于ALU仅理解机器语言，因此人们开发了专用软件（称为汇编器），以将汇编语言程序转换为机器语言程序。

1.3　过程式语言

典型的过程式语言比汇编语言更接近英语。例如，下面是过程式语言 Pascal 中的语句：

```
If (rollNumber = 147) Then Write ('Entry denied.');
```

这个语句的含义非常明显：如果 rollNumber 的值为 147，则在屏幕上显示消息“Entry denied.”。为了将过程式语言程序翻译成机器语言程序，人们使用称为**编译器**的软件。过程式语言是高级语言。

程序员将过程式语言与结构化编程技术结合使用。什么是结构化编程？从广义上讲，**结构化编程**这一术语指的是将编程艺术转化为理性科学的运动。这一切都始于 Edsger Dijkstra 在 1968 年 3 月出版的《Communications of the ACM》期刊上发表的“Go To Statement Considered Harmful”（Go To 语句是有害的）。结构化编程依赖于以下基石：

- **模块化**：不是编写一个大程序，而是将程序拆分为多个子程序或模块。
- **信息隐藏**：模块的接口应仅显示尽可能少的信息。例如，考虑一个计算数字平方根的模块。该模块的接口将接受一个数字并返回此数字的平方根。此模块的详细信息将对此模块的用户隐藏。
- **抽象**：抽象是隐藏细节的过程，以便于理解复杂的系统。在某种程度上，抽象与信息隐藏有关。

然而，随着程序越来越大，很明显结构化编程技术虽然是必要的，但还不够。因此，计算机科学家转向面向对象编程，以便管理更复杂的项目。

1.4　面向对象的语言

我们使用计算机程序来解决现实问题。结构化范式的问题在于，无法使用它方便地在计算机上模拟实际问题。在结构化范式中，使用数据结构来模拟现实生活中的对象，但这些数据结构在模拟真实对象方面远远不够。汽车、房屋、狗和树是现实生活中对象的例子，我们期望编程语言能够模拟这些对象以解决现实生活中的问题。面向对象的范式简单地通过提供软件对象来模拟现实生活中的对象，从根本上解决了这个问题。面向对象范式提供的对象是类的实例，并拥有像现实生活中的对象那样的身份、属性和行为。例如，如果 Bird（鸟）是一个类，那么 parrot、peacock、sparrow 和 eagle（鹦鹉、孔雀、麻雀和鹰）就是对象或 Bird 类的实例。此外，如果 Mammal（哺乳动物）是一个类，那么 cat、dog、lion 和 tiger（猫、狗、狮子和老虎）是对象或 Mammal 类的实例。与结构化范式相比，面向对象范式更能够重用现有代码。**代码**是指程序或其中的一部分。

面向对象的范式与结构化范式一样古老。结构化范式的运动开始于 1968 年 Dijkstra 的著名的文章“Go To Statement Considered Harmful”，而面向对象范式自己的编程语言 SIMULA 67 出现于 1967 年，然而，SIMULA 67 的面向对象能力不是很强。第一个真正面

向对象的语言是 Smalltalk。事实上，**面向对象**这个术语正是通过 Smalltalk 文献创造的。C 不是面向对象的语言，它只是一种过程式语言。1983 年，Bjarne Stroustrup 为 C 语言添加了面向对象的功能，并将这种新语言命名为 C++，这是计算机行业广泛使用和重视的第一种面向对象语言。今天，最流行的面向对象语言是 Java。面向对象语言是高级语言。

1.5 计算机术语

在几乎所有科学中，术语都源自希腊语或拉丁语等语言。为什么？如果你从英语中导出术语，则存在技术含义与该术语的当前用法之间出现混淆的风险。然而，计算机科学中的术语源自英语，这会使初学者混淆。诸如**树**（tree）、**内存**（memory）、**核心**（core）、**根**（root）、**文件夹**（folder）、**文件**（file）、**目录**（directory）、**病毒**（virus）、**蠕虫**（worm）、**垃圾**（garbage）等英语单词被用作计算机领域中的技术术语。你可能不知道除了当前的非技术含义之外，特定术语还附带了一些技术含义。为避免混淆，请始终在桌面上备一本好的计算机词典。无论何时有疑问，都请查词典。

1.6 编译和解释语言

当计算机科学家设计新的编程语言时，主要问题是在各种平台上实现该语言。实现语言有两种基本方法：

- **编译**：高级语言的代码被翻译成低级语言。创建一个文件来存储编译或翻译后的代码。然后，你需要通过提供适当的命令来执行已编译的代码。
- **解释**：代码中的指令由虚拟机（或解释器）逐条解释（执行）。不创建文件。

现在详细讨论这两种方法。

编译

在编译方法中，高级语言的源代码被翻译成实际机器的机器语言。FORTRAN、Pascal、Ada、PL/1、COBOL、C 和 C++ 都是编译语言。例如，考虑一个在屏幕上显示文本“Hello”的 C 程序。假设 hello.c 是包含此程序源代码的文件（C 源代码文件的扩展名为 .c）。C 编译器**编译**（或翻译）源代码并生成可执行文件 hello.exe。文件 hello.exe 包含实际机器的机器语言指令。你现在需要通过提供适当的命令来执行文件 hello.exe，并且执行文件 hello.exe 不是编译过程的一部分。在 Windows 平台上准备的可执行文件 hello.exe 只能在 Windows 平台上执行，根本无法在 UNIX 平台或 Linux 平台上执行此文件。但是，可以使用适用于所有平台的 C 编译器。因此，可以在 UNIX 或 Linux 平台上加载适当的 C 编译器编译文件 hello.c，以生成可执行文件 hello.exe[⊖]，然后在该平台上执行它。

⊖ UNIX 或 Linux 平台的可执行文件通常不用 .exe 扩展名。——译者注

编译语言的主要好处是编译程序的执行速度很快。编译语言的主要缺点是程序的可执行版本依赖于平台。

解释

在解释方法中，通过添加期望数量的软件层来创建虚拟机，使得高级语言的源代码是该虚拟机的“机器语言代码”。例如，BASIC 语言是一种解释语言。考虑一个在屏幕上显示文本“ Hello”的 BASIC 程序。假设此程序的源代码存储在 hello.bas 文件中。hello.bas 中的源代码被送到 BASIC 虚拟机，并且 BASIC 虚拟机逐条解释（执行）hello.bas 中的指令。另请注意，hello.bas 中的编程语句是 BASIC 虚拟机的机器语言指令。在解释过程中不会创建新文件。

解释语言的主要好处是程序与平台无关。解释语言的主要缺点是程序的解释（执行）很慢。BASIC、LISP、SNOBOL4、APL 和 Java 都是解释语言。

在实践中，很少使用纯粹的解释，如 BASIC 的情况。在几乎所有解释语言（例如 Java）中，都使用编译和解释的组合。首先，使用编译器将高级语言中的源代码转换为中间级代码。其次，创建虚拟机，使得上述中间级代码是该虚拟机的机器语言代码。然后将中间级代码送到虚拟机以进行解释（执行）。最后，请注意所有脚本语言（例如 Perl、JavaScript、VBScript、AppleScript 等）都是纯粹的解释语言。

1.7　第一个 C 程序

作为一种传统，典型的 C 编程书中的第一个程序通常是“ Hello, world”程序。我们遵循这一传统，创建并运行（执行）第一个程序。此程序将在屏幕上显示“ Hello, world”文本。在 C 文件中键入以下文本（程序）并将其保存在文件夹 C:\Code 中名为 hello.c 的文件中：

```
#include <stdio.h>
main()
{
  printf("Hello, world\n") ;
  return(0) ;
}
```

编译并执行此程序，屏幕上会显示以下文本：

```
Hello, world
```

如果语言的编译器或解释器区分大写和小写字母，则称该语言为**区分大小写的**。Pascal 和 BASIC 不是区分大小写的语言。C 和 C++ 是区分大小写的语言。

- C 是区分大小写的语言，因此不应混淆大写和小写字母。例如，如果键入 Main 而不是 main，则会导致错误。

- 不要混淆文件名和程序名。这里，hello.c 是包含程序源代码的文件的名称，而 hello 是程序名称。

为了解释这个程序（或者任何其他程序）是如何工作的，需要引用这个程序中的代码行（LOC），因此，需要对代码行进行编号。我重写了程序 hello，其中添加了行号作为注释（这些是多行注释），如下所示。此程序产生与程序 hello 相同的输出。

```
/* 此程序将产生与程序 hello 相同的输出。唯一的区别是此程序包含注释。注释仅为
   程序员提供方便。编译器直接忽略这些注释。*/
                                                    /* BL */
#include <stdio.h>                                  /* LOC 1 */
                                                    /* BL    */
main()                                              /* LOC 2 */
{                                                   /* LOC 3 */
  printf("Hello, world\n");                         /* LOC 4 */
  return(0);                                        /* LOC 5 */
}                                                   /* LOC 6 */
```

C 中有两种类型的注释：多行注释（也称为**块注释**）和单行注释（也称为**行注释**）。单行注释来自 C++，自 C99 起正式引入 C 语言。

现在注意用插入单行注释重写的程序 hello，如下所示。此程序产生与程序 hello 相同的输出。

```
// 此程序将生成与程序 hello 相同的输出。唯一区别在于该
// 程序包含注释。注释仅为程序员提供方便。
// 编译器直接忽略这些注释。
                                                    // BL
#include <stdio.h>                                  // LOC 1
                                                    // BL
main()                                              // LOC 2
{                                                   // LOC 3
  printf("Hello, world\n");                         // LOC 4
  return(0);                                        // LOC 5
}                                                   // LOC 6
```

传统上，C 语言教科书仅使用多行注释并避免单行注释。我将在本书中遵循这一惯例。

1.8 C 的突出特点

C 是一种流行语言。它大受欢迎归功于以下功能：

- C 是一种小型语言。它只有 32 个关键字。因此，可以很快学会它。
- 它具有强大的内置函数库。
- 它是一种可移植的语言。为一种平台（例如，Windows）编写的 C 程序可以移植到另一种具有微小变化的平台（例如，Solaris）。
- C 程序执行速度快。因此，C 程序用在效率很重要的地方。

- 结构化编程所需的所有构造都可以在 C 中获得。
- C 语言中提供了低级编程所需的大量构造，因此 C 可用于系统编程。
- C 语言中提供指针，这增强了它的功能。
- 在 C 中递归功能可用于解决棘手的问题。
- C 具有扩展自身的能力。程序员可以将自己编写的函数添加到函数库中。
- C 几乎是一种强类型语言。

1.9　隐式类型转换

在赋值语句中，右侧显示的量称为**右值**（r-value），左侧显示的量称为**左值**（l-value）。在每个赋值语句中，都要确保左值的数据类型与右值的数据类型相同。有关示例请参阅此处给出的赋值语句（假设 intN 为 int 变量）：

```
intN = 350;                    /* L1, 现在intN的值是350 */
```

这里，L1 表示 LOC 1，为了节省空间，我使用字母 L 来表示代码中的 LOC。在 LOC 1 中，左值为 intN，右值为 350，它们的数据类型是相同的：int。当编译器编译这样的语句时，它从不会忘记检查赋值语句两边的类型。编译器的这个任务称为**类型检查**（typechecking）。如果双方的类型不一样会怎样？会发生类型转换！在类型转换中，右侧的值的类型在赋值之前被更改为左侧的值的类型。类型转换可以分为两类。

- 隐式或自动类型转换
- 显式类型转换

注意这里给出的 LOC（假设 dblN 是 double 变量）：

```
dblN = 35;                     /* L2, 好的, 现在dblN的值是35.000000 */
```

在此 LOC 中，dblN 的类型为 double，数值常量 35 的类型为 int。这里，编译器将数据类型 35 从 int（源类型）提升为 double（目标类型），然后将 double 类型常量 35.000000 赋给 dblN。这称为**隐式类型转换或自动类型转换**。在隐式（或自动）类型转换中，类型转换是自动发生的。

在类型转换中，右值的类型称为**源类型**，左值的类型称为**目标类型**。如果目标类型的范围宽于源类型的范围，则此类型的转换称为**扩大类型转换**。如果目标类型的范围窄于源类型的范围，则此类型的转换称为**缩小类型转换**。LOC 2 中的类型转换是扩大类型转换，因为 double（目标类型）的范围比 int（源类型）的范围宽。

这是隐式类型转换的另一个例子（假设 intN 是 int 变量）：

```
intN = 14.85;                  /* L3, 好的, 现在intN的值是14 */
```

在此 LOC 中，数字常量 14.85 的类型是 double，intN 的类型是 int。这里，编译器将 14.85 的数据类型从 double 降级为 int，它截断并丢弃其小数部分，然后将整数部分 14 赋值

给 intN。LOC 3 中的类型转换是缩小类型转换。

这是隐式类型转换的另一个例子：

```
dblN = 2/4.0;                    /* L4, 好的, 现在dblN的值是0.500000 */
```

在此 LOC 中，右值是一个表达式，该表达式又由数值常量 2 除以数值常量 4.0 组成。但是数值常量 2 的类型是 int，而数值常量 4.0 的类型是 double。这里，编译器将数值常量 2 的类型从 int 提升为 double，然后执行浮点数 2.0 / 4.0 的除法。结果 0.5 被赋值给 dblN。

■注意 在表达式或赋值语句中混合使用不同类型时，编译器会在计算表达式或赋值时执行自动类型转换。在执行类型转换时，编译器会尽力防止信息丢失。但有时候信息丢失是不可避免的。

例如，在 LOC 3 中，存在信息丢失（double 类型数值常量 14.85 转换为 int 类型数值常量 14）。在扩大类型转换中没有信息丢失，但在缩小类型转换时存在一些信息丢失。编译器总是允许扩大类型转换。编译器也允许缩小类型转换，但有时编译器会显示警告。不允许无意义的转换。某些类型转换在编译期间允许，但在运行期间会报告错误。例如，请注意这里给出的代码段：

```
double dblN1 = 1.7e+300;                    /* LOC K */
float fltN1;                                /* LOC L */
fltN1 = dblN1;                              /* LOC M */
printf("Value of fltN1  %e\n", fltN1);      /* LOC N */
```

编译器成功编译了这段代码，没有任何警告。但是，当你执行这段代码时，屏幕上会显示以下文本行而不是预期的输出：

```
Floating point error: Overflow.
Abnormal program termination
```

程序在执行 LOC M 期间“崩溃”，这行代码尝试缩小类型转换。当一个程序在运行时突然终止时，我们用程序员的语言说程序崩溃了。

不同的语言允许将类型混合到不同的程度。允许不受限制地混合不同类型的语言称为**弱类型的**（weakly typed）语言或具有**弱类型**的语言。不允许混合不同类型的语言称为**强类型的**（strongly typed）语言或具有**强类型**的语言。

■注意 C 几乎是一种强类型语言。

C 的强类型检查在函数调用中很明显。如果函数需要一个 int 类型参数，并且将一个字符串作为参数（而不是 int 类型参数）传递给该函数，那么编译器会报告错误并停止编译程序，这证实了 C 是一种强类型语言。

请注意，前面使用了**几乎**这个术语，因为在某种程度上，C 语言中允许隐式类型转换，这使得 C 成为“几乎”强类型的语言，而不是完全强类型的语言。