全国高职高专规划教材·计算机系列

C YUYAN CHENGXU SHEJI SHIYAN JIAOCHENG

C语言程序设计实验教程

陈　欣◎主编

北京大学出版社
PEKING UNIVERSITY PRESS

全国高职高专规划教材·计算机系列

C语言程序设计实验教程

陈　欣　主　编
张洪瑞　副主编

北京大学出版社
PEKING UNIVERSITY PRESS

内 容 简 介

本书是C语言程序设计的辅助教材，通过大量的编程实践，培养读者的程序设计能力。本书共分两大部分，第一部分是C语言的实验指导，第二部分是C语言的实验内容。实验指导部分简单总结了上机的指导思想和要求、程序的调试和测试方法以及常见的错误和常见编译的出错信息；实验内容部分共包括15个实验，涵盖了C语言程序设计的主要内容，每个实验主要包括：读程序写出运行结果、补充程序使其能正确运行、改错和综合程序设计4个任务。程序由易到难，循序渐进地引导学生理解程序的语法和算法的思想，从而习惯C语言编程的要求，掌握C语言的基本知识点。

本书在编写上力求实用，让读者可直接在上机指导上填写实验的结果，方便老师和学生使用。本书适合作为高职高专院校计算机、通信、电子信息等专业的基础课辅导教材，也可作为编程人员的参考用书。

图书在版编目(CIP)数据

C语言程序设计实验教程/陈欣主编. —北京：北京大学出版社，2012.1
（全国高职高专规划教材·计算机系列）
ISBN 978-7-301-20025-4

Ⅰ.①C… Ⅱ.①陈… Ⅲ.①C语言－程序设计－高等职业教育－教材 Ⅳ.①TP312

中国版本图书馆CIP数据核字（2011）第278055号

书　　名：C语言程序设计实验教程
著作责任者：陈　欣　主编
策 划 编 辑：温丹丹
责 任 编 辑：温丹丹
标 准 书 号：ISBN 978-7-301-20025-4/TP·1209
出 版 发 行：北京大学出版社
地　　址：北京市海淀区成府路205号　100871
电　　话：邮购部62752015　发行部62750672　编辑部62765126　出版部62754962
网　　址：http://www.pup.cn
电 子 信 箱：zyjy@pup.cn
印　刷　者：山东省高唐印刷有限责任公司
经　销　者：新华书店
787毫米×1092毫米　16开本　12印张　292千字
2012年1月第1版　2012年1月第1次印刷
定　　价：24.00元

前　言

C 语言是一种功能强大、运用灵活的程序设计语言，在我国的高等学校中普遍开设了 C 语言课程。学习 C 语言的目的不只是单纯地掌握一门程序设计语言，而是要学会编写程序的方法。要学好 C 语言，不仅要学习它的基本概念、方法、语法规则，更重要的是要进行实践，通过实践，读者可以逐步积累编程经验，培养编程能力。为此，我们编写了本书。

学习程序语言的主要方法就是实践。本书就是为 C 语言程序设计的学习者提供了一本实践指导教程，通过介绍 C 语言的基础知识点和形式多样的上机实验，引导读者在读懂程序、分析程序的基础上，可以独立编写程序。

本书共分两大部分，第一部分是 C 语言的实验指导，第二部分是 C 语言实验内容。

（1）实验指导

实验指导包括：上机的指导思想和要求、程序的调试和测试方法以及常见的错误和常见编译出错信息 3 方面内容。

（2）实验内容

程序设计是一门实践性很强的课程，在学习这门课程的过程中离不开上机实验。本书充分考虑了实验个数、题目难易程度、知识分布情况等教学要求，一共设计了 15 个实验内容，每个实验主要包括：读程序写出运行结果、补充程序使其能正确运行、改错和综合程序设计 4 个任务。其中，每一个任务中包含有多个程序，程序由易到难，循序渐进地引导读者理解程序的语法和算法的思想，从而习惯 C 语言编程的要求，掌握 C 语言的基本知识点，提高 C 语言编程的能力。

本书由陈欣任主编，具体编写工作如下：实验指导、实验 4～11 和实验 14～15 由辽宁装备制造职业技术学院的陈欣编写，实验 1～3 和实验 12～13 由辽宁装备制造职业技术学院的张洪瑞编写。本书的全部程序在 Visual C ++ 6.0 下调试通过。

在本书的编写过程中，虽然经过多次的校对，但由于作者水平和精力有限，难免有不足之处，敬请广大读者批评指正。

编者

2012 年 1 月

目　录

第一部分　实验指导

上机的指导思想和要求

1. 上机实验的目的

对于“C 语言程序设计”这门课程的学习不能仅限于“懂了”，不能只满足于能读懂书上的程序而已，而是要熟练地掌握程序设计的全过程，包括独立完成源程序的编写，独立进行上机程序的调试，独立运行程序和分析结果。计算机的语言类课程的实践性很强，必须要十分重视实践环节，必须保证有足够的上机实践时间。学习“C 语言程序设计”课程应该至少要有 27 小时的上机时间，在课程学时分配上最好能做到上机学时与理论课学时比例为 1∶1。同时，除了课程安排的上机实验学时以外，学生自己应该在课余时间多多进行上机实践。

上机实验不仅仅是为了验证教材和讲课的内容，或者验证学生自己编写的程序是否正确，而是有以下几个原因。

（1）让学生加深对授课内容的理解，尤其是一些语法的规则。学生在理论课堂上通过教师讲授的方式来学习这些语法规则，既显得枯燥无味，又难以记住，但是它们在程序设计中又是很重要的部分。解决的方法之一是让学生通过多次上机反复练习，自然而然地、熟练地掌握这些语法规则。

（2）让学生熟悉 C 语言程序开发的环境。一个程序必须在一定的外部环境下才能运行，所谓“环境”，就是指所用的计算机系统的硬件和软件条件。学生应该了解为了运行 C 语言程序需要哪些必要的外部条件（例如硬件配置和软件配置），可以利用哪些系统的功能来帮助自己开发程序。每一种计算机系统的功能和操作方法并不完全相同，但只要学生熟练掌握一两种计算机系统的使用，再遇到其他的系统时便会触类旁通，很快就能学会。

（3）让学生熟练掌握上机调试程序的方法。也就是善于发现程序中的错误，并且能很快地排除这些错误，使程序可以正确运行。学生要学会根据编译过程中出现的“出错信息”，分析并找出错误所在，并能正确改正。要真正掌握计算机应用技术，不仅要了解和熟悉有关的理论和方法，还要自己亲自动手实现。对于程序设计人员来说，要会编写程序并能上机调试通过。因此调试程序也是程序设计课程的一个重要的内容，要给予充分的重视。调试程序可以借鉴他人的现成经验，但通过自己亲自实践来积累经验更为重要。调试程序的能力是每一个程序设计人员应当掌握的一项基本功。因此，在做实验时，千万不要在程序编译过程中没有出现错误和警告就认为万事大吉、完成任务了，而应当在已通过的程序基础上做一些改动（例如修改一些参数、增加程序的一些功能、改变输入数据的方法等），再进行编译、连接和运行；甚至于“自设障碍”，也就是把正确的程序改为有错的（例如，用 scanf 函数输入变量时，漏写“&”符号；使数组下标出界；使整数溢出等），以便观察和分析所出现的情况。这样的学习方法才会有真正的意义，才是灵活主动的学

习，而不是呆板被动的学习。

2. 上机实验前的准备工作

学生在上机实验之前要事先做好准备工作，这样可以提高上机实验的效率，准备工作包括以下几方面。

（1）了解上机实验所要用的计算机系统（包括 C 编译系统）的性能和使用方法。

（2）复习和掌握与本次上机实验有关的教学内容。

（3）准备好上机实验所需的程序。如果是手编程序，则要书写整齐，并且自己检查无误后才能上机，这样可以提高上机的效率。

注意：对于初学者来说，切忌在上机之前不编写程序，或者照抄他人的程序直接上机实验。

（4）对于自己编写的程序在运行中可能出现的问题要事先做出预估；并且对程序中自己有疑问的地方，应做上标记，以便在上机时予以注意。

（5）准备好调试和运行时所需要的数据。

3. 上机实验的步骤

上机实验时应该做到一人一组，独立上机。上机过程中出现的问题，除了是系统的问题以外，一般应自己独立处理，不要轻易举手问老师。尤其对“出错信息”应善于自己分析判断，这是学习调试程序的良好机会。

上机实验一般要包括以下几个步骤。

（1）调出 C 编译系统，进入 C 工作环境（例如 Visual C ++ 6.0）。

（2）输入自己编写好的程序。

（3）检查一遍已输入的程序是否有错（包括输入时打错的和编程当中的错误），如果发现有错误，则及对改正。

（4）进行编译。如果在编译和连接过程中发现错误，输出窗口会出现“出错信息”，根据提示找到出错位置和原因，加以改正，再进行编译、连接……如此反复，直到顺利通过编译和连接为止。

（5）运行程序，并分析运行结果是否合理和正确。在运行时要注意当输入不同的数据时所得到的结果是否正确（例如，在求解一元二次方程 $ax^2 + bx + c = 0$ 的根时，不同的 a、b、c 组合应该得到相应的不同结果）。此时应多运行几次，分别检查在不同情况下程序是否正确。

（6）输出程序清单和运行结果。

4. 写实验报告

实验报告应包括以下内容。

（1）题目。

（2）实验目的，实验设备。

（3）程序清单（编写的源程序）。

（4）实验数据（从标准输入设备上输入的验证数据）。

（5）实验过程报告。

（6）运行结果（必须是上面程序清单所对应的输出结果）。

（7）实验小结（对运行情况的分析，以及本次调试程序所取得的经验。如果此次程序

未通过，则应该分析未通过的原因）。

5. 实验内容的安排

结合所学内容布置上机练习题。本书给出 15 个实验内容，每个实验内容对应一个 C 语言程序设计的模块。教师可以根据实验的内容和上机时间的长短，指定全部或一部分的内容作为上机练习题。其中实验 15 是综合实验，让学生最后从整体上对 C 语言进行编写设计。

注意： 应该要求学生在实验之前将教师指定的题目编好程序，然后上机直接进行输入和调试。

关于程序的调试和测试方法

1. 程序错误的类型

为了帮助学生调试程序和分析程序，下面简单介绍程序出错的种类，主要有以下几种。

（1）语法错误。语法错误是指不符合 C 语言编写的语法规则。例如，将 scanf 错写为 scan，括弧不匹配，语句最后漏掉了分号等，这些都会在编译过程中被发现并指出。这些都属于“致命错误”，不改正就不能通过编译。对一些在语法上有轻微毛病但不影响程序运行的问题（例如在定义了变量之后始终未使用），编译时会发出“警告”。虽然程序能通过编译，但不应当使程序“带病工作”，应该将程序中所有导致“错误（error）”和“警告（warning）”的因素都排除掉，再使程序投入运行。

关于在编译过程中可能会出现的语法错误，将在后面具体介绍。

（2）逻辑错误。逻辑错误是指程序没有语法错误，也能正常运行，但是结果不对。例如，求 s = 1 + 2 + 3 + … + 100，有的人写出以下程序：

```
#include <stdio.h>
main()
{  int s,i;
   s = 0;i = 1;
   while(i < 100)
   {
      s = s + i;
      i ++;
   }
   printf("s = % d",s);
}
```

上面语句的语法并没有错，但是求出的结果却是 1 + 2 + 3 + … + 99 之和，而不是想要的 1 + 2 + 3 + …100 之和，查看程序后发现错误的原因是少执行了一次循环体。这类错误可能是设计算法时的错误，也可能是算法正确而在编写程序的时候出现疏忽所致。对于这种错误计算机是无法检查出来的。如果是算法有错，则应该先修改算法，再修改程序；如果是算法正确而程序编写的不对，则直接修改程序。

（3）运行错误。有时程序既没有语法错误，也没有逻辑错误，但程序仍不能正常运行或输出的结果不对。这种情况多数是数据不对，包括数据本身不合适以及数据类型不匹配。例如有以下程序：

```
#include <stdio.h>
main()
{  int  x,y,z;
   scanf("% d,% d",&x,&y);
   z=x/y;
   printf("% d\n",z);
}
```

程序的功能是求 x 整除 y 的结果并输出。在程序运行时，当输入的 y 为非 0 值时，运行没有问题；但是当输入的 y 为 0 时，程序运行时会出现“溢出（over-flow）”的错误。又比如在执行上面程序的 scanf 函数语句时输入如下数据：

56.78,3.4↙

则输出 z 的值为 16，显然是不对的。这是由于输入的数据类型与输入格式符“%d”不匹配而引起的。

2. 程序的测试

程序调试的任务是排除程序中的错误，使程序能够顺利地运行并得到预期的效果。程序的调试阶段不仅要发现和去除语法上的错误，还要发现和去除逻辑错误和运行错误。除了可以利用编译时提示的“出错信息”来发现和改正语法错误以外，还可以通过程序的测试来发现逻辑错误和运行错误。

程序测试的目的是尽量找出程序中可能存在的错误，检查程序有无“漏洞”。在程序测试时要设想到程序运行时会出现的各种情况，测试在各种情况下的运行结果是否正确。程序测试是程序调试的一个组成部分。

从上面的例子可以看出，有的时候程序在某些情况下能正常运行，但是在另外一些情况下不能正常运行或者是得不到正确的结果。因此，一个程序即使通过了编译并可以正常运行而且得到的结果也正确时，也不能认为程序就没有问题了。而是要考虑是否在任何的情况下程序都能正常运行，并且可以得到正确的结果。测试的任务就是要找出那些不能正常运行的情况和原因。下面举个例子进行说明。

在求一元二次方程 $ax^2+bx+c=0$ 的根时，根据数学求根公式：$x_{1,2}=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$

编写以下的程序：

```
#include <stdio.h>
#include <math.h>
main()
{
   double a,b,c,disc,x1,x2,p,q;
   printf("please input a,b,c:");
   scanf("% lf,% lf,% lf",&a,&b,&c);
   disc=b*b-4*a*c;
   p=-b/(2*a);
   q=sqrt(disc)/(2*a);
   x1=p+q;
   x2=p-q;
   printf("x1=% .2lf,x2=% .2lf\n",x1,x2);
}
```

当输入 a、b、c 的值为 1，-3.3，2 时，输出 x1=2.50，x2=0.80，结果是正确的；当输入 a、b、c 的值为 0，4，1 或输入 a、b、c 的值为 3，4，5 时，屏幕上出现的数据是

错误的，原因是作为分母 a 的值为 0，对负数求了平方根。

由此可以看出，上面的程序只适用于求 $b^2-4ac>0$ 和 $a\neq0$ 的情况。这样程序不能说是错误的，只能说程序考虑的情况不全面，程序不能在任何条件下都可以得到正确的结果。因为这个程序获得正确结果的前提是 $b^2-4ac>0$ 和 $a\neq0$，这样就给使用这个程序的人带来了不便。在输入数据之前，使用者必须要先计算一下 b^2-4ac 是否大于 0，同时 $a\neq0$。一个好的程序应该在各种情况下都能正常运行并得到相应的结果。

下面来分析一元二次方程 $ax^2+bx+c=0$ 的根。

（1）当 $a\neq0$ 的情况

① 当 $b^2-4ac>0$ 时，方程有两个不相等的实根

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

② 当 $b^2-4ac=0$ 时，方程有两个相等的实根

$$x_1=x_2=-\frac{b}{2a}$$

③ 当 $b^2-4ac<0$ 时，方程有两个不相等的共轭的复根

$$x_{1,2}=\frac{-b}{2a}\pm\frac{\sqrt{b^2-4ac}}{2a}$$

（2）当 $a=0$ 的情况

方程就变成了一元一次线性方程：$bx+c=0$。

① 当 $b\neq0$ 时，$x=-\frac{c}{b}$；

② 当 $b=0$ 时，方程变为：$c=0$。

当 $c=0$ 时，x 可以为任何值；

当 $c\neq0$ 时，方程没有解。

综合上面的分析，可以看出一元二次方程 $ax^2+bx+c=0$ 的根有以下 6 种情况：

① $a\neq0$，$b^2-4ac>0$；

② $a\neq0$，$b^2-4ac=0$；

③ $a\neq0$，$b^2-4ac<0$；

④ $a=0$，$b\neq0$；

⑤ $a=0$，$b=0$，$c=0$；

⑥ $a=0$，$b=0$，$c\neq0$。

测试程序时应当用以上 6 种情况分别进行测试，观察它们是否符合要求。因此，应该准备 6 组数据，用这 6 组数据去测试程序的“健壮性”。上面的程序显然只有满足了情况①和②才能使程序得到正确的结果，而如果输入满足情况③～⑥的数据时，程序会出现错误的结果。这就说明程序不“健壮”。因此，应当修改程序，使之能适应以上 6 种情况，修改代码如下。

```
#include<stdio.h>
#include<math.h>
main()
{
   double a,b,c,disc,p,q;
```

```
    printf("please input a,b,c:");
    scanf("% lf,% lf,% lf",&a,&b,&c);
    if (a ==0)
       if(b ==0)
            if(c ==0)
                 printf("解可以为任何值. \n");
            else
                 printf("方程没有解. \n");
       else
            {
                 printf("方程是一元一次方程,只有一个解: \n");
                 printf("x = % .2lf \n",-c/b);
            }
    else
    {
    disc = b * b - 4 * a * c;
    p = -b/(2 * a);
    q = sqrt(fabs(disc))/(2 * a);
       if(disc > =0)
            if(disc ==0)
            {
                 printf("方程有两相等的实根: \n");
                 printf("x1 = x2 = % .2f \n", -b/(2 * a));
            }
            else
            {
                 printf("方程有两个不相等的实根: \n");
                 printf("x1 = % .2lf,x2 = % .2lf \n",p + q,p - q);
            }
       else
       {
          printf("方程有两个不相等共轭的复根: \n");
          printf("x1 = % .2lf + % .2fi,x2 = % .2lf - % .2fi \n",p,q,p,q);
       }
    }
}
```

为了测试程序的“健壮性”，准备 6 组数据进行测试：

① 3，4，1 ② 2，4，2 ③ 5，3，2 ④ 0，2.2，4 ⑤ 0，0，0 ⑥ 0，0，3.5

分别用上面 6 组数据作为 a、b、c 的值输入，得到以下运行结果：

① please input a,b,c:3,4,1 ↙

方程有两个不相等的实根：

x1 = -0.33,x2 = -1.00

② please input a,b,c:2,4,2 ↙

方程有两相等的实根：

x1 = x2 = -1.00

③ please input a,b,c:5,3,2 ↙

方程有两个不相等共轭的复根：

x1 = -0.30 +0.56i,x2 = -0.30 -0.56i

④ please input a,b,c:0,2.2,4 ↙

方程是一元一次方程，只有一个解：

```
x = -1.82
```

⑤ please input a,b,c:0,0,0↙

解可以为任何值.

⑥ please input a,b,c:0,0,3.5↙

方程没有解.

经过上面的测试，可以看到程序对任何输入的数据都可以正常运行，并且得到正确的结果。

以上是根据数字知识知道输入数据有6种，但在有些情况下却没有现成的数学公式作为依据。例如，一个学生管理程序，要求对各种不同的检索做出相应的反应。如果程序中包含多条路径（比如用if语句写的程序），那么应当设计多组测试数据，使程序中每一条路径都有机会被执行，观察其运行是否正常。

以上是程序测试的初步知识。测试的关键是正确地准备好测试数据。上面的程序如果只准备4组数据，那么即使程序都能正常运行，也不能认为此程序没有问题。只有将程序运行时所有可能情况都做测试，才能做出程序没有问题的判断。

测试的目的是检索程序有无“漏洞”。对于一个简单的程序，要找出程序运行时全部可能执行到的路径，正确地准备数据并不困难。但是，如果要测试的是一个复杂的大程序，要找到全部可能的路径，准备出所需要的测试数据并不是件容易的事。例如，有两个非嵌套的if语句，每个if语句有2个分支，它们所形成的路径数目为$2\times2=4$；如果一个程序包含50个if语句，则可能的路径数目为$2^{50}=1.1259\times10^{15}$，而实际上进行测试的只是其中一部分。因此，经过测试的程序一般不能轻易说“没有问题”，而只能说：“经过测试的部分没有问题。”

学生应当了解测试的目的，学会组织测试数据，并根据测试的结果修改和完善程序。

常见错误和常见编译出错信息

1. 常见错误分析

下面列出初学者在使用C语言时常见的错误。

（1）变量未定义就使用。

```
main()
    {  a=4;b=5;
          printf("%d\n",a+b);
    }
```

（2）输入输出的数据类型与所用格式说明符不一致。

例如：

```
int a=3;float f=4.5;printf("%f,%d\n",a,f);
```

编译时不会出错，但运行结果与预期不符。

上面代码不是按照赋值的规则进行转换（如把4.5转换为4），而是将数据在存储单元中的形式按格式符的要求组织输出。输出结果为：0.000000，1074921472。

（3）输入变量时忘记使用地址运算符。

例如：

```
scanf("%d%d",x,y);
```

（4）输入时数据的组织与要求不符。

对 scanf 函数中格式字符串中除了格式说明符外，对其他字符必须按原样输入。例如：

```
scanf("% d,% d",&x,&y);
```

有人按下面的格式输入数据：

```
30 40↙
```

这是错误的。输入数据时应该用“,”做分隔。应该用下面的格式输入：

```
30,40↙
```

（5）误把“=”作为“等于”比较运算符。

在 C 语言中，“=”为赋值运算符，“==”为比较运算符。

（6）输入数据时，企图规定精度。

例如：

```
scanf("% 7.2f",&f);
```

不合法，输入数据不能规定精度。

（7）语句后面漏掉分号。

C 语言规定每一个语句末尾必须有分号。例如：

```
x=4
y=9
```

编译时，编译程序在“x=4”后面未发现分号，就把一下行“y=9”视为上一行语句的一部分，这样就出现语法错误。

（8）在不该加分号的地方加了分号。

例如：

```
if(a<b);
        printf("b is larger than a. \n");
```

上面代码段原意是当 a<b 的时输出“b is larger than a.”的信息。但由于在 if（a<b）后面加了分号，因此 if 语句到此结束，即当（a<b）为真时，执行的是一个空语句。这样无论 a<b 还是 a>b 都要输出“b is larger than a.”。

（9）对应该有“{ }”的复合语句，忘记加“{ }”。

例如：

```
sum=0;
    i=1;
 while(i<=100)
sum+=i;
i++;
```

上面代码段本意是要实现 1+2+3…+100，但上面的语句只是重复执行 sum+=i; 并且是死循环。

（10）括弧不配对。

例如：

```
while((c=getchar()=='#')
 putchar(c);
```

（11）书写标识符时，忽略了大小写的区别。

例如：

```
main()
```

```
{
   int x,y,z;
   x=2;
   y=3;
   Z=X+Y;
   printf("%d+%d=%D",X,Y,Z);
}
```

编译出错，编译时程序把 z 和 Z 作为两个不同的变量名。

（12）引用数组元素时下标运算符误用了圆括弧。

例如：

```
main()
{
   int i,a(10);
   for(i=0;i<10;i++)
    scanf("%d",&a(i));
}
```

（13）引用数组时，将数组定义的“元素个数”当成是可使用的最大下标值。

例如：

```
main()
{
   int a[5]={1,2,3,4,5};
   int i;
   for(i=1;i<=5;i++)
    printf("%d",a[i]);
}
```

这种错误编译时是不会提示的。

（14）对二维或多维数组的定义和引用的方法不对。

例如：

```
main()
{
   int a[3,5];
   …
   printf("%d",a[1+2,2+2]);
   …
}
```

对多维数组在定义和引用时必须将每一维的数据分别用“[ ]”括起来。上面的程序应该改为 a[3][5] 和 a[1+2][2+2]。

（15）误以为数组名代表数组中全部元素。

例如：

```
{  int a[4]={1,2,3,4};
   printf("%d%d%d%d",a);
}
```

在 printf 输出语句中企图用数组名代表整个数组元素。C 语言规定，数组名只能代表数组首地址，不能通过数组名输出所有数组元素。

（16）混淆字符数组与字符指针的区别。

例如：

```
main()
```

```
{
   char s [40];
   s = "Computer and c";
   printf("% s \n",s);
}
```

编译会出错。因为数组名 s 代表数组首地址，是一个常量，不能再被赋值。

（17）在引用指针变量之前没有对它赋予确定的值。

例如：

```
main()
{
   char *p;
   scanf("% s",p);
}
```

没有给指针变量 p 赋值就引用它，编译时给出警告信息。

（18）混淆字符和字符串的表示形式。

例如：

```
…
char sex;
sex = "M";
…
```

字符常量应该用单引号括起来，字符串常量用双引号括起来。

（19）switch 语句的各分支中漏写 break 语句。

例如：

```
switch(grade)
{
   case 'A':printf("优");
   case 'B':printf("良");
   case 'C':printf("中");
   case 'D':printf("及格");
   case 'E':printf("不及格");
}
```

上面代码执行后是想根据 grade，输出成绩的等级。但当 grade 的值为“A”时，输出为“优良中及格不及格”。

不过上述代码原意不是这样的，而是希望只输出一个成绩等级。此程序应改为：

```
switch(grade)
{
   case 'A':printf("优");break;
   case 'B':printf("良"); break;
   case 'C':printf("中"); break;
   case 'D':printf("及格"); break;
   case 'E':printf("不及格");
}
```

（20）自加（++）和自减（--）运算符使用时的错误。

例如：

```
main()
{
   int *p,a[5] = { 1,3,5,7,9 };
```

```
    p=a;
    printf("%d",*p++);
}
```

有的人认为 *p++ 的作用是先让 p+1，然后再进行间接访问，输出的是 a[1] 的值，但实际上是先用 p 的原值，再进行间接访问，输出的是 a[0] 的值。

(21) 所调用的函数在调用语句之后才定义，而又在调用前未加说明。

例如：

```
main()
{  float a,b,c;
   a=4.5;b=-8.6;
   c=max(a,b);
   printf("%f",z);
}
float max(float x,float y)
{
   return (x>y?x:y);
}
```

编译时会出错，因为 max 函数是在 main 函数之后定义的。

(22) 误认为形参值的改变会影响实参的值。

例如：

```
swap(int x,int y)
{  int t;
   t=x;x=y;y=t;
}
main()
{  int a,b;
   a=3;b=4;
   swap(a,b);
   printf("%d,%d\n",a,b);
}
```

原意是通过调用 swap 函数使 a 和 b 的值对换，然后在 main 函数中输出已对换后的值，但事实不是这样的。

(23) 函数的实参和形参类型不一致。

例如：

```
fun(float x,float y)
main()
{
   int a=3,b=4;
   c=fun(a,b);
   …
}
```

(24) 不同类型指针混用。

例如：

```
main()
{
   int i=3,*pi;
   float a=1.5,*pa;
   pi=&i;pa=&a;
```

```
    pa = pi;
    printf("% d,% d \n", * pi, * pa);
}
```

这个程序是使 pa 也指向 i，但 pa 是实型指针，不能指向整型变量。解决办法是进行强制转换。

（25）没有注意函数参数的求值顺序。

例如：

```
int i = 3;
printf("% d,% d,% d \n",i, ++i, ++i);
```

结果为 5，5，4。因为 Visual C ++ 6.0 是采取自右至左的顺序求函数的值，C 标准没有具体规定函数参数求值的顺序。

（26）混淆数组名与指针变量的区别。

例如：

```
main()
{  int i,a[10];
   for(i = 0;i < 10;i ++)
   scanf("% d",a ++);
}
```

上面程序是想通过数组名 a 的变化读取数组中的元素。这是错误的，应当用指针变量来指向数组元素，程序应改为：

```
main()
{  int a[10], * p;
   p = a;
   for(int i = 0;i < 10;i ++)
   scanf("% d",p ++)
}
```

（27）混淆结构体类型与结构体变量的区别。

例如：

```
struct student
{  long int num;
   char name[20];
   char sex;
   int age;
};
student.num = 187045;
strcpy(student.name,"ZhangFun");
student.sex = 'M';
student.age = 18;
```

这是错误的，不能对类型赋值只能对变量赋值。程序应改为：

```
struct student
{  long int num;
   char name[20];
   char sex;
   int age;
};
struct student stu;
stu.num = 187045;
strcpy(stu.name,"ZhangFun");
```