第一部分　实 验 指 导

实验 1　熟悉 C 语言集成开发环境

【实验目的】

（1）熟悉 C 语言集成开发环境 C-Free 3.5。掌握 C-Free 3.5 的启动和退出方法，以及在该集成环境下 C 语言源程序文件的新建、打开、保存和关闭等基本操作；掌握开发一个 C 程序设计的基本步骤，包括源程序编辑、编译、连接和运行。

（2）理解程序调试的基本思想，熟悉常用的语法错误提示信息，并根据系统提供的错误提示信息修改 C 程序。

（3）了解 C 程序的基本框架，能够编写简单的 C 程序。

（4）通过运行简单的 C 程序，初步了解 C 语言源程序的特点。

【实验内容】

1. 调试样例 1

在屏幕上显示短句“This is a C program.”。

C 语言源程序如下：

```
#include <stdio.h>
int main()
{    printf("This is a C program.\n");
     return 0;
}
```

以上述 C 语言源程序为例，在 C-Free 3.5 集成环境下，运行一个 C 程序的基本步骤如下所示。

（1）建立自己的文件夹。在磁盘上新建一个文件夹，如 E:\C_programm，用于存放 C 语言源程序。

（2）启动 C-Free 3.5。双击桌面上的 C-Free 快捷方式或依次选择“开始”→“所有程

序”→C-Free 3.5 →C-Free 3.5 菜单命令，进入 C-Free 3.5 集成开发环境（如图 1-1-1 所示）。

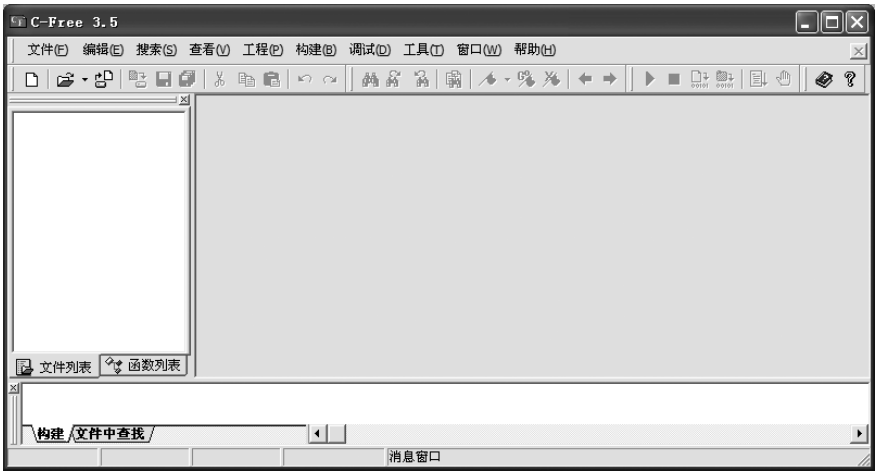

图 1-1-1　C-Free 3.5 启动界面

（3）新建文件。选择“文件”→“新建”菜单命令或按 Ctrl＋N 键（如图 1-1-2 所示）。新建文件后，再选择“文件”→“另存为”菜单命令（如图 1-1-3 所示）。然后在弹出的“另存为”对话框中的“保存在”下拉列表框中选择用户已经建立的文件夹，如 E:\C_programm；再选择保存类型为“C Files，(*.c)”或“.cpp”，接着在“文件名(N)：”下拉列表框中输入文件名，如“test.c”（见图 1-1-4），最后单击“保存”按钮，在 E:\C_programm 文件夹下就新建了文件 test.c，并显示文件列表窗口、编辑窗口和消息窗口（如图 1-1-5 所示）。这时文件列表窗口显示文件的路径及文件名，编辑窗口和消息窗口均为空。

图 1-1-2　新建源文件

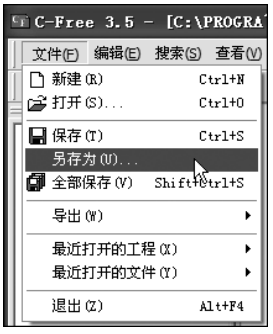

图 1-1-3　保存源文件

（4）编辑和保存。在编辑窗口中输入源程序或在编辑窗口右击，在弹出的菜单中选择“插入代码模板”→C template 命令（如图 1-1-6 所示），会在编辑区产生如下代码：

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    return 0;
}
```

修改以上代码，在语句“return 0;”的上面插入语句“printf("This is a C program.\n");”，并删除 main 函数括号中的参数（如图 1-1-7 所示），然后选择“文件”→“保存”菜单命令，也可

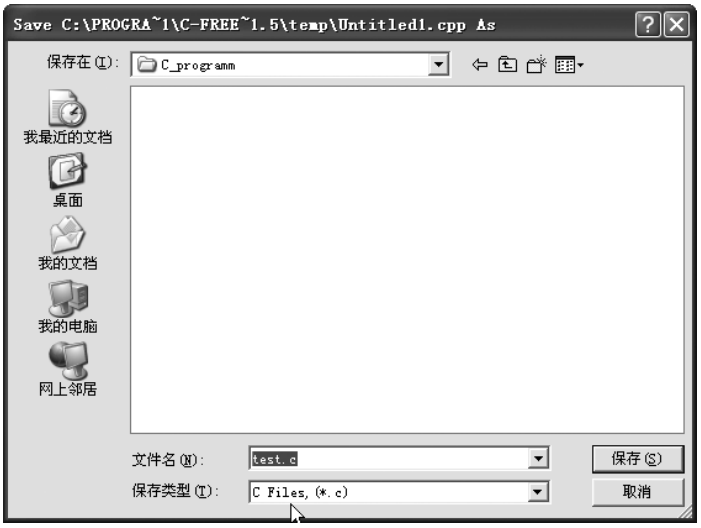

图 1-1-4　文件另存

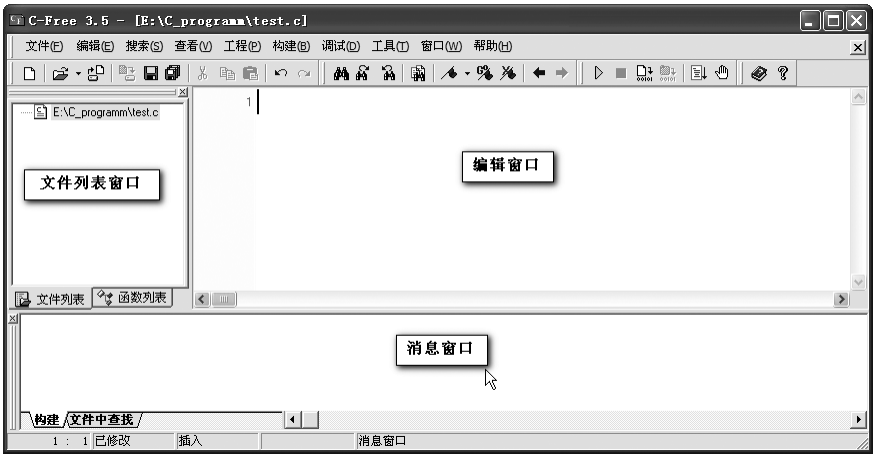

图 1-1-5　新建 C 语言源程序后的界面

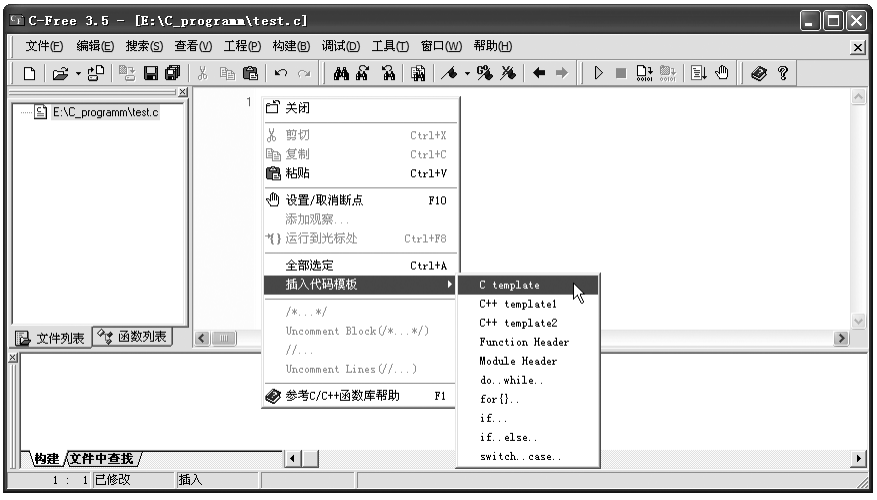

图 1-1-6　插入临时代码

以按 Ctrl＋S 键或单击工具栏上的“保存”按钮来保存源程序。

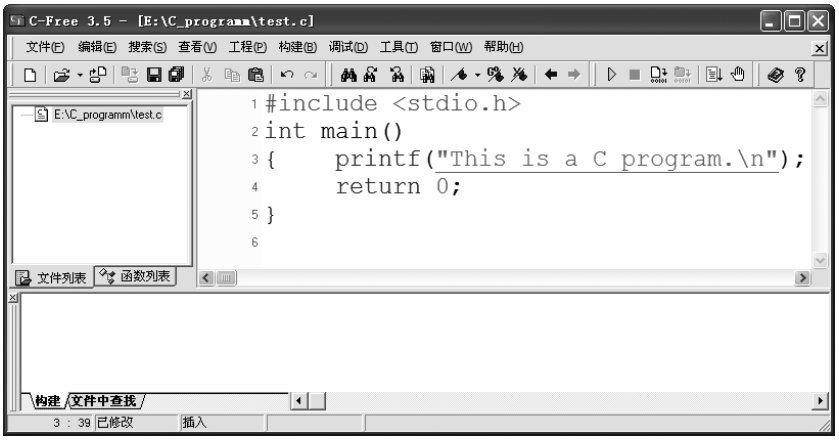

图 1-1-7　编辑源程序

(5) 编译。选择“构建”→“编译 test.c”菜单命令或按 F11 键或单击工具栏上的“编译”按钮进行编译(如图 1-1-8 所示)。系统在编译前自动将程序保存,然后开始编译,并在消息窗口中显示编译信息(如图 1-1-9 所示)。在图 1-1-9 所示的消息窗口中出现的“0 个错误,0 个警告”表示编译成功,没有发现(语法)错误和警告,并生成了目标文件 test.o。

图 1-1-8　编译源程序

注意：如果显示错误信息,说明程序中存在语法错误,必须改正。编译有错误,可以双击提示的错误信息,则在源程序中高亮显示错误行,此时应该检查高亮显示所在行或前面行的程序,找出错误并改正。另外,有时一个简单的语法错误,编译系统可能会提示多条错误信息。此时要找出第一条错误信息,改正后重新编译;再找出重新编译后的第一条错误信息,改正后重新编译;依次找出其他错误并改正,直到没有发现错误并能生成目标文件为止。如果显示警告信息,说明这些错误并未影响目标文件的生成,但通常也应该改正。

(6) 连接。选择“构建”→“构建 test.c”菜单命令,开始连接,并在消息窗口中显示连接信息(如图 1-1-10 所示)。在图 1-1-10 的消息窗口中出现的“0 个错误,0 个警告”表示连接成功,并生成了可执行文件 text.exe。

(7) 运行。选择“构建”→“构建并运行”菜单命令或按 F5 键或单击工具栏上的“运行”按钮(如图 1-1-11 所示),自动弹出运行窗口(如图 1-1-12 所示),显示运行结果“This is a C

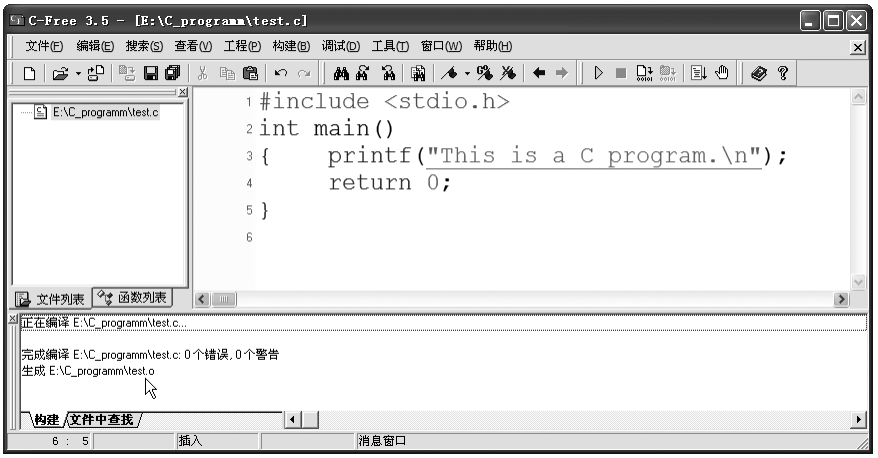

图 1-1-9　编译正确

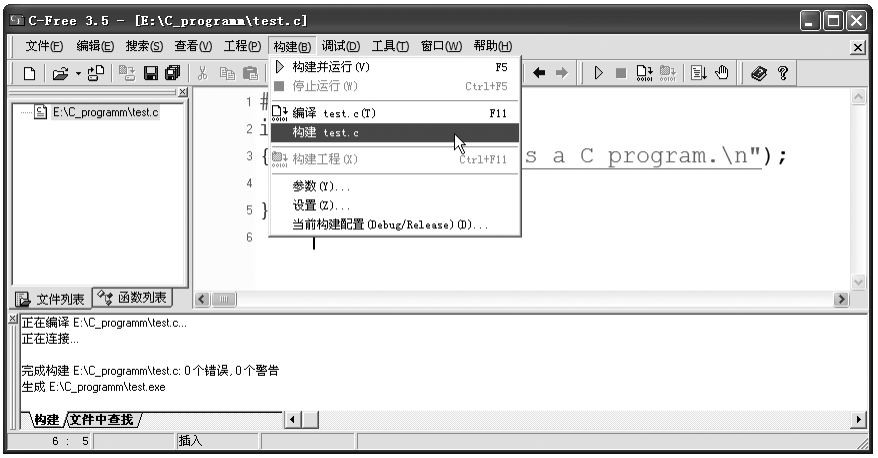

图 1-1-10　连接成功并产生运行文件

program.”。其中“Press any key to continue”是系统自动加上的，提示用户可以按任意键退出运行窗口，返回到 C-Free 3.5 编辑窗口。单击工具栏的“停止运行”按钮（如图 1-1-13 所示），或按 Shift＋F5 键，或直接单击运行窗口控制按钮中的“关闭”按钮（即图 1-1-12 中鼠标指针所指处），都可关闭运行窗口。

图 1-1-11　运行程序

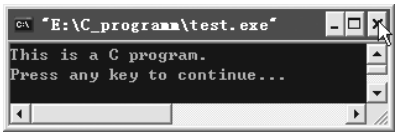

图 1-1-12　运行窗口

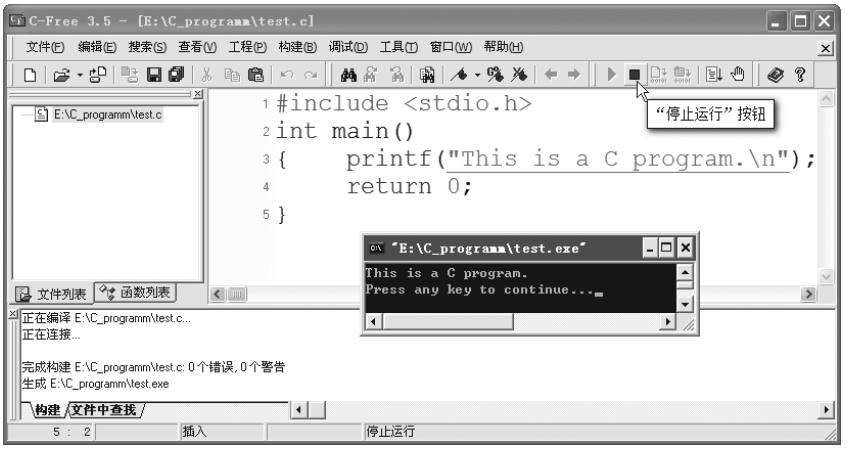

图 1-1-13　关闭运行窗口

(8) 关闭文件。单击工具栏上最右端的“关闭窗口”按钮(如图 1-1-14 所示),即可关闭当前文件 test.c。

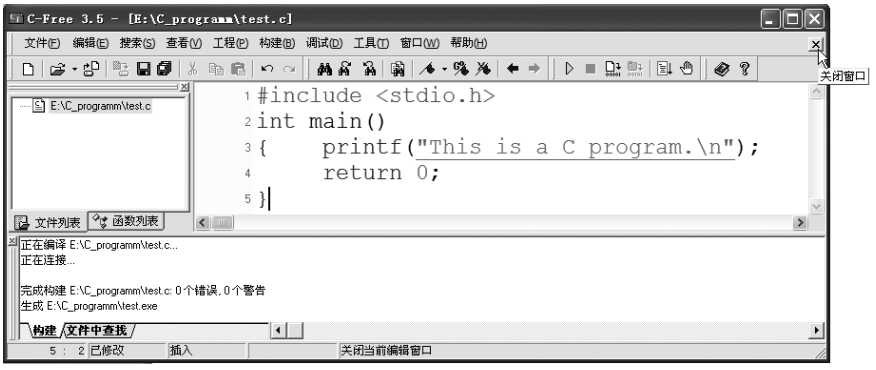

图 1-1-14　关闭当前文件 test.c

(9) 查看 C 语言源程序、目标文件和可执行文件的存放位置。经过编辑、编译、连接和运行后,在文件夹 E:\C_programm 中存放着相关文件,即源程序文件test.c、目标文件 test.o 和可执行文件 test.exe(如图 1-1-15 所示)。

(10) 打开文件。如果要再次打开 C 语言源程序文件,可以选择“文件”→“打开”菜单命令或按 Ctrl + O 键,在弹出对话框的“查找范围”下拉列表框中选择文件夹 E:\C_programm,然后选择文件 test.c,并单击“打开”按钮;或在文件夹 E:\C_programm 中直接双击文件 test.c,都可再次打开源程序文件 test.c。

2. 调试样例 2

改正下列程序中的错误。在屏幕上显示短句“This is a C program.”。

有错误的源程序 error1_1.c:

图 1-1-15　文件夹 E:\C_programm

```
#include <stdio.h>
int mian()
{    printf(This is a C program.\n")
     return 0;
}
```

(1) 编辑。按照“1.调试样例 1”小节中介绍的步骤(10),打开源程序 error1_1.c;或按照调试样例 1 的方法新建一个文件并输入程序 error1_1.c。

(2) 编译。选择“构建”→“编译 error1_1.c”菜单命令,开始编译。编译后,消息窗口中显示“2 个错误,0 个警告”。

(3) 找出错误。在消息窗口双击第一条出错信息,编辑窗口的源程序中就会高亮显示错误行(如图 1-1-16 所示)。一般在高亮显示所在行或前面行,可以找到出错语句。图 1-1-16 中高亮显示的是源程序的第 3 行,其对应的消息窗口中显示“unterminated string or character constant”,出错信息指出字符串或字符常量缺少结束符。出错的原因是使用字符串或字符常量缺少配对的引号,应检查所有字符串是否都使用了成对的双引号,所有字符常量是否都使用了成对的单引号。仔细观察后,发现 This 字符串前少了一个前双引号。

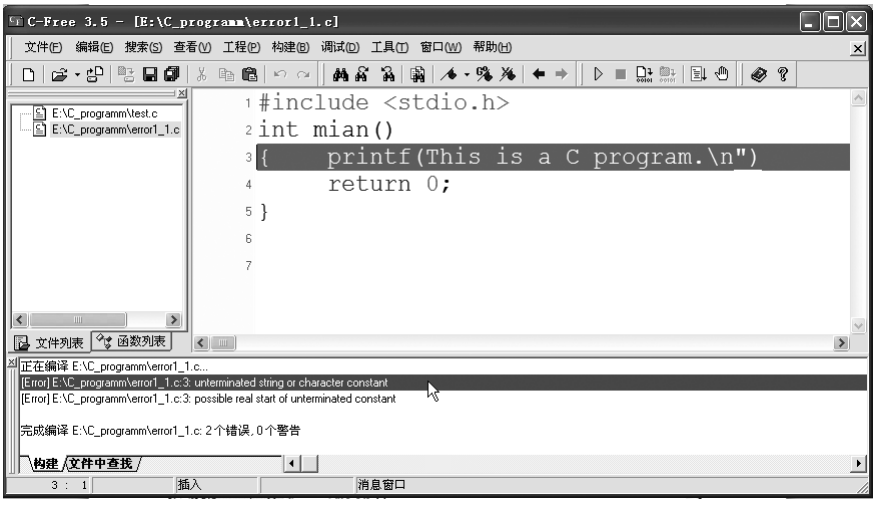

图 1-1-16　程序 error1_1.c 编译产生的错误提示信息

(4) 改正错误。在 This 前加上前双引号。

(5) 重新编译。因为有些错误往往是前一处错误引起的,所以修改程序错误时,最好先修改第一处错误,且修改后要重新编译。重新编译后,消息窗口中显示“1 个错误,0 个警告”。双击消息窗口中的错误提示信息“parse error before 'return'”,则在编辑窗口的源程序中高亮显示第 4 行(如图 1-1-17 所示),错误提示信息指出在 return 前有语法错误。仔细观察,引起错误的原因是 return 前一条语句缺少分号。改正错误,在 return 前一条语句最后补上一个分号。

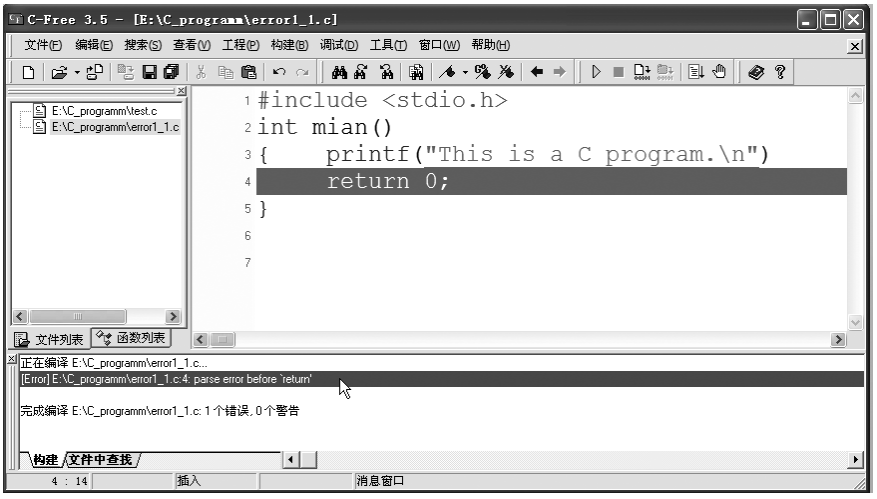

图 1-1-17 重新编译后产生的错误提示信息

(6) 再次编译。消息窗口中显示编译正确。

(7) 连接。选择“构建”→“构建 error1_1.c”菜单命令,开始连接,并在消息窗口显示连接错误提示信息“undefined reference to 'WinMain@16'”(如图 1-1-18 所示)。错误提示信息指出没有定义参数 WinMain@16,仔细观察后,发现主函数名 main 拼写错误,被误写为 mian。

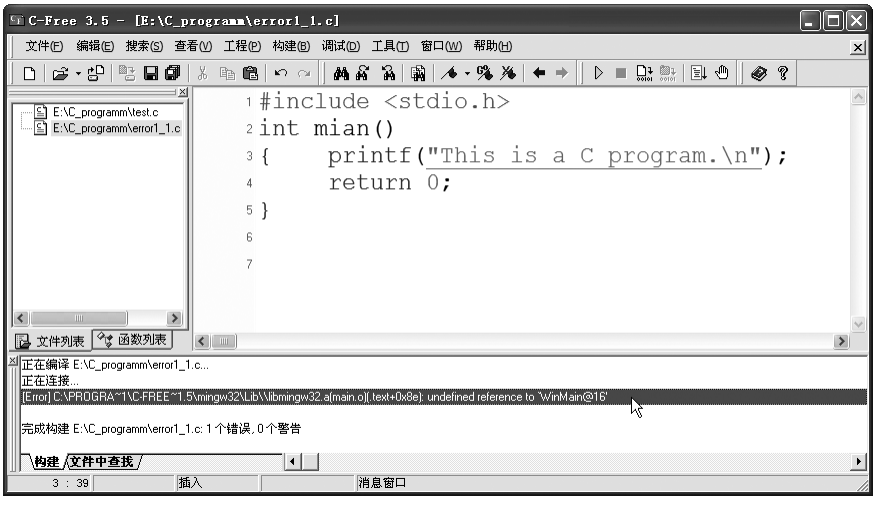

图 1-1-18 连接产生的错误提示信息

(8) 改正错误。把 mian 改为 main 后，重新编译和连接，消息窗口中没有出现错误和警告信息。

(9) 运行。选择“构建”→“构建并运行”菜单命令、按 F5 键或单击工具栏上的“运行”按钮，自动弹出运行窗口(如图 1-1-19 所示)，显示运行结果，按任意键返回。

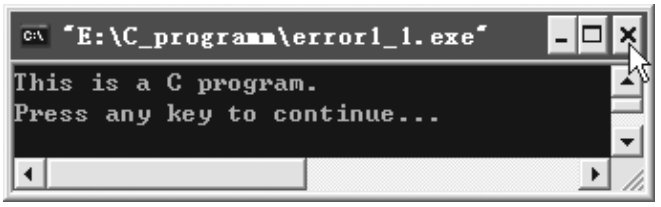

图 1-1-19　改正后的程序 error1_1.c 运行窗口

3. 程序修改题

模仿调试样例 2 的方法，改正下列程序中的错误。在屏幕上显示以下 3 行信息。

```
**************
Very good!
**************
```

有错误的源程序 error1_2.c：

```
#include <stdio.h>
int mian()
{  printf("**************\n");
   printf(" Very good!/n")
   printf("**************\n);
   return 0;
}
```

4. 程序设计题

模仿调试样例 1 的方法，完成下列程序设计题。

(1) 在屏幕上显示短句“One World，One Economic!”。

扩展：如何在屏幕上显示数字、英文字母和汉字等信息？例如“你是住 B 区 8 栋吗?”

(2) 在屏幕上显示下列网格。

```
+---+---+
|   |   |
+---+---+
```

扩展：如何在屏幕上显示自己设计的名片？例如：

```
|~~~~~~~~~~~~~~~~~~~|
|   My name is XXX   |
|___________________|
```

(3) 在屏幕上显示下列由各种字符组成的图案。

```
      A
   B    C
D    E    F
```

【实验结果和分析】

（1）将C语言源程序、运行结果写在实验报告上。

（2）分析源程序和运行结果，并将遇到的问题和解决问题的方法写在实验报告上。

实验 2 熟悉 C 语言的基本元素

【实验目的】

（1）掌握 C 语言简单数据类型及不同的数据类型之间的转换规则。

（2）掌握变量和常量的定义与使用。

（3）掌握 C 语言的运算符及各种运算符运算规则，以及包含这些运算符的表达式。

（4）掌握输入函数 scanf()和 getchar()、输出函数 printf()和 putchar()的使用，并能调用 C 语言的数学函数。

（5）掌握 C 程序的基本结构，并能够编程实现简单的数据处理。

（6）掌握使用工具栏进行编辑、编译和运行操作的方法，进一步理解编译错误信息的含义，熟悉简单 C 程序的查错方法。

【实验内容】

1. 调试样例

改正下列程序中的错误。输入华氏温度 f，输出对应的摄氏温度 c。转换公式如下：

$$c = \frac{5 \times (f - 32)}{9}$$

有错误的源程序 error2_1.c：

```
#include <stdoi.h>
int main()
{  int c;f;
   printf("Enter f:");
   scanf("%d",f);
   c=5*(f-32)/9;
   printf("f=d, c=%d\n,f,c");
   return 0;
}
```

现在介绍使用工具栏上的按钮完成编辑、编译和运行操作。若工具栏上相应的按钮没有显示，可选择“查看”→“工具条”菜单命令，然后选中相应工具条前的复选框即可显示（如图 1-2-1 所示）。

（1）打开文件。单击工具栏上的“打开”按钮，在弹出对话框的“查找范围”下拉列表框中找到要打开文件的路径，然后双击文件 error2_1.c，即可打开源程序 error2_1.c。

（2）编译。单击工具栏上的“编译”按钮，在消息窗口会出现提示信息（如图 1-2-2 所

图 1-2-1　C-Free 3.5 工具栏和“查看”菜单

示)。双击消息窗口中的第一条出错信息“stdoi.h:No such file or directory”,编辑窗口就高亮显示源程序的第 1 行,出错信息指出没有 stdoi.h 这样的文件或目录。仔细观察后,发现错误原因是 stdoi.h 拼写错误,应将它改为 stdio.h。改正后重新编译,在消息窗口会出现新的提示信息(如图 1-2-3 所示),双击新产生的第一条出错信息“‘f’undeclared”,编辑窗口会高亮显示源程序的第 3 行,出错信息指出变量 f 没有定义,变量必须先定义后使用。仔细观察后,发现 f 前的分号应该为逗号。将 f 前的分号改为逗号后,重新编译,编译正确。

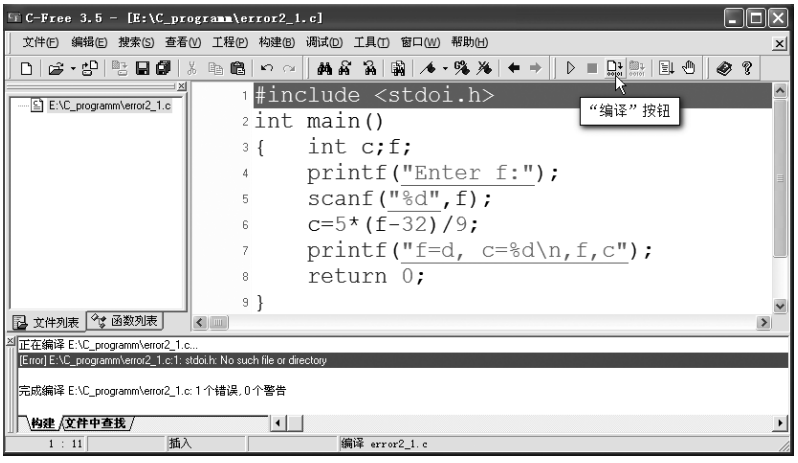

图 1-2-2　程序 error2_1.c 第一次编译产生的错误提示信息

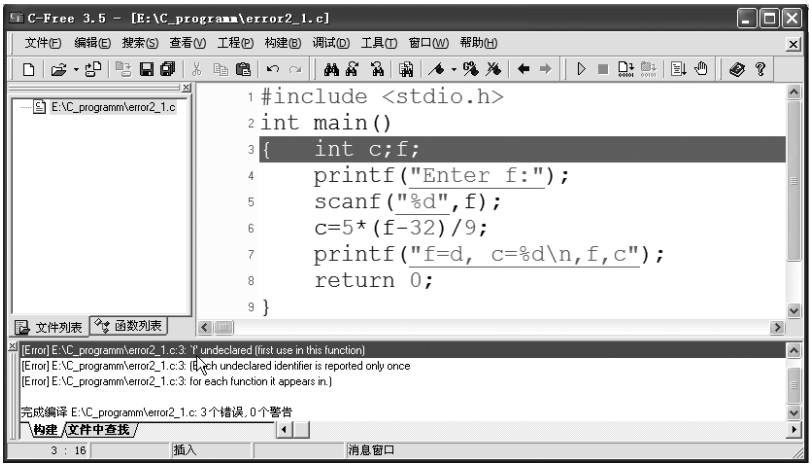

图 1-2-3　程序 error2_1.c 重新编译产生的错误提示信息

(3) 连接。选择“构建”→“构建 error2_1.c”菜单命令，开始连接，连接正确。

(4) 运行。单击工具栏上的“运行”按钮，在弹出的运行窗口中输入“150”后按 Enter 键，出现系统提示信息(如图 1-2-4 所示)，这是地址越界引起的错误。遇到这种现象应该考虑输入变量 f 的输入格式是否正确，本程序应将输入语句改为“scanf("%d",&f);”。修改后再运行程序，发现运行结果为：f=d，c=37814104，结果不符合题目的要求，仔细检查源程序，发现输出函数 printf 中不仅 f=d 的 d 应改为%d，而且 printf 函数中括号内后双引号(")的位置也错了，应改为“printf("f=%d，c=%d\n",f,c);”。改正后，重新编译、连接和运行，在弹出的运行窗口中仍然输入“150”后按 Enter 键，显示结果与题目要求的结果一致(如图 1-2-5 所示)。单击工具栏上的“停止运行”按钮(如图 1-2-1 所示)返回。

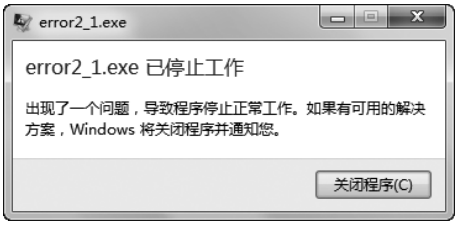

图 1-2-4　系统提示信息

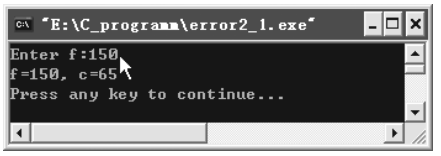

图 1-2-5　改正后的程序 error2_1.c 运行窗口

注意：

① 输入函数 scanf、输出函数 printf 的输入输出参数必须和格式控制字符串中的格式控制说明相对应，即它们的类型、个数和位置都要一一对应，且书写形式也要正确。但在 scanf 函数中，输入参数的形式为：变量名前面要用地址符 &，表示变量在内存中的地址，且变量地址要求有效。

② 顺序结构中常出现的错误一般有：变量定义的位置不正确；定义变量的形式不正确，如语句“int c;f;”；丢分号、大括号、双引号和单引号等；scanf 函数中少写 &；大小写错误等。

扩展：如果华氏温度和摄氏温度都是双精度浮点型数据，应如何修改程序？

注意：输入 double(双精度浮点)型数据一定要使用格式控制说明符%lf，其中的 l 是 long 的首字母，不是数字 1；但输出 double 型数据使用格式控制说明符%lf 和%f 都可以。

2. 程序修改题

模仿以上调试样例的方法，改正下列程序中的错误。

(1) 计算某个数 x 的平方 y，并分别以 $y=x*x$ 和 $x*x=y$ 的形式输出 x 和 y 的值。请不要删除源程序中的任何注释。

输出样例(假设 x 的值为 3)：

```
9=3*3
3*3=9
```

有错误的源程序 error2_2.c：

```
#include <stdio.h>
int main()
{  int y;
   y=x*x
```

```
    printf("%d=%d*%d\n",x);     /*输出
    printf("d*%d=%d\n",y);
    return 0;
}
```

提示：

① 检查 C 语句是否都由分号结束。

② 注释的方法有两种：一种是块注释，必须用/*和*/配对使用，二者之间为注释内容，可以包含多行；另一种是行注释，注释范围从//起至换行符止。注释部分的内容均不会产生目标代码。

③ 变量必须先定义，并经过初始化后才可使用该变量，否则变量值无法预计。

(2) 输入两个实数，计算并显示这两个实数之和的平方根。

有错误的源程序 error2_3.c：

```
#include <stdio.h>
#include <math>
int main()
{  double x,y,s;
   scanf("%f%f",x,y);
   s=sqrt(x+y);
   printf("s=%f\n,s");
   return 0;
}
```

提示：

① 使用 sqrt 函数计算平方根时，在程序的开头加命令行 # include<math.h>引入 math.h 头文件。

② 输入 double 型数据使用格式控制说明符%lf，其中的 l 是 long 的首字母，不是数字 1；但输出 double 型数据使用格式控制说明符%lf 和%f 都可以。

③ 在 scanf 函数中括号内变量 x、y 的前面，要用地址符 &，表示变量在内存中的地址。

④ 注意 printf 函数中括号内双引号的位置。

(3) 输入摄氏温度 c，输出对应的华氏温度 f。转换公式如下：

$$f=\frac{9}{5}\cdot c+32$$

有错误的源程序 error2_4.c：

```
#include <stdio.h>
int main()
{  double c=0,F=0;
   printf("Enter c:");
   scanf("%lf",c);
   f=(9/5)·c+32;
   print("c=%lf,f=%lf\n,c,f");
   return 0;
}
```

提示：

① C 语言是区分大小写的。

② C 语言表达式中的乘号必须用“*”表示。

③ 使用输入输出函数时，各参数的书写形式要正确。

④ 两个整型数相除，其运算结果也是整型。

3. 程序填空题

注意： 下列程序中标有①②③④的部分为需要填空的部分。在填空时，先删除填空标志，再根据程序功能填空，然后调试运行程序。在本书的所有程序填空题中，都遵循这一规定。

完善下列程序。将输入的角度转换成弧度。

有待完善的源程序 fill2_1.c：

```
#include <stdio.h>
  ①
int main()
{  int degree;
   float radian;
   printf("Enter degree:");
   ②  ;
   radian=PI * degree/180;
   printf("\n  ③  degrees equal to  ④  radians.\n", degree,radian);
   return 0;
}
```

提示： 圆周率 π 在 C 语言中是不合法的标识符，必须定义符号常量（如 PI）或直接用 3.1415926 代表圆周率；注意输入输出函数的使用。

4. 程序设计题

（1）编写程序输出 $5\sin60^\circ+12.5\times3.4+\sqrt{16.88}$ 的值。

要求： 不使用变量。

（2）输入两个整数，计算并输出这两个数的和、差、积、商、余数和平均值。

要求：

① 输入数据前，给出输入提示信息；输入数据占一行，由两个整数组成，数据之间用一个空格隔开。

② 输出为 5 行，分别输出这两个数的和、差、积、商、余数和平均值（取整数部分），并以算术的形式显示。

扩展： 如果输入的两个数是 double 型数据，应如何修改程序？题目的要求都能达到吗？

（3）输入两个点坐标（x1，y1）和（x2，y2），计算并输出两点间的距离。

要求：

① 输入数据前，给出输入提示信息；输入数据占一行，由 4 个 double 型数据组成，分别表示为 x1、y1、x2、y2，数据之间用一个空格隔开。

② 输出为一行，并有输出说明，且结果保留两位小数。

提示： printf 函数的格式控制说明符 %f 指定以小数形式输出 double 型数据（保留 6 位小数），而 %.2f 则指定输出时保留两位小数。

(4) 当 n 为 152 时，计算并输出 n 的个位数字 d1、十位数字 d2 和百位数字 d3 的值。

要求： 输出样例为“整数 152 的个位数字是 2，十位数字是 5，百位数字是 1.”。

提示： n 的个位数字 d1 的值是 n%10，十位数字 d2 的值是(n/10)%10，百位数字 d3 的值是 n/100。

扩展：

① 逆序输出任意一个三位正整数的每一位数字，应如何实现？

② 如果 n 是任意一个四位正整数，如何求出它的每一位数字？

③ 输入一个五位正整数，分解出它的每位数字，并将这些数字间隔 3 个-的形式输出。例如，输入 12345，则输出 1---2---3---4---5。应如何实现？

(5) 输入一个正整数 n(n 表示分钟数)，通过程序实现把 n 分钟用小时和分钟显示。

要求：

① 输入数据前，给出输入提示信息。

② 输出为一行，并有输出说明，例如，输入“500”，则输出“500 minutes:8 hours and 20 minutes.”。

(6) 输入两个实数 r 和 h，计算并输出以 r 为底面半径以 h 为高的圆柱体的体积(体积＝底面积×高，底面积＝πr^2)。

要求：

① 输入数据前，给出输入提示信息；输入数据占一行，由两个 double 型数据组成，数据之间用一个空格隔开。

② 输出为一行，并有输出说明，且结果保留两位小数。

③ 调用数学函数 pow()求幂。

④ 定义符号常量 PI 代表圆周率。

(7) 用 getchar 函数读入两个字符给变量 c1、c2，然后分别用 putchar 函数和 printf 函数输出这两个字符。并思考以下问题：

① 变量 c1、c2 应定义为字符型还是整型？或二者皆可？

② 要求输出 c1 和 c2 值的 ASCII 码，应如何处理？用 putchar 函数还是 printf 函数？

③ 整型变量与字符型变量是否在任何情况下都可以互相代替？例如，“char c1,c2;”与“int c1,c2;”是否无条件等价？

要求：

① 输入数据前，给出输入提示信息；输入数据占一行，由两个字符型数据组成，两个数据之间无任何符号。

② 输出为两行，分别用 putchar 函数和 printf 函数输出 c1、c2 这两个字符，并有输出说明。

注意： 在用连续两个 getchar 函数输入两个字符时，只要输入了“a”后按 Enter 键，系统就会认为用户已经输入了两个字符。所以应当连续输入“ab”后再按 Enter 键，这样就保证了 c1 和 c2 分别得到字符 a 和 b。

(8) 输入两个数字字符并分别存放在字符型变量 a 和 b 中，通过程序将与这两个字符对应的数字相加后输出。例如，输入字符型数字 7 和 5，输出的则是整型数 12。

要求：

① 输入数据前，给出输入提示信息。

② 通过 scanf 函数或 getchar 函数输入字符型变量 a、b 的值，输入数据占一行，由两个字符型数据组成，字符数据之间无任何符号。

③ 输出为一行，并要求输出求和算术式，例如，输出“7＋5＝12”。

提示：通过“数字字符-'0'”得到对应数字。

扩展：将连续输入的 4 个数字字符拼成一个整型的数值。如输入 4 个字符分别是'1'、'2'、'4'、'8'，应该得到一个整型数值 1248。应如何编程实现？

(9) 输入两个字符，分别存放在变量 x 和 y 中，通过程序交换它们的值。

要求：

① 输入数据前，给出输入提示信息。

② 通过 scanf 函数或 getchar 函数输入字符变量 x、y 的值，输入数据占一行，由两个字符型数据组成，两个数据之间无任何符号。

③ 输出为一行，并有输出说明。

【实验结果和分析】

(1) 将 C 语言源程序、运行结果写在实验报告上。

(2) 分析源程序和运行结果，并将遇到的问题和解决问题的方法写在实验报告上。

实验 3 分支结构程序设计

【实验目的】

（1）掌握 C 语言关系运算符和关系表达式、逻辑运算符和逻辑表达式的使用。

（2）熟练掌握各种类型 if 语句的使用方法。

（3）掌握 switch 语句以及其中的 break 语句的使用方法。

（4）掌握条件运算符和条件表达式的使用。

（5）掌握基本输入输出函数的使用，能正确调用 C 语言提供的数学函数（math.h）和常用字符函数（ctype.h）。

（6）掌握简单的单步调试、断点调试和使用 Debug 工具栏调试程序的方法。

【实验内容】

1. 调试样例

使用单步调试、断点调试和 Debug 工具栏调试程序的方法，改正下列程序中的错误。输入参数 a、b、c，求一元二次方程 $ax^2+bx+c=0$ 的根。

有错误的源程序 error3_1.c：

```
#include <stdio.h>
#include <math.h>
int main()
{  double a,b,c,d;
   printf("Enter a,b,c:");
   scanf("%f%f%f",&a,&b,&c);
   d=b*b-4*a*c;                    /*调试时设置断点*/
   if(a=0)
   {  if(b==0)
      {  if(c==0)
            printf("参数都为 0,方程无意义!\n");
         else
            printf("a 和 b 为 0,c 不为 0,方程不成立\n");
      }
      else
         printf("x=%0.2f\n",-c/b);
   }
   else
      if(d>=0)                     /*调试时设置断点*/
```

```
        {  printf("x1=%0.2f\n", (-b+sqrt(d))/(2*a));
           printf("x2=%0.2f\n", (-b-sqrt(d))/(2*a));
        }
        else
        {  printf("x1=%0.2f+%0.2fi\n", -b/(2*a), sqrt(-d)/(2*a));
           printf("x2=%0.2f-%0.2fi\n", -b/(2*a), sqrt(-d)/(2*a));
        }
    return 0;                    /*调试时设置断点*/
}
```

(1) 打开源程序 error3_1.c,对程序进行编译和连接,没有出现错误和警告信息。但运行程序时,在弹出的运行窗口中输入 a、b、c 的值(2.1 8.9 3.5)后按 Enter 键,发现运行结果(如图 1-3-1 所示)显然错误,说明程序存在逻辑错误,需要调试修改。

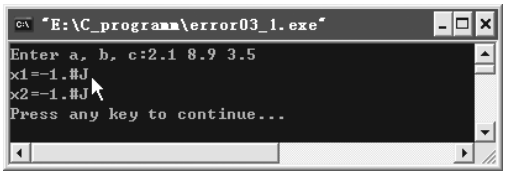

图 1-3-1　程序 error3_1.c 运行窗口

(2) 调试步骤如下。

首先介绍断点的使用。断点的作用就是使程序执行到断点处暂停,用户可以观察当前变量或表达式的值。要设置断点,最方便快捷的方法是将鼠标指针移到代码区中某一条代码的左边(灰色区域),光标由 I 字形变成黑色圆形断点形状(如图 1-3-2 所示),然后单击,看到红色断点就设置完成。另一种方法是先将光标移到你想要设置的行,然后单击工具栏上的"设置/取消断点"按钮(如图 1-3-2 所示)。对于已经设置断点的行,对该行重复进行上面的设置断点的操作,将取消断点。

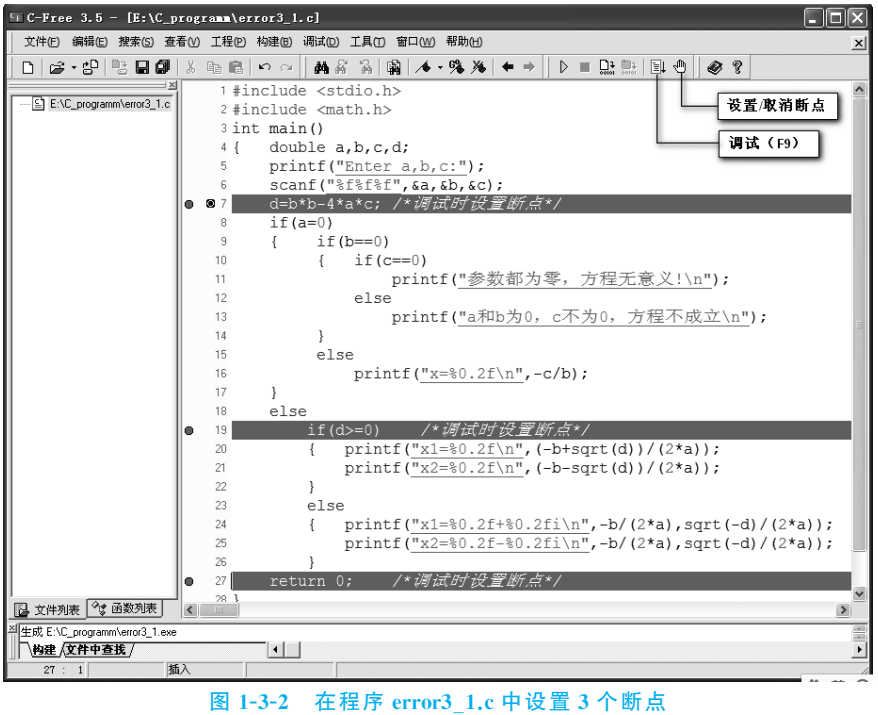

图 1-3-2　在程序 error3_1.c 中设置 3 个断点

① 调试程序开始，设置 3 个断点(如图 1-3-2 所示)，具体位置见源程序的注释。

② 单击工具栏上的“调试”按钮或按 F9 键，程序开始调试。一旦程序开始调试，C-Free 会自动显示 Debug 工具栏(如图 1-3-3 所示)。

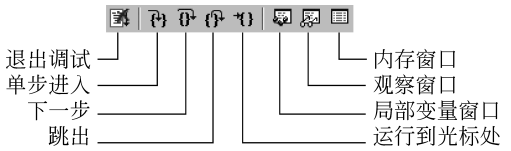

图 1-3-3　Debug 工具栏

注意：程序开始调试，执行到某一个断点前，这时“调试”按钮的功能变为“继续”。单击该按钮，程序从该断点处继续执行，直到碰到下一个断点。

③ 在弹出的运行窗口中输入 a、b、c 的值(2.1 8.9 3.5)后按 Enter 键，程序执行到并停在第一个断点处，单击 Debug 工具栏上的“局部变量窗口”按钮(如图 1-3-3 所示)，然后在局部变量窗口(如图 1-3-4 所示)中查看变量 a、b、c 的值，此时，这些变量的值与输入的值不一致。由此可知第一个断点处之前肯定有错误发生，且变量 a、b、c 的取值不正确，应该检查输入函数 scanf 是否正确。仔细检查本程序中的 scanf 函数，发现 scanf 函数中 double 型数据的输入格式控制说明符%lf 错写成%f 了。

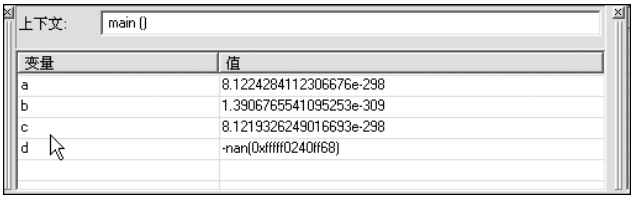

图 1-3-4　局部变量窗口

注意：当程序处于调试停止状态，局部变量窗口将显示当前运行环境下所有局部变量的值。对于源程序 error3_1.c 的程序运行状态，有 3 个局部变量。局部变量窗口如图 1-3-4 所示，其中上下文显示的是当前程序运行的函数环境，包括参数的值。

④ 单击 Debug 工具栏上的“退出调试”按钮(如图 1-3-3 所示)或按 Ctrl+F9 组合键，结束程序调试。然后将本程序中的输入语句改为“scanf("%lf%lf%lf",&a,&b,&c);”。改正后，重新对程序进行编译和连接。再单击工具栏上的“调试”按钮，在弹出的运行窗口中同样输入 a、b、c 的值(2.1 8.9 3.5)后按 Enter 键。程序执行到第一个断点处，同样在局部变量窗口中查看变量 a、b、c 的值。此时，这些变量的值与输入的值一致(如图 1-3-5 所示)。但在图 1-3-5 所示的局部变量窗口中，查看到此时变量 d 的值显然不正确，原因是程序执行到第一个断点处时，断点处的语句并未执行。

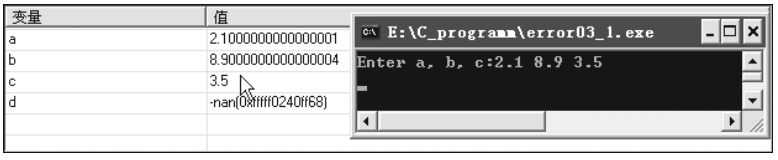

图 1-3-5　局部变量窗口中变量的值与输入的值对照

⑤ 单击 Debug 工具栏上的“下一步”按钮(如图 1-3-3 所示)或按 F8 键后，在图 1-3-6 所示的局部变量窗口中观察到变量 d 的值变为 49.81，此时 d 值是正确的。“下一步”按钮的功能是单步执行，即单击一次执行一行(如图 1-3-6 所示)，编辑窗口中的箭头指向某一行，表示程序将要执行该行。单击 Debug 工具栏上的“观察窗口”按钮(如图 1-3-3 所示)，即可打