李学武 编著 吴文虎 审

中学生学C语言

清华大学出版社
北京

序

就我在大学教书的体会，对于计算机知识和技能，“学过一点”和“一点没学”大不一样，特别是涉及程序设计的知识。有关计算机语言的学习被许多中学生视为畏途，原因是许多书不是给中学生写的，既引发不了兴趣，学后又不会用。戴着近视眼镜的先生们曾说，现成软件那么多，又好学又能解决问题，在中学教程序设计是误区。前几年有些外国专家也如是说。听听本书的作者李学武教授怎么说：“不会编程的人，只能局限在别人划定的圈内工作，对于那些划在圈外，计算机能够胜任，而你又渴望去做的工作，不会编程的人只能‘望圈兴叹’。”李老师语重心长地说“不能让我们的孩子输在起跑线上”。我赞赏李老师的远见卓识，更愿意加上一句话：“我们有责任让那些有理想有抱负有灵气又肯吃苦的中国孩子在起跑时就能占有先机。”

李老师花了许多精力研究怎样让C语言的教学既能引发兴趣又能深入浅出，他将自己多年的教学体验，特别是辅导信息学奥林匹克选手的经验融入这本教材中，自然会达到有的放矢、学以致用的效果。

也许有的小朋友会说：“C语言这么难，我能学会吗？”我的逻辑是：“和你同样年龄同样水平的人能学会，你肯定也能学会。”不过我要告诉你：“化难为易的金钥匙掌握在你自己手里——上机动手做”。就像学英语那样，光看语法不行，要大声说，大声练。学习C语言要上机编写程序，越编就会越有信心、有兴趣和有成就感，实际上参加信息学奥赛的那些高手们在起步的时候都是这样做的。

青少年是国家的希望，不断提高青少年的科学素养是中华民族永远昂首屹立在世界东方的根基所在。“精心育桃李，切望青胜蓝”是我和李学武老师的共同心愿。

国际信息学奥林匹克中国队总教练
清华大学计算机系教授博士生导师
吴文虎

2004年7月4日

前言

显然，这是一本为中学生写的书。笔者深知，写一本真正让中学生喜欢的专业性较强的书，并不是一件容易的事情。它的内容应该通俗易懂而又不失于肤浅，它的文字应该生动有趣而又能准确地表述科学的内涵，这些是笔者一直在努力追求的目标；至于能不能达到或接近这一目标，笔者只能诚惶诚恐地等待读者的检验。衷心希望本书中一段又一段的C语言代码，在读者阅读本书之后，不再是一堆枯燥乏味的符号，而是能激起读者深造欲望和创造灵感的火花。

C语言是一种在计算机上使用的程序设计语言。当今我国大多数的中学生都使用过计算机，虽然部分地区限于条件，目前还没有在中学开设计算机课程，但这很快会成为历史；因为21世纪的竞争，在很大程度上是围绕着信息技术的广泛应用与迅猛发展而展开的，“不能让我们的孩子输在起跑线上”已经成为人们的共识。国家早已制定并正在执行着一个全面普及信息技术教育的宏伟计划。这里要指出：“使用计算机”与“计算机程序设计”是两个完全不同的概念。计算机是按照给定的指令完成各种操作的，而程序是一系列指令的集合，计算机的使用是离不开程序的。一些不会程序设计的人，照样也能用计算机完成许多重要的工作，这又怎么解释呢？其实很简单，这实际上反映了计算机发展的一个重要原则：用越来越简单的操作，完成越来越复杂的任务，惟有这样，计算机才能走进千家万户，走进社会各个角落。而连接“简单的操作”与“复杂的任务”的桥梁，是一个通常称为“软件”的很复杂的程序组，只不过它是由专业人员而不是用户编写的。不会编程的人，只能局限在别人划定的圈内工作，对那些划在圈外、计算机能够胜任而你又渴望去做的工作，不会编程的人只能“望圈兴叹”。我们提倡中学生学习一些程序设计语言，就是为了让我国青少年在中学阶段就打下良好的基础，以便将来能驰骋在计算机应用技术的自由王国之中，成为各行各业的栋梁之材。

从20世纪90年代初开始，C语言就不再是开发软件的主要工具，人们已经研制了多种适用于不同需要的功能强大的开发工具，如Visual Basic、Visual C++ 、Delphi、Java、微软公司的.net系列，以及各种用于数据库系统的开发工具和其他许多专业性较强的开发工具。这些开发工具的广泛使用，使人们驾驭计算机的能力得到了空前的提高。面对这种情况，笔者以为，中学生学习C语言仍然是很有必要的。因为任何一个实用软件的开发，都有很多具体实现的细节，这些细节具有某种特殊性，靠开发工具提供的功能是无法解决的，必须要靠研制人员自己去设计算法和编写代码，而没有C语言程序设计的功底是难以胜任这些任务的。同时，笔者也不主张每个中学生都花费很多精力学习C语言，

中学是人生的重要阶段，要打好各方面的基础，学习的任务相当繁重，学会用C语言解决一些简单的问题就可以了。对于基础较好的同学，则应学得更全面、更深入一些，力图能解决竞赛级别的一些难题，这方面的学习和训练，将会受益终身。

在本书写作过程中，时常困扰笔者的一个问题是：如何在叙述的简明性与严谨性之间进行取舍。当然应该努力追求二者的统一，但限于笔者的学识和具体问题的复杂程度，二者的冲突是难以避免的。对此，笔者一般这样处理：如果忽略一些严格性的描述不至于从主流上影响读者的理解，那么就以简明的描述为主，否则就必须做严格的描述。另外，细心的读者（特别是计算机专业的教师）可能会发现本书关于C语言语法规定的某些叙述与流行的同类书籍中的描述不完全一致，对此，可能有这样几种情况：首先，本书尽量遵循经典的教科书中的有关描述，主要依据B. W. Kernighan与D. M. Ritchie合著的《C程序设计语言（第2版）》（有中译本，详见参考文献[1]）中的“附录A参考手册”的描述，它是按C语言的最后一个标准“ANSI C X3.159-1989”（以后简称为“ANSI C 89”）编写的。例如，目前许多C语言教材中都介绍“浮点数一律按双精度数进行运算”，这是C语言早期的规定，在“ANSI C 89”中已经做了本质性的修改。其次，本书的所有算例都是在Turbo C 2.0环境下运行的，该系统有少量处理不完全符合“ANSI C 89”，况且，很多处理方式在ANSI标准中并没有明确的规定，不同的系统采用了不同的处理，自然也导致了不同的解释。第三，为了更通俗易懂，在不会产生实质性错误的前提下，有些语法现象没有给出严格的描述。最后，当然不能排除笔者在理解上的失误，对此，笔者诚恳地希望得到读者的批评指正。

我国著名计算机专家、计算机教育家吴文虎教授为本书的顺利出版发挥了至关重要的作用，吴先生在极其繁忙的工作之余认真地审阅了本书全稿，并热心地为本书写了序。相信吴先生的指导和帮助必将使本书能够吸引更广泛的读者。笔者对吴先生表示万分感谢。

我院杨彬同学认真地在文字和内容上对本书初稿进行了校对，并对笔者编写和调试过的程序实例逐一上机复核，提出了许多重要的修改意见。在此，笔者对杨彬同学的辛勤付出表示衷心的感谢。此外，作者还感谢李斌同志为本书精心绘制了全部插图。

李学武

2004年4月

目录

第1篇 入 门 篇

第2篇 提 高 篇

第3篇 应 用 篇

第1篇

入门篇

学习C语言从这里起步

作为教师，笔者曾为不同程度不同要求的学生讲过C语言，一次在一个短训班即将结束的时候，一名学生突然发问："老师，您讲了半天，C语言在计算机里到底有什么用？"听完这话，我不禁一愣，自然这个学生的这门课几乎是白上了。但他的话一语中的，道出了我在这次讲课中忽略了一个重要的事实：人们对计算机的理解是千差万别的，在许多人眼里，使用计算机，无非就是操纵鼠标，至多再用键盘输入几个汉字。对于我的教学，这种忽略无疑是一个致命的错误(Fatal error! 这是在运行C程序时最忌讳出现的一种计算机系统的提示)。

在此，我先不忙于直接回答这个学生的问题。大家先看一段C语言的程序。

例 1-1

```
/* 程序 1-1 */
#include <stdio.h>
main()
{printf ("Hello,My Computer! \n");
}
```

你可能会问：什么是程序？例1-1中的程序(即程序1-1)要做些什么？

当我们要让计算机完成某些操作时，都要先给它下达命令，平时我们点击鼠标、敲键盘，广义上讲，都是在下命令。这种命令要通过事先安装在计算机内部的软件完成既定的操作。然而在许多情况下，我们要让计算机完成的操作并没有相应的支撑软件。例如你突然对圆周率π发生了兴趣，想看一下π的前一万位小数，对于目前的计算机，这只不过是小事一桩，但是如果你的计算机里没有π的计算程序，也没有保存π的一万位小数的数据，而且也不知道怎样在因特网上搜索，要想达到自己的目的，就必须直接对计算机下达一系列正确的命令。这种命令通常被称为指令。一台计算机所能执行的全部指令的集合，称为指令系统；而可以被计算机处理的指令序列就是程序。但是，直接书写计算机指令程序是一件十分烦琐的事情，为此，人们通常采用其他程序设计语言间接地为计算机下达命令，例如汇编语言、各种高级语言和第4代语言等。C语言是一种高级计算机程序设计语言(有些书称之为中级语言，因为它可以直接完成类似于汇编语言的一些操作)，用C语言编写的程序(即命令序列)要被C语言编译系统(也叫做C编译器)转换成计算机能直接接受的指令才能执行。和其他高级语言(如FORTRAN语言，PASCAL语言，BASIC语言等)一样，用C语言书写的程序接近于自然语言和数学上习惯的表达方式，因

而更容易学习和使用。C语言是1973年由Dennis Ritchie在Ken Thompson和Martin Richars开发的BCPL语言的基础上研制的，经过30年实践的考验，C语言已成为当今广泛流行的一种计算机高级语言。

程序设计是根据所提出的任务，用某种程序设计语言编制一个能正确完成该任务的计算机程序。诚然，不是每一个使用计算机的人都要学习程序设计，但也不是只有计算机专家才需要掌握程序设计，不论将来你从事什么职业，精通程序设计必将使你出类拔萃。

下面回过头来看程序1-1。这是一段简单的C语言的程序，执行结果是在屏幕上显示一串字符：

```
Hello,My Computer!
```

希望这个程序能成为初学者运行的第一个C语言程序。为此要做以下几件事情：

(1) 你的计算机应事先安装一个C语言开发环境。例如Turbo C或rhide C，这两个软件在因特网上都很容易找到，它们都是具有多种功能的集成开发环境(integrated development environment，IDE)，其核心是C语言编译系统。如果你准备参加信息学程序设计竞赛，最好使用后者，因为它可以充分利用当前的计算机系统资源，rhide C有for Windows和For Linux两个版本。但Turbo C目前在教学中使用更广泛。本书的所有实例都是在Turbo C环境下通过的。本书附录A介绍了Turbo C的具体操作方法，在上机之前，应该认真阅读附录A。

(2) 在IDE的编辑窗口(Edit)键入程序1-1的4行程序，书写格式不要随意改动。

(3) 将刚输入的程序进行编译(选Compile项)，如果报错，要认真核对修改，直到编译通过为止。

(4) 将程序文件存盘(选Save项)，并指定程序文件名，例如lx01.C，C语言程序文件的扩展名一律规定为C。

(5) 运行该程序(选run项)，然后进入用户屏幕(User Screen)，就会看到预期的结果。

程序1-1虽然简单，但它是一个完整的C语言程序。每一个C程序，不论大小，都是由若干个函数和变量组成的。函数是一个相对独立的程序模块，其中包含若干语句，这些语句指定了让计算机进行的某种操作。而变量则用于在计算过程中存储有关的值。在程序1-1中，只有一个名字为main的函数(一般称为主函数)，main是一个特殊的函数名，每一个程序都从名为main的函数的第一条可执行语句开始执行。这就是说，每个程序都必须包含一个main函数，使用还是不使用其他函数则根据需要而定。

函数的一般结构如下：

```
函数名(参数表)
{语句1
语句2
      ⋮
语句k
}
```

除了main和标准库函数以外，其他函数名都是根据个人需要选取的，表示函数开始

和结束的一对花括号{}是必不可少的。参数表可以为空，这时圆括号()不能省略。第7章将对函数的定义和使用给出详细的说明。

程序 1-1 的第 1 行

```
/ * 程序 1-1 * /
```

是一个注释，在 C 语言中，写在一对标志符号"/ * "与" * /"内部的字符都属于注释部分。计算机并不理睬注释的内容，但注释对于程序设计人员却很重要，有助于对程序的理解。以后在讨论程序设计风格时，还要详细讨论注释的作用。

程序 1-1 的第 2 行

```
#include <stdio.h>
```

是固定的用法，建议初学者的每个程序都以它作为第 1 行。它的作用是告诉编译程序在本程序中要用到标准输入输出库的有关信息。

程序 1-1 的第 4 行

```
printf("Hello,My Computer! \n");
```

是一条函数表达式语句。其中 printf()是个标准库函数，f 是 format(格式)的缩写，故 printf()也称为格式打印函数，其功能是在输出设备(例如屏幕)上显示一系列字符，本例中显示的是"Hello,My Computer!"，这些字符要放在一对双引号之间，但在屏幕上并不显示两端的双引号。"\n"在 C 语言中表示一个字符，即换行符，在打印时它用于指示从下一行的左边继续打印。printf()函数加上分号";"便构成了函数表达式语句。但通常不把它说成是 print 语句或打印语句。与其他高级语言不同，C 语言没有定义自己的用于输入或输出的语句。关于函数 printf()的使用方法，在第 3 章再详细介绍。

由程序 1-1 可知，C 语言程序是由若干个函数和变量组成的，每个函数包含了若干条语句。一般地，一行只写一条语句，但这不是必须的，C 语言程序的书写格式是比较随意的。有时一行也可以写多条语句，有时一条语句也可以分写在不同的行上。但也不是在任何地方都可以断开换行，例如，如果将程序 1-1 的第 4 行改写为：

```
printf("Hello,
  My Computer! \n");
```

或

```
printf("Hello,My Computer!
");
```

C 编译器都会报错。原因在第 3 章再讲。

那么，应该提倡什么样的书写格式呢？本书以后再给出一些建议。在此，仅请大家记住：程序的书写格式是衡量程序质量的重要标准之一。因而，养成良好的书写程序的习惯是至关重要的。

为了让大家了解更多的 C 语言的元素，再看一个例子。

例 1-2

```
/*程序 1-2 */
#include <stdio.h>
main()
{int a,b,c,d;
 printf ("Input a,b:\n");
 scanf("%d%d",&a,&b);
 c=a+b;
 d=a*b;
 printf("a+b=%d,a*b=%d \n",c,d);
}
```

程序 1-2 的功能是先由键盘输入两个整数 a 和 b，然后计算这两个数的和与积，最后在屏幕上输出计算结果。为了运行这个程序，要和程序 1-1 一样，完成键入程序、编译、存盘几个环节，在程序运行阶段(run)，屏幕上将显示：

```
Input a,b:
```

见到这个提示后，用键盘输入两个整数，中间用空格隔开，再“回车”。例如：

```
30 40
```

这时，屏幕应显示：

```
a+b=70,a*b=1200
```

下面对程序 1-2 做一些说明。

程序 1-2 的第 4 行

```
int a,b,c,d;
```

是个类型说明语句。其功能是说明 a,b,c,d 是 4 个变量，它们用于存放整型数值。在 C 程序执行过程中用到的所有的“数”(我们暂且用这个词，实际上它比我们在数学计算意义下的数的含义要广泛得多)都要存放在内存单元中，有些数在程序运行期间是始终不变的，称为常量。有些数可能会改变(例如本例中的 a,b 在每次运行时可以输入不同的值)，这样的量称为变量。在 C 语言中，所有的变量都必须先说明后使用，说明通常放在函数开始处的可执行语句之前。其中，类型标识符 int 表示后面所列出的变量为整型变量。关于类型说明的目的和有关细节，将在第 2 章介绍。

程序 1-2 的第 6 行

```
scanf("%d%d",&a,&b);
```

与“printf();”类似，也是一条函数表达式语句。其中 scanf()是标准库函数，一般称为格式输入函数，其功能是从输入设备(例如键盘)上读入一些数据，并存放到指定的变量之中。其中“%d%d”称为格式说明符，它指定数据要按整型格式输入。而“%d%d”表示要输入两个整数。“&a,&b”表示要用变量 a,b 存放输入的数据，前缀“&”是必须的，不能省

略。同样，通常把第 6 行称为函数表达式语句，而不是输入语句。

程序 1-2 的第 5 行

```
printf ( "Input a,b:\n" );
```

是为下一行的输入做提示的。在任何输入之前，都应该在屏幕上有所提示，否则面对屏幕上的输入光标，往往不知道要做些什么。尽管 scanf()函数本身也具有提示的功能，例如将两行合并为：

```
scanf("Input a,b:\n%d%d",&a,&b);
```

仍可实现原有的功能，但在具体输入数据时，输入格式很容易出错，从而数据不能正确地读入，不提倡这种程序设计风格。建议初学者尽量采用输入提示与实际输入分开书写的方式。

程序 1-2 的第 9 行

```
printf("a+b=%d,a*b=%d \n",c,d);
```

中的两个“%d”与 scanf()函数中的用法类似，都是格式说明符，表示后面的两个数 c,d 要按整型格式输出。

程序 1-2 的第 7、8 两行

```
c=a+b;
d=a*b;
```

是赋值表达式语句。“c=a+b”是个赋值表达式，符号“=”称为赋值运算符，也叫做赋值号。它不同于数学中的等号，“c=a+b”也不是数学中的等式。尽管作为高级语言，C 语言采用了许多数学上习惯的表述方式，但 C 语言程序绝不是一段数学公式的推导，千万不要混淆。在 C 语言程序中，“c=a+b”表示先将变量 a,b 的值取出来，求和，然后将结果放入变量 c 中。这个过程称为赋值。为了更形象一些，在你的脑海里，可把“c=a+b”转换为“c⇐a+b”，但千万不要在程序中也这样转换，程序设计语言中的语法规定是十分严格的，往往一个符号的失误就要导致满盘皆输。

程序 1-2 还给出了多数实用程序的一个框架：

(1) 变量说明；
(2) 输入数据；
(3) 计算；
(4) 输出结果。

当然这些并不是必须的。但一个正确的程序，总要有结果输出，以满足人们的某种需要，否则何必要运行它呢？对于输出的结果，要有广义的理解，许多情况下，是显示或打印一些数、字符或图像；但有时是其他的形式，例如播放一段音乐或执行计算机的某种操作，甚至看不出有什么反应，仅仅是为了延迟一段时间，等等。

还有许多编程方法可以实现程序 1-2 同样的功能，例如：

```
#include <stdio.h>
```

```
main()
{int a,b;
 printf ( "Input a,b:\n" );
 scanf("%d%d",&a,&b);
 printf("a+b=%d,a*b=%d \n",a+b,a*b);
}
```

或许比程序 1-2 更精练一些。

这里，涉及到一个程序质量与程序设计风格的问题。设计风格与前面提到的程序书写格式是两个不同的问题。对于初学者，详细讨论这个问题现在为时尚早，这里只强调一点：在保证程序正确的前提下，程序应该是容易阅读和容易理解的，也就是说，程序的可读性应作为衡量程序质量的首要标准。

学习程序设计必须理论与实践并重。理论上的学习使我们了解必须严格遵守的各种语法上的规定，通过范例学习规范的程序设计方法，学习前人创造的大量的程序设计技巧。而实践的环节更为重要。在此，我们把程序设计与数学解题的过程对照一下。为了解决一个数学问题，例如比较难的数学竞赛题，首先要有必要的数学基础知识，同时还要有一个正确的解题思路，这是靠平时的学习和训练培养出的一种能力。这个过程基本上是单向的，即只要本人具有良好的素质并付出相应的努力就可以获得成功。而程序设计的过程是双向的，除了需要个人的素质和努力，还需要能正确地与计算机沟通，因为你的任务是要让计算机去做一些事情，而不是你替计算机做一些事情。让计算机不折不扣地听从你的安排并不是一件容易的事情。这种沟通只有靠大量的不断的上机实践才能实现。这与学游泳必须下水是一个道理，我们过去即使在经济比较困难的情况下，也始终反对程序设计的无机教学，因为那是纸上谈兵学不到程序设计的精髓。

下面一段对话在我辅导学生上机的经历中不时地重演：

学生："老师，我的程序一点都没错，可就是结果不对，怎么回事？"

笔者："结果不对就一定有错。"

学生："谁的错？是不是机器工作不正常？"

笔者："错误出在机器上的概率不到百分之一，出在你的程序中的概率不低于百分之九十九。"

学生："可是……"

上机实践除了编写和录入程序，更重要的是调试程序，即修正程序中的错误，提高程序的质量，这是提高程序设计能力的必经之路。出错是常有的事情，应把它当作一件好事，因为改错的过程就是提高自己分析能力和编程能力的过程。当我们的程序不能达到预期的目的时，可能会有以下一些原因：

(1) 程序中的语法错误。

(2) 输入的原始数据有错误，特别是输入格式不正确。

(3) 程序的逻辑结构有错，或算法上有错。

(4) 程序是正确的，但计算机软件或硬件环境存在故障。

语法错误是初学者最容易犯的错误，这类错误是指程序中存在着违反语法规定的问

题，这时 C 编译器会指出出错的位置及错误的性质供你参考。要认真分析这些信息，把错误找准，并加以纠正。这类错误一般会随着经验的积累而逐步减少。系统给出的一般是英文的错误提示信息，应认真弄清每个英文提示的含义，这对于提高调试程序的能力至关重要，附录 A 中给出了部分提示的中英文对照。对于第 2、第 3 类错误，计算机一般没有什么提示，我们在后面将给出一些解决办法，但这些方法都不是万能的，解决这两类特别是第 3 类错误的根本出路在于综合素质的提高。至于第 4 类情况则是很少发生的，因此，不要一出现错误就归咎于计算机。有一个很简单的办法可检验是不是计算机的故障：把程序放到另一台计算机上再试一下，因为两台计算机出现完全相同故障的情况毕竟是很少见的。

习　题

1. 编程分 4 行打印以下一段文字：

C is a general-purpose programming language. It has been closely associated with the UNIX system where it was developed, since both the system and most of the programs that run on it are written in C.

2. 编程完成下面的工作：由键盘输入一个圆的半径，然后输出圆的面积与周长。圆周率 π 取为 3.1416。

提示：对照程序 1-2，这里的数（如半径、面积、周长等）不再是整数，这时，格式说明符应使用“%f”，不能用“%d”，说明变量时，例如半径 r，应使用“float r”，不能用“int r”。