

从“错误”中学到的东西，往往比从“正确”中学到的多……



抛弃C程序设计 中的谬误与恶习

薛非 著

源自 ChinaUnix

论坛著名技术长贴《以C语言使人堕落》

ChinaUnix论坛和博客网

技术博客《C解惑》系列



$a+=a-=a*a$ 

一个语句必须在最后有一个分号，分号是语句中不可缺少的组成部分 

所有float型数据都先转换为double型，然后进行运算 

清华大学出版社



抛弃C程序设计 中的谬误与恶习

薛非 著



清华大学出版社
北京

内 容 简 介

这是迄今为止国内仅有的一部全面且集中地讨论 C 程序设计中常见错误的书籍。其主旨在于帮助读者从错误中更准确地理解 C 语言，并在实践中更好地运用 C 语言。

全书分为上下两部：上部“形而下学”讨论 C 代码中常见的错误和编程恶习；下部“形而上学”讨论的是对 C 语言本身认识方面的常见误区。这些问题特别具有普遍性，很多错误不仅仅存在于初学者之中，即使是职业程序员多半也会在本书中找到自己的错误。因此，对于任何想纠正自己对 C 语言的错误认识以及想不断提高 C 语言水平的读者来说，本书都提供了一种独到且重要的视角，是一本不容错过的、具有重要参考价值的技术资料。

本书并非那种长篇大论环环相扣的书籍，它的各个主题独立成篇，使人阅读起来轻松愉快。读这本书并不需要正襟危坐在书房里，它可以随时随地地拿过来翻阅。

本书可作为各高校 C 语言或相关课程的教学参考书，也适合那些具有一定经验的 C 语言程序员阅读使用。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

图书在版编目(CIP)数据

品悟 C：抛弃 C 程序设计中的谬误与恶习 / 薛非著. —北京：清华大学出版社，2012.10

ISBN 978-7-302-29626-3

I. ①品… II. ①薛… III. ①C 语言—程序设计 IV. ①TP312

中国版本图书馆 CIP 数据核字(2012)第 185168 号

责任编辑：栾大成

装帧设计：杨玉芳

责任校对：胡伟民

责任印制：王静怡

出版发行：清华大学出版社

网 址：<http://www.tup.com.cn>, <http://www.wqbook.com>

地 址：北京清华大学学研大厦 A 座 邮 编：100084

社 总 机：010-62770175 邮 购：010-62786544

投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈：010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者：清华大学印刷厂

装 订 者：三河市新茂装订有限公司

经 销：全国新华书店

开 本：188mm×260mm 印 张：32.75 插 页：1 字 数：920 千字

版 次：2012 年 10 月第 1 版 印 次：2012 年 10 月第 1 次印刷

印 数：1~5000

定 价：69.00 元

产品编号：043948-01

编者序——敢于向权威挑战

经常听到我的作者朋友说类似这样的话：“看我的代码，多帅气！”何谓帅气、漂亮的代码？

在我看来，先不谈代码丑俊，首先代码必须没有残疾。在健康的代码基础上，才能谈网络间流传的各种称号：漂亮代码、优质代码、帅气代码、性感代码……

这就是本书要讲的内容——摒弃一些错误的观念，重新认识 C 语言的严谨与华丽。

计算机语言是人类操控计算机硬件的主要手段，是语言创造者智慧的浓缩。计算机语言在中国的普及，是伴随着中国经济腾飞的这几十年。无数中国的程序员伴随着很多经典教材顺利入门并成长了起来，随着互联网的发展，更多的朋友在网上发表自己的看法，所以互联网也成为了大家获取编程技术的重要渠道。

但是，金无足赤，随着 C 语言标准的不断完善，C 开发理念的不断进步，市面上的一些教材和网上一些说法确实存在一些值得商榷的问题，或者谬误，因为这些资料传播甚广，有些反而成为了“标准”，这种现象有时候可能会误导初学者，让初学者无法从一开始就建立相对准确的开发观念。

本书就是这样一本敢于向“权威”挑战的书，通过作者多年的积累，以及众多热心网友的鼎力支持，对一些 C 语言中经常出现的问题进行讨论。

在中国的教育体制下，不可否认，很多创造力被限制甚至扼杀，很多路被强制标注为正确途径，很多人的学习兴趣在一开始就被磨灭……。但是很高兴能出现这么一本书，在阅读过程中，可以体会一众先锋程序员对开发观念的重视，对细节的精益求精，对逻辑的反复推敲，甚至上升到——对真理的求索。

我通常拿到一本书，会先浏览一下，把我认为重要的文字用特殊形式标出来，比如“提示”、“注意”、“TIPS”之类的样式以及文字加粗、代码加黑之类的重点标注，但是本书我在做出努力之后，发现无法进行这样的处理，因为似乎每句话都是重点，每段代码都是精心挑选的，无论是正确代码还是错误代码。

最后一点不得不提一下，作者的文笔非常犀利，对一些问题的批判毫不留情，可能是作者本人以及相关参与编写的网友受到过太多问题资料的影响，以至于走了不少弯路，所以面对问题资料，个别时候会流露出来一种愤青一般的愤怒。这可以理解，而且我在编辑加工过程中也尽量保持了文风，但是还是希望作者和所有读者做一名淡定的程序员。

如果本书冒犯了某些朋友，请不要大动肝火，可以发邮件与作者讨论，毕竟从一个角度看问题，存在片面的可能性。大家讨论技术，以代码会友。

权威之所以是权威，必然有其可取之处，对于权威，我们除了要有挑战的勇气，更多的是要理性的尊重，毕竟“权威”带领太多人进入 C 语言门槛。没有 BUG 的代码是不存在，同样没有问题的资料也是不存在的，希望大家在学习过程中，多思考，多交流，早日树立端正的开发思路。

在此感谢包括作者在内的热心人，有监督才会有活力，有挑战才会有发展！

前 言

这是一本什么样的书

这是一本关于 C 程序设计常见错误的书。书中详尽地剖析、讨论了学习或使用 C 语言过程中常见的一些错误观念和错误实践。目的是帮助读者更准确地理解并运用 C 语言。

为什么要写这样一本书

因为错误^①与程序设计一向如影随形，密不可分。只要编写程序，就无法避免错误。这是一条铁律。无论是初出茅庐的新手还是经验丰富的老手，无一例外。从某种意义上来说，软件业的主要工作有三项：制造错误、改正错误和掩饰错误。因此不深入了解和认识错误就不可能真正懂得编程。

孔子说：“未知生，焉知死？”程序设计也是这个道理，不懂得错误就不可能真正理解程序设计语言的要点及编程的真谛。实践表明，人们从错误中学习到的东西往往比从正确中学到的多得多。这就是人们常说的“吃一堑，长一智”。

事实上每个程序员的成长历程都是一部不断认识错误并予以改正的纠错史。每一个优秀的程序员无一不是从无数的错误中脱颖而出的，并且始终要与错误进行不懈地斗争。闪光的思想一向少不得经历错误的磨砺。

从这个角度来看，对程序设计中的错误进行讨论比那些一本正经地正确叙述更有助于程序设计人员水平的提高，本书的目的就在于此。

然而尽管市场上 C 语言的书籍多得可以车载斗量，但遗憾的是几乎鲜有专门论述 C 程序设计中错误的书籍，除了 Koenig 所著的《C 陷阱与缺陷》。不过我觉得作者 Koenig 先生“阐之未尽，我有我的看法”。

实际上 Koenig 先生并不了解中国国情，他不知道程序设计中的错误其实分为两种，一种是错误，另一种是中国式错误。一句话，他根本无法想象我们是用质量何等奇特的教科书、用何等落后的方式学习 C 编程。因此他的书只提到了普通的一般性错误，而对形形色色的具有中国特色的错误却只字未提。这是一个巨大的、需要填补的空白（我不知道这算不算学术空白，从某种意义上来说这更像是清除技术污染）。这项工作目前来看还没有人做。既然古代的圣贤曾经教诲过我们应该“当仁不让”，所以现在只好由本书来勉为其难了。

错误是怎样“炼”成的

众所周知，所谓编程在本质上无非是用特定的形式语言向计算机描述一个问题的解

^①错误有很多文雅含蓄或直截了当的称呼，比如：缺点(defect)，偏差(variance)，故障(fault)，失败(failure)，问题(problem)，矛盾(inconsistency)，错误(error)，特殊(feature)，事件(incident)，缺陷(bug)，异常(anomaly)，不良反应(idiosyncrasy)，崩溃(breakdown)……，不一而足。本书后面通称为错误或 BUG。

决方法。在此之前，对这种语言的学习也是一个必不可少的环节。在整个过程中的一系列环节的任何环节上都可能会出现错误。

首先，语言的学习过程中会产生许多错误。这些错误不仅是因为学习者自身的原因，同时也是由于教科书方面的原因。

许多虔诚的学习者内心的潜意识里会以为教科书不会有什么错误，他们一向跪在教科书上学习，遇到错误时只是不断地检讨自己。在这个连婴儿奶粉都可能含有三聚氰胺的社会，怀有这种念头实在是 too naive 了。没有任何理由指望你的教科书比婴儿奶粉更纯净。

教科书同样可能有错，这种“水源”的污染才是最严重的污染。在这种情况下，只查找自身出错的原因是不可能解决问题的。这种先入为主的错误非常难以纠正，而且其影响往往也最为深远。例如，很多人在成为职业程序员之后也还不清楚

```
a+=a-=a*a
```

这个表达式是不正确的。

对语言不够熟悉是初学者犯错误的另一个主要原因。通常表现为代码里不是缺点儿什么就是多点儿什么。这种错误是一种最初级的错误，连编译器能够指出，因而比较容易得到纠正，所以多半不足为虑。

学习一种语言，本质上就是为了学习一种新的思维方式。将平时不严谨的自然语言思维习惯带入程序经常会导致对程序设计语言的误解，例如把“x 不等于 1 或 2”错误地表达为

```
x!=1||x!=2
```

甚至我们所熟悉的严谨的数学语言，由于语境发生了变化，也会在程序中带来错误。例如，很多人都犯过把“==”（“等于”）写成数学中的“=”（“赋值”，在数学领域表示“等于”）这种错误。这种错误一般需要通过长期的自觉训练来纠正。

往往被初学者所忽视的是不良编程习惯的问题，他们一心只想着写正确的代码，而从不考虑如何正确地写出正确的代码。这往往会导致许多低级错误一犯再犯，比如一再出现括号不成对这样的问题。

缺乏良好的编程习惯是写不好程序的，任何一个工人都懂得：良好的产品质量需要由合理的工艺来保证。不懂得正确地编写程序就几乎不可能写出正确的程序。

初学者忽视不良习惯的另一个主要原因是，这些编程陋习有时并不能在短期内显示出其恶果，比如“int a,b,c;”这种烂得不能再烂的变量名。实际上良好编程习惯的意义不仅在于避免错误，还在于它意味着少犯错误，或者在出现错误时很容易检查到并修正，这些长远的利益在短时间内一般是体会不到的。

良好的习惯要靠自觉培养，但是如果教科书没告诉你这些，那是教科书的问题。因为初学者不可能在短时间内完全靠自己的悟性领会前人几十年积累下来的编程规范方面的经验。顺便说一句，有些教科书的代码风格本身就极滥无比，对初学者误导很大。

仅仅熟悉语言元素对于编程来说是远远不够的，因为每一个程序的目标都是为了解决问题。而从问题到程序要经历问题的提出、问题的分析到问题的解决，最后落实为代码，在这一过程中每一个步骤都可能会出现错误。

正确地提出问题是解决问题的先决条件，错误的问题是不可能解决的。例如，要求“输出 $1 \sim n \times n$ 的自然数构成的魔方阵”就是一个错误的问题，因为 2 阶的魔方阵根本不存在。求解这样的问题，可以说是未解先错。

问题提出模糊不清的要求也是一种错误，例如“有三个数 a 、 b 、 c ，要求按大小顺序把它们输出”，这里的三个数究竟是什么数，是模糊不清的，在不同的情况下代码截然不同。要求应该明确。要求不清是提出者的错误，但擅自猜测问题的要求则是程序员的错误。

编写求解错误问题或不严谨问题的程序有害无益，而且贻害无穷，这会使程序员在不知不觉中丧失必要的职业严谨，对错误的需求渐渐变得麻木不仁。要知道，在真实的开发中，因为软件需求错误而导致的软件错误占到错误总数的一半以上，而且那些错误的需求往往不那么清楚明显、容易察觉。不具备审视问题或要求是否合理的习惯和能力，在真正的软件开发中就如同盲人骑瞎马，不可能不掉到沟里。

初学者中很少有人意识到问题本身可能就是错误的。很多初学者常常不管三七二十一地解决错误的问题。这对他们的逻辑思维能力是一种巨大的慢性戕害。

即使正确地提出了问题，也还可能被程序设计者所误解。错误地理解问题的要求，同样是错误产生的一个重要原因。

分析问题的过程中最容易产生逻辑错误，或者对问题分析得不全面导致的顾此失彼。这种不全面的逻辑漏洞往往比完全的逻辑错误更难发现，因为这时程序的运行有可能会“显得”是正确的。

即使没有前面提到的那些错误，距离写出优秀的代码还有很长的路。写代码和写文章一样，需要精心地进行布局谋篇，用术语来说就是进行设计。

设计不仅包括算法设计还包括对数据结构的设计。在“算法——程序的灵魂”这种片面观点的误导下，很多人轻视数据结构的设计，冥思苦想所谓的算法。实际上算法和数据结构是密不可分的，那些艰深晦涩的糟糕算法，大多是因为垃圾的数据结构设计。

有些人写程序毫无设计意识，上来就开始匆忙写代码，东一榔头西一棒槌，写到哪算哪，写出的代码笨拙无比、僵硬造作、逻辑混乱、啰唆重复或者效率低下。这样的代码即使能输出结果，也是不合格的丑陋代码。

难道程序长得丑陋也是一种错误吗？是的，作为一种艺术，程序有自己的美学标准。从微观上看，这种美感表现为充分体现语言的优美和风格的简洁；从宏观来看则表现为一种层次清晰和条理分明。这种标准并不出于纯粹的审美，也有其功用价值——不容易出错，发生错误容易改正。而要达到这一目标，程序设计实践过程离不开正确的哲学指导，这种哲学的一个核心理念就是结构化程序设计思想。

至此，尽管还有很多种类错误没有谈到，但是已经不难得出结论，程序设计错误很多。需要始终对错误保持警惕。

应该如何对待程序设计中的错误

首先，不要惧怕错误。错误尽管繁多，但只要了解认识错误就可以改正、避免错误。最可怕的是对错误的无知与鸵鸟般的无视。认识错误是改正错误的前提。

其次,了解错误越早越好,了解得越多越好。了解得越早进步得越快,了解得越多进步越大。

第三,错误也是一种财富。既然从错误中积累经验是提高程序设计水平的必经之路,那么从他人的错误中吸取教训则是一种提高程序设计水平的捷径,因为所谓捷径无非就是少走弯路或不走弯路而已。

本书的主要内容及特点

本书汇集了大量 C 语言的错误和编程陋习,逐个予以详细讨论,这些错误主要取自国内发行量和影响力较大的 C 语言教科书及其他一些非教科书 C 语言书籍以及网络上比较流行的观点,主要包括□□□□□[为节约篇幅,此处删去 286 个字],也有一些则是 C 语言本身的一些不易理解的特点以及学习者不具备良好的编程习惯所导致的错误。应该说本书所举出的反面例子非常具有代表性。

本书分为上下两部,上部“形而下学”讨论代码中的常见错误和不良风格,下部“形而上学”讨论的是很多人对 C 语言特有的错误观点和认识。

书中各个小节基本上是互相独立的,因此读这本书并不需要循序渐进,可以翻到哪页读哪页。

什么人应该及什么人不应该看这本书

对于使用国内教科书学习 C 语言的人来说,无论是初学者还是职业程序员,这本书应该是很有价值的参考读物。因为本书的内容恰恰是教科书不讲的内容或恰恰是教科书讲错了的内容。

如果是为了应付什么等级考试之类的话,就不需要浪费钱财了。看了此书你可能会发现那些考试本身也有很多错误。

阅读指南

如下形式,标注为“样本”的虚线框中的内容,是采样自教材或者互联网上值得商榷的说法或者代码。

```
#include <stdio.h>
#include <string.h>
int main()
{ char s1[80],s2[80];
  int i;
  printf("input s2:");
  scanf("%s",s2);
  .....
```



如下形式，虚线框中的内容，是存在问题，或者说的不那么完美的说法或者代码。

```
printf("enter data:\n");
scanf("%d",&a[0]);
i=1;
while(i<N)
{scanf("%d",&a[i]);
  if(a[i]>=a[i-1])
    i++;
  .....
```

如下形式，无虚线框中的内容，是没有问题的优质代码或相对准确的说法。

```
#include <stdio.h>
int main( void )
{
  char str1[5],str2[5],str3[5];

  str1[4]='X';
  str2[4]='X';
  .....
```

虽然为写这本书花了近三年的时间，但最后完稿时，我发现这段时间依然显得非常短促。因此尽管这是一本讨论错误的书，但是毋庸置疑，它本身可能也免不了存在错误。

您若在阅读时发现任何错误或不妥之处，欢迎与作者联系并通知作者：
pmerofc@126.com。这里先谢了！

本书的勘误将发布在作者的博客：<http://blog.chinaunix.net/uid/22996974.html>
和 <http://www.cnblogs.com/pmer/>。

致谢

本书的写作过程中得到了很多人的帮助和支持，在此作者表示深深的感谢。

特别感谢 starwing83 网友的一贯支持和帮助，他对程序设计语言的深刻理解和丰富的实践经验不但让笔者受益匪浅，而且也使本书在许多方面得到了提高和充实。本书的第 11 章问题 28、第 18 章问题 10 和第 19 章问题 1 为 starwing83 网友所撰写。

特别感谢 OwnWaterloo 网友的一贯支持和帮助，他对标准的准确理解和对程序设计深刻的洞悉解开了笔者的很多疑惑，使作者在很多问题上都茅塞顿开。第 5 章“问

题 19” 为 OwnWaterloo 网友所撰写。

特别感谢幻の上帝网友, 在多年的共同探讨中, 他热情而无私地给予作者很多支持、帮助和启迪。本书的很多观点受惠于幻の上帝网友的启迪, 他还为本书收集了很多资料, 为本书做了许多翔实的考证工作, 并纠正了作者的很多错误。

特别感谢诸多网友在长期的讨论中所给予的道义支持、精神鼓励和技术帮助, 他们是: 变异老鼠、shan_ghost、蔡万钊、狗气球、狗蛋、三月二十七、tempname2、x5miao、madoldman、walleeee、ChiyuT、davelv、huangzhenfan、花瓣雪、一介村夫、wait_rabbit、MMMX、rover12421、良化纲领、assiss、gccer、8pm、supermegaboy、A.com、noword2k、hahajerry007、ztz0223、Ray001、fera、unixlinuxsys、cokeboL、koolcoy、sh19871122、群雄逐鹿中原、fender0107401、jerryz920、hellioncu、blueheavenljn、x2、lylesong、supersuper8、gaara99、Kabie、pkkj、rossini23、wolfkin、milujite、weixuejun、cnhbdu、yesBSD、kouu、chinesedragon、water_wf、gz80、toniz、eminem112、sunjiakuang、airjordanforce、yumaofsj、yug1129、campuspuzzle、www1862、zxrjkl、peijue、jhzhz_snps、x75yan、zhaohongjian000、tinysniper、xxwpk007、geel、字母二十六、奶茶 dsk、abc976031617、rain_fish、amarant、财版、没本、grand508、star1983653、bill15、makeit、liexusong、pandaia、txg531、kingwolf520、梅川内依酷、bukkake、wuxb45、hobbs136、cjaizss、greensnow、jhui66、luojiannx、耐心学习、xiaobenniao514、L_kernel、geruihai、erlangs、A13433758072、CUXXXCU、ip200、Hongqiyaodao、dglwx、jhinux、xwxfirst、timesu、srdgame、septem776、liupingforarm、kellenforever、192redwolf、cobras、small_bee、gyarenas、jack4010、startn、rainysky、jimmyixy、litanhe、rubyish、Demon—Hunter、wuliming_sc、bigxu、xfoucs、时尚农民、yangxue1206、hukb_cu、jnbxzl0200157、tianshx、sukora、ux400、crane-fly、autoasm、时间看来、Jokday、asuka2001、namiii、jeung、btdm123、lilery、wangzhen11aaa、nketc、KanonInD、ybh37、dahan16、digdeep126、keytounix、光明-使者、gtv、SoforthHe、djsxut、oilgeo、呆呆的等候救赎、hbmhalley、china_ssl、solu、dajiangyou77、wl85125771、魔兽_LOVER、0xC1988、inzahgi、武林萌猪、liliny225、wgm001、royalzhang、千年老狼、musezh2、gooderfeng、HAL9000、Jack_Jack、大石头、万仓一黍、Ivony、愚溪、nscboy、minvt、underwings、Artech、ini_always、小彬、xujif、Leon Sharp、刘博平、linyilong、浩然正气 89、backag、happycat1988、ygcao、rhs、Zlinux、小_金_鱼、泡泡腾、SongSharp、szwe、eeeyes、+v、linxr、andreas、冠吸柏汁霆疯、九原山人、火禾、luotong、Programmer K、hachihe、rockyoung、czcz1024、Edgar Wang、magicDict、hoodlum1980、空明流转、随风浪迹天涯、不死鸟之魂、木耳、木野狐(Neil Chen)、散客游、sunriseyuen、chasefornone、飞浪、glshader、Credo、wcut、houqidian、Parry、xiekun605746、哥哥.Net、Virus-BeautyCode、庄金峰、我想我是风、Mien Ng、soap、greenhand2008、Homer_Simpson、zsea、臻、ctou45、shamo0303、gussing、eflay、王小兵、l23cy、winter-cn、Lee.Kevin、阳光明媚 Ryan、地狱门神、richardzeng、pulihe、寒露、cncolder、陈玉国、是否存在、key yao、codebumb、☆凯子、darklx、草珊瑚、鹤冲天、Jake Lin、Lithium、干拔三分、leizisdu、zzuxiaolei、BitSky、Jeffrey Zhao、桀骜的灵魂、海南。

胡勇、sm11e、我写的不是代码 是寂寞、lzyzizi、肖璟、嗷嗷、木+头、zy498420、No完成、五星、Simon-Zhu、BeckFun、Tony Zhou、lxlylm、陈梓瀚(vczh)、zdd、xchat、frank_hust、漂浮的雨、六芒星、MagicHu、天方、深蓝流沙、vipercaos、swfc_qinmm、潇湘雨歇、riccc、Jeff Wong、wincss、reavics、weiwelcome0、Muse、lx458004975、镜美如、volcanol、路过秋天、徐少侠、南京王清培、诺贝尔、蔺燕梅.....这个名单可能有所遗漏，在此向被遗漏的网友表示歉意。

特别感谢 ChinaUnix 论坛，在这个论坛的《以其昏昏使人昭昭》帖子中我完成了这本书的主要素材的积累。这个论坛良好的技术氛围使得本书素材的积累得以顺利进行。

特别感谢博客园，作者在这个论坛发表的《C 解毒》系列博文是本书的部分内容或草稿。在这个论坛得到了很多网友有益的反馈。

本书在写作过程中参阅了大量网上技术资料，从中得到了很多启发，郑重向这些作者表示谢意。

一位朋友根据《以其昏昏使人昭昭》帖子整理了一份文档，给本书的写作带来了极大的便利和帮助，在此深深表示感谢。

薛非

目 录

上部 形而下学——代码错误及陋习

第1章 低级错误	3
问题1 C啊,多少C++假汝之名而行——C、C++不分	4
问题2 环节缺失——关于C程序开发过程	7
问题3 错误的“标点”——混用中英文字符	9
问题4 错误的“单词”	11
问题5 “合并单词”产生的问题	12
问题6 形形色色的“分号”误用	13
问题7 “一仆二主”——两个main()函数	15
问题8 忘记“&”	17
问题9 who's who	18
问题10 同形异质——为符号常量赋值	19
问题11 词不达意	20
问题12 使用switch语句的常见错误	21
第2章 初级错误	23
问题1 随手写出的大数	24
问题2 关于数据类型的潜规则	25
问题3 使用垃圾值	26
问题4 对数组的误用与误解	28
问题5 好心办坏事——修改代码引发的错误	30
问题6 “右移运算等价于除法运算”	32
问题7 “中国人民很行”——C语言中的错别字	33
问题8 学会数数——循环次数错误	34
问题9 忽视函数原型	36
问题10 张冠李戴——错误的格式转换	37
问题11 只进不出——缺少输出	39
问题12 鸠占鹊巢——数组越界	41
第3章 逻辑错误	45
问题1 喧宾夺主——换行问题	46
问题2 结构体链表的一个例题——输出错误	47
问题3 一些幼稚的写法	50
问题4 c=getchar()!=EOF——优先级错误	50
问题5 判断三个整数相等——“=”运算符的误用	51
问题6 悬挂else问题——怎样写出正确的if语句	52

问题 7 浮点循环变量	53
问题 8 不顾常识	55
问题 9 自然语言的陷阱	57
问题 10 <code>c==' '\n' '\t'</code>	58
问题 11 乱做习题你伤不起啊	59
问题 12 误改循环变量	62
第 4 章 似是而非	65
问题 1 <code>void main()</code>	66
问题 2 生硬的嫁接, 滑稽的实参	66
问题 3 输出之误——输出数组应注意的问题	70
问题 4 多此一举	73
问题 5 笨拙的 <code>for</code> 语句	74
问题 6 只用 <code>printf()</code> 函数输出	76
问题 7 容易招致编译器误解的代码	76
问题 8 使用 <code>/*</code> 注释注意事项	77
问题 9 使用野指针	78
问题 10 用驴子拖宝马——滥用结构体	79
问题 11 似是而非的 <code>k=sqrt(n)</code>	84
问题 12 <code>scanf("%s",&str)</code>	88
问题 13 已知两边长求三角形面积——不完全函数参数	89
问题 14 对 <code>'c'</code> 转换说明符的误解	92
问题 15 多余的计算	93
问题 16 到处忙活与一劳永逸——函数类型声明位置问题	96
问题 17 半身不遂和粗中有细	99
问题 18 忽视前提	103
问题 19 <code>char text[3][80]</code> 不是一篇文章, 也并非 3 行文字	104
问题 20 不安全的 <code>gets()</code> 函数	108
问题 21 文件名残缺	109
问题 22 <code>exit(0)</code>	110
问题 23 舍近求远	112
问题 24 重复无效的代码	114
问题 25 拙劣的外部变量	114
第 5 章 画蛇添足	129
问题 1 <code>KISS</code>	130
问题 2 为赋新词强说愁	131
问题 3 啰唆重复	132
问题 4 表达式是什么	133
问题 5 不识指针	134

问题 6	脱裤子放屁	136
问题 7	怎样利用 <code>scanf()</code> 函数自虐	138
问题 8	自寻烦恼的 <code>scanf()</code> 函数调用方式	139
问题 9	输入指定范围的整数——Basic 风格写法和 C 风格写法	140
问题 10	“豆浆买两碗，喝一碗，倒一碗”——多余的数组元素	143
问题 11	筛法“四不像”	146
问题 12	不彻底的思考	152
问题 13	代码写得要“拽”(DRY)	156
问题 14	作茧自缚的“%3d”	158
问题 15	大腹便便——臃肿与重复	160
问题 16	舍简就繁	161
问题 17	程序的劣化与优化	163
问题 18	滥用变量综合症	172
问题 19	<code>free</code> 空指针	174
第 6 章	疑难杂症	177
问题 1	混乱是怎样炼成的	178
问题 2	无知乱吃药—— <code>strcpy()</code> 函数的误用	183
问题 3	边界测试——让 BUG 现形	186
问题 4	内裤外穿——错位及不伦不类	192
问题 5	flag 标志：代码馊了（一）	196
问题 6	flag 标志：代码馊了（二）	202
问题 7	flag 标志：代码馊了（三）	210
问题 8	鸡窝里飞出伪凤凰	216
问题 9	不顾常识	221
问题 10	不易察觉的 BUG	222
问题 11	含糊之过、多做之过及乱做之过	224
问题 12	怎样调戏程序	230
问题 13	糟蹋好题——魔方阵问题	236
第 7 章	风格问题	249
问题 1	如何进行数组初始化	250
问题 2	括号不配对错误及如何免疫	251
问题 3	不良代码编辑风格引起的错误——怎样写控制语句	252
问题 4	将 <code>main()</code> 函数进行到底	253
问题 5	赤裸的 <code>scanf()</code>	257
问题 6	c + 32 的问题	257
问题 7	怎样整理房间	259
问题 8	贪小便宜——省略函数类型声明等问题	262
问题 9	常数 Magic Number	265

问题 10 搏二兔	269
-----------	-----

下 部 形而上学——概念错误及误区

第 8 章 概念的缺失	277
问题 1 C 语言的历史及 C 标准	278
问题 2 只知变量不知对象	278
问题 3 C 语言中 Byte 的含义	280
问题 4 “不知所云”的无定义行为 (undefined behavior)	281
问题 5 “可以清心也”应如何解读——不确定行为 (Unspecified behavior)	283
问题 6 实现 (Implementantation)	285
问题 7 “编译器自己做主”——实现定义的行为 (implementation-defined behavior)	285
问题 8 整数提升 (Integer promotion)	285
问题 9 副效应 (side effects)	286
问题 10 “左值就是可以出现在赋值运算符的左侧”	287
第 9 章 基本词法	289
问题 1 字符集问题	290
问题 2 “ α , β , δ , ε , I, II, III, IV 等不是 C 语言的合法字符”	291
问题 3 “各种字符集的基本集都包括了 127 个字符”	291
问题 4 C 源程序的组成——“单词”视角	292
问题 5 “a”、‘a’ 与 a 辨析	293
问题 6 “C 语言关键字都是小写的”	294
问题 7 “C 语言允许一行内写多个语句”	294
问题 8 “标识符由字母、下划线和数字组成”	295
问题 9 标识符的长度限制	295
问题 10 “括号、赋值、逗号都是运算符”	296
第 10 章 数据类型及类型转换	297
问题 1 “类型就是对数据存储单元的安排”	298
问题 2 1.%3 之惑	298
问题 3 “int 为 2 个字节或 4 个字节”	299
问题 4 int 类型的范围问题	299
问题 5 混乱不堪的“整型”	300
问题 6 “-345 是整型常量”	300
问题 7 “字符数据以 ASCII 码存储”	301
问题 8 “char 类型就是 signed char 类型”	301
问题 9 通过程序测定 char 类型的性质	302

问题 10	“C 用 1 个字节 (8 位) 存储一个字符”	302
问题 11	“i='A' 是字符型数据赋给整型变量”	303
问题 12	“字符常量只能是一个字符”	303
问题 13	“从常量的表示形式可以判断其类型”	303
问题 14	1、'1' 与 "1"	304
问题 15	“变量值在存储单元中都是以补码形式存储的”	304
问题 16	“不应向无符号整型变量赋予一个负值”	305
问题 17	3.0 乘以 4.2 怎么就变成了 12.59999	306
问题 18	“实型变量”剖析	307
问题 19	使用 float 招来的警告及“因噎废食”的对策	307
问题 20	123f——怎样写 float 类型的常量	309
问题 21	“实型常量有两种表示形式”	309
问题 22	“浮点数类型包括 float、double 和 long double”	310
问题 23	“实数就是浮点数”	310
问题 24	“编译系统为每一个 float 型变量分配 4 个字节”	310
问题 25	“浮点型”、“双精度浮点型”和“实型”	311
问题 26	“Turbo C 对 long double 型分配 16 个字节”	311
问题 27	“必然以规范化的指数形式输出”	311
问题 28	“浮点型数据是用来表示具有小数点的实数的”	312
问题 29	对类型转换运算的误解	312
问题 30	“整型量与字符型数据以及逻辑型数据可以通用”	312
问题 31	“C 语言把所有的实数都作为双精度数处理”	314
问题 32	“所有 float 型数据都先转换为 double 型”	314
问题 33	“int 型与 float 或 double 型数据进行运算, 结果是 double 型”	315
问题 34	关于 char 类型数据的运算规则	316
问题 35	对表达式“10+'a'+i*f-d/3”的分析	316
问题 36	“字符数据既可以以字符形式输出, 也可以整数形式输出”	318
问题 37	3 ? 1 : 2	319
问题 38	“运算符两侧的数据类型不同, 先自动进行型转换”	319
第 11 章	运算和表达式 (Expression)	321
问题 1	关于“整型常量”	322
问题 2	“变量名是以一个名字代表一个存储地址”	323
问题 3	“缺省情况下所有变量都是 auto 的”	324
问题 4	“不能对一个类型赋值、存取或运算”	324
问题 5	“变量必须先定义、后使用”	324
问题 6	“常量是没有名字的不变量”	325
问题 7	“标识符用来标识一个对象”	325
问题 8	啥叫“中间变量”	326