第一部分　C 语言上机指导

第 1 章　Visual C++ 6.0 上机操作

C++语言是在 C 语言的基础上发展而来，它增加了面向对象的编程，成为当今最流行的一种程序设计语言。Visual C++ 6.0 是由微软公司开发的面向 Windows 编程的 C++语言工具。它不仅支持 C++语言的编程，也兼容 C 语言的编程。由于 Visual C++ 6.0 被广泛地用于各种编程，使用面很广。这里简要介绍如何在 Visual C++ 6.0 下运行 C 语言程序。

1.1　使用 VC++ 6.0 调试程序一般步骤

在使用 Visual C++ 6.0 调试程序之前，必须先了解项目和项目工作区两个概念。项目（Project）是一组相互关联的文件，是将一个应用程序中需要的所有文件组成一个有机的整体，一般包括源文件、头文件和资源文件等。而项目文件必须在某个项目工作区（Workspace）的管理之下，一个项目工作区可以包含一个以上的项目。

使用 Visual C++ 6.0 调试程序的大致步骤是：

（1）创建一个项目工作区（Workspace）。

（2）在项目工作区中新建一个项目（Project）。

也可以将上面两步合并，在建立新项目的同时创建项目工作区。

（3）在项目中建立源文件和头文件等（或将这些文件添加到项目中）。

（4）编辑项目中的源文件和头文件。

（5）连接、编译应用程序。

（6）运行应用程序。

注意：如果不先创建项目工作区和项目，而直接创建源文件，当编译该源文件时会提醒没有项目工作区和项目，单击“确定”按钮会自动创建一个默认的项目工作区和项目。

1.2　启动 VC++

Visual C++ 6.0 是一个庞大的语言集成工具，经安装后将占用几百兆磁盘空间。从“开始”→“程序”→Microsoft Visual Studio 6.0→Microsoft Visual C++ 6.0，可启动 Visual C++ 6.0，屏幕上将显示如图 1.1 所示的窗口。

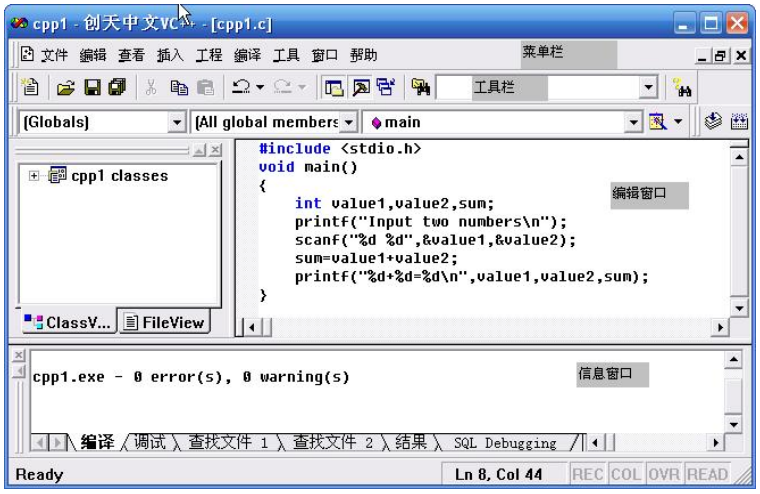

图 1.1　VC++ 6.0 窗口

1.3　新建或打开 C 程序文件

选择“文件”菜单中的“新建”命令，单击如图 1.2 所示的“文件”标签，选中“C++ Source File”，单击“确定”按钮，然后在编辑窗口中输入程序。

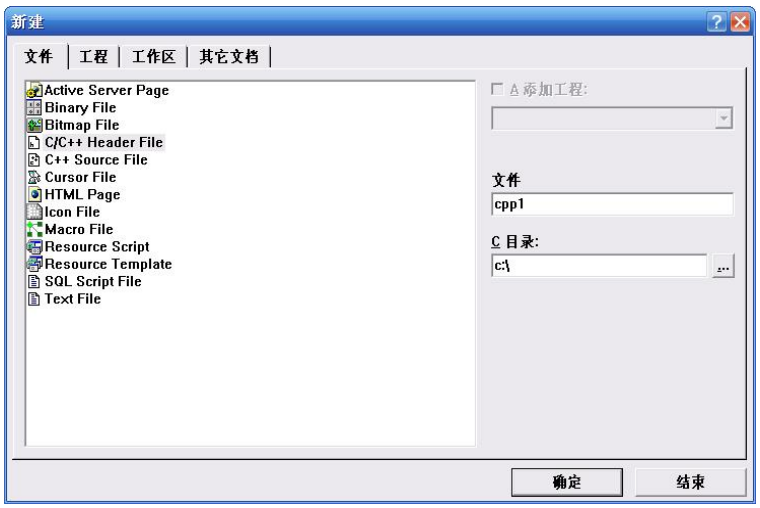

图 1.2　“新建”对话框

如果程序已经输入过，可选择“文件”菜单中的“打开”命令，并在查找范围中找到正确的文件夹，调入指定的程序文件。

1.4　程序保存

在打开的 VC++界面上，可直接在编辑窗口输入程序，由于完全是 Windows 界面，输入及修改可借助鼠标和菜单进行，十分方便。当输入结束后，保存文件时，应指定扩展名为“.C”，

否则系统将按 C++扩展名“.CPP”保存。程序保存对话框如图 1.3 所示。

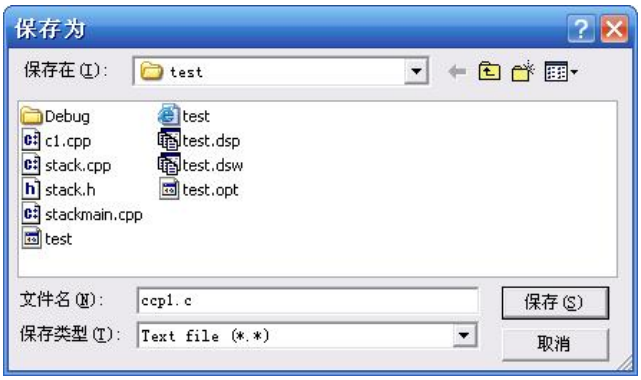

图 1.3　“保存为”对话框

1.5　执行程序

首先要生成可执行文件。使用 VC++“编译”菜单中的“构件”命令，如图 1.4 所示，也可使用快捷键 F7。在编译、链接过程中 VC++将保存新输入的程序，并生成一个同名的工作区。保存文件时需填入文件名，如“4-1.C”。假如不指定扩展名“.C”，VC++会把扩展名自动定义为“.CPP”，即 C++程序。

图 1.4　编译菜单

如果程序没有错误，将在如图 1.5 所示的信息窗口中显示内容：

0 error(s), 0 warning(s)

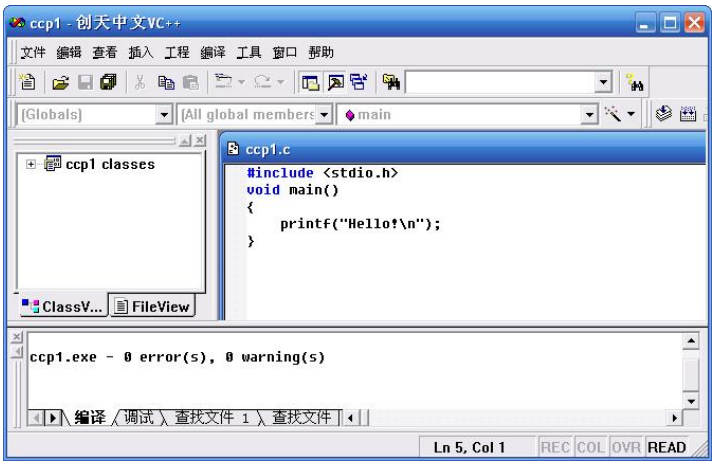

图 1.5　编译链接正确

表示没有任何错误。有时出现几个警告性信息（warning）并不影响程序执行。假如有致命性错误（error），如图 1.6 所示，双击某行出错信息，程序窗口中会指示对应的出错位置，可根据信息窗口的提示分别予以纠正。然后用“编译”菜单中的“执行”命令（或快捷键 Ctrl+F5）执行程序。

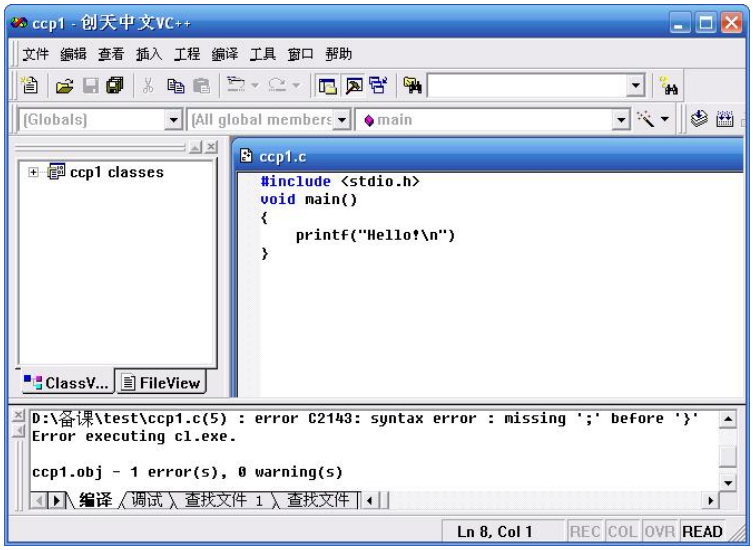

图 1.6　编译链接出错

当运行 C 程序后，VC++将自动弹出程序运行窗口，如图 1.7 所示。按任意键将关闭该窗口。

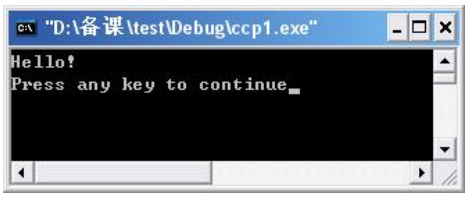

图 1.7　程序运行窗口

对于编译、链接和执行操作，VC++还提供了一组工具按钮，如图 1.8 所示。

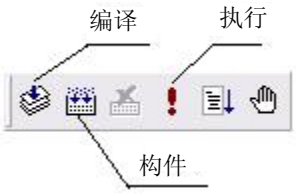

图 1.8　编译连接执行工具按钮

1.6　关闭程序工作区

当一个程序编译链接后，VC++系统自动产生相应的工作区，以完成程序的运行和调试。若想执行第二个程序时，必须关闭前一个程序的工作区，然后通过新的编译链接，产生第二个程序的工作区。否则，运行的将一直是前一个程序。“文件”菜单提供关闭程序工作区功能，如图 1.9 所示，执行“关闭工作区”命令，在如图 1.10 所示的对话框中单击“否”按钮。如果单击“是”按钮将同时关闭源程序窗口。

图 1.9　“文件”菜单

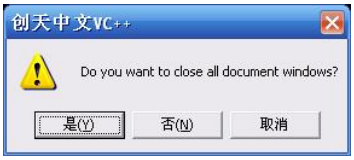

图 1.10　文件关闭对话框

VC++ 6.0 中常用快捷键如下：

Ctrl+N：新建程序；　　Ctrl+O：打开程序；

Ctrl+S：保存程序；　　Ctrl+F7：Compile（编译）；

F7：Build （编译链接）；　　Ctrl+F5：Execute Program（执行程序）；

F5：Go（开始调试）；　　F9：Insert/Remove BreakPoint（插入/删除断点）；

F11：Step into（单步调试，可进入函数体内部）；

F10：Step over（单步调试，不能进入函数体内部）；

Ctrl+Break：Stop Build（停止编译链接）；

Ctrl+F10：Run to Cursor（运行到光标处）；

Shift+F5：Stop Debugging（取消调试）。

1.7　命令行参数处理

VC++是一个基于窗口操作的 C++系统，没有提供命令行参数的功能，需要在 Windows 的“MS-DOS 方式”窗口里以命令方式实现。具体步骤参考如下：

（1）正确编译连接，生成可执行程序。

（2）通过“我的电脑”或“资源管理器”找到要运行的 C 源程序（设为 a.c）。

（3）进入 debug 文件夹（它包含 a.c 程序的可执行文件 a.exe）。

（4）执行“开始”菜单的“运行”命令，填入 a，然后单击“确定”按钮。

（5）在打开的“MS-DOS 方式”窗口中输入：a 参数 1 参数 2…，带参数运行程序。

1.8　程序调试简介

除了较简单的情况之外，一般的程序都很难一次就能完全正确。在上机过程中，根据出错现象找出错误并改正的过程称为程序调试。我们在学习程序设计的过程中，逐步培养调试程序的能力是非常重要的。这是一种经验的积累，不可能单单凭几句话就可以描述清楚，要靠读者在上机练习中不断进行摸索、总结。

程序中的错误大致可分为三类：

（1）编译错误。编译错误是指程序编译时检查出来的语法错误。编译错误通常是编程人员违反了 C 语言的语法规则，如大括号不匹配、语句少分号等。

（2）链接错误。链接错误是指程序链接时出现的错误。链接错误一般是由于未定义或未指明要链接或包含的函数，或者函数调用不匹配等因素而引起的。

对于编译错误和链接错误，C 语言系统会提供出错信息，包括出错位置（行号）、出错提示信息。编程人员可以根据这些信息，找出错误所在。

注意：有时系统会提示一大串错误信息，但并不表示真的有那么多的错误。这往往是因为前面的一、两个错误带来的。所以当纠正了前几个错误后，可再编译一次，然后根据最新的出错信息继续纠正。VC++ 6.0 中 C 语言编程的常见错误可参考本书附录 D。

（3）运行错误。运行错误是指程序执行过程中的错误。有些程序虽然通过了编译链接，并能够在计算机上运行，但得到的结果不正确。这类错误相对前两种错误较难改正，须要求编程人员认真分析程序的执行过程，从而找出错误所在。

错误的原因一种可能是程序书写错误带来的，例如应该使用变量 x 的地方写成了变量 y，虽然没有语法错误，但意思完全错了；另一种可能是程序的算法不正确，解题思路不对。还有一些程序的计算结果有时正确，有时错误，这往往是编程时对各种情况考虑不周所致。解决运行错误的首要步骤就是错误定位，即找出出错的位置，才能予以纠正。通常先设法确定错误的大致位置，然后通过调试工具找出真正的错误。

程序调试可通过 Build 菜单下 Start Debug 子菜单进行，或在工具栏空白处右击再选择快捷菜单中的 Debug 工具条选项，还可以使用调试微型条辅助进行，如图 1.11 所示。

图 1.11　调试微型条

以下通过三个例子来讲解如何进行程序调试。

【例 1.1】 调试编译错误和链接错误。

下面的程序已经按照要求完成了编辑过程，命名为：Hello World.c，以该源文件为例进行编译错误调试。

调试前，故意把倒数第二行末尾的分号删除后，再进行编译。如图 1.12 所示，得到错误提示信息为：“语法错误，在‘}’前缺少‘;’”。

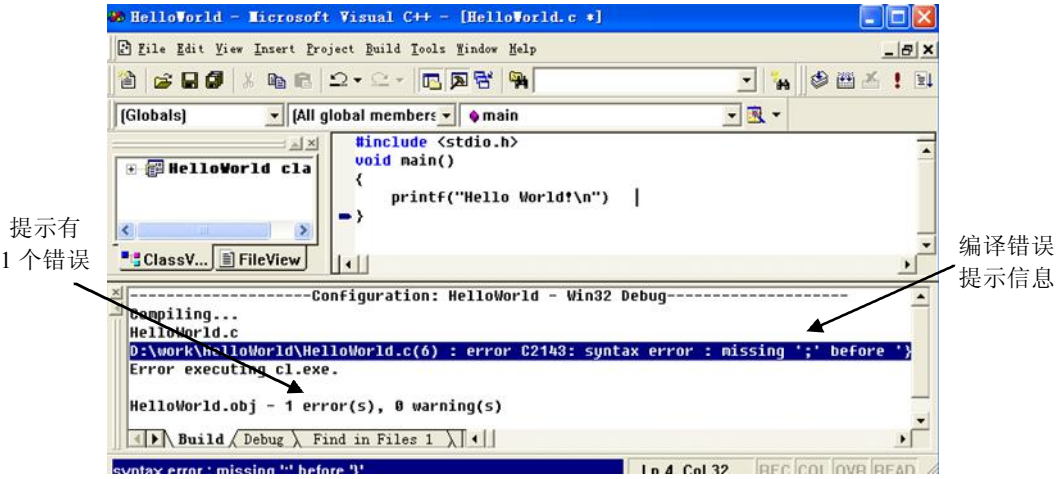

图 1.12　编译错误提示窗口

此时，可在错误信息上双击，VC++ 6.0 将在输出窗口高亮显示该行提示信息，并切换到

出错的源文件编辑窗口。可以看到，在编辑窗口左侧的蓝色箭头指向了错误所在行。根据错误提示信息，把“;”加在错误行前一行的末尾，修改后再进行调试。

注意：根据错误信息直接修改错误是改正编译错误和链接错误的通用方法。

【例 1.2】 调试运行错误。

程序的源文件为：

```
#include <stdio.h>
void main( )
{
    int a, b, c;                    //定义 3 个整型变量 a，b，c
    a = 3;                          //把数值 3 存入变量 a 中
    b = 2;                          //把数值 2 存入变量 b 中
    c = a + b;                      //把 a+b 的结果存入变量 c 中
    printf("%d + %d = %d\n", a, b, c);   //在屏幕上输出 c 的值，屏幕显示为：3+2=5
}
```

以该源文件为例进行运行错误调试。

调试前，故意把程序倒数第三行的“+”号改成“-”号后，再进行编译。编译和链接均通过，但运行结果为 1，而不是 5。因此，初步估计错误发生在倒数第三行 c=a-b。下面通过调试找出真正的错误。

① 在该行位置上单击，选择图 1.11 所示调试微型条的按钮，设置一个断点。此时，该行前面出现一个红色圆点标志。也可以先将鼠标定位在此行，再选择调试微型条的按钮也可以达到同样效果。

注意：断点通常用于调试较长的程序，且程序可同时设置多个断点；而按钮的功能是程序运行到光标处暂停。

② 选择调试微型条的按钮开始进行调试。当程序运行到该行时就会暂停，如图 1.13 所示，编辑窗口中左侧的彩色箭头表示当前程序暂停的位置。

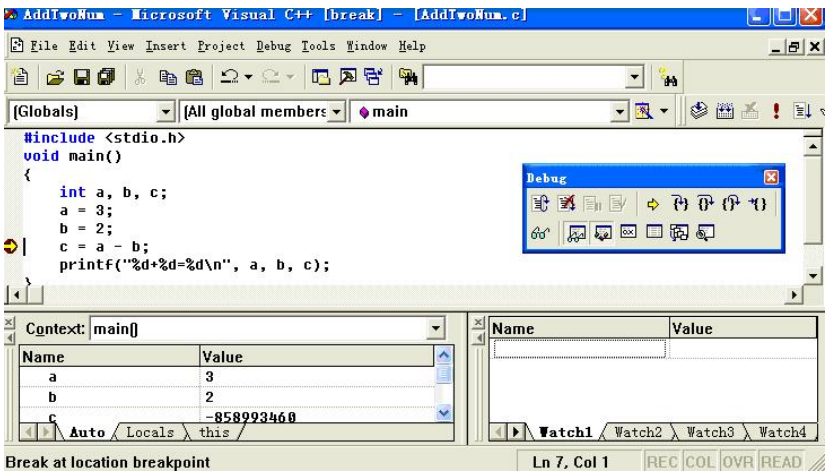

图 1.13　程序在断点处暂停

此时，在图 1.13 的左下角窗口中系统自动显示了有关变量的值，其中 a 和 b 的值分别是 3、2，但变量 c 的值是任意值。这是因为程序刚运行到此处，并未执行 c=a-b 语句，因而还未对

变量 c 赋值。

③ 选择调试微型条的{}按钮进行单步调试。此时，箭头下移一行。如图 1.14 所示，左下角窗口中 c 的值被更新为 1（注意：变量的值更新后用红色表示）。真正的错误就是发生在这一行。因为程序需要完成的是 3+2 的运算，而非 3-2 的运算。

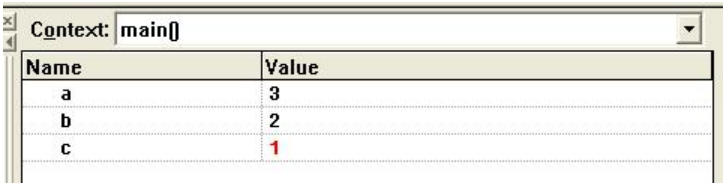

图 1.14 变量观察窗口

④ 找到真正的错误后，单击调试微型条上的按钮结束调试，同时返回到程序编辑窗口进行修改。

如程序仍需再次调试，可重复以上步骤。

注意：使用断点可以使程序暂停。一旦设置了断点，无论是否还需要调试，每次执行程序时都会在断点上暂停。因此调试结束后应取消所定义的断点。方法是先把光标定位在断点所在行，再单击“调试微型条”中的按钮。该按钮是一个开关，第一次单击是设置断点，第二次单击是取消断点。如果想取消全部断点，可单击“Edit”菜单中的“Breakpoints”菜单项，屏幕上会显示“Breakpoints”窗口。窗口下方会列出程序中设置的所有断点，单击“Remove All”按钮，将取消所有断点。

如果一个程序设置了多个断点，按一次快捷键 Ctrl+F5 会暂停在第一个断点，再按一次快捷键 Ctrl+F5 会继续执行到第二个断点暂停，依次执行下去。

【例 1.3】 函数跟踪调试。输入两个数，输出其中的较大值。

为讲解此部分，给出下面的例子，程序命名为 max.c。

注意：函数相关知识点将在后面章节介绍，此处仅介绍函数的调试跟踪方法。

```
#include <stdio.h>
int max(int x, int y)                    //自定义 max 函数，x、y 为形参
{
    int z ;
    if (x > y)
        z = x;
    else
        z = y;
    return(z);
}
void main( )
{
    int a, b, c;
    scanf("%d, %d", &a, &b);
    c = max(a, b);                       //调用 max 函数，求 a 和 b 中大数，其中 a 和 b 为实参
    printf("max = %d\n", c);
}
```

输入及程序运行结果如下：

2, 3↵

max = 3

① 单击主函数的 scanf 语句所在行，再单击调试微型条的{}按钮。程序运行到此处会暂停。

② 单击调试微型条的{}按钮，进行单步调试，程序执行“scanf("%d, %d", &a, &b);”语句。此时，需要在已经打开的控制台窗口中输入变量 a,b 的值，如输入 2,3↵。输入完成后，编辑窗口中光标已经下移一行，指向“c = max(a, b);”语句。

③ 由于函数调用的实质是实参传递给形参，因此需要跟踪进入到函数体内部进行检查。此时，单击调试微型条的{}按钮进入 max 函数，如图 1.15 所示，光标进入 max 函数体。同时，在左下角窗口中可看到形参 x、y 接收了实参 a、b 传递过来的值，因此 x=2，y=3。

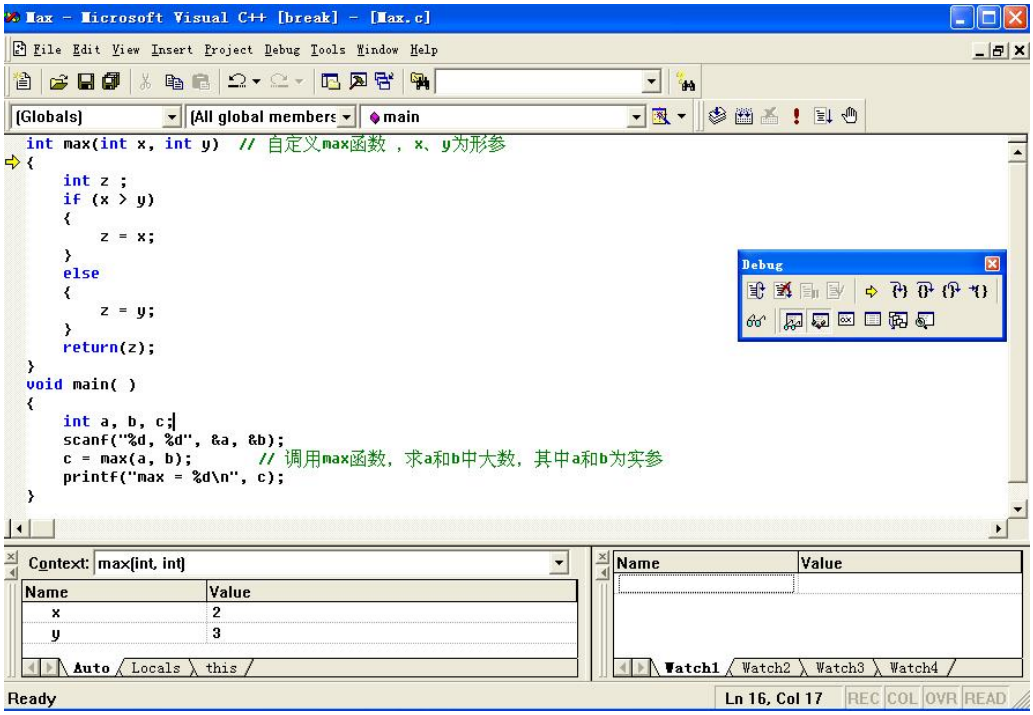

图 1.15　跟踪进入函数体内部

注意：两个单步调试按钮的区别。

{} step into 单步调试，进到函数体内部。

{} step over 单步调试，不进到函数体内部。

④ 单击{}按钮，继续单步调试。由于 x<y，所以程序运行到双分支 if 语句结构时将执行 else 后的语句。

⑤ 单击{}按钮，程序执行“z = y”语句。此时，z 被赋值为 3。

⑥ 单击{}按钮，程序执行“return(z)”语句，把 z 的值返回给调用函数，即 main 函数。

⑦ 单击{}按钮（或单击{}按钮，跳出 max 函数体），程序已返回到 main 函数“c = max(a, b);”语句行。

⑧ 再次单击按钮，如图 1.16 所示，c 的值被更新为 3，即 max 函数返回值。

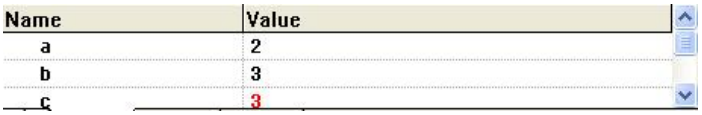

图 1.16　函数调用后的变量观察窗口

⑨ 单击按钮，程序执行“printf("max = %d\n", c);”语句，控制台窗口显示“max=3”。

⑩ 单击调试微型条上的按钮，结束调试。