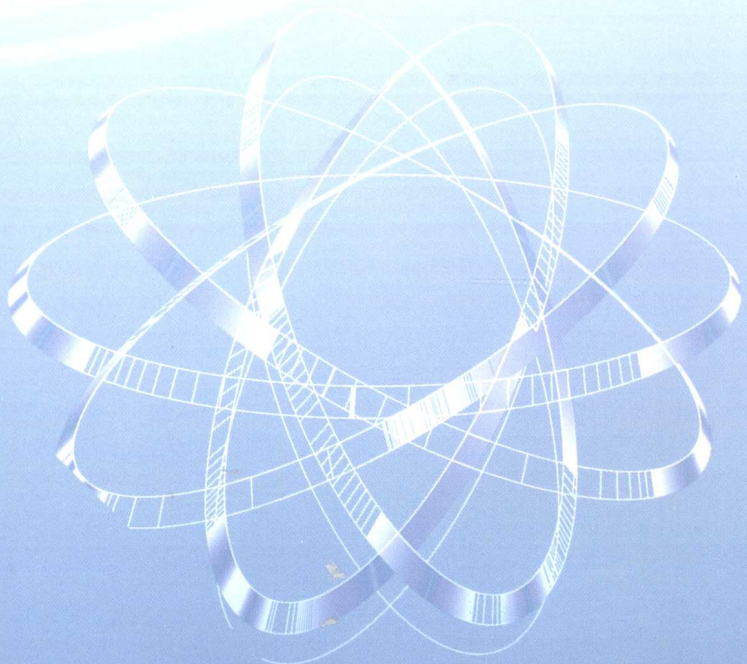
普通高等教育“十一五”规划教材

C程序设计实验指导

张建勋　金　艳　主编



中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

普通高等教育“十一五”规划教材

C程序设计实验指导

张建勋　金　艳　主编

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书主要包括6部分：第1部分阐述了C语言上机实验的目的和要求；第2部分为上机指导，介绍了Turbo C 3.0集成环境下的编辑、编译、调试和运行C程序的方法以及常见的C语言编写过程中的错误和编译信息提示；第3部分为主要实验内容，共设计了14个实验，每个实验根据难易程度和掌握要求分为基本内容和选择内容两部分，实验题目的题型与二级上机考试相似，对于难以理解的地方，大部分添加了注释；实验程序的选择既考虑了知识点的覆盖面，以培养程序设计能力为主线，又重点兼顾了计算机等级考试的能力训练，旨在培养综合能力；第4部分为上机实验补充参考程序，旨在拓宽知识面和加大深度，便于进一步学习；第5部分为笔试模拟测试题；第6部分为全国计算机二级考试大纲和模拟试题。

本书具有较强的针对性与实用性，内容丰富、条理清晰，是学习C程序设计的实用参考用书，本书适合作为高等院校本科教学的辅助教材，也可作为高等职业学校的辅助教材，同时还可作为参加全国计算机等级考试二级C语言的参考用书。

图书在版编目（CIP）数据

C程序设计实验指导/张建勋，金艳主编. —北京：中国铁道出版社，2009.1
普通高等教育“十一五”规划教材
ISBN 978-7-113-08890-3

Ⅰ.C… Ⅱ.①张… ②金… Ⅲ.C语言－程序设计－高等学校－教学参考资料 Ⅳ.TP312

中国版本图书馆CIP数据核字（2009）第001649号

书　　名：C程序设计实验指导
作　　者：张建勋　金　艳　主编

策划编辑：严晓舟　吕燕新
责任编辑：王雪飞　　**编辑部电话：**（010）63583215
编辑助理：杜　鹃
封面设计：付　巍　　**封面制作：**白　雪
责任印制：李　佳

出版发行：中国铁道出版社（北京市宣武区右安门西街8号　　邮政编码：100054）
印　　刷：北京鑫正大印刷有限公司
版　　次：2009年1月第1版　2009年1月第1次印刷
开　　本：787mm×1092mm　1/16　**印张：**7　**字数：**160千
印　　数：3 000册
书　　号：ISBN 978-7-113-08890-3/TP・2901
定　　价：15.00元

前　言 FOREWORD

C 语言因其语言简洁紧凑、使用灵活方便、功能强、应用广泛等诸多优点而成为学习计算机程序设计语言的首选语言。然而，正是由于其功能强、编程限制少、灵活性高，也就意味着不好把握、易出错、难检错、调试困难，所以对使用者要求较高，尤其是初学者会感到很难掌握。

本书是《C 程序设计实用教程》（张建勋、纪钢主编）的配套实验指导书和参考用书。编写本实验指导书的目的是通过一些循序渐进的范例，使学生逐渐熟悉编程思想，熟练掌握程序设计和编码的方法。本书中的所有实验都经过精心设计，着重突出概念，以有助于读者理解 C 程序设计的基本思想和基本方法。每个实验都精心设计实验范例，以助读者深刻理解计算机程序设计中的重要概念，循序渐进地掌握程序设计方法。本书所有实验范例都在 Turbo C 3.0 或 Visual C++ 6.0 中调试通过。

全书由重庆工学院“C 程序设计”精品课程建设小组的教师集体编写完成，张建勋、金艳任主编，纪钢、刘春萌、陈渝、张红、杨长辉、洪雄、李娅、何进参与了本书的编写。本书的作者都是长期在高校从事“C 程序设计”教学的一线教师，有丰富的教学经验和软件开发能力。

本书适合作为高等院校本科教学的辅助教材，也可作为高等职业学校的辅助教材，同时还可作为自学 C 程序设计的参考用书。希望此书能成为读者学习 C 程序设计的工具、能力培养的助手。

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

由于编者水平和时间有限，书中难免存在疏漏和不足之处，恳请广大师生及读者不吝赐教，批评指正。

编　者

2008 年 10 月

目　录 CONTENTS

第 1 部分　C 语言上机实验的目的和要求

1.1　上机实验的目的

程序设计是一门实践性很强的课程，必须十分重视实践环节，保证有足够的上机实验时间，最好能做到上机时间与授课时间之比为 1∶1。由于课内上机时间一般较少，所以除了教学计划规定的上机实验以外，还提倡学生课余时间多上机实践。

学习程序设计课程不能满足于能看懂书上的程序，而应当熟练地掌握程序设计的全过程，即独立编写出源程序，独立上机调试程序，独立运行程序和分析结果。上机实验绝不仅仅是为了验证教材和讲课的内容，其目的是：

（1）了解和熟悉 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”，就是指所用计算机系统的硬件和软件。每一种系统的功能和操作方法不完全相同，但只要熟练掌握一两种系统的使用方法，就可以举一反三。

（2）加深对课堂讲授内容的理解。一些语法规定是 C 语言的重要组成部分，仅凭课堂讲授，既枯燥无味又难以记住，但通过多次上机实践，就能熟练地掌握；一些基本算法对于初学者来说也可能很抽象，通过上机运行，可以理解其奥妙，掌握其技巧。

（3）学会上机调试程序，即善于发现并纠正程序中的错误，使程序能正确运行。经验丰富的人，在编译、连接过程中出现“出错信息”时，一般能很快地判断出错误所在并进行改正。而缺乏经验的人即使在明确的“出错提示”下也往往找不出错误而只能求救于别人。有些经验只能“意会”难以“言传”，仅靠教师的讲授是不够的，还必须亲自动手实践。调试程序的能力，是程序设计人员的基本功，是自己实践的经验累积。

此外，在做实验时，当程序正确运行后，不要急于做下一个题目，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、增加一些功能、改变输入数据的方法等），从多个角度完善程序，再进行编译、连接和运行，观察和分析所出现的情况。在这个过程中要积极思考，主动学习。

1.2　上机实验基本要求

1. 实验前的准备工作

为了提高上机实验的效率，上机实验前必须了解所用系统的性能和使用方法（若用 Turbo C，可详细阅读第二部分；若用 Visual C++ 6.0 可参考其他相关资料），事先做好准备工作，内容至少应包括：

（1）复习和掌握与本实验有关的教学内容。

（2）准备好上机所需的程序。手编程序应该书写清楚，并经过检查认为无误。初学者切忌不编程序或抄袭其他人的程序上机，从一开始就要养成严谨的科学作风。

（3）对运行中可能出现的问题应事先做出估计；对程序中有疑问的地方，应做上记号，以便在上机时注意和解决。

（4）准备好调试和运行时所需的数据（即设计测试方案）。

（5）写出实验预习报告。报告内容包括实验题目、编写好的程序、可能存在的问题和测试数据。

2．实验中的操作步骤

在实验过程中，步骤如下：

（1）调出C编译系统，进入C语言工作环境。

（2）输入自己编好的源程序。检查已输入的程序是否有错，如发现有错，则及时改正。

（3）进行编译和连接。若有语法错误，屏幕上会出现“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。如此反复，直到顺利通过编译和连接为止。

（4）运行程序并分析运行结果。若不能正常运行或结果不正确，则说明有逻辑错误，经过调试、修改，再进行步骤（3），直到得出正确的运行结果为止。

（5）输出程序清单和运行结果（若无打印条件，要记录下调试后的源程序和运行结果）。

（6）若还有时间，应尽可能在调试后的程序中补充一些功能，以提高自己的实践能力。

上机过程中出现的问题，一般应自己独立处理，不要轻易问教师。尤其对“出错信息”，应善于自己分析判断，学会举一反三。这样，处理问题和调试程序的能力就能提高。

3．实验后的分析和整理

实验结束后，需要分析和整理出实验报告。实验报告应包括以下内容：

（1）实验题目。

（2）程序清单。

（3）运行结果（必须是上面程序列表所对应的输出结果）。

（4）对运行情况所做的分析以及本次调试程序所取得的经验。如果程序未能通过，则应分析其原因。

第 2 部分 C 语言的运行环境

2.1 C 语言的运行环境

目前，C 语言的运行环境主要以 DOS 操作系统为主，在程序编译、运行时应在 DOS 环境下进行；其主要编译软件为 Turbo C 2.0（简称 TC2）、Turbo C 3.0（简称 TC3），如果要在 C 语言中进行汉字的处理，须在 DOS 中运行相应的 DOS 汉字操作系统。

1. Turbo C 2.0/ Turbo C 3.0 操作方式

在 Windows 98/Windows 2000/XP 操作环境中，运行 Turbo C 2.0/ Turbo C 3.0 软件有两种方式：在 Windows 中进入 DOS 环境，运行 TC 软件；在 Windows 环境中，直接单击 TC。

Turbo C 3.0 是目前 DOS 系统中可以运行 C++源程序的编程软件，也可以运行 C 语言源程序。如果运行 C++程序，其文件扩展名为*.cpp，运行 C 语言程序，其文件扩展名为*.c。Turbo C 3.0 系统是一个集成系统，能够编辑、编译、连接、运行、查错、求助与操作系统切换于一身。

2. Turbo C 3.0 集成开发环境的组成

图 2-1 显示出了 Turbo C 3.0 集成开发环境，由上至下分成 4 个部分，包括主菜单、编辑窗和信息窗和快速参考行。

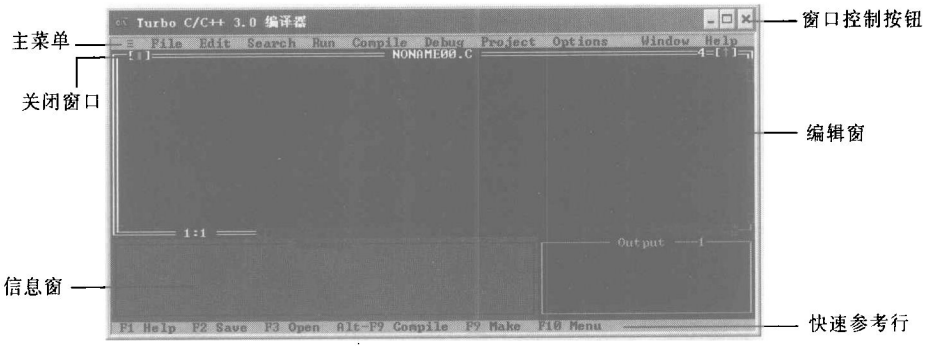


图 2-1 Turbo C 3.0 集成开发环境

Turbo C 3.0 支持多窗口操作，每一个窗口都有“关闭”和“还原/最大化”按钮。可通过 Windows 菜单或相应的快捷键打开任意的窗口。

（1）主菜单：包含 10 个选择项，可对文件进行编辑、修改、运行、存盘等操作。

File：装入或保存文件，管理目录，转入 DOS 和退出 Turbo C。

Edit：建立和编辑文件。

Search：查找、替换和定位等操作。

Run：编译、连接和运行装入环境下的当前程序。

Compile：编译在当前环境下的程序。

Debug：设置各种调试选择项，允许用户增加、删除和编辑监视表达式，可设置和清除断点。

Project：管理多文件工程。

Options：设置编译程序和连接程序的各种选择项。

Window：多窗口操作的设定。

Help：Turbo C 3.0 帮助信息。

（2）编辑窗：源程序输入窗口。在编辑窗口的底部有一个编辑状态行，给出了正在编辑位置的行号和列号。

（3）信息窗口：编译操作发现源程序有错误时，系统将自动地激活信息窗口，信息窗口将列出被编译文件的每个警告或出错信息，同时用高亮度光条在编辑窗口中标出该程序出错的位置。

（4）快速参考行：说明当前位置功能键的功能。

【F1】—Help：打开 HELP 窗口，提供有关的编辑命令信息。

【F2】—Save：保存当前编辑窗口的内容。

【F3】—Open：打开程序文件。

【Alt+F9】—Compile：对当前编辑窗口中的程序进行编译。

【F9】—Make：编译连接生成可执行文件。

【F10】—Menu：活动窗口与菜单切换。

窗口中主要下拉菜单选项

（1）File 菜单：如图 2-2 所示，可对文件进行存储、打开、新文件的建立、DOS 的切换、路径的设置等操作。

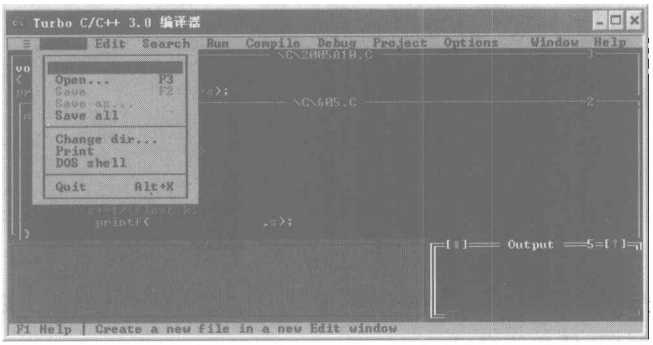


图 2-2 Turbo C 3.0 集成开发环境的文件菜单

（2）Run 菜单：如图 2-3 所示，可以进行文件运行等操作。

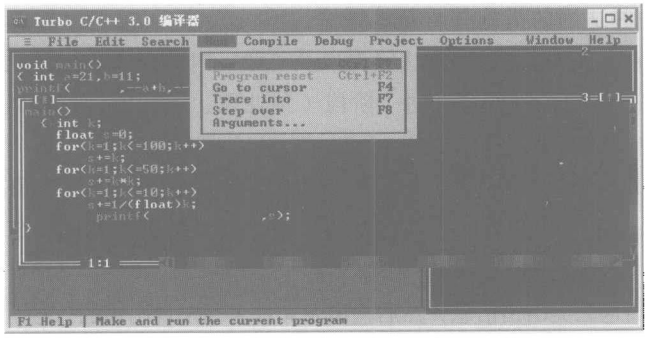


图 2-3　Turbo C 3.0 集成开发环境的运行菜单

（3）Compile 菜单：如图 2-4 所示，可以进行文件的编译和连接等操作。

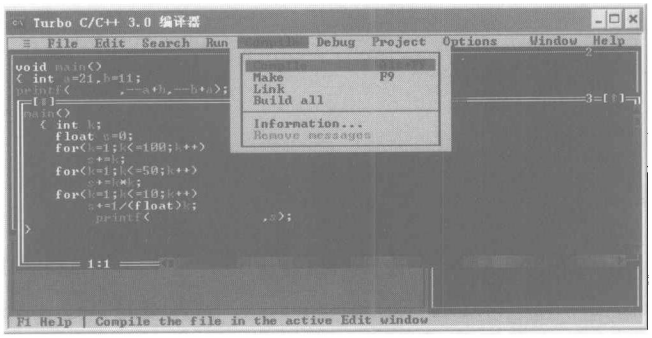


图 2-4　Turbo C 3.0 集成开发环境的编译菜单

（4）Options 菜单：如图 2-5 所示，可以设置编译程序和连接程序的各种选择项文件的编译和连接等操作。

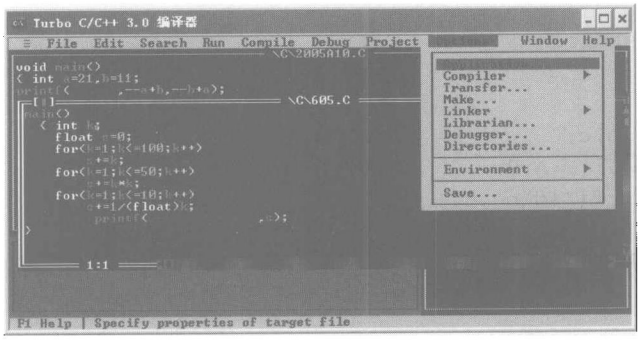


图 2-5　Turbo C 3.0 集成开发环境的选项设置菜单

注意：如果要在源程序中输入汉字，请运行 UCDOS 汉字操作系统。

2.2　C 语言调试、运行操作中的常见错误

1．源程序错误信息的分类

Turbo C 编译程序查出的源程序错误分为严重错误、一般错误和警告 3 类。

（1）严重错误（fatal error）：很少出现，它通常是内部编译出错。在发生严重错误时，编译立

即停止，必须采取一些适当的措施并重新编译。

（2）一般错误（error）：指程序的语法错误以及磁盘、内存或命令行错误等。编译程序将完成现阶段的编译，然后停止。编译程序在每个阶段（预处理、语法分析、优化、代码生成）将尽可能多地找出源程序中的错误。

（3）警告（warning）：不阻止编译继续进行。它指出一些值得怀疑的情况，而这些情况本身又可以合理地作为源程序的一部分。一旦在源文件中使用了与机器有关的结构，编译程序就将产生警告信息。

编译程序首先输出这三类出错信息，然后输出源文件名和发现出错的行号，最后输出信息的内容。

2. Turbo C 程序中的常见错误

（1）说明变量容易出现的错误。

① 忘记定义变量。

例如：

```
#include<stdio.h>
void main()
{
    x=5;y=10; printf("%d\n",x+y);
}
```

C 语言要求在程序中所用到的每一个变量都必须先进行定义。上面的程序中对变量 x 和 y 就没有进行定义。应该在程序的开头处进行定义："int x，y"，这是学过 BAS1C 语言和 FoxBASE 语言的读者最容易出现的一个错误。

② 标识符的大小写字母混用。

C 语言对标识符的大小写一般是非常敏感的，除非将其对应的开关设置为不敏感。

例如：

```
#include<stdio.h>
void main()
{
    int a,b,c;
    a=5;b=6; C=A+B;
      printf("%d\n",C);
}
```

C 语言的编译程序把 A 和 a，B 和 b，C 和 c，分别当做不同的变量。它会提示出 A、B、C 变量是未定义的变量。

③ 字符和字符串的使用混淆。

C 语言规定字符是只具有一个字符的常量或变量。而字符串不管是常量还是变量，它起码要有两个以上的字符。

例如：

```
char sex;
sex="f";
...
```

这里，字符变量 sex 系统只为其分配一个字节的内存空间，无法使用两个字节的空间，应将 sex="f"; 改为 sex='f';。

④ 指针型函数和指向函数的指针之间的混淆。

C 语言中的指针型函数是指该函数的返回值是地址量，而指向函数的指针是指本指针只能接受函数名对它的赋值，通过对指向函数的指针的取内容运算可以将程序的控制流程切换到这个函数。

例如：

```
int(*ptr1) ();
int  *ptr2 ();
```

第一个程序代码说明 ptr1 是一个指向函数的指针，而第二个程序代码说明 ptr2 是一个指针型函数。

（2）使用运算符容易出现的错误。

① 误把“=”当做“==”运算符。

在许多高级语言中，都把“=”既作为赋值运算，又把它作为关系运算符中的“等于”；BASIC 语言和 Pascal 语言都是将它们混用的。

例如：

```
if(a=h)  printf("a 等于 b!");
```

C 语言程序规定：“=”是赋值运算符，“==”才是关系运算符中的“等于”，在 C 语言编译程序中，将（a=b）当做赋值表达式进行处理，它将 b 的值赋值给 a，然后判断 a 的值是否为 0，如果 a 的值是非零，则输出“a 等于 b!”这个字符串；否则，将执行本语句的后继语句。而不是在 a 等于 b 时，就输出“a 等于 b!”这个字符串，否则就执行其后继语句。

② 使用增 1 和减 1 运算符容易出现的错误。

例如：

```
#include<stdio.h>
void main()
{
    int *ptr,a[]={1,3,5,7,9};
    ptr=a;
    printf("%d",*ptr++);
}
```

在这个程序中，有人认为其输出是 a 数组的第一个元素 a[1]的值为 3。其实是由于*ptr++表达式中的增 1 运算是后置加 1，所以要先输出 ptr 指针所指向的 a[0]的内容，即是 1；然后，再调整指针，使指针 ptr 指向 a[1]。如果是*（++ptr）；则先使指针 ptr 指向数组 a[1]，然后输出其值。

例如：

```
#include<stdio.h>
void main()
{
    int a=2;
    printf("%d:%d:%d\n",a,++a,a--);
}
```

上面程序的结果很容易判定为“2，3，3”；其正确的结果是“2，2，2”。其原因是：在执行 printf（）语句时，C 语言的编译程序将参数自右至左依次压入栈中；即是先压入 a--参数的值，再是++a 参数的值，最后是 a 参数的值。出栈时，弹出的顺序是 a、++a、a--，所以其结果是 2、2、2。

③ a>>2 操作并不能改变 a 的值。

例如：

```
#include<stdio.h>
void  main()
{
    unsigned char a;
    a=0x10;
    while(a) printf("%0x右移两位的值是%0x\n",a,a>>2);
}
```

上面的程序是一个死循环。其原因是循环体内 a 的值并未发生变化。注意，像 a>>2 这样操作并不会使操作数 a 的值发生变化。在做 a=a>>2 操作时，变量 a 的值才能发生变化。

④ 错把 & 运算符当做 & & 运算符。

例如：

```
#include<stdio.h>
void main()
{
     int a=5,b=7,c=9,d=11;
     if((a>b)&(c>d)) printf("a>b并且c>d!\n");
     else printf("a不大于b而且c也不大于d!\n");
}
```

本例中，将位逻辑运算符 & 当做了逻辑运算符&&。此种错误是 C 语言初学者最容易犯的错误。

（3）使用 I / O 函数容易出现的错误。

① 输入/输出数据的类型与所用的格式控制符不一致。

例如：

```
#include<stdio.h>
void main()
{
    int a;float b;
    a=3;
    b=4.5;
    printf("%f %d\n",a,b);
}
```

上述程序在编译时并不给出错误信息，但是运行的结果确有点令人不可理解：

```
0.000000     16403
```

它们在存储时是按照赋值转换的规则进行转换的，而输出时是将数据在存储单元中的形式按输出格式控制符的要求将其输出。

② 忘记使用地址运算符。

例如：

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

这是许多 C 语言初学者容易犯的错误；更是其他语言中的习惯所造成的习惯性错误。C 语言要求使用某些变量时，应该指明其地址标记。上述语句正确的表述为：

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

③ 输入数据与要求不符。

使用 scanf()语句时，应该特别注意输入数据与语句要求数据的一致性。

例如：

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

如果输入数据为 6　7 < CR > ，这是错误的；正确输入为 6，7 < CR > 。

（4）使用函数容易出现的错误。

① 未对被调用的函数进行必要的说明。

例如：

```
#include<stdio.h>
void main()
{
    float x,y,z;
    x=3.5; y=-7.5; z=max(z,y);
    printf("较大的数是 %f\n",z);
}
float max(float x, float y)
{
    return(x>y?x:y);
}
```

这个程序看起来并没有错误，但在编译时将给出出错信息。其原因是 max()函数是个浮点型函数，并且是在 main()函数之后定义，而调用本函数时并未对其进行说明，所以出现错误。其改正的方法有两种：

方法一：在函数调用之前用“float　max()”进行说明，其位置最好是在主函数的变量定义和说明部分进行。

方法二：将 max()函数的定义移动到 main()函数的前边。

② 将函数的形式参数及其局部变量一起说明。

例如：

```
min()
int x,y,z;
{
   z=x<y?x:y; return(z);
}
```

这里的错误是将函数的形式参数和函数的局部变量混为一谈了。函数的形式参数具有局部的存储属性，也需要对其进行说明，其位置应该在函数名和函数的主体之间进行。而函数内部的局部变量应该在函数体的首部，变量定义和变量或函数的说明部分进行定义，本例应该将函数的局部变量 z 单独在函数体内进行定义。

③ 认为函数的形式参数可以影响调用函数的实际参数。

例如：

```
#include <stdio.h>
void main()
{
    int x,y;
    x=5;y=9;swap(x,y);
    printf("%d,%d\n",x,y);
}
  swap(int x,int y)
  {
    int t;
```

```
    t=x; x=y; y=t;
}
```

其本意是想通过调用swap()函数来使得main()函数中变量x和y的值能得到交换，但结果是未能达到预期的效果，其原因是由于这种传递数值方法的函数调用，实际参数和形式参数是分别在不同的独立单元，每一组单元的操作并不能影响另一组单元。要想达到上述目的，只有采用传递地址方式，即：

```
#include <stdio.h>
void main()
{
    int x,y;
    x=5; y=9; swap(&x,&y);
    printf("%d,%d\n",x,y);
}
swap(int *x,int *y)
{
    int t;
    t=*x; *x=*y; *y=t;
}
```

④ 实际参数和形式参数类型的不一致。

例如：

```
#include <stdio.h>
void main()
{
    int a=4, b=9, c; c=fun(a,b); …
}
  fun(float x,float y)

{
  …
}
```

上面程序的实际参数a和b是整型变量，而形式参数x和y却是浮点型变量。C语言要求实际参数和形式参数的数据类型必须一致。

⑤ 函数参数的求值顺序造成的差异。

例如：

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

如果变量i在此前的值是3，人们一般认为其输出是“3，4，5”。其实并不一定。在有些计算机系统中输出确是“5，5，4”。其原因是有些系统采用的是自右至左的顺序求函数参数的值，即输出为“5，5，4”。而有的系统则是从左至右求函数参数的值，故其输出是“3，4，5”。

⑥ 用动态的地址作为函数的返值。

例如：

```
#include <stdio.h>
char *strcut(char *s,int m,int n);
void main()
{
    static char s[]="Good Morning!"; char *ptr;
    ptr=strcut(s,3,4);
```

```
    printf("%s\n",ptr);
}
char *strcut(char *s,int m,int n)
{
    char substr[20]; int i;
    for(i=0;i<n;i++)
    substr[i]=s[m+i-1];
    substr[i]='\0';
    return(substr);
}
```

函数 strcut()是个指针型函数，它的返回值是个地址量——自动型字符数组 substr[]的地址。由于字符数组 substr[]的地址在函数 strcut()返回到主调函数时，其地址空间已被释放了，所以指针 ptr 所指的地址就是一个不定的，对应字符串的输出也就是莫名其妙的，严重时系统可能会瘫痪。其解决的方法是把字符数组定义为静态型，代码如下：

```
static char substr [20];
```

⑦ 对指向函数的指针赋值有错误。

例如：

```
#include<stdio.h>
char p1();
void main()
{
    char(*s1)(); char ch;
    s1=p1(); ch=(*s1)("abcde");
    printf("%c\n",ch);
}
char p1(char *s2)
{
   return(s2[1]);
}
```

本例中的指针 sl 是一个指向函数的指针，它所接收的是某函数的地址。而函数的地址就是函数名的本身，本例中就是 p1，而不是 pl()。在此例中的 s1=pl()；是将函数 pl()的返回值赋值给一个指向函数的指针 s1，这是绝对不允许的。

⑧ 函数的返值与期望的不一致。

例如：

```
#include<stdio.h>
void main()
{
    float x,y;
    scanf("%f%f",&x,&y);
    printf("%d\n",addup(x,y));
}
  float addup(float x,float y)
  {
     return(x+y);
  }
```

本例中的主函数期望从函数 addup()的返回值得到一个整型数，而函数 addup()返回值的确是一个浮点数。