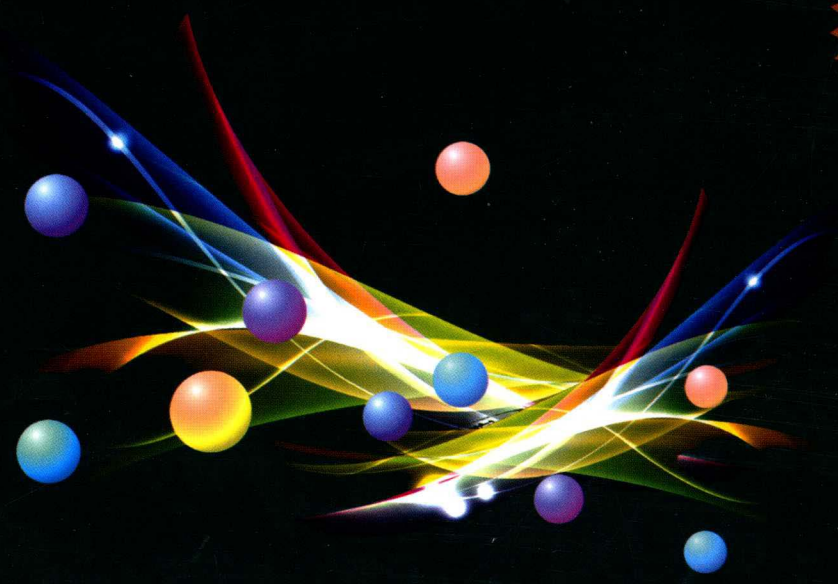

电气信息类
精品系列

C Chengxu Sheji

# C程序设计

杨　远◎主编

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

# C 程序设计

杨 远 主 编
赵 先 库 波 副主编

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

## 内 容 提 要

本书全面、系统地介绍了C语言的基本概念、语义和语法，精心设计和挑选了大量具有代表性的例题，从问题实际出发进行详细阐述，选择合理的数据结构、构造算法、编码的结构化程序设计过程，引导读者逐步掌握程序设计的思想、方法和技巧。本书的特点是问题覆盖面广，求解方法分析深入浅出，条理清晰，注重对读者程序设计能力的训练。

本书作为学习程序设计的入门教材，以对读者进行基本训练为出发点，以提高综合运用C语言进行程序设计的能力为目标，可作为非计算机专业教材，也可供计算机专业学生使用。



**图书在版编目（CIP）数据**

C程序设计／杨远主编. —北京：北京理工大学出版社，2015.5
ISBN 978-7-5682-0485-9

Ⅰ.①C…　Ⅱ.①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073629号

出版发行／北京理工大学出版社
社　　址／北京市海淀区中关村南大街5号
邮　　编／100081
电　　话／(010)68914775(办公室)　68944990(批销中心)　68911084(读者服务部)
网　　址／http：// www.bitpress.com.cn
经　　销／全国各地新华书店
印　　刷／三河市天利华印刷装订有限公司
开　　本／787毫米×1092毫米　1/16
印　　张／16.25
字　　数／388千字
版　　次／2015年5月第1版　2015年5月第1次印刷　责任校对／陈玉梅
定　　价／52.00元　责任印制／周瑞红

# 前　言

C 语言是使用最广泛的一种高级语言，其发展也相当神速。它拥有众多的编译器，其中也不乏优秀者。C 语言以其灵活性强、效率高、可移植性好的特点，深得人心。后来发展起来的 C++、Java 等语言，无不是在其基础上进行扩充的。

现在的 C 语言离其广泛应用期已经很远了，但这里不是要讨论学习什么以及有用无用的问题。对于 C 语言，在现在的很多领域仍然有其用武之地，即使你以后不用 C 语言写东西，甚至不从事程序编写工作，C 语言的教育意义仍然存在，比如学习面向过程的编程思想、为实际问题建立模型等。使用 C 语言要面对很多底层的东西，比如说指针，这对学习更“高级”的语言很有帮助。但是，对于 C 语言的过分要求也是不必的。现在很多学校的考试重点仍然是运算符优先级、printf 的参数表这些东西，这些东西在实际中非常不好用，或者说是可以不必理会的。对于运算符优先级，多用几个括号就可以了。我认为需要理解的东西是这些“过时”东西背后体现的本质（指针、内存），而不是老旧知识的本身。

学习 C 语言，起初会觉得要记的东西太多，这是由于它太灵活了，但只要学到一定程度，就会尝到甜头了。这种灵活性带来的是其可读性好、语法简单、效率高。当然，C 语言最大的特色还是它的指针，对指针的透彻理解将为今后的开发工作提供巨大帮助。在 C++中指针无处不见，很多参数就完全是指针化的。Java 中虽然摒弃了指针，那是从安全性方面考虑，但从性能上来说，那就损失大了。所以指针是一个核心。要学好 C 语言，需要透彻理解书本概念，并辅之以大量上机编程。要想提高应用水平，就要多看些应用方面的书，比如看看《数据结构》，自己想办法来实现其中的算法。总之，编程很重要，而且编程是靠编出来的，不是靠看出来的。在调试程序时，遇到问题应尽量自己解决，实在解决不了，可以请教老师，有条件的话，在网上提问题可收到事半功倍的效果。坚持下去，相信不久你就会成功的。以上所述，旨在抛砖引玉，若有不当，敬请见谅！

本书以目前计算机上流行的 ANSI C 为版本，兼顾集成化环境 Turbo C 编译程序，系统地、循序渐进地介绍了 C 语言的数据类型和运算符、语句格式和功能、结构化程序设计思想和方法等内容。

本书由杨远主编，赵先、库波副主编。全书由杨远统稿、审稿。

在本书编写的过程中，参考了大量有关 C 语言程序设计的书籍和资料，编者在此对这些作者表示感谢。

由于编者水平有限，书中难免有不足之处，敬请广大读者不吝赐教，以便再版时修改。

编　者

# 目　录

# 第1章 引 论

对于计算机，也许你是老手，也许你是新人……

但不管怎样，如果你现在要学习编程，那么你应该多多少少知道点什么叫硬件、什么叫软件。

美国一个电脑神童说："凡是摔到地上会坏的就是硬件。"我深感不妥，众所周知，如果把硬盘摔到地上，硬盘坏了，里头的那些数据也一样坏得让人心疼。

倘若按字面意思去理解，那就更加矛盾重重：硬盘硬是硬件；软盘软也是硬件。

还一种说法是：看得见摸得着的为硬件，看不见摸不着的为软件。刚觉得它说得不错，但马上又能发现它的破绽：我现在用的 Word 2000，它就在屏幕上，界面美观，操作方便……

金山词霸中有这样的解释："硬件：计算机及其他直接参与数据运算或信息交流的物理设备。"挺好，硬件就是设备。平常生活中的各种设备，洗衣机、冰箱、电视、还有螺丝刀、钳子、都是硬件。

软件呢？"软件：控制计算机硬件功能及其运行的指令、例行程序和符号语言"。指令、程序和符号语言是什么且不说，至少我们知道了软件是用来控制硬件的运行的。

这就好办了。我们可以打比方：譬如汽车，其本身自然是硬件，但关于驾驶车的那一套技术及有关交通规则，我们可称之为软件，因为后者控制了前者的运行。

现在来谈"指令、程序和符号语言"。交通方面的"软件"确实就是这些东西。不管你会不会驾车，但你至少应该坐过车吧？当你看到警察在你坐的车前用指头一指，就令司机脸色发青，之后，一套既定的处罚程序被执行。很快，听说那司机又在学习那些用来表示"单行"、"只许右拐"、"不许停车"、"禁鸣"等奇奇怪怪的符号语言了……

事实上，说软件看不见摸不着其实也正确，因为它们是思想、精神、规则、逻辑，本身是抽象的，确实不可触及。但软件总是要有载体来存放的，要有表达或表现方式，这些使得它们变得形象具体起来。在此意义上，说软件是摔在地上坏不了的东西，也相当行得通，神童毕竟是神童。

最后，什么是程序？我们来给它下个定义：

程序是一组按照一定的逻辑进行组合的指令。

因此，在以后的学习过程中，很多时候，我们会觉得程序就是指令；同样很多时候，我们会觉得程序就是逻辑。

当然，更多的时候，我们并不区分程序和软件。也许前者更趋于抽象，而后者更趋于具体。比如在写那些表达我们思想逻辑的指令时，我们喜欢说"写程序"；而当程序完成，可以待价而沽时，我们称它为软件产品。

## 1.1 计算机语言

程序是用计算机语言写成的。编程的实质就是用计算机语言来表达要解决的问题的逻辑。

那么，什么叫计算机语言呢？

先不必去解释。因为，计算机是机器，机器不是生物，它怎么能有语言呢？小猫小狗有语言我尚可相信。机器也有语言，还要我们去学习，这似乎有渎人类之尊严。

如果不把这个结解开，可能部分特别在意人类尊严的学生对学习编程从此产生心理障碍，无法继续学习……

狭义上说，我们讲的语言包括汉语、英语、广东话，它是语言，有声音。小鸟之间的唧唧喳喳，大抵也是语言。但其实语言二字虽都带口，却不是说非得有声才称为语言；哑语无声，但它也是语言。广义上讲，语言是沟通、交流的一种手段。基于此，我们认为所有的机器或工具，也包括计算机，都有它们自己的语言。比如锤子，它的语言是敲打；比如螺丝刀，它的语言是拧。如果你非要拧锤子，非要敲打螺丝刀，那么就像你用法语和广东人交谈，用粤语和法国人说话一样莫名其妙。

一般来说，越复杂的机器，人类与其沟通的语言也越复杂。比如汽车，你想驾驭它，你就必须去驾校参加学习。想一想，开车的时候，我们的确是在和车进行沟通。如果你俩之间的沟通出现差错——你心里右转，手却一个劲向左转方向盘，向机器发出了错误的命令——这将多么可怕呀！

至此，我们的心理障碍可以消除了。小猫小狗有语言是因为它们聪明，而机器有语言却是因为它们的笨。它们笨，没办法像动物一样可以通过培训来理解人类的意愿，所以，让人类反过来为它们制定一套沟通的规则，然后去学会这些语言，从而方便控制机器。

可以说，凡是机器语言都是笨笨的语言。机器语言可以分低级语言和高级语言，但无论何者，都笨得可爱——学得越多你就会越发现它的笨和可爱。另外，当我说越复杂的机器其语言也越复杂时，我用“一般来说”加以修饰。这是因为，发明和发展机器的智者们会为机器制造出越来越高级的语言，这些高级语言，最终越来越接近人类的自然语言。就像计算机，我们有信心相信，终有一天，它能听懂我们的语言——这就是流传在程序员中的一个梦。当程序员熬红眼敲打出数万行代码时，他们便会想起这个梦。闭上双眼，伸腰，对 PC 说：“BEGIN……”，深呼吸一次，然后说：“END”，睁眼时发现计算机已完成了所有工作……

下面回到计算机，它是机器，也是人类有史以来，继发明使用火、电、电子这些改善人类生活的工具后，最为重要、最为先进、最为广泛使用的工具。它的机器语言的复杂程度可想而知，已经复杂到必须成为大学的一门专业课程。然而别忘了前面的结论，语言只是沟通的手段。在这个意义上，当你用鼠标或键盘在计算机上进行输入时，只要你输入的是正确操作，我们都认为你在使用计算机语言，因为你确实是在用一种特定的方式或动作进行着和计算机的交流。

当然，这里的课程并不特意教你任何有关计算机的基本操作。计算机的基本操作主要是指如何使用计算机内已有的软件产品，比如 Windows（操作系统是软件，称为系统软件）、办公系统 MS Office 或 WPS Office（这些实现工作生活中具体应用需求的软件称为应用软件）、游戏（一种特定的，只拿来玩的应用软件，称为游戏软件）。但我们不同，我们学的是如何编写软件。也就是说，我们将是发明人、设计师、创造者；而他们（到今天仍拒不学习编程的家伙）都只是使用者。

程序（或软件）是用计算机语言写出来的。

写一个程序，大致是这么一个过程：

（1）人有一个问题或需求想用计算机解决……

（2）人想出解决问题或实现需求的思路……

（3）人将思路抽象成数学方法和逻辑表达或某种流程的模式……

（4）程序员将数学方法、逻辑表达中的数据和流程用计算机语言表达，称为编码……

用计算机高级语言写成的代码被语言的实现工具（Turbo C、VC、VB、Delphi 或 C++Builder）转换成计算机的最低级机器语言，这就完成了人与机器在程序制定上的最后沟通。

可见，你的思路是先用人类自己的语言思考，然后用一门计算机语言写成代码，最终需要一个语言工具来将它转换成机器可以理解的机器语言。我们要学的就是一门承上启下的计算机语言。这样语言有很多：BASIC，Pascal，C、C++，Java，C#…，我们学的 C 语言是使用最多的语言。有关 C 语言的更多特点，将在以后的章节谈到。

尽管你完全可以直接用最低级的计算机语言——机器语言来写代码，那样就不需要语言工具了，但在这里你要弄清楚，我们不是教机器语言。下一节，你会明白用机器直接能懂的语言——不妨称之为原始的机器语言写软件，在今天是多么的不现实呀。

## 1.2　语言和实现语言的工具

### 1.2.1　机器语言

你知道香蕉叫什么吗？就叫香蕉？叫 banana?

错，都错。

香蕉叫“牙牙”。

这是一个婴儿的语言。一个婴儿还没学会人类的主要语言，所以面对喜欢的东西总是发出咿咿呀呀的声音，也许你听不懂，但这是他的语言，符合小孩特点的语言。

计算机的机器语言也一样，必须符合计算机的硬件特点，而问题就在这里，越符合机器的特点，同时也就越不符合人类的特点。

计算机全称为电子计算机。20 世纪 40 年代以来，无线电技术和无线电工业的发展为电子计算机的研制提供了物质基础。1943—1946 年美国宾夕法尼亚大学研制的电子数字积分和计算机 ENIAC（ElectroicNumericalIntegratorAndComputer）是世界上第一台电子计算机。ENIAC 计算机共用了 18 000 多个电子管、15 000 个继电器，占地 170 $m^2$……

这是计算机的始祖，一堆电子管。随后，电子计算机进入第二时期，小巧的晶体管取代了电子管；再后来，集成电路又取代了晶体管，电子计算机进入第三时期。

但无论是哪一时期（以后也许不是），计算机始终采用电子器件作为其基本器件，因此电子器件的特点就是计算机的特点。

为什么使用电子元件？为什么木头不能做计算机？还真别说不能，你也应该知道，最早出现的用于计算的机器，真是木头的。你用过计算尺吗？算了，这玩意儿太简单。以前有人用木头作成齿轮，经过设计，当表示个位数的齿轮转动 1 圈时，就会带动表示十位数上的齿轮转动 1 格。以此为原理，只要你转动转轴，木头机器就会算出 123+456=579……

电子元件没有齿轮，但它们的特点是它们有两种很稳定的状态：导电或不导电。假如用


图 1–1

不通电时表示 0，通电时表示 1，再通过集成电路实现进位的机制，于是，计数功能就有了基础。可以用图 1–1 表示。

我们生活中常用的数是逢十进一，称为十进制数。而计算机由于其电子元件的特点，使用的是二进制数。

十进制数：最低位称为个位，高一位称为十位，再高一位称为百位。为什么这样称呼呢？因为在个位上，0 表示 0，1 表示 1，2 表示 2，3 表示 3，……；在十位上，0 表示 0，1 表示 10，2 表示 20，3 表示 30，……，总之，每高一位长十倍，为十进制。

二进制数：最低位仍可称为个位，但这里称为 1 位。1 位上，0 表示 0，1 表示 1。2 呢？没有 2，因为逢 2 就得进 1（后面同）。高一位称为 2 位，0 表示 0，1 表示 2。再高一位称为 4 位，0 表示 0，1 表示 4。可以看出，每高一位长 2 倍，为二进制。

现在看图 1–1，00、01、10 是 3 个二进制数。根据上面的进位方法，你可以算出它们分别表示十进制数的 0、1、2 来吗？首先，当面对二进制数时，先要认识到它们从低到高（从右到左）的位依次不再是个位、十位、百位，而是 1 位、2 位和 4 位。

00：都是 0，所以它就是 0；

01：2 位为 0，1 位为 1，表示 0 个 2 和 1 个 1，所以是 1；

10：2 位为 1，1 位为 0，表示 1 个 2 和 0 个 1，所以是 2。

计算机的机器语言正是由这些 0 和 1 组成的。事实上，计算机里的所有数据，无论是一个程序、一篇文稿、一张照片还是一首 MP3，最终都是 0 和 1。

世界就是这样奇妙，万事万物五彩缤纷，但进了计算机，却只是 0 和 1 的组合。

机器语言尽是 0 和 1，于是可以想象当时（还没有其他语言时）的程序员是如何编写程序的。他们写程序不用坐在计算机前，而在家里或什么地方，拿笔在纸上画圈，一圈、两圈、三圈（感觉有点像阿 Q），圈够了就给专门的打孔小姐照着在纸带上打成孔，最后这些纸带被计算机“吃”进去并读懂，然后执行。如用有孔表示 1，无孔表示 0，则图 1–2 表示 3 行数 110、011 和 101。

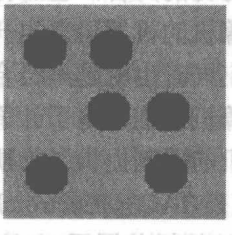


图 1–2

面对这样的“程序”你是否表示怀疑？别以为我瞎说，也许你的电脑很先进，但在你的电脑上，仍有那种程序的历史遗迹，即软驱。如果有软驱，那你应该能找到一张软盘。知道软盘有写保护吗？仔细看看那个写保护的开关——就是一个方孔。打开，告诉软驱本张软盘不能进行写操作；关闭，告诉软驱本张软盘可以进行写操作。

## 1.2.2 汇编语言

前面说机器语言尽是 0 和 1，那么是不是随便写一串 0 和 1 就算是程序呢？不是。就像汉语是由汉字组成的，可我要是说下面这一串汉字：

天爱我京门北安。

你觉得我是在说人话吗？

机器也有自己的固定词汇，在机器语言里称为机器指令。程序是由指令及数据组成的。

这些指令是一些固定的 0 和 1 的组合（不同厂商不同型号的机器，其指令又有不同）。作为程序员，就得将这些指令一次次正确地用 0 和 1 拼写出来。

你决不会将“我爱北京天安门”说成上面的话，但极有可能将 10101101 写成 10010101。所以很自然地出现了用符号来表示这些固定的二进制指令的语言，这就是汇编语言。

下面是一段代码，它表示的是：已知 b 等于 1；c 等于 2，计算 b+c 的值，并将该值赋给 a。

把这段代码的机器语言（左）和汇编语言（右）进行对照，可看出两者各自的特点。

```
100010100101010111000100      mov edx,[ebp-0x3c]
000000110101010111000000      add edx,[ebp-0x40]
100010010101010111001000      mov [ebp-0x38],edx
```

汇编语言仅是机器语言的一种助记符，两者之间没有本质的区别，所以很多时候人们把两者等同视之。

无论是机器语言还是汇编语言，都让人看了头痛。

### 1.2.3 高级语言

汇编语言和机器语言虽然很难记难写，但它们的代码效率高、占用内存少，这相当符合当时计算机的存储器昂贵、处理器功能有限等硬件特点。

众所周知，计算机的发展迅速，功能越来越强大。一方面，它有能力，人们也要它能完成越来越复杂或庞大数据量的计算功能，机器/汇编语言已经无法满足这些需求；另一方面，硬件的发展和关键元件价格的降低，使得程序员不需要在程序的降低内存占用、减少运算时间等方面花太多的精力，这样，各门高级语言便接二连三地出现了。

一门计算机语言“越符合机器的特点，同时也就越不符合人类的特点”。最早有 Pascal、C、C++、BASIC 等数百种高级语言，现在又有 Java、C#等。高级语言的高级之处在于它总是尽量接近人类的自然语言和思维方式。

当然，一门语言再好，如果没有其实现工具，一切都是空谈。对于 C 语言，我们推荐使用 Turbo C 2.0 集成开发环境。当然，如果你有意学习 C++等后续版本，你也可以选择使用微软的 VC，也可以使用 Borland C++ Builder，两者都相当优秀。

### 1.2.4 语言实现工具

高级语言比较接近人类的语言，用起来得心应手，但更得意的一定是让程序代码变成可执行文件。

无论是在写代码的过程，还是最后要编译成可执行文件，都需要有一个工具存在。这一工具一般称为编程集成环境（IDE）。之所以称为集成，是因为从写代码到最后软件的出炉，我们需要它的地方实在太多了。下面列出其中最重要的功能项。

（1）方便的代码编辑功能。尽管你可以使用记事本、Word 或其他任何文本编辑器来写代码，但除非特殊需要，否则那将是极为低效的方法。现在的编程集成环境都相当的智能，在很多情况下可以自动完成人们所需的代码，既准确又迅速。

（2）程序编译这功能。前面已讲过，人们写的代码在成为机器能懂的可执行程序前，必须通过编译。

（3）程序调试功能。如何尽量减少程序的 Bug 呢？没有编程集成环境提供的强大调试功

能，我们做的程序将毫无质量保证。

（4）其他辅助功能。安装程序的创建已属于另外一种工具的范畴，但人们仍可以通过编程集成环境来决定是最终生成单一可执行文件，还是带有其他动态库。如果是后者，还可以通过集成环境来检查程序运行时调用了哪些动态库文件。

当然，现在市面上可以见到的语言实现工具所提供的功能远不止上面所说的。对于一个工具，只有动手使用了，才会真正了解它。

## 1.3 C语言简介

C语言是贝尔实验室Dennis Ritchie在1973年设计的一种程序设计语言，其目的是用于写操作系统和系统程序，初期用在PDP-11计算机上写UNIX操作系统。20世纪70年代后作为UNIX的标准开发语言，C语言随着UNIX系统流行而得到越来越广泛的接受和应用。20世纪80年代后它被搬到包括大型机、工作站等许多系统上，逐渐成为开发系统程序和复杂软件的一种通用语言。随着计算机的蓬勃发展、处理能力的提高和应用的日益广泛，越来越多的人参与了计算机应用系统的开发工作，这就需要适合开发系统软件和应用软件的语言。C语言能较好地满足人们的需要，因此在计算机软件开发中得到了广泛的应用，逐渐成为最常用的系统开发语言之一，被人们用于开发微型机上的各种程序，甚至是非常复杂的软件系统。

在使用最多的以Intel及其兼容芯片为基础的计算机上，也有许多性能良好的C语言系统可用。包括Borland公司早期的Turbo C和后续Borland C/C++系列产品，Microsoft（微软）公司的Microsoft C和后续Visual C/C++系列产品，还有其他的C/C++语言系统产品，使用较广的有Watcom C/C++和SymantiC/C++等，此外还有许多廉价的和免费的C语言系统。其他计算机也有多种C语言系统。各种工作站系统大都采用UNIX和Linux，C语言是它们的标准系统开发语言。各种大型计算机上也有自己的C语言系统。

### 1.3.1 C语言的特点

C语言之所以能被世界计算机界广泛接受是由于其自身的特点。总体上说，其设计把直到20世纪70年代人们对于程序语言的认识和开发复杂系统程序（例如操作系统等）的需要成功地结合起来。C语言的主要特点包括以下几点。

C语言比较简单，是一个比较小的语言。学习时入门相对容易，知道很少东西就可以开始编程。C语言很好地总结了其他语言提出的程序库概念，把程序设计中许多需要的功能放在程序库（称为标准函数库）里实现，如输入输出功能等，这就使语言本身比较简单，编译程序的实现比较容易。人们早已用C语言写了它自己的编译程序，这种程序很容易移植到各种不同的计算机上，促进了C语言的传发展。

C语言提供了丰富的程序机制，包括各种控制机制和数据定义机制，能满足构造复杂程序时的各种需要。方便易用的函数定义和使用机制使人可以把复杂的程序分解成一个个具有一定独立性的函数，以分解程序的复杂性，使之更容易控制和把握。

C语言提供了一套预处理命令，支持程序或软件系统的分块开发。利用这些机制，一个软件系统可以较方便地先由几个人或几个小组分别开发，然后再集成，构成最终的系统。这

种工作方式对于开发大软件系统是必须的，人们用 C 语言开发了许多规模很大的系统。

C 语言的另一特点是可以写出效率很高的程序。人们之所以在一些地方继续使用汇编语言，就是因为高级语言写出的程序效率低一些。这样，开发效率要求特别高的程序时就只能使用汇编语言。C 语言的基本设计使得用它开发的程序具有较高的效率，它还提供了一组比较接近硬件的低级操作，可用于写较低级、需要直接与硬件打交道的程序或程序部分。这些特点使 C 语言常被当做汇编语言的“替代物”，从而大大提高了开发低层程序的效率。

C 语言的设计得到世界计算机界的广泛赞许。一方面，C 语言在程序语言研究领域具有一定价值，它引出了不少后继语言，还有许多新语言从 C 语言中汲取营养，吸收了它不少的东西。另一方面，C 语言对计算机工业和应用的发展也起了很重要的推动作用。正是由于这些贡献，C 语言的设计者获得世界计算机科学技术界的最高奖——图灵奖。

## 1.3.2 C 语言的发展和标准化

在设计 C 语言时，设计者主要把它作为汇编语言的替代品，作为写操作系统的工具，因此更多强调的是其灵活性和方便性，语言的规定很不严格，可以用许多不规则的方式写程序，因此也留下了许多不安全因素。使用这样的语言，就要求编程序者自己注意可能的问题，程序的正确性主要靠人来保证，语言的处理系统（编译程序）不能提供多少帮助。随着应用范围的扩大，使用 C 语言的人越来越多（显然其中大部分人对语言的理解远不如设计者），C 语言在这方面的缺限日益突出。由此造成的后果是，人们用 C 语言开发的复杂程序里常带有隐藏很深的错误，难以发现和改正。

随着 C 语言应用的发展，人们更强烈地希望 C 语言能成为一种更安全可靠、不依赖于具体计算机和操作系统（如 UNIX）的标准程序设计语言。美国国家标准局（ANSI）在 20 世纪 80 年代建立了专门的小组研究 C 语言标准化问题，这项工作的结果是 1988 年颁布的 ANSI C 标准。这个标准被国际标准化组织和各国标准化机构所接受，同样也被采纳为中国国家标准。此后人们继续工作，于 1999 年通过了 ISO/IEC 9899:1999 标准（一般称为 C99）。这一新标准对 ANSI C 做了一些小修订和扩充。

语言改造非常困难。虽然人们已经认识到原来 C 语言中的一些不足之处。但一方面已有的东西，主要是新标准出现前人们已开发的各种程序和软件是一笔巨大财富，不能轻易丢掉，彻底改造要耗费极大的人力和物力，也不可能做到；另外，一批老用户已养成习惯，不可能在一朝一夕改变。因此，即使想建立一个新标准，也要尽可能保持与原形式的兼容性，作为对现实的让，ANSI C 标准基本上容许原形式的 C 程序步。但新标准中强调：旧事物终将被抛弃，希望写程序的人尽量不要再使用它们。

今天学习 C 语言和程序设计，理所当然应该采用新的形式，不应该学习那些过时的东西。原因主要有两条：① 这些旧东西终归将被抛弃，养成使用它们的习惯后将来还要改，那时将更加费时费力，也毫无意义；② 这些过时的东西确实不好，虽然有时用它们能少写几个字符，但往往会阻碍编译系统对程序的检查。人很容易犯错误，在从事写程序这种复杂工作时尤其如此。阻止编译检查就是拒绝计算机帮助，其实际后果无法预料，可能代价惨重并且为程序中实际存在的隐藏错误耗费更多的时间和精力。

## 1.4 一个简单的C程序

本书将从第2章开始详细讨论各种C程序结构，讨论程序设计的各方面情况和问题。在开始这些讨论之前，先看一个简单的程序例子，看看用C语言写出的程序是什么样子，然后从它出发解释一些程序开发中的问题。下面是一个简单程序：

```
#include <stdio.h>
main()
{
    printf("Good morning!\n");
}
```

几点说明：

- #include 称为文件包含命令。
- 扩展名为.h 的文件称为头文件。
- main 是主函数的函数名，表示这是一个主函数。
- 每一个C源程序都必须有且只能有一个主函数（main 函数）。
- 一个函数包括函数首部（如 main ()）和函数体（程序中{}括起来的部分）两部分。
- 函数调用语句，printf 函数的功能是把要输出的内容送到显示器去显示。
- printf 函数是一个由系统定义的标准函数，可在程序中直接调用。

用C语言写的程序简称为C程序。上面这个简单程序可分为两个基本部分：第一行是个特殊行，说明程序用到C语言系统提供的标准功能，为此要参考标准库文件 stdio.h，有关细节在以后的章节中介绍；其余行是程序基本部分，描述程序所完成的工作。该程序的运行结果就是在显示器屏幕上输出一行文字“Good morning!”。

### 1.4.1 C程序的加工和执行

C语言是高级程序语言，用C语言写出的程序通常称作源程序。C程序人容易使用、书写和阅读，但计算机却不能直接执行，因为计算机只能识别和执行特定二进制形式的机器语言程序。为使计算机能完成某个C源程序所描述的工作，就必须首先把这个源程序（如上面简单例子）转换成二进制形式的机器语言程序，这种转换由C语言系统完成。由源程序到机器语言程序的转换过程称为“C 程序的加工”。每个C语言系统都具有加工C源程序的功能，包括“编译程序”、“连接程序”等，系统里还可能有一些其他的程序或功能模块。

程序加工通常分两步完成：第一步，由编译程序对源程序文件进行分析和处理，生成相应的机器语言目标模块，由目标模块构成的代码文件称为目标文件。目标文件还不能执行，因为其中缺少C 程序运行所需要的一个公共部分——C程序的运行系统。此外，一般C程序里都要使用函数库提供的某些功能，例如前面例子中用到标准函数库的一个输出函数（printf 是该函数的名字）。为构造出完整的可以运行的程序，还需要第二步加工——连接。这一工作由连接程序完成，将编译得到的目标模块与其他必要部分（运行系统、函数库提供的功能模块等）拼装起来，做成可执行程序。图 1–3 说明了C程序的基本加工过程。

对前面简单 C 程序的例子进行加工后，就能得到一个与之对应的、可以在计算机上执行的程序。启动运行这个可执行程序，将能看到它的执行结果。这个程序的执行将得到一行输出，通常显示在计算机屏幕上或者图形用户界面上的特定窗口里：

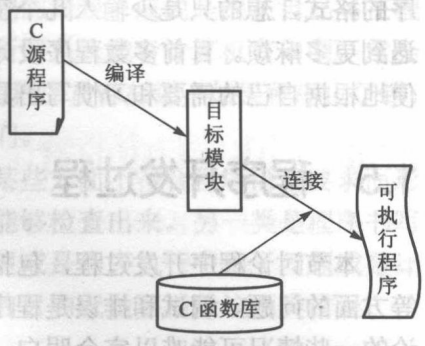

图 1-3

```
Good morning!
```

如果修改程序，将双引号里的一串字符换成其他内容，就可以让它输出那些内容。例如：

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

这一程序加工后执行，就会输出：

```
Hello, world!
```

C 程序加工过程的启动方式由具体 C 语言系统确定，具体情况请查看有关的系统手册，有时需要直接用操作系统命令形式启动各种基本加工工作（启动编译系统、连接系统等）。具体过程是先用一个命令要求编译源程序，再用另一个命令做连接。其中除了要把源程序文件名作为命令参数外，还常常需要输入一些其他参数。这些命令的书写形式比较复杂，使用不太方便，此外，为了输入、编辑和修改程序，还需要另用一个编辑系统。

## 1.4.2　程序格式

实际的 C 程序可能比前面的简单例子长得多。一般来说，一个 C 程序是由一系列可打印（可显示）字符构成的，人们一般用普通编辑器或者用专门的程序开发系统写程序、修改程序。组成程序的字符序列通常按照人阅读的习惯被分为一些行（就是在字符序列中插进一些换行符），每行长度不必相同。注意，上面把花括号内的部分看做下一层次内容后退几格写出，就是希望程序的表面形式能较好反映程序的内部层次结构。

C 语言是一种“自由格式”语言，除了若干简单限制外，写程序的人完全可以根据自己的想法和需要选择程序格式，选择在哪里换行，在哪里增加空格等。这些格式变化并不影响程序的意义。没规定程序格式并不说明格式不重要。程序的一个重要作用是给人看的，首先是写程序的人自己要看。对于阅读而言，程序格式非常重要。在多年的程序设计实践中，人们在这方面取得了统一认识：由于程序可能很长，结构可能很复杂，因此程序必须采用良好的格式写出，所用格式应能很好体现程序的层次结构，反映各个部分间的关系。

关于程序格式，人们普遍采用的方式是：① 在程序里适当加入空行，分隔程序中处于同一层次的不同部分；② 同层次不同部分对齐排列，下一层次的内容通过适当退格（在一行开始加空格），使程序结构更清晰；③ 在程序里增加一些说明性信息，这方面情况将在后面介绍。上面程序例子的书写形式就符合这些要求。

开始学习程序设计时就应养成注意程序格式的习惯。虽然对开始的小程序，采用良好格式的优势并不明显，但对稍大一点的程序，情况就不一样了。有人为了方便，根本不关心程

序的格式，想的只是少输入几个空格或换行，这样做结果是使自己在随后的程序调试检查中遇到更多麻烦。目前多数程序设计语言（包括C语言）都是自由格式语言，这就使人能够方便地根据自己的需要和习惯写出具有良好格式的程序来。

## 1.5 程序开发过程

本节讨论程序开发过程，包括程序调试（Testing）和排除错误（简称排误，Debugging）等方面的问题。调试和排误是程序实现的必经阶段。在读者刚开始学习程序设计时，下面讨论的一些情况可能难以完全明白，因为还缺乏程序设计实践，但是这些问题确实需要说明。我们把有关讨论集中在这里，希望读者能在学习了后面章节、做了些程序后再回来重读这些说明，这样反复几次就能弄清楚了。

### 1.5.1 程序的开发过程

用计算机解决问题的过程可以用图1–4描述，这种过程大致如下。

（1）分析问题，设计一种解决问题的途径。

（2）根据所设想的解决方案，用编辑系统（或IDE）建立程序。

（3）用编译程序对源程序进行编译。正确完成就进入下一步；如发现错误，就需要设法确定错误，返回到第2步去修改程序。

（4）反复工作直到编译能正确完成，编译中发现的错误都已排除，所有警告信息都已处理（一些排除，其余已弄清不是错误），这时就可以做程序连接了。如果连接发现错误，就需返回前面步骤，修改程序后重新编译。

（5）正常连接产生了可执行程序后，就可以开始程序的调试执行了。此时需要用一些实际数据来考查程序的执行效果。如果执行中出了问题或发现结果不正确，那么就要设法确定错误原因，返回到前面步骤：修改程序，重新编译，重新连接，等等。

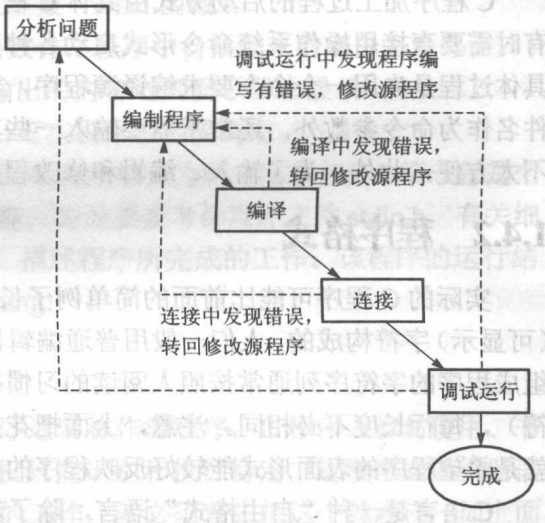


图1–4 调试运行中发现问题分析本身有错误，重新分析问题

### 1.5.2 程序错误

关于排除程序错误的术语Debugging还有一个故事。在美国计算机发展早期，有一天一台计算机出故障不能运行了。经仔细检查，人们发现计算机里有一个被电流烧焦的小虫（bug），它造成了电路短路，是这次故障的祸根。从此，检查排除计算机故障的工作就被称为Debugging，就是“找虫子”。后来人们也这样看待和称呼检查程序错误的工作。

实际上，对程序设计而言这个词并不贴切。因为程序中的错误都是编程者所犯的错误，并没有其他客观原因，也没有虫子之类的小东西捣乱，学习程序设计首先应该认清这一情况。