PEARSON

C和C++实务精选
品味岁月积淀，读享技术菁华

C语言解惑

[美] | Alan R. Feuer | 著
杨涛 王建桥 杨晓云 | 译

The C Puzzle Book

- 最佳C语言进阶图书
- 来自C语言摇篮贝尔实验室的秘籍
- 测试你的C语言段位

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

PEARSON

[美] | Alan R. Feuer | 著
杨涛 王建桥 杨晓云 | 译

C语言解惑

The C Puzzle Book

人民邮电出版社
北京

图书在版编目（CIP）数据

C语言解惑 / (美) 富勒 (Feuer, A. R.) 著 ; 杨涛, 王建桥, 杨晓云译. -- 北京 : 人民邮电出版社, 2016.3
ISBN 978-7-115-35114-2

Ⅰ. ①C… Ⅱ. ①富… ②杨… ③王… ④杨… Ⅲ. ①C语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2015)第007124号

◆ 著　　[美] Alan R. Feuer
译　　杨　涛　王建桥　杨晓云
责任编辑　傅道坤
责任印制　张佳莹　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：10.25
字数：189 千字　　2016 年 3 月第 1 版
印数：1－3 000 册　　2016 年 3 月北京第 1 次印刷
著作权合同登记号　图字：01-2007-0544 号

定价：35.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315

内 容 提 要

本书脱胎于作者在C语言的摇篮——贝尔实验室教授C语言的讲稿，几乎涵盖了C语言各个方面的难点，并包含了一些其他书籍很少分析到的问题。在每个谜题后面都有详尽的解题分析，使读者能够清晰地把握C语言的构造与含义，学会处理许多常见的限制和陷阱，是一本绝佳的C语言练习册。

本书结构清晰，循序渐进，适合于C语言的初学者，可用作高校计算机相关专业的辅助教材，同时也可供具有一定C语言编程经验的读者复习提高之用。

前　言

C语言并不大——如果以参考手册的篇幅作为衡量标准的话，C语言甚至可以归为一种“小”语言。不过，这种“小”并不意味着C语言的功能不够强大，而是说明了C语言里的限制性规则比较少。C语言本身的设计非常简洁精妙，这一点相信C语言的使用者早已有所体会。

不过，C语言的这种精妙对C语言的初学者来说，似乎是故作神秘。因为限制较少，C语言可以写成内容丰富的表达式，这可能会被初学者认为是输出错误。C语言的紧凑性允许以简洁凝炼的方式实现常见的编程任务。

学用C语言的过程，与学用其他的程序设计语言一样，大致可以分为三个阶段（这样的分段想必读者在其他的教科书里已见过很多次了）。第一阶段是理解这种语言的语法，这至少需要达到编译器不再提示程序存在语法性错误的程度。第二阶段是了解编译器将赋予正确构造的结构什么含义。第三阶段是形成一种适合这种语言的编程风格；只有到了这一阶段，才能编写出清晰简洁而又正确的程序。

本书中的谜题是我们为了帮助广大读者迅速通过C语言学习过程中的第二阶段而准备的。它们不仅可以检验读者对C语言语法规则的掌握程度，还可以引导读者接触一些很少涉及的问题，绕过一些常规的限制，跳过几个打开的陷阱。（我们必须承认，C语言也有一些需要一定的编程经验才能掌握的难点，在这方面与其他程序设计语言没有什么两样。）

请不要把本书的谜题视为优秀的代码范例，事实上，本书的某些代码相当不容易理解。但这也正是我们编写本书的目的之一。编写失当的程序往往却能成为一个有意义的谜题：

- 表达含混，必须参照一本语法手册才能看懂；
- 结构过于复杂，数据结构和程序结构不够清晰，难以记忆和理解；
- 某些用法晦涩难懂，在运用某些概念的时候不遵守有关的标准。

本书中的谜题全部基于ANSI标准的C语言，涉及的某些功能可能有某些编译器不支持。

不过，因为ANSI C是绝大多数C语言版本的超集，所以即使你们的编译器不支持书中涉及的某项功能，它也很可能会以另外一种方式实现。

如何使用这本书

本书是标准C语言教材[1]的绝佳配套读物。本书分为9章，每章探讨一个主题，各章均由C语言代码示例构成，分别对该章主题的各个方面进行探讨。在那些代码示例里有许多print语句，而本书的主要工作就是分析每段示例代码的输出到底是什么。书中的示例程序都是彼此独立的，但后面的谜题需要把前面的谜题搞清楚之后才容易理解。

每个程序的输出紧接在相应的程序代码的后面给出。这些程序都已经在“Sun工作站+Unix操作系统”和“PC + MS/DOS操作系统”环境下调试通过。少数例子在这两种平台上有不同的输出，我们分别给出这两种输出。

本书的大部分篇幅是一步一步地解释各类谜题的答案，相关的C语言编程技巧就穿插在解释内容里。

做谜题的一般顺序是这样的：

- 阅读C语言教科书里该主题的相关内容。
- 阅读本书与该主题相关的章节里的每一段示例程序：
  - 做各段示例程序相关的谜题；
  - 把你的答案与书中给出的程序输出进行对照；
  - 阅读本书对解决方案的解释。

致谢

本书脱胎于我在C语言的诞生地——贝尔实验室教授C语言的讲稿。来自听课人员的踊跃反应使得我有勇气把这些谜题及其答案整理成书。有许多朋友和同事对本书的草稿提出了宝贵的建议和指教，他们是：Al Boysen，Jr.、Jeannette Feuer、Brian Kernighan、John Linderman、David Nowitz、Elaine Piskorik、Bill Roome、Keith Vollherbst和Charles Wetherell。最后，我要感谢贝尔实验室为我提供的有利环境和大力支持。

Alan Feuer

1. 推荐人民邮电出版社即将出版的美国主流教材《C程序设计：现代方法》（K. N. King著）。——译者注

目　录

第 1 章　操作符……1
谜题 1.1　基本算术操作符……1
谜题 1.2　赋值操作符……6
谜题 1.3　逻辑操作符和增量操作符……10
谜题 1.4　二进制位操作符……16
谜题 1.5　关系操作符和条件操作符……23
谜题 1.6　操作符的优先级和求值顺序……27

第 2 章　基本类型……33
谜题 2.1　字符、字符串和整数类型……33
谜题 2.2　整数和浮点数的转换……37
谜题 2.3　其他类型的转换……43

第 3 章　头文件……49

第 4 章　控制流……51
谜题 4.1　if 语句……51
谜题 4.2　while 和 for 语句……57
谜题 4.3　语句的嵌套……62
谜题 4.4　switch、break 和 continue 语句……67

第 5 章　编程风格……73
谜题 5.1　选用正确的条件……73
谜题 5.2　选用正确的结构……76

第 6 章　存储类……81
谜题 6.1　块……81
谜题 6.2　函数……85
谜题 6.3　更多的函数……89
谜题 6.4　文件……94

第 7 章 指针和数组 …… 99
谜题 7.1 简单的指针和数组 …… 99
谜题 7.2 指针数组 …… 105
谜题 7.3 多维数组 …… 110
谜题 7.4 难解的指针 …… 113

第 8 章 结构 …… 119
谜题 8.1 简单的结构、嵌套结构 …… 119
谜题 8.2 结构数组 …… 124
谜题 8.3 结构指针数组 …… 131

第 9 章 预处理器 …… 139
谜题 9.1 C 语言的预处理器的宏命令替换功能 …… 139
谜题 9.2 宏的副作用 …… 145

附 录 …… 149
附录 A 操作符优先级表 …… 149
附录 B 操作符汇总表 …… 150
附录 C ASCII 字符表 …… 153
附录 D 类型转换表 …… 154

第1章 操作符

C语言程序由语句构成，而语句由表达式构成，表达式又由操作符和操作数构成。C语言中的操作符非常丰富——本书的附录B所给出的操作符汇总表就是最好的证据。正是因为这种丰富性，为操作符确定操作数的规则就成为了我们理解C语言表达式的核心和关键。那些规则——即所谓的“优先级”和“关联性”——汇总在本书附录A的操作符优先级表里。请使用该表格来解答本章中的谜题。

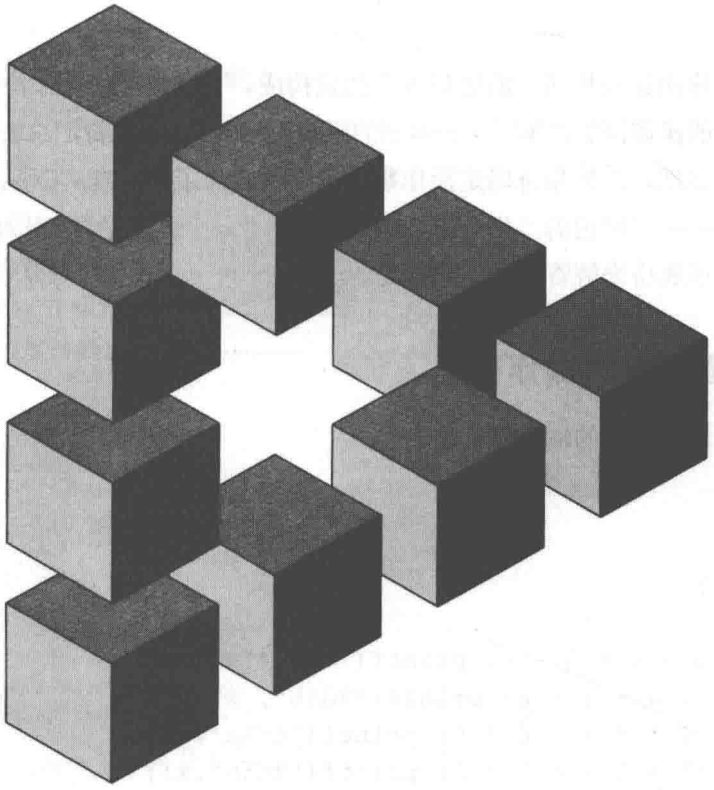
谜题 1.1 基本算术操作符

请问，下面这个程序的输出是什么？

```
main()
{
    int x;

    x = -3 + 4 * 5 - 6; printf("%d\n",x);          (1.1.1)
    x = 3 + 4 % 5 - 6; printf("%d\n", x);          (1.1.2)
    x = - 3 * 4 % - 6 / 5; printf("%d\n",x);       (1.1.3)
    x = ( 7 + 6 ) % 5 / 2; printf("%d\n",x);       (1.1.4)
}
```

输出：

```
11                                                    (1.1.1)
1                                                     (1.1.2)
0                                                     (1.1.3)
1                                                     (1.1.4)
```



解惑 1.1　基本算术操作符

1.1.1

```
x = -3 + 4 * 5 - 6
```

在解答这道谜题之前，请大家先把附录A中的操作符优先级表从头到尾好好看一遍。

```
x = (-3) + 4 * 5 - 6
```

在这个表达式里，优先级最高的操作符是一元操作符“-”。我们将使用括号来表明操作符与操作数的绑定情况。

```
x = (-3) + (4*5) - 6
```

在这个表达式里，优先级第二高的操作符是乘法操作符“*”。

```
x = ((-3) + (4*5))-6
```

“+”和“-”操作符的优先级是一样的。如果操作符的优先级相同，它们与操作数的绑定顺序将由该级别的关联规则来决定。具体到“+”和“-”，关联是从左到右。所以先对“+”操作符进行绑定。

```
x = (((-3) + (4*5))-6)
```

接下来，对“-”操作符进行绑定。

```
(x = (((-3) + (4*5))-6))
```

最后绑定的是出现在操作符优先级表末尾的“=”操作符。把每个操作符的操作数都确定下来之后，我们就可以对表达式进行求值了。

```
(x = ((-3 + (4*5))-6))
```

对于这个表达式，求值过程将按由内而外的顺序进行。

```
(x = ((-3+20)-6))
(x = (17-6))
(x = 11)
```

把每一个子表达式替换为相应的计算结果。

```
11,一个整数
```

对赋值表达式而言，它的值是“=”操作符右边的计算结果，类型是“=”操作符左边的变量的类型。

关于*printf*：printf是C语言标准函数库提供的一个格式化输出函数。printf函数的第一个参数是一个格式字符串，它描述了后面的参数将如何输出。“%”引出参数的输出规范。在谜题1.1这段程序里，“%d”将使得printf函数对第二个参数进行分析然后将其输出为一个十进制整数。printf函数也可以用来输出字面字符。在这个程序里，还输出了一个换行符，这需要在格式字符串里给出换行符的名字（\n）。

1.1.2

```
x = 3 + 4 % 5 - 6
```

这个表达式与前一个很相似。

```
x = 3 + (4 % 5) - 6
x = (3 + (4 % 5)) - 6
x = ((3 + (4 % 5)) - 6)
(x = ((3 + (4 % 5)) - 6))
```

按照操作符的优先级和关联规则进行绑定（求余操作符“%”在这里的用途是求出4除以5的余数）。

```
(x = ((3 + 4) - 6))
(x = (7 - 6))
(x = 1)
1
```

按由内而外的顺序逐步求出表达式的值。

1.1.3

```
x = -3 * 4 % - 6 / 5
```

这个表达式比前两个要稍微复杂一点儿，但只要严格按照操作符的优先级和关联规则就可以求出。

```
x = (-3) * 4 % (- 6) / 5
x = ((-3) * 4) % (- 6) / 5
```

“*”、“%”和“/”操作符的优先级是一样的，它们将按从左到右的顺序与操作数相关联。

```
x = (((-3) * 4) % (- 6)) / 5
x = ((((-3) * 4) % (- 6)) / 5)
(x = ((((-3) * 4) % (- 6)) / 5))
```

```
(x = (((-3 * 4) % - 6) / 5))
(x = ((-12 % - 6) / 5))
(x = (0 / 5))
(x = 0)
0
```

按由内而外的顺序逐步求出表达式的值。

1.1.4

```
x = (7 + 6) % 5 / 2
```

当然，我们并非不能改变预先定义好的操作符优先级。我们总可以用括号来明确地表明我们想先进行哪些计算。

```
x = (7 + 6) % 5 / 2
```

括号里的子表达式将首先绑定。

```
x = ((7 + 6) % 5) / 2
```

接下来，像刚才那样根据操作符的优先级和关联规则进行绑定。

```
x = (((7 + 6) % 5) / 2)
(x = (((7 + 6) % 5) / 2))
(x = ((13 % 5) / 2))
(x = (3 / 2))
```

求值。

```
(x = 1)
1
```

整数运算的结果将舍弃小数部分。

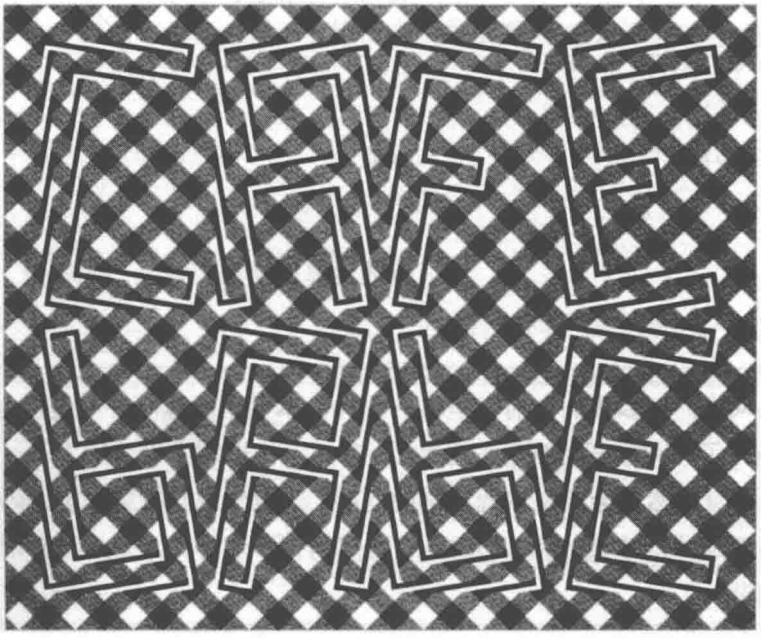
> **关于编程风格：**正如我们在前言里讲的那样，本书里的程序并不是应当仿效的范例。这些程序只是为了让大家开动脑筋去思考C语言的工作机制而设计的。不过，既然也是程序，这些谜题本身当然也会涉及编程风格的问题。一般来说，如果一段代码总是需要程序员求助于参考手册才能读懂的话，那它要么是编写得不够好，要么是需要增加一些注释来提供缺少的细节。

本谜题的这组题传达了这样一个信息：在复杂的表达式里，使用括号有助于读者搞清楚操作符与操作数之间的关联关系。

谜题 1.2　赋值操作符

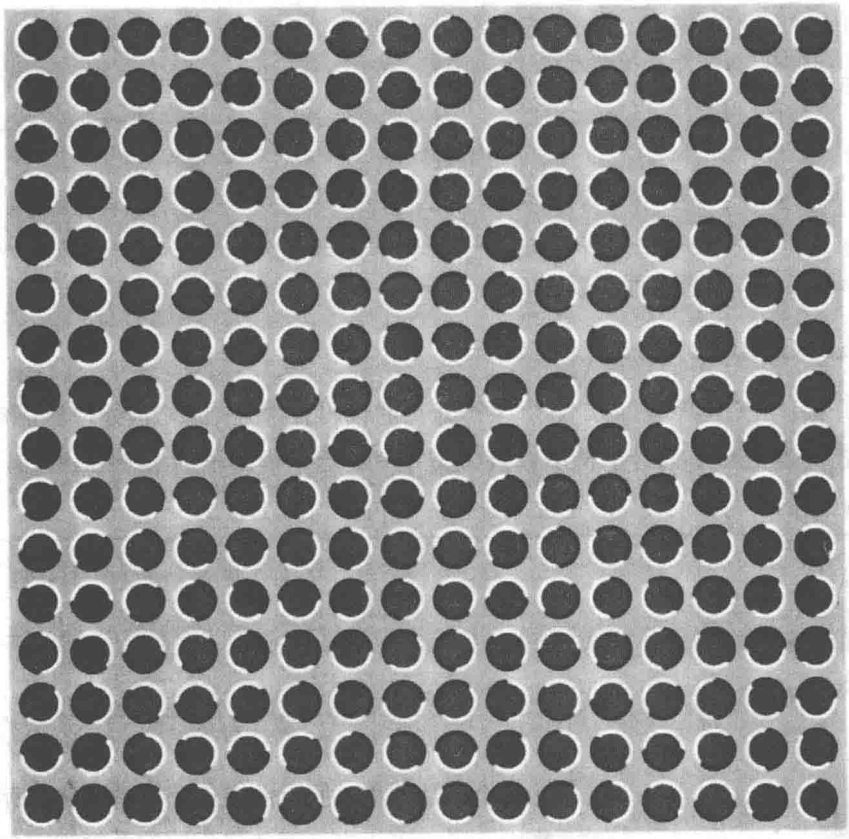
请问，下面这个程序的输出是什么？

```
#define PRINTX printf("%d\n",x)

main()
{
    int x = 2,  y,  z;

    x *= 3 + 2; PRINTX;                    (1.2.1)
    x *= y = z = 4; PRINTX;                (1.2.2)
    x = y == z; PRINTX;                    (1.2.3)
    x == ( y = z ); PRINTX;                (1.2.4)
}
```



输出：

```
10                                                    (1.2.1)
40                                                    (1.2.2)
1                                                     (1.2.3)
1                                                     (1.2.4)
```



解惑 1.2　赋值操作符

1.2.1

初始值：x=2

```
x *= 3 + 2
```

按照操作符的优先级和关联规则进行绑定。

```
x *= (3 + 2)
```

正如我们刚才看到的那样，赋值操作符的优先级低于算术操作符（“*=”是一个赋值操作符）。

```
(x *= (3 + 2))
```

```
(x *= 5)
```

求值。

```
(x = x*5)
```

把赋值操作扩展为它的等价形式。

```
(x = 10)
10
```

> 关于*define*：本谜题的程序以
>
> ```
> #define PRINTX  printf("%d\n", x)
> ```
>
> 这条语句开头。在C程序里，以“#”字符开头的代码行都是一条C预处理器语句。预处理器的工作之一是把一个记号字符串替换为另一个。这个程序里的define语句，告诉预处理器把代码里的“PRINTX”记号全部替换为字符串“printf("%d\n", x)”。

1.2.2

初始值：x=10

```
x *= y = z = 4
```

```
x *= y = (z = 4)
```

在这个表达式里，所有的操作符都是赋值，所以绑定顺序将由关联规则决定。赋值操作符的关联规则是从右到左。

```
x *= (y = (z = 4))
(x *= (y = (z = 4)))
```

```
(x *= (y = 4))
(x *= 4)
40
```

求值。

1.2.3

初始值：y=4,z=4

```
x = y == z
x = (y == z)
```

“=”（赋值）与“==”（是否相等）之间的区别经常让一些C语言的初学者感到困惑。根据C语言的操作符优先级表，“==”操作符的优先级要高于“=”操作符。

```
(x = (y == z))
(x = (TRUE))
(x = 1)
1
```

关系操作符和相等操作符的结果要么是TRUE（用整数值1表示）要么是FALSE（用整数值0表示）。

1.2.4

初始值：x=1,z=4

```
x == (y = z)
(x == (y = z))
```

在这个表达式里，因为使用了括号，所以赋值操作将优先于相等比较。

```
(x == 4)
```

求值。

```
FALSE,即0
```

这个表达式的求值结果是0。但因为变量x的值没有发生变化（“==”操作符不改变其操作数的值），所以PRINTX语句仍将输出1。