**VS2015调试工具使用报告**

**班级：计算机一班**

**姓名：王哲源**

**学号：1652228**

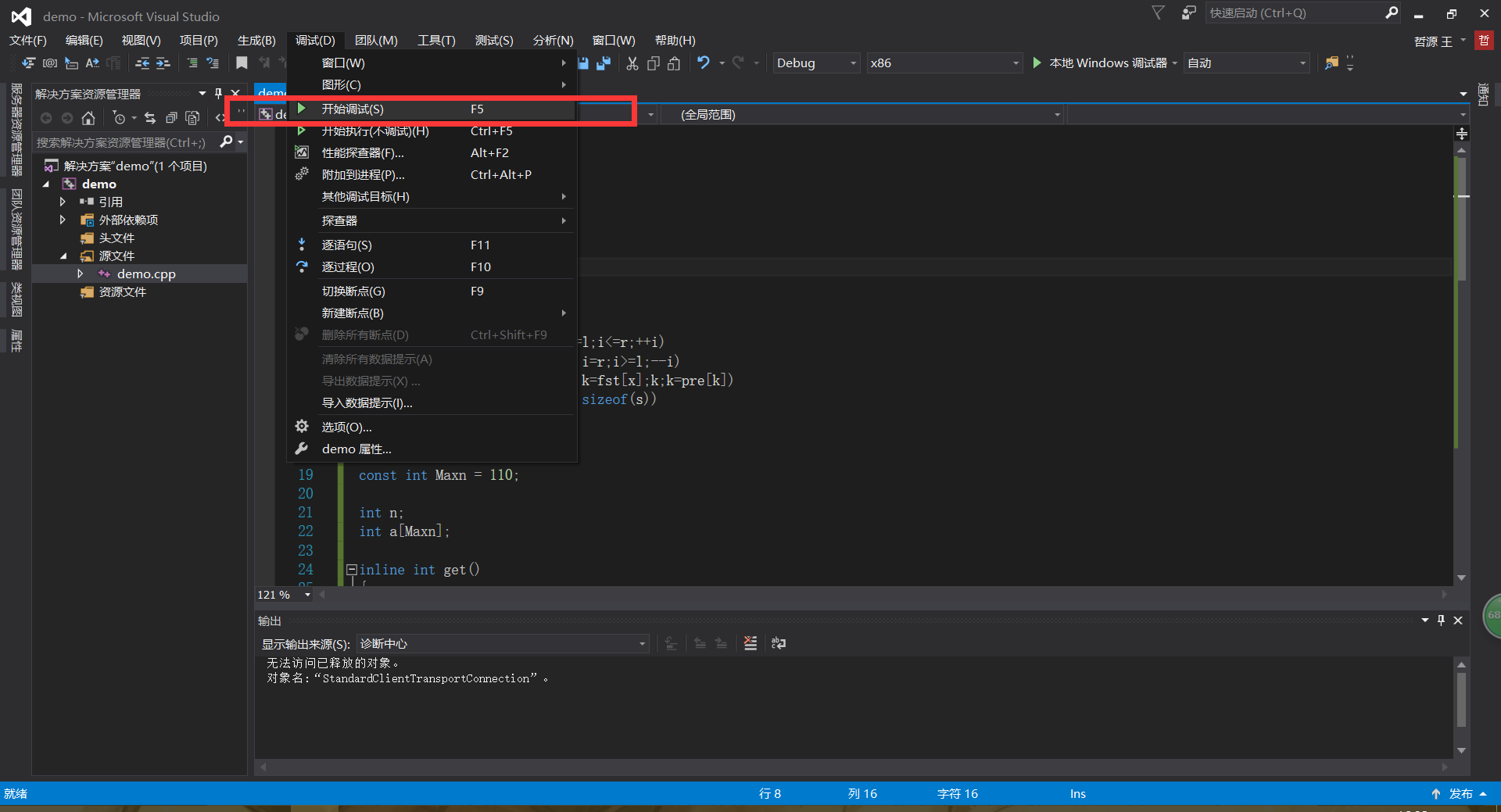
**完成日期：2016.12.22**

1.VS2015环境下调试工具的基本使用

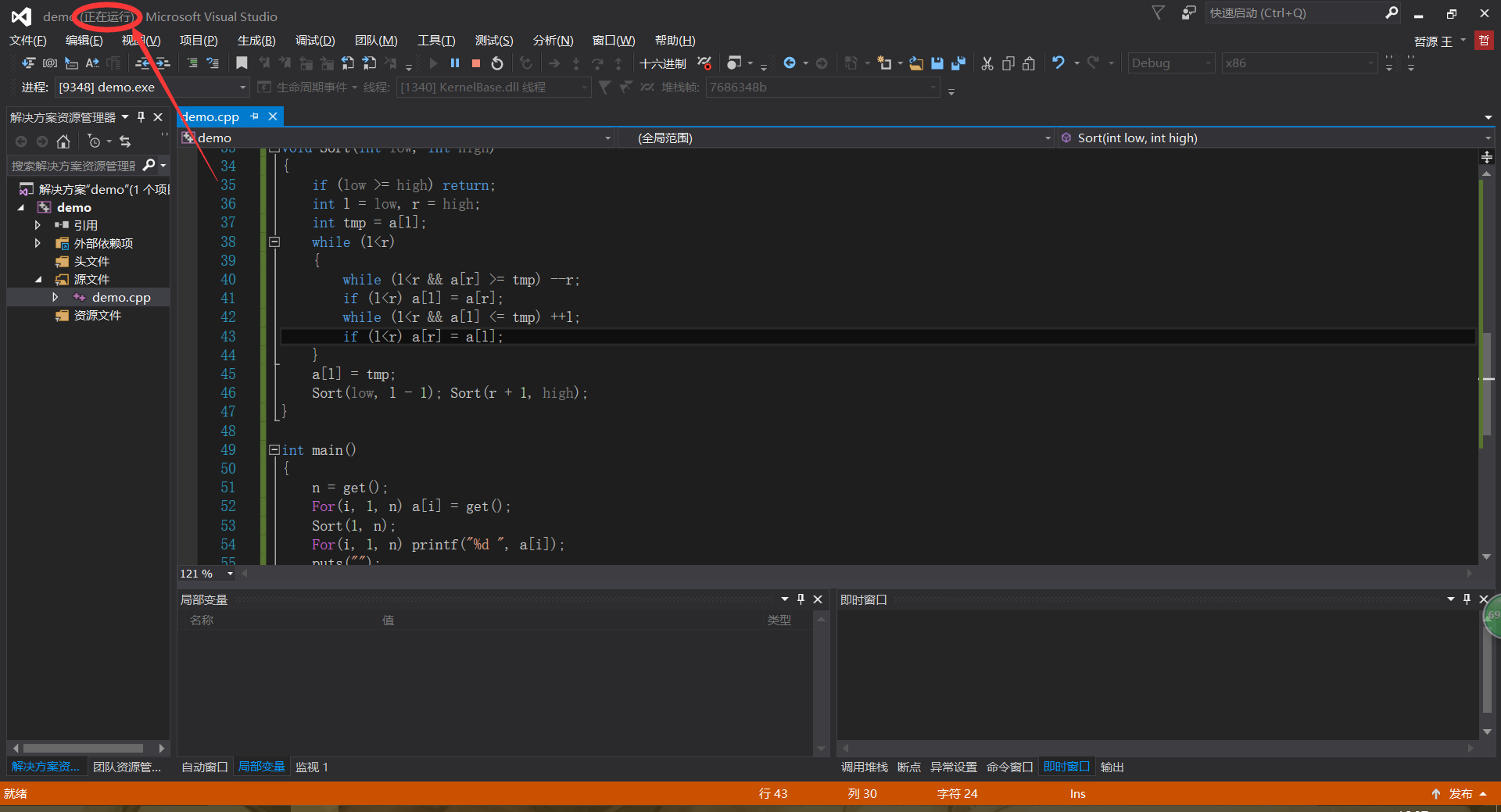
**1.1调试的开始及结束**

**1.1.1如何开始调试**

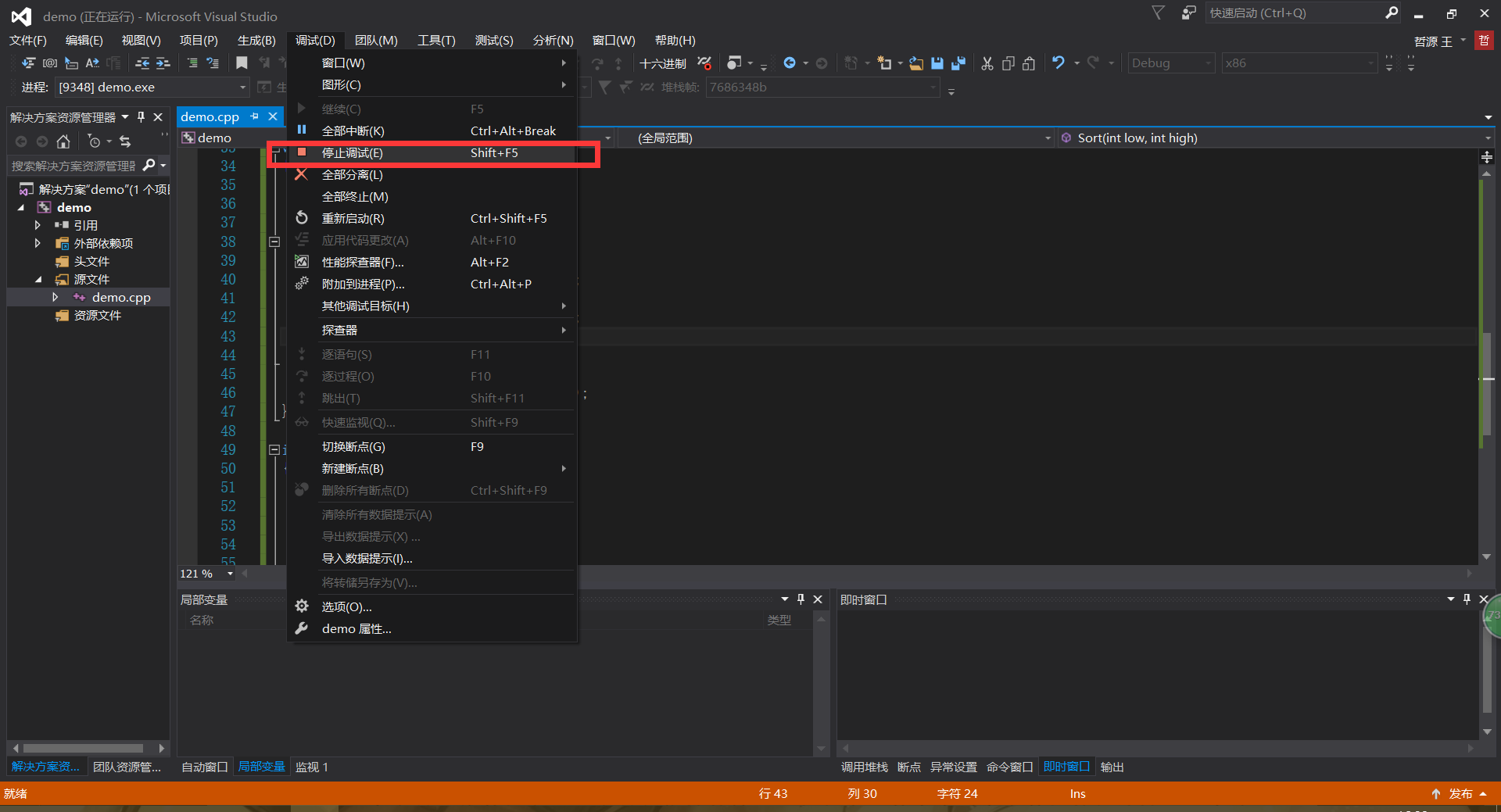
程序编写结束之后，选择工具栏中“调试-开始调试”（或者按下快捷键F5）进入调试



接下来若显示该界面，则说明已进入调试模式



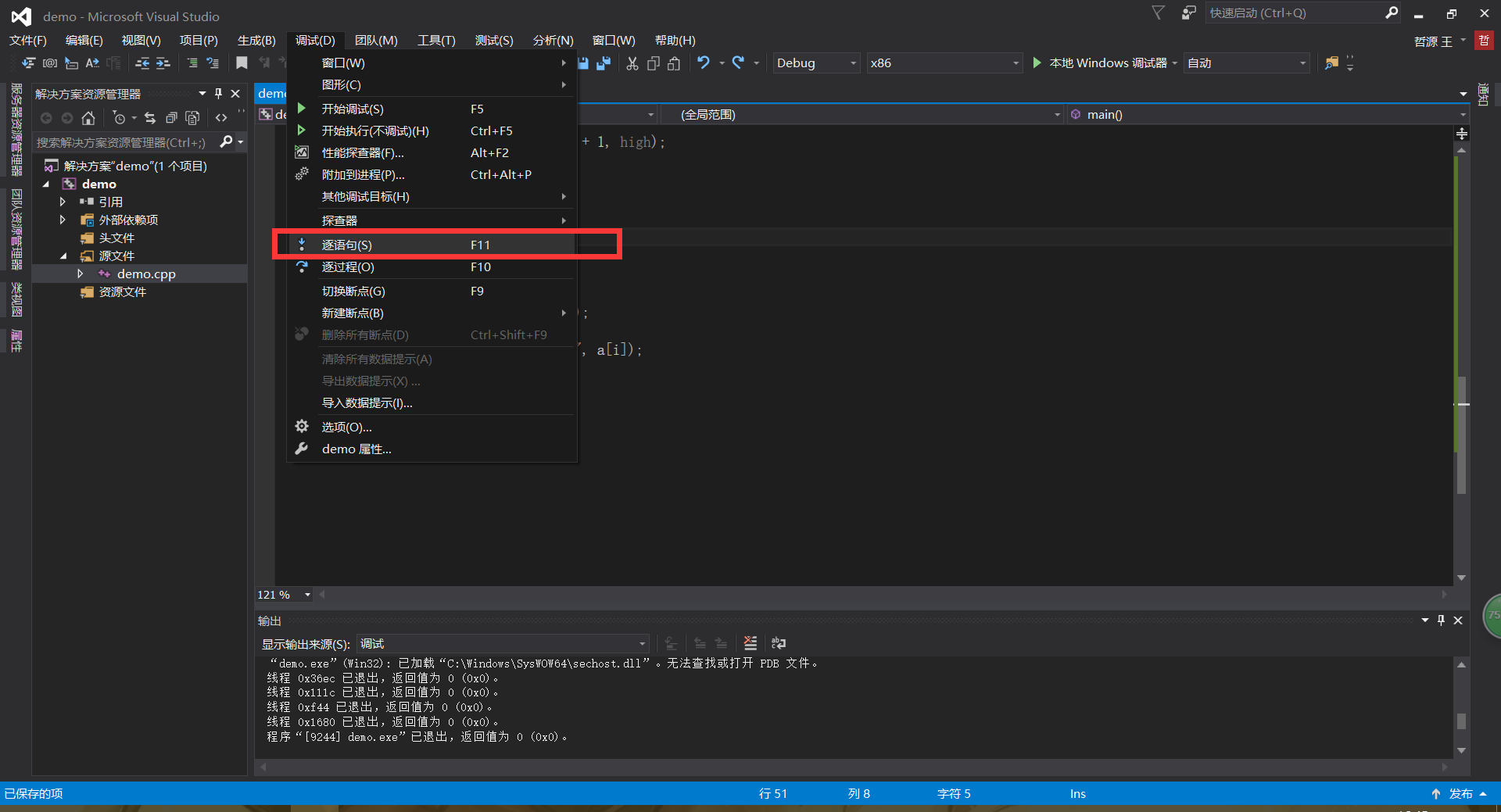
**1.1.2如何结束调试**

****

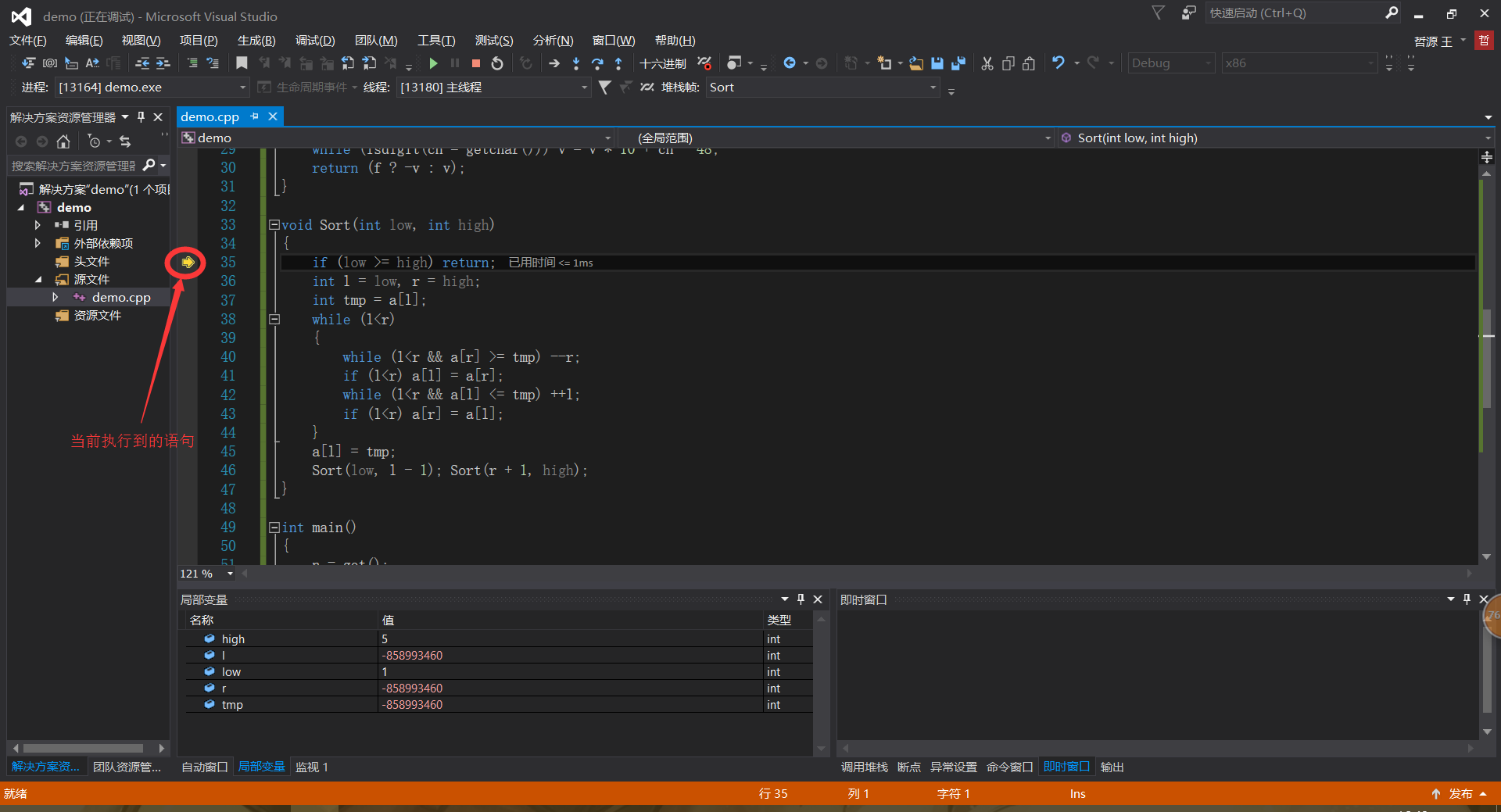
选择工具栏中“调试-停止调试”（或使用快捷键Shift+F5）即可退出调试模式

**1.2逐语句单步执行**

程序编写完毕之后，选择工具栏“调试-逐语句”（或快捷键F11）即可进入调试模式并从程序开头进行调试。

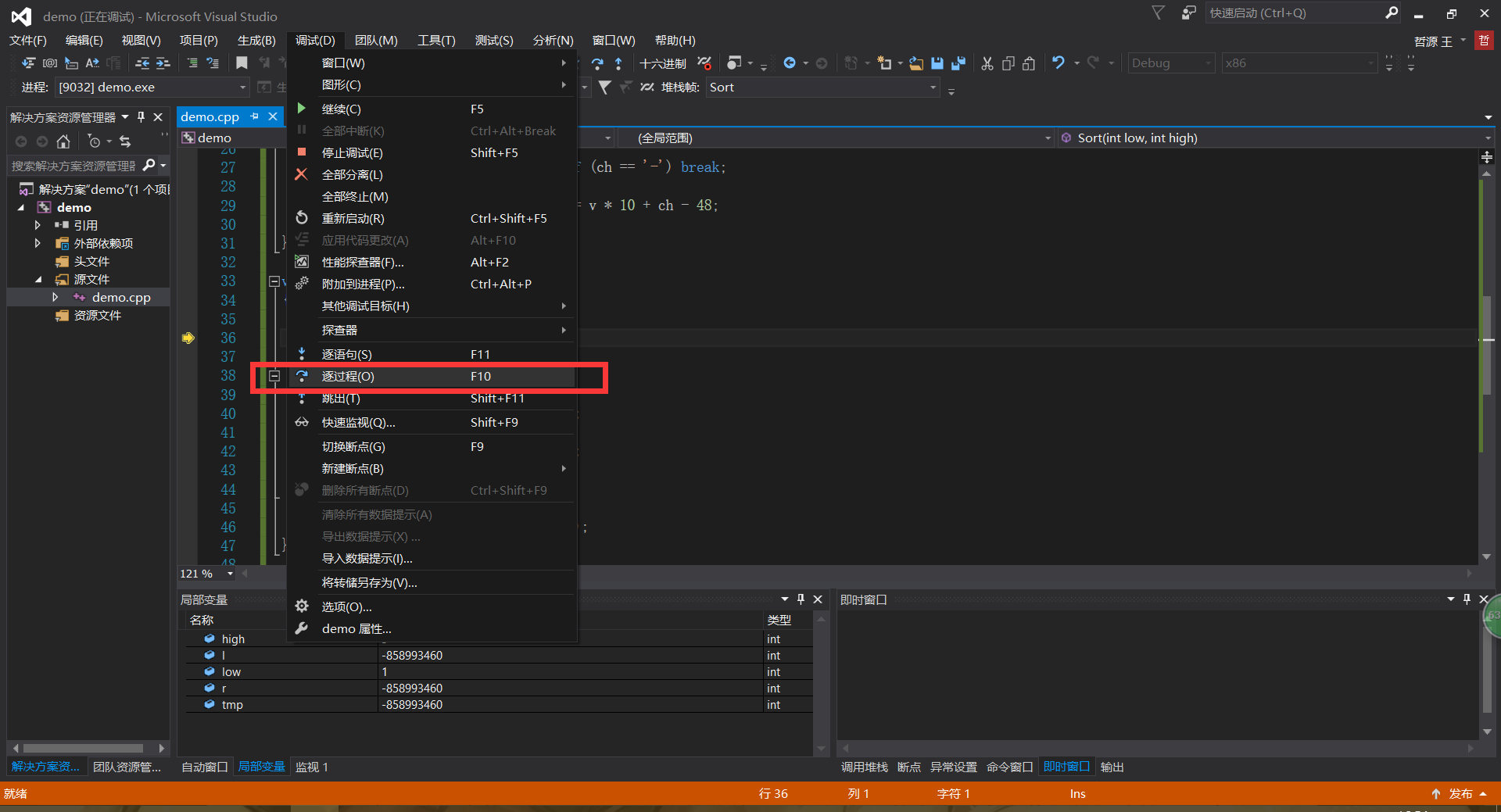


在函数中按F11（推荐快捷键，后同）便可实现逐语句单步执行



**1.3在碰到 cout/sqrt 等系统类/系统函数时，一步完成这些系统类/系统函数的执行而不进入这些系统类/函数的内部单步执行的方法**

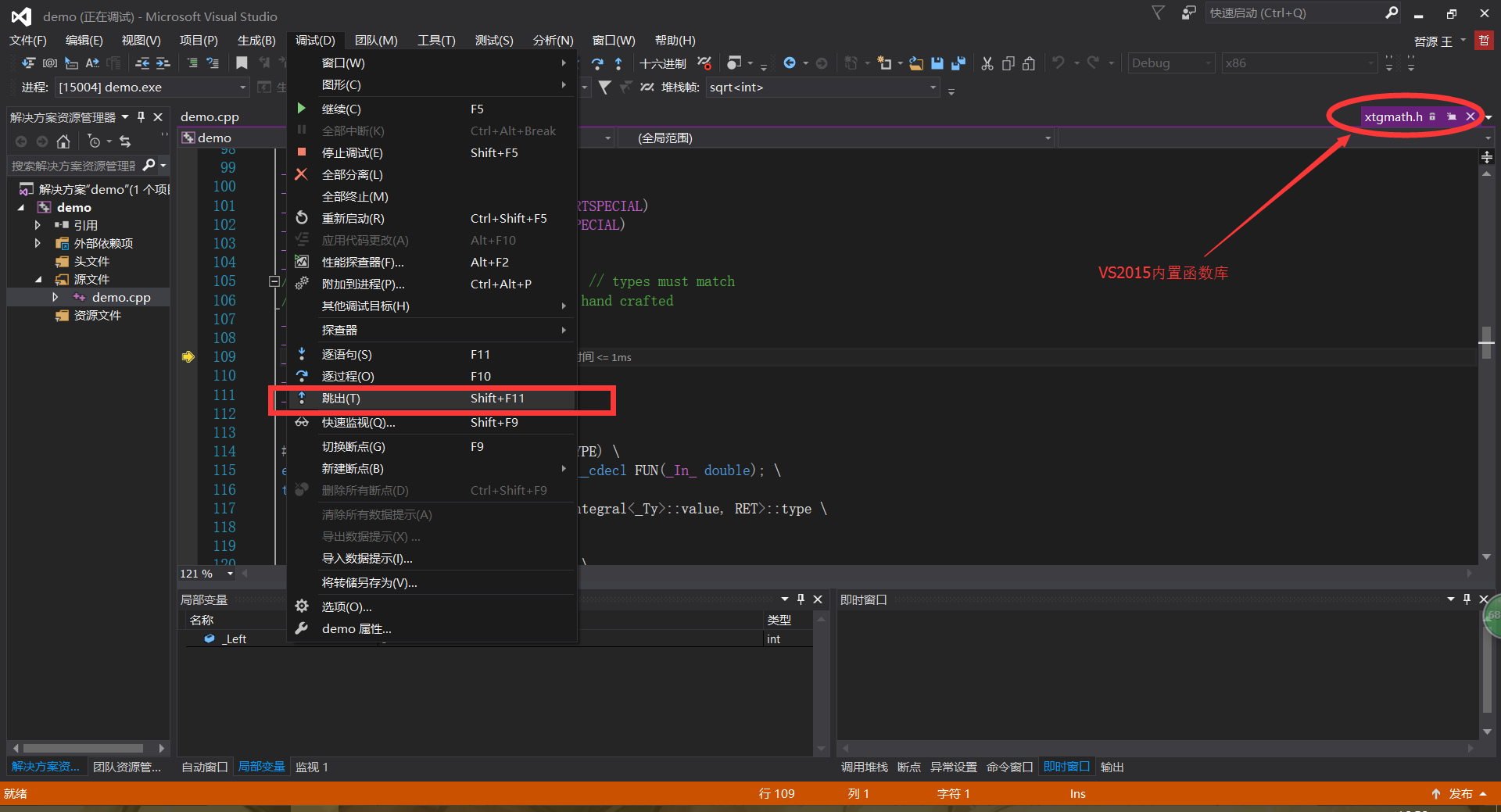
在遇到这些函数时，选择工具栏中“调试-逐过程”（或快捷键F10）即可。同1.2，代码完成后按F10也可进入调试模式，之后不再赘述。

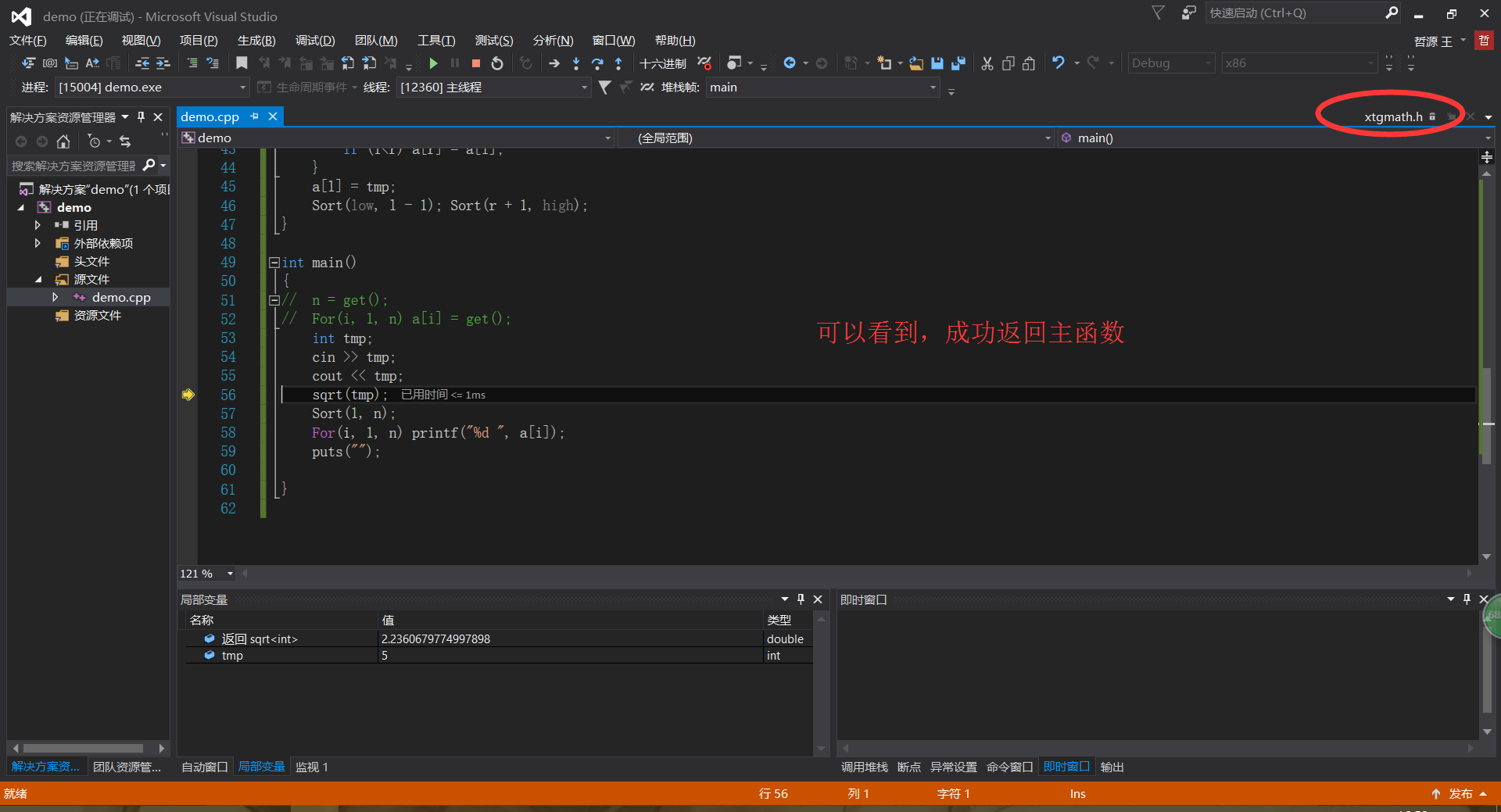


**1.4不慎进入到 cout/sqrt 等系统类/系统函数的内部，跳出并返回自己的函数的方法**

这里以不慎进入sqrt函数为例

在进入系统类/函数内部时，选择工具栏中“调试-跳出”（或快捷键Shift+F10）即可





**1.5碰到自定义函数调用时一步完成函数而非进入函数进行执行**

操作方式同1.3中遇到系统类/函数的方法一致，可参考1.3中的操作方法。这里不再赘述。

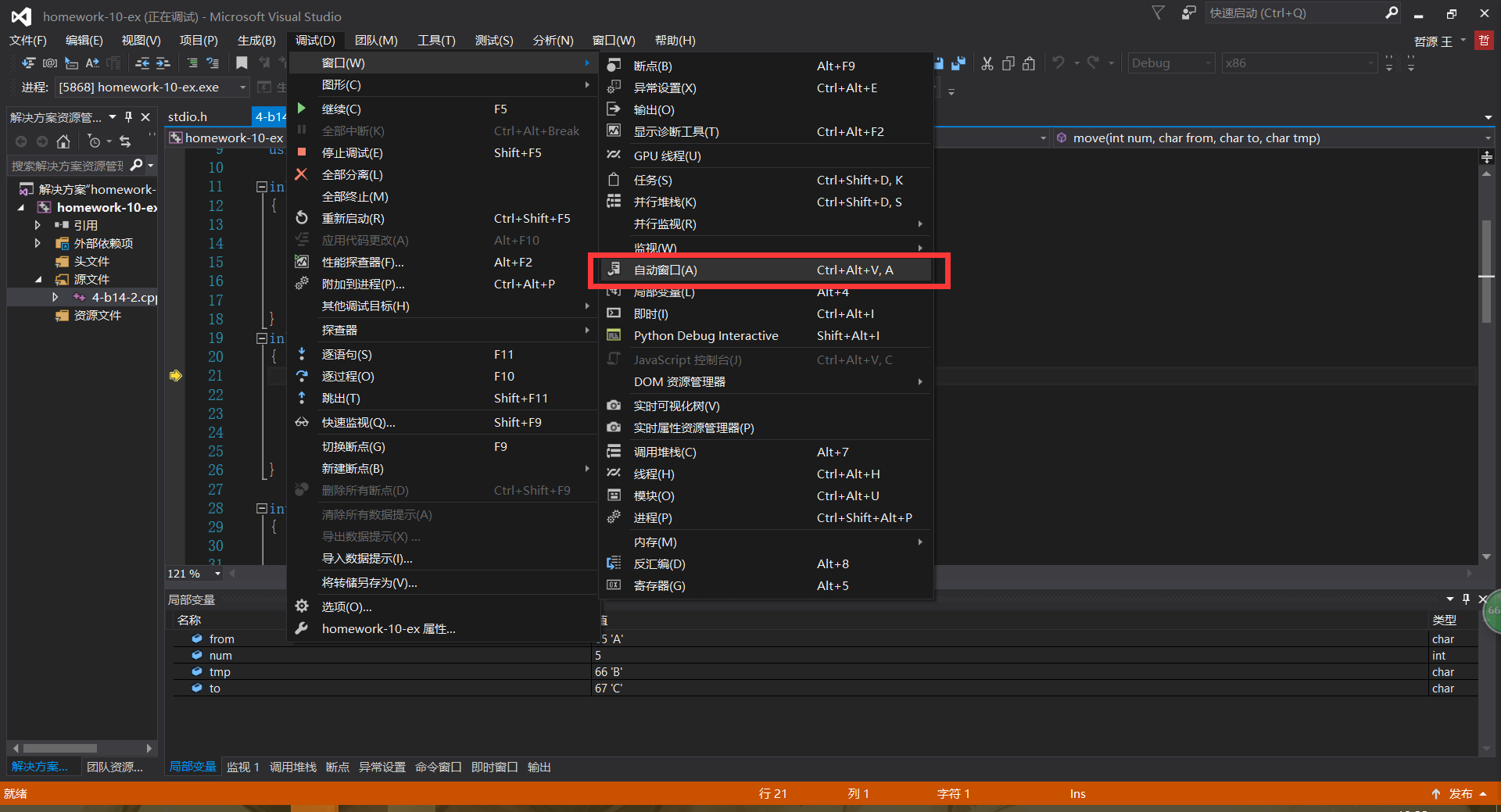
**1.6不慎进入自定义函数调用语句时的处理方法**

操作方法同1.4中不慎进入系统类/函数的解决方法一致。可参考1.4中的解决方法。这里不再赘述

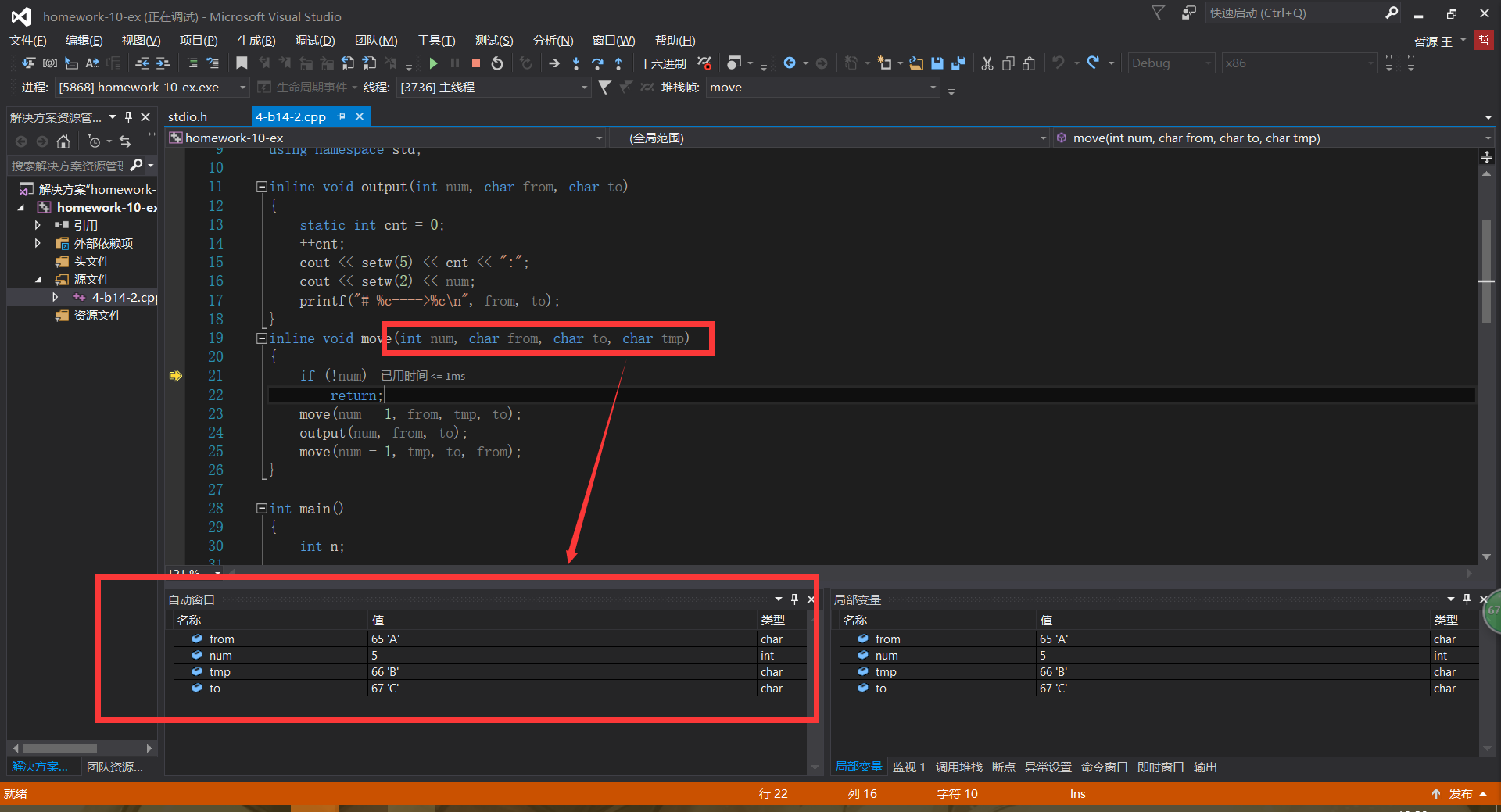
2. 用 VS2015 的调试工具查看各种生存期/作用域变量的方法

**2.1 查看形参/自动变量的变化情况**

在调试状态下选择工具栏“调试-窗口-自动窗口”（或快捷键 Ctrl+Alt+V后再按A）

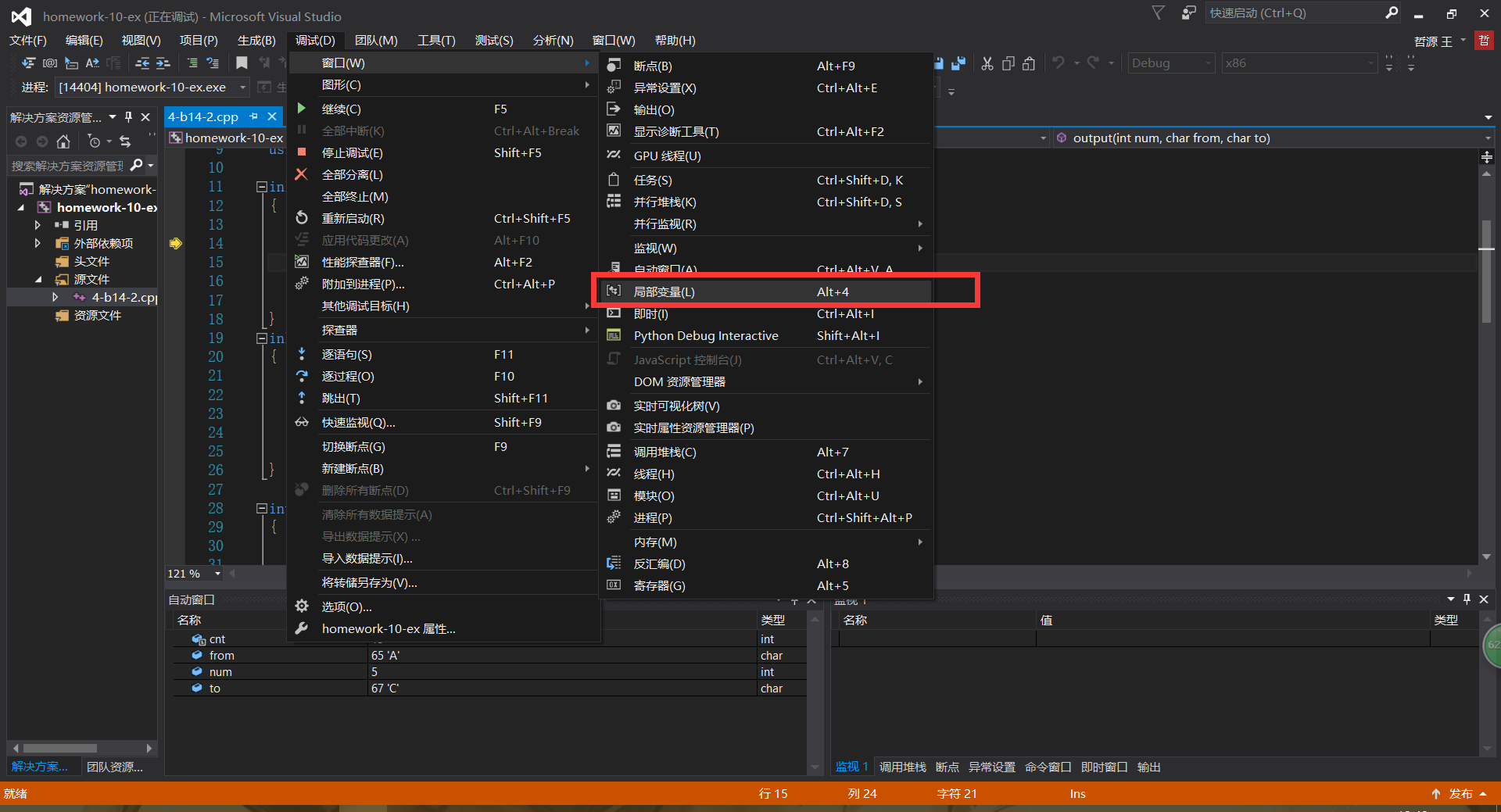


如下图出现形参数值



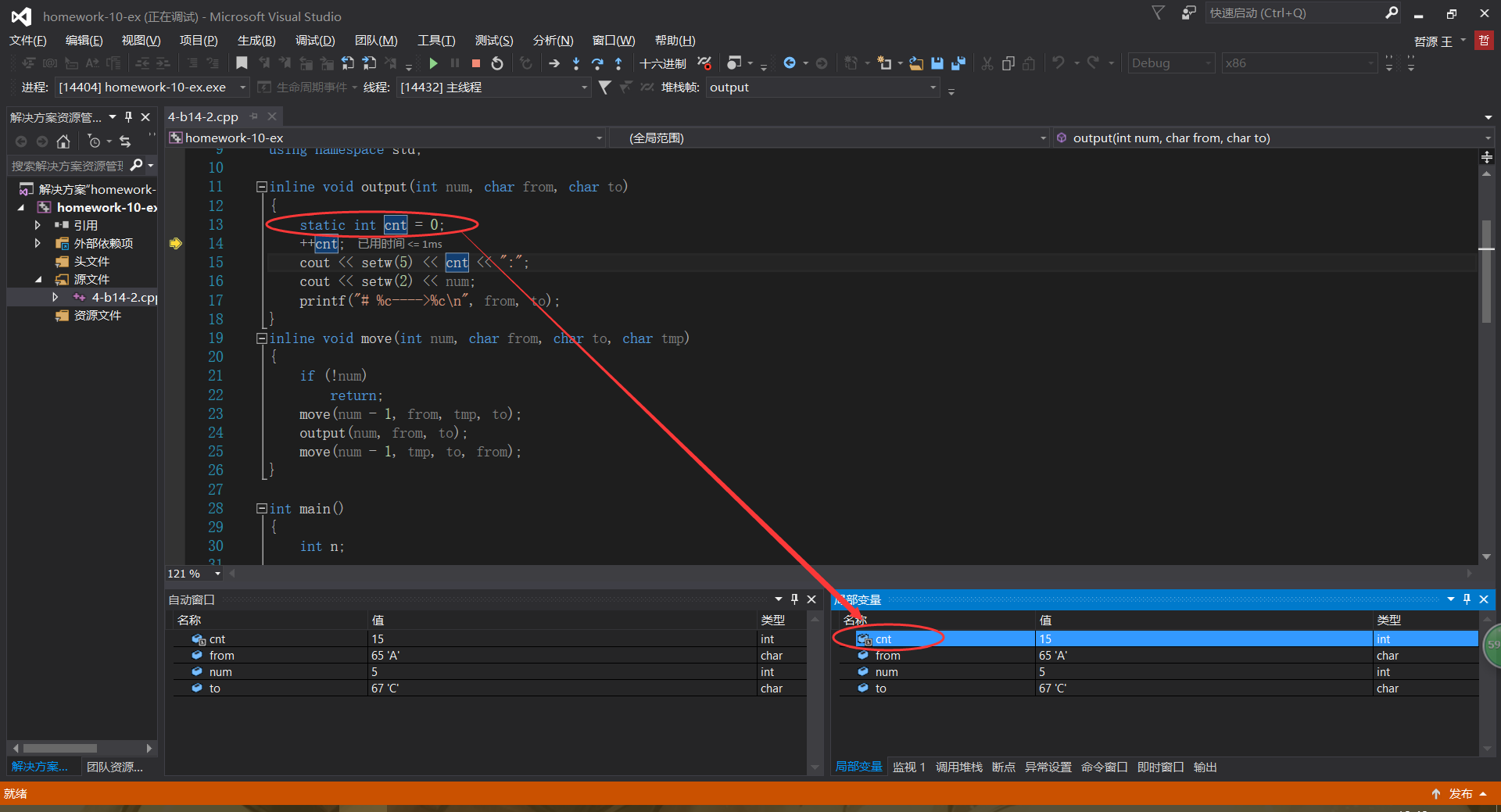
**2.2 查看静态局部变量的变化情况（/函数体外）**

在调试模式下，选择工具栏“调试-窗口-局部变量”（或快捷键Alt+4），将局部变量窗口打开



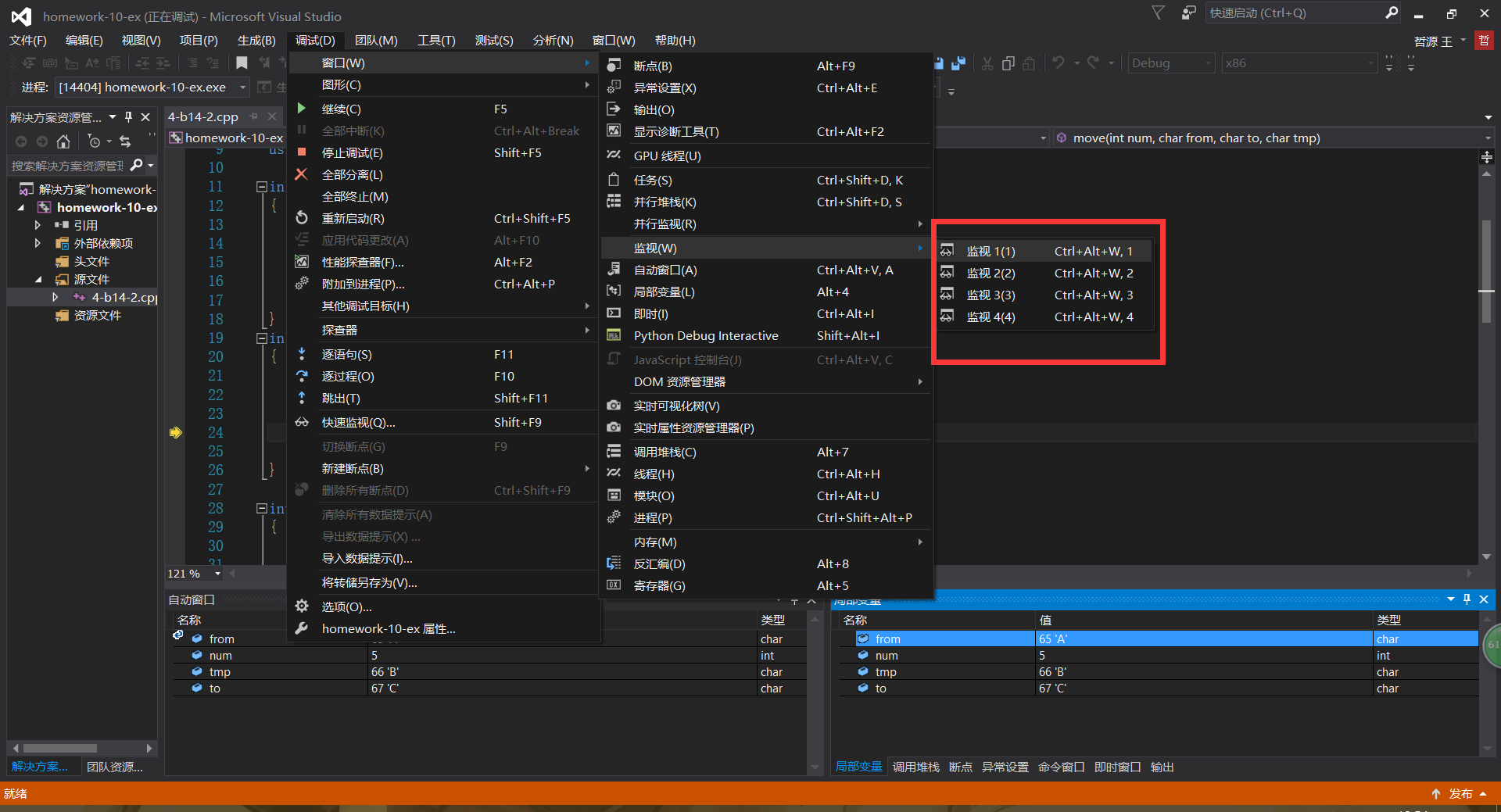
**2.2.1 该静态局部变量所在的函数体内**

出现以下界面，其中可以发现静态局部变量与动态局部变量的图标有所差别

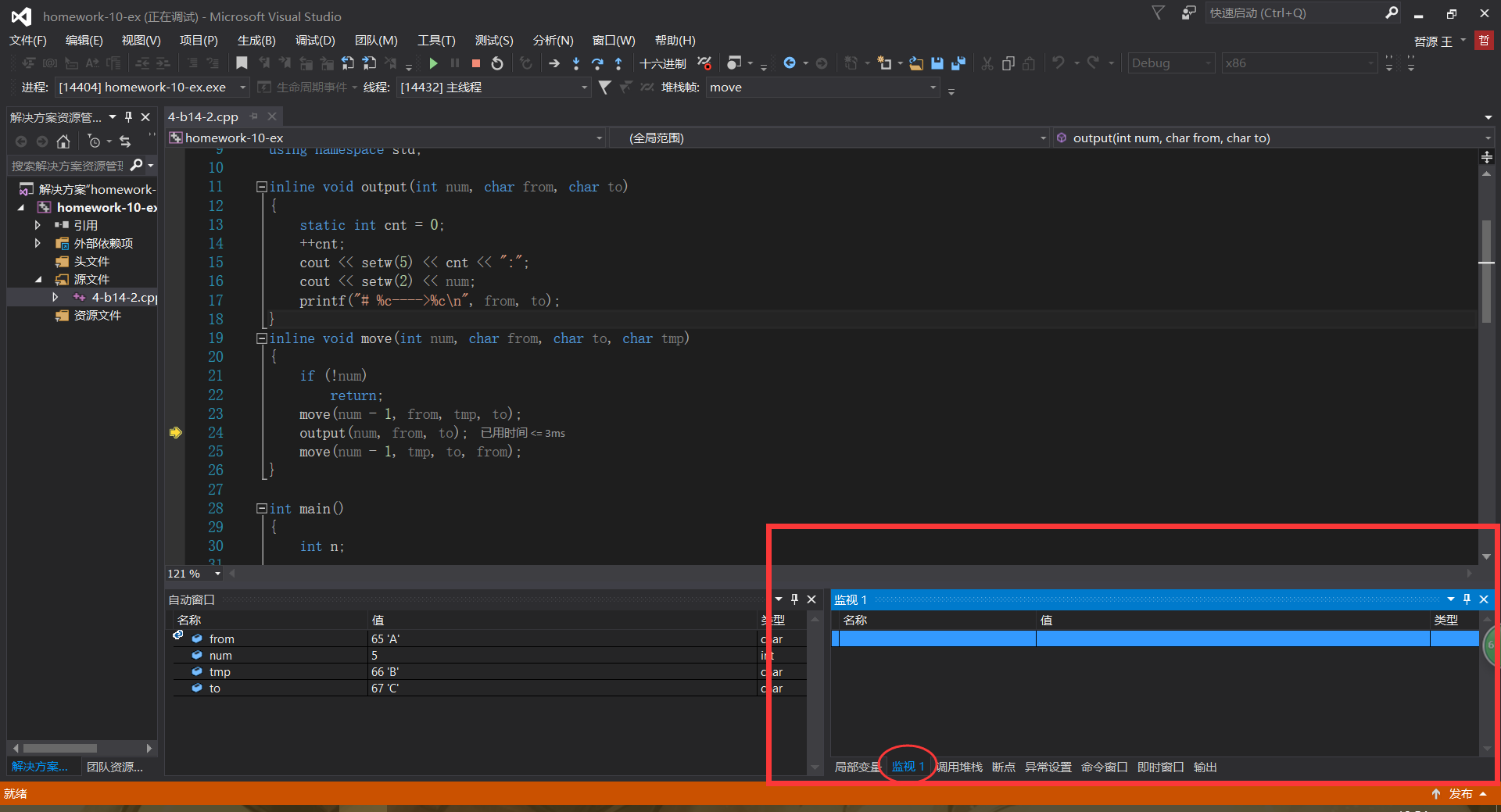


**2.2.2 该静态局部变量所在的函数体外**

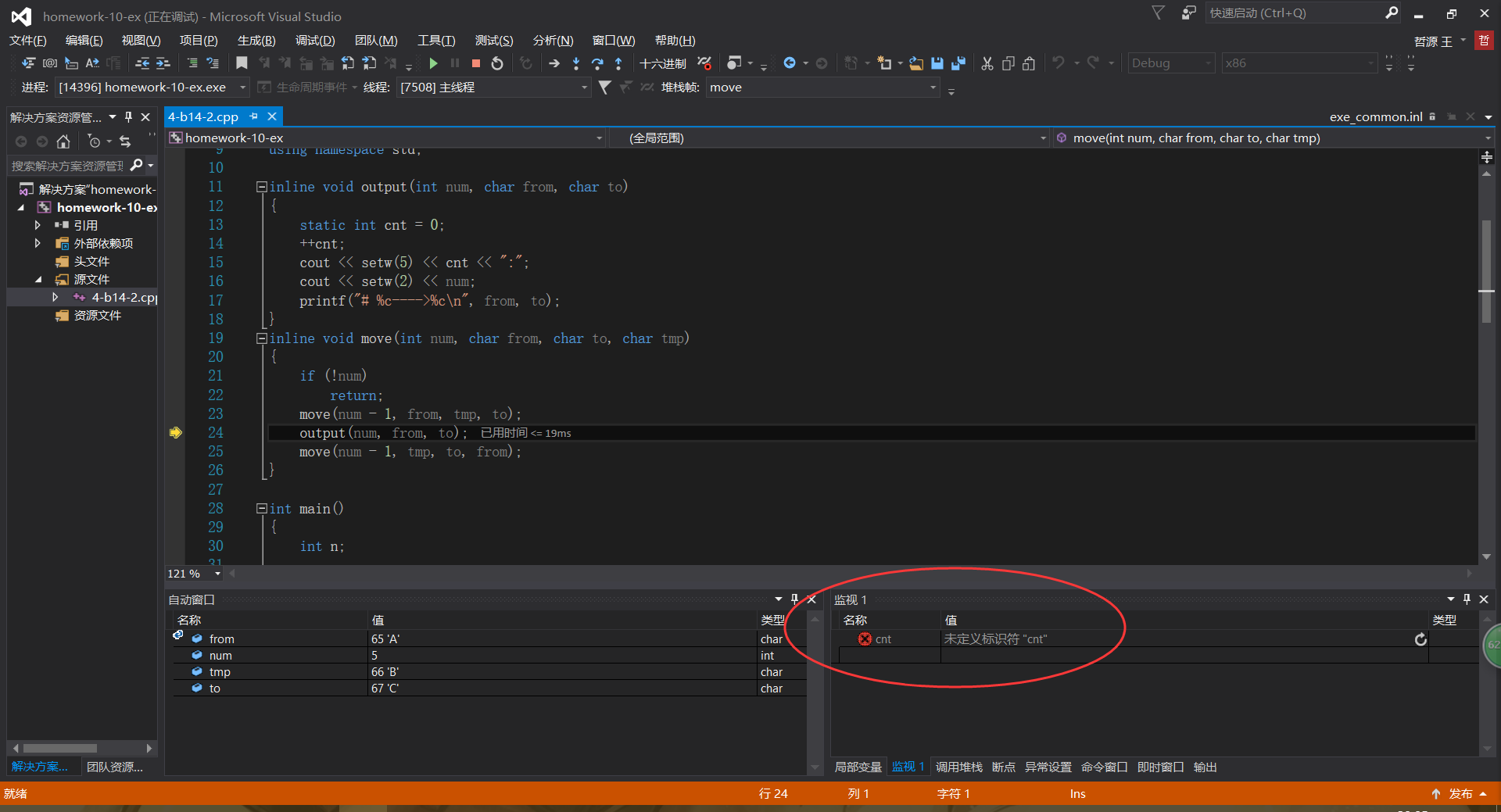
选择工具栏“调试-窗口-监视”中任意一个监视选项（快捷键Alt+Ctrl+1/2/3/4）（这里以监视1为例）



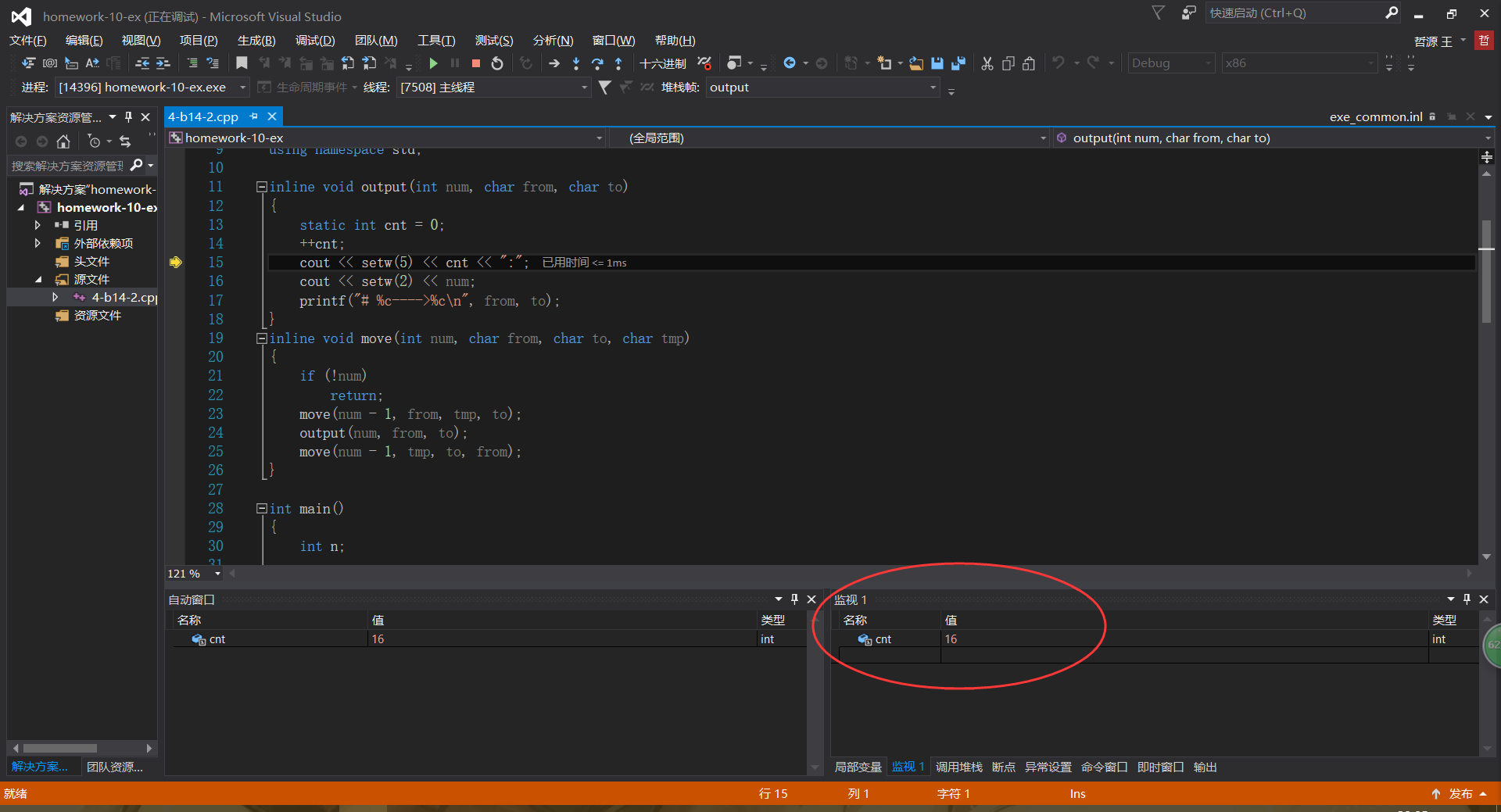
即可打开监视窗口，如下



在其中输入想要查看的静态局部变量名后敲回车



此时会显示未定义标识符，没关系，我们执行该静态局部变量存在的函数一次



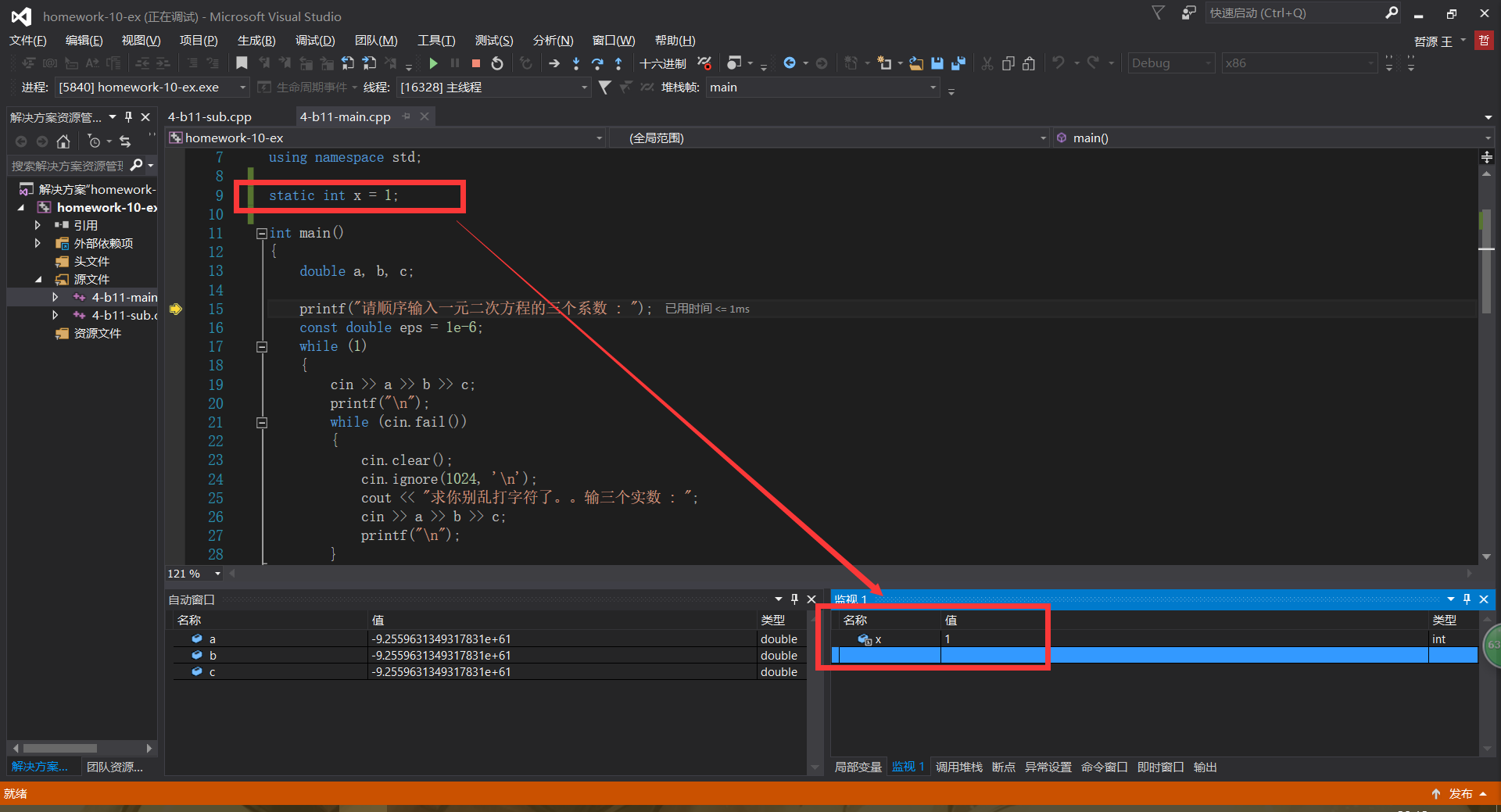
这时变量数值开始显示。然后当我们退出该函数后



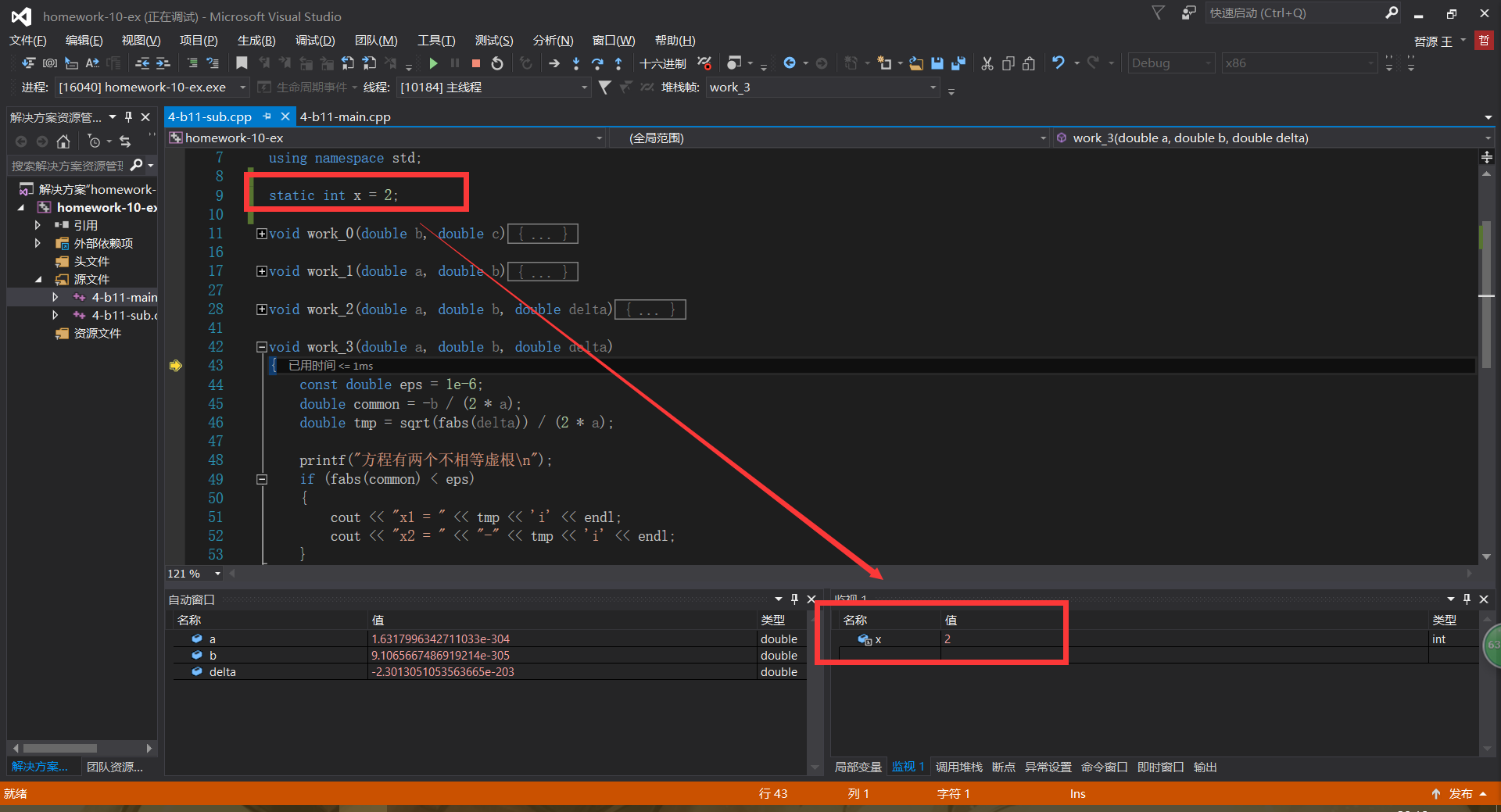
可以发现，这时函数名已经变为灰色，说明函数值仍保留，但在其他函数体中无法访问

**2.** **3 查看静态全局变量的变化情况（两个源程序文件，有静态全部同名）**

在调试模式下，在监控界面输入静态全局变量名称



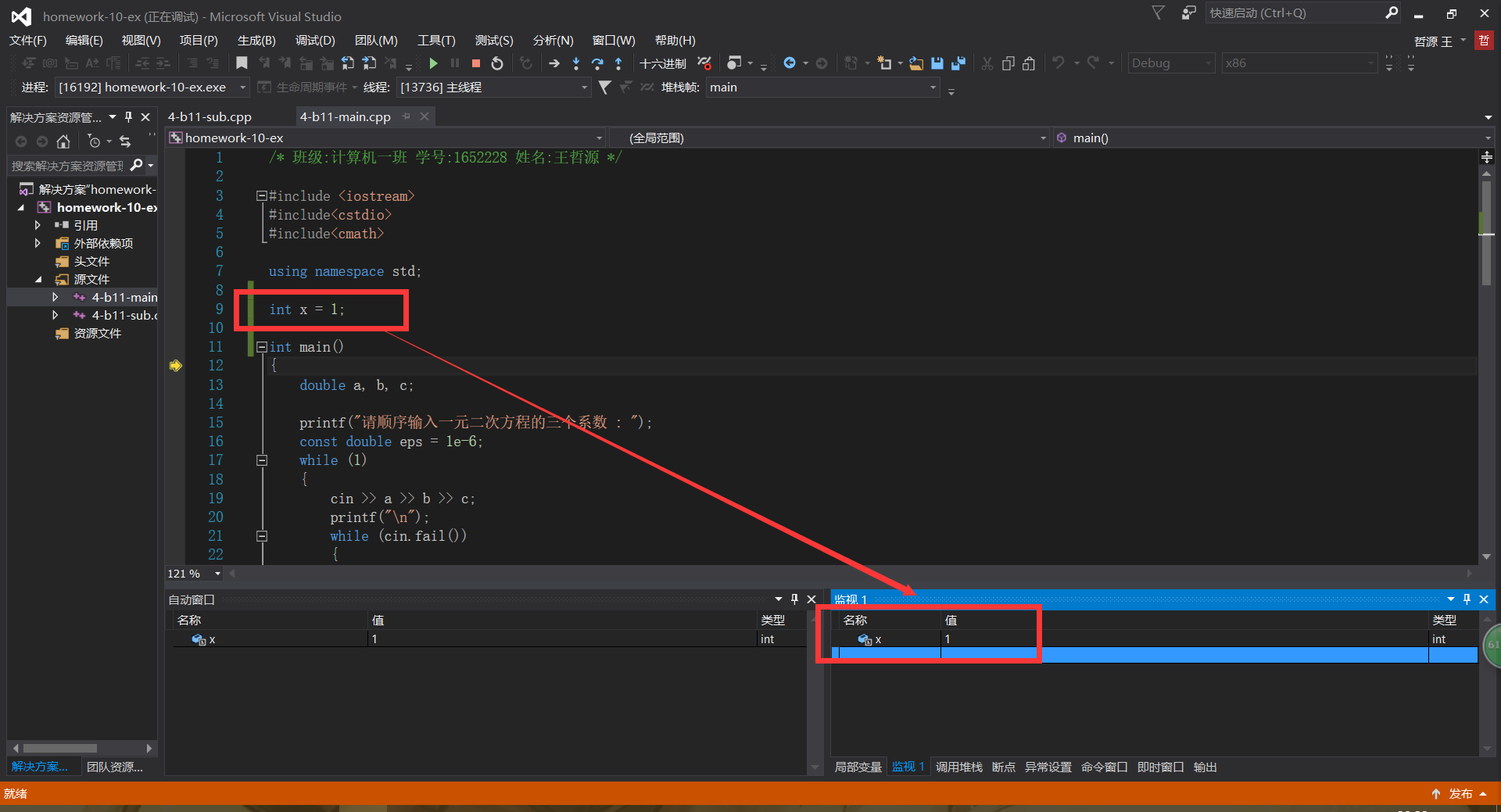
当程序运行到另一源文件中时，出现如下界面



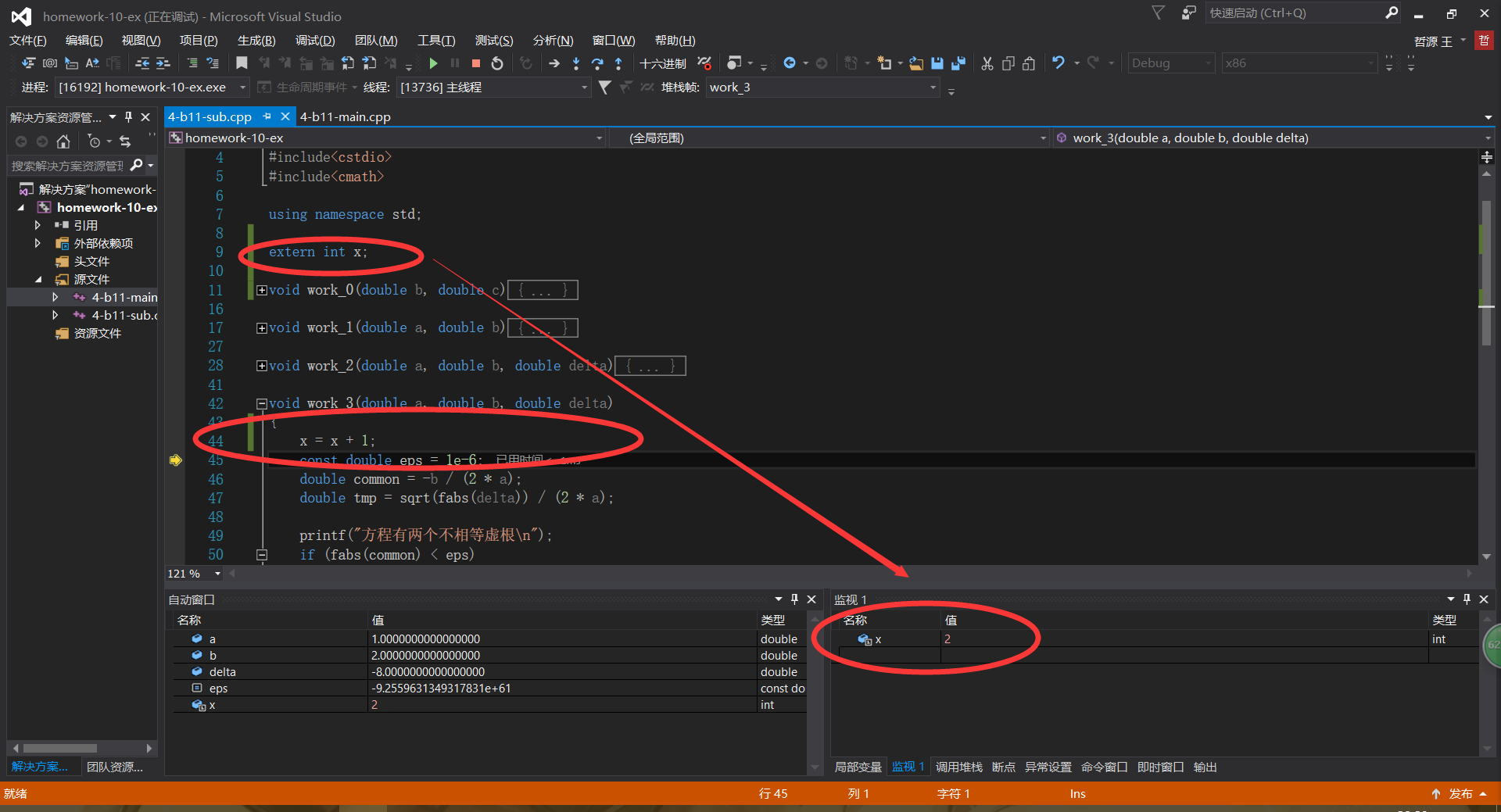
我们可以发现x的值发生改变，说明此处显示的是该源程序中的x

**2.4 查看外部全局变量的变化情况（两个源程序文件，一个定义，另一个有 extern 说明）**

同2.3，在监视中输入x



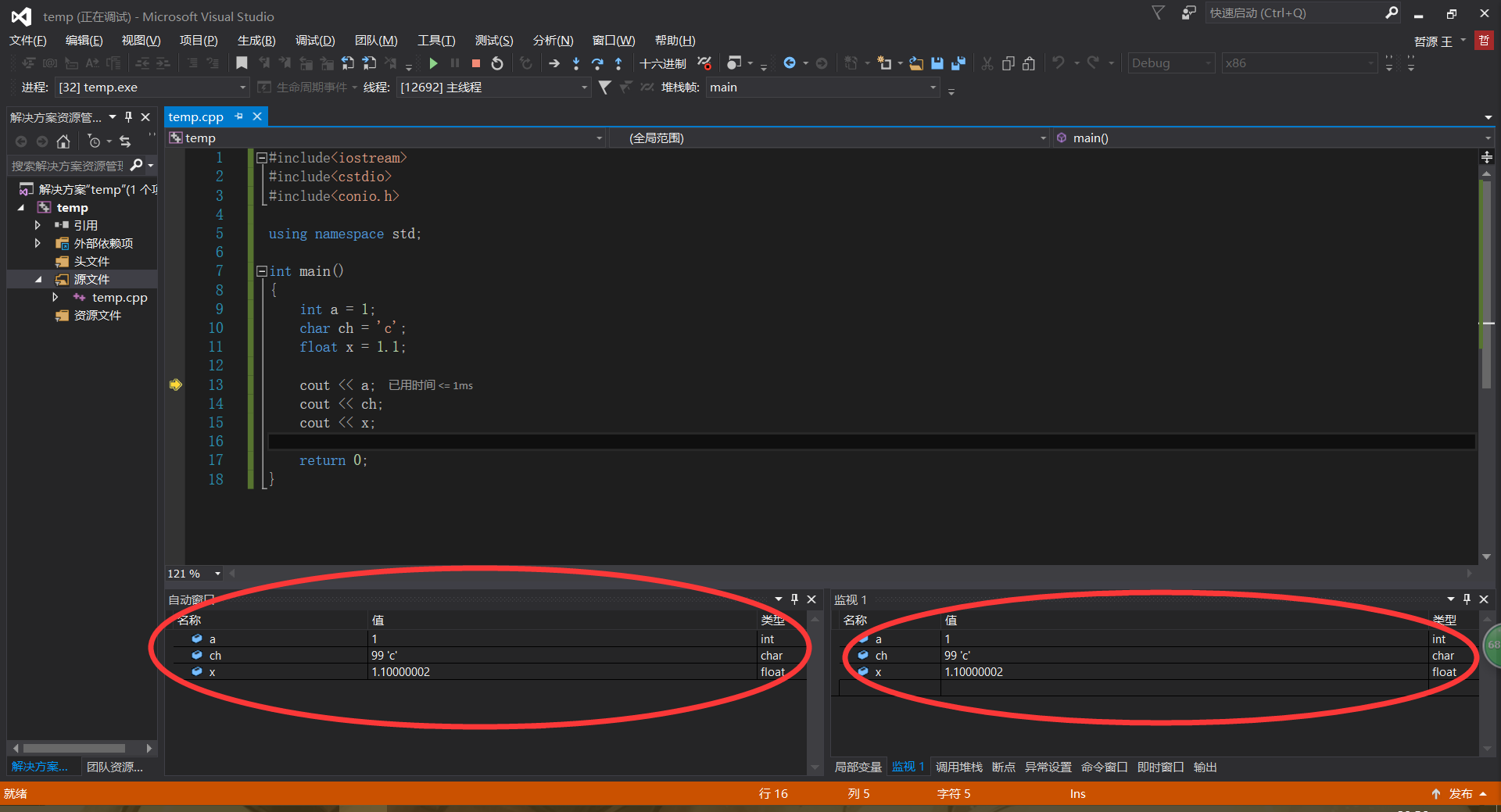
进入-sub源程序，发现监视1中的x的值发生改变



3. 用 VS2015 的调试工具查看各种不同类型变量的方法

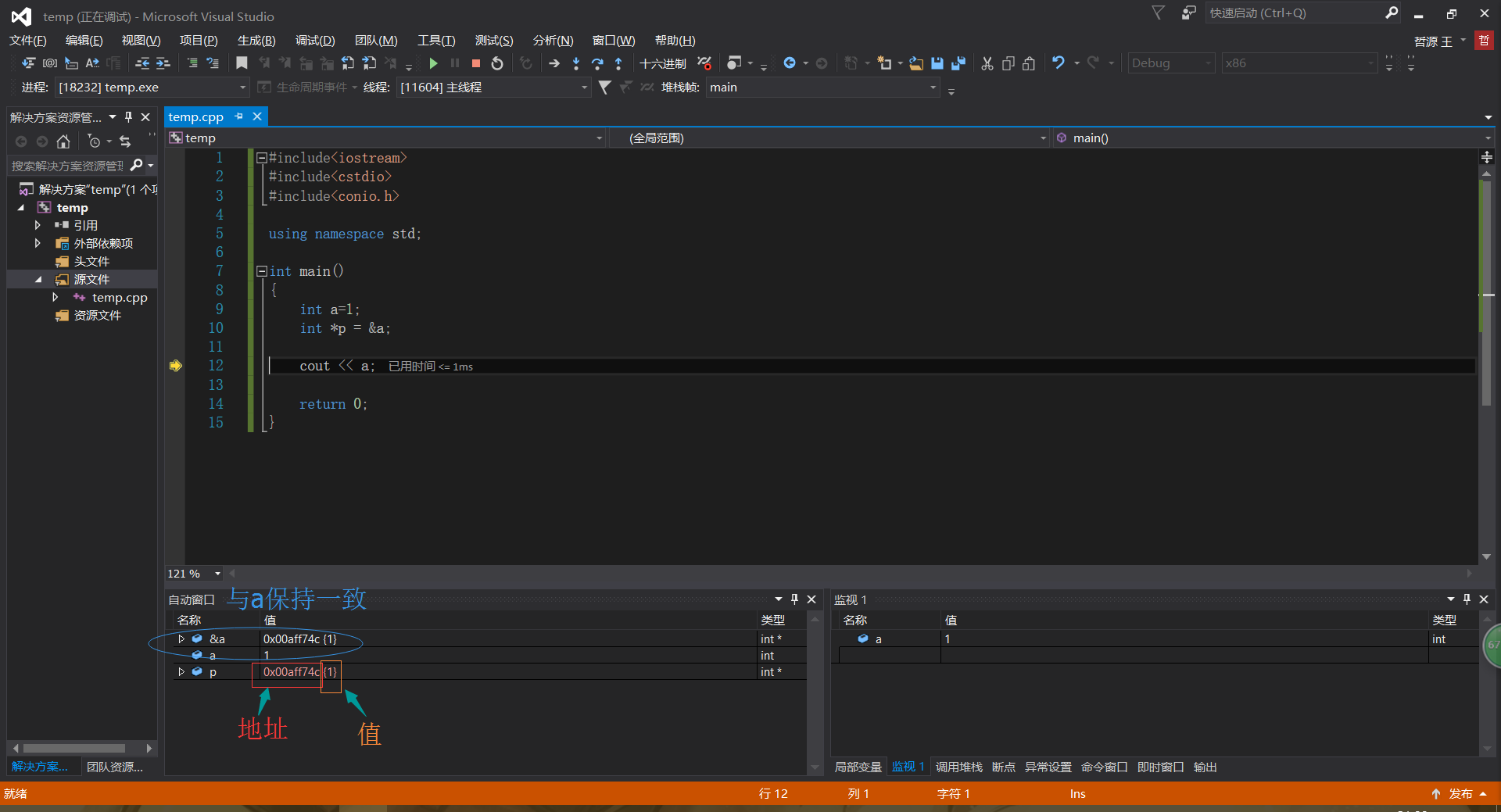
**3.1 char/int/float 等简单变量**

在调试模式下采用2.2中的方法打开局部变量窗口，或者采用2.3的方法打开监视窗口后添加监视即可



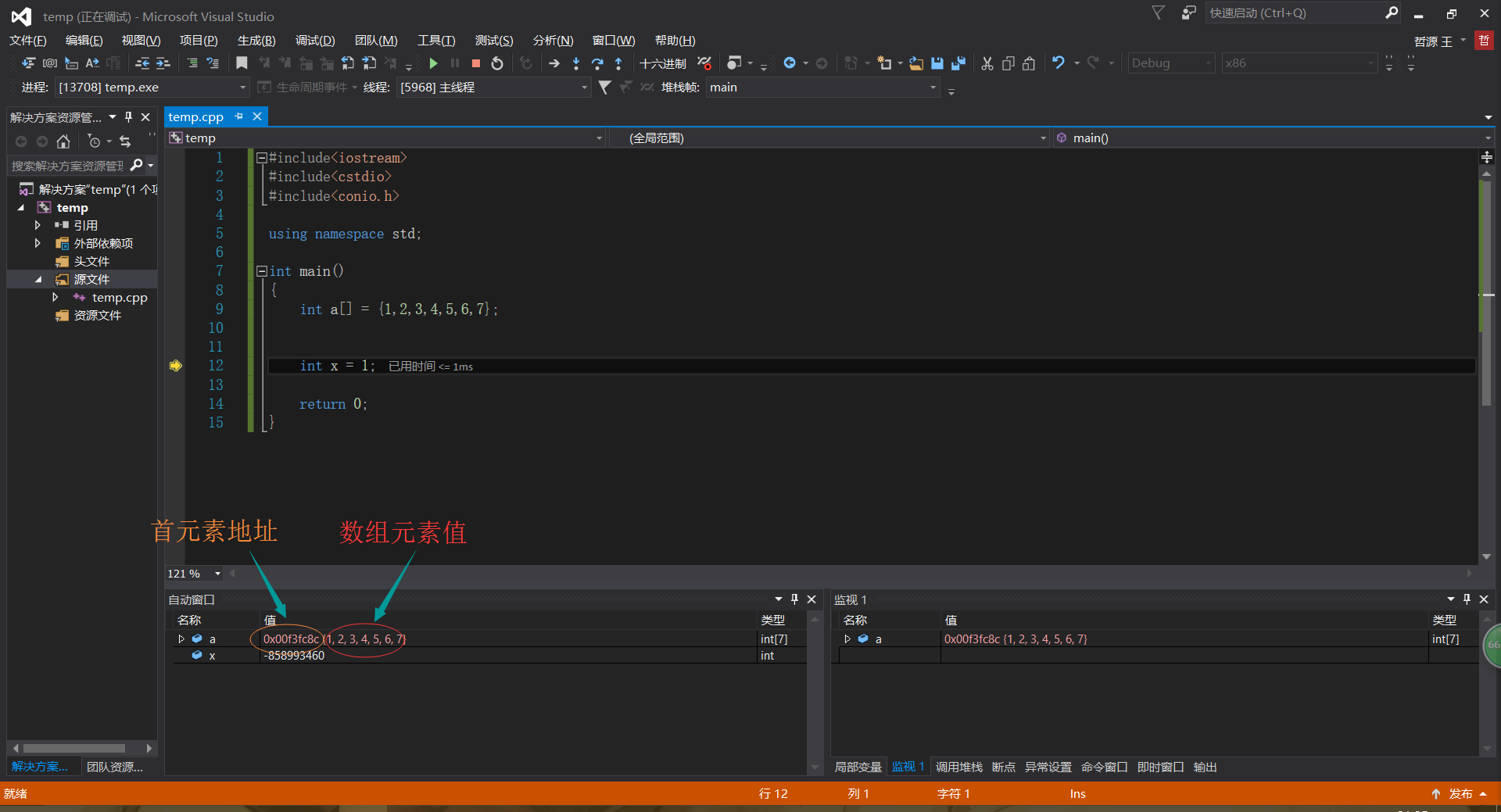
**3.2 指向简单变量的指针变量**

同3.1中的方法添加即可（以下均以局部变量为例）



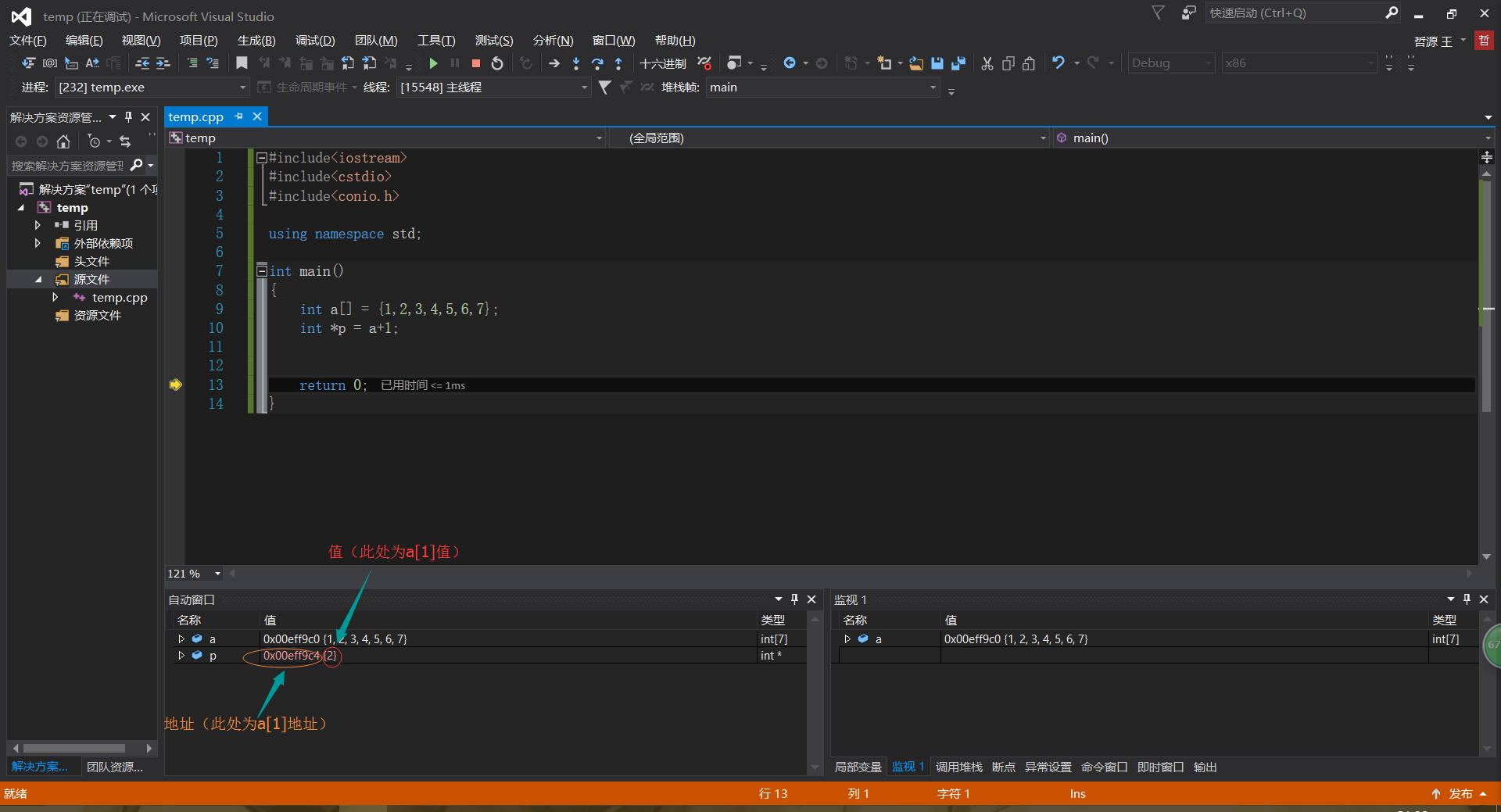
**3.3 一维数组**

同3.1，查看情况如下



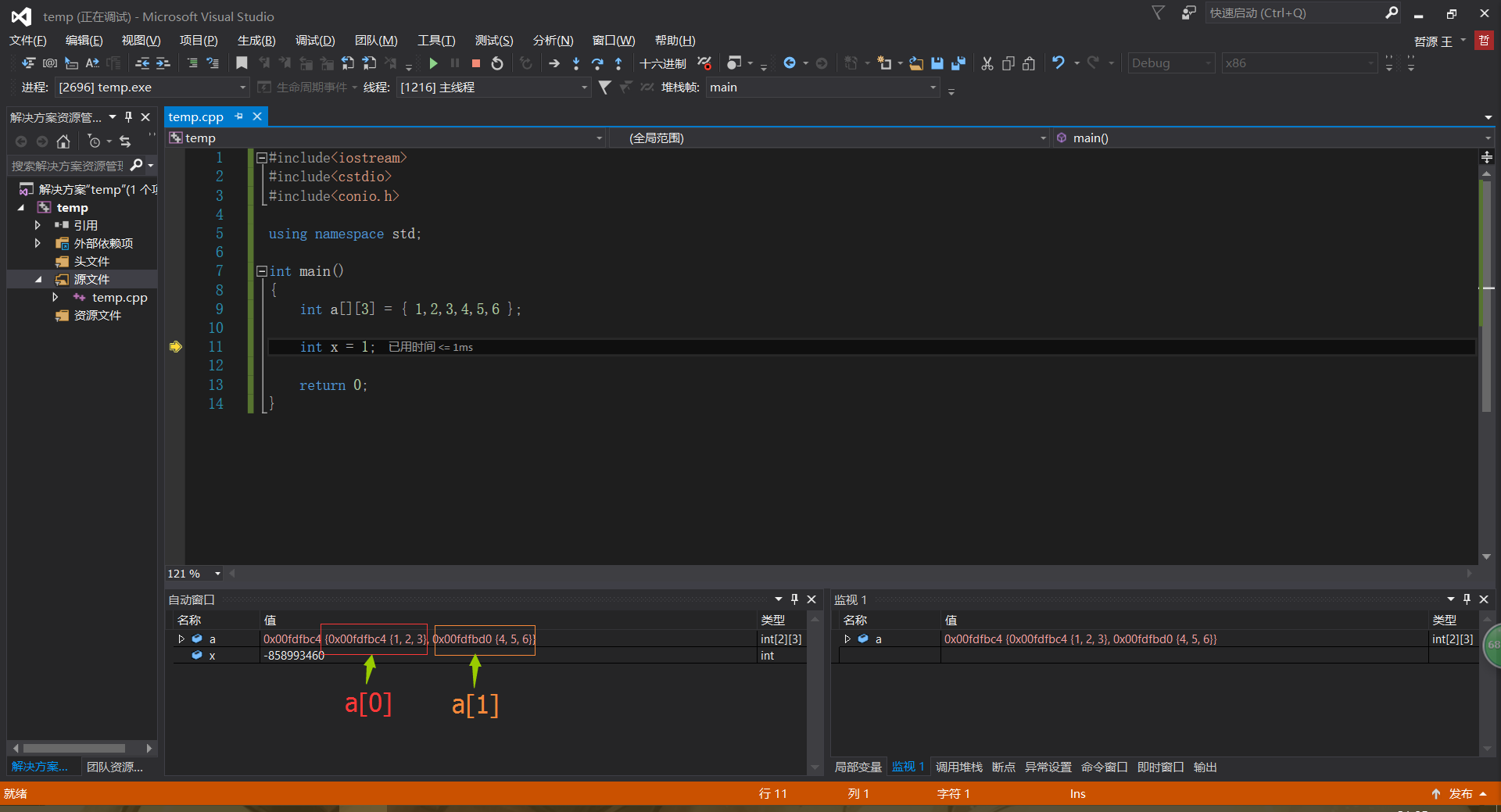
**3.4 指向一维数组的指针**

同3.1，查看情况如下



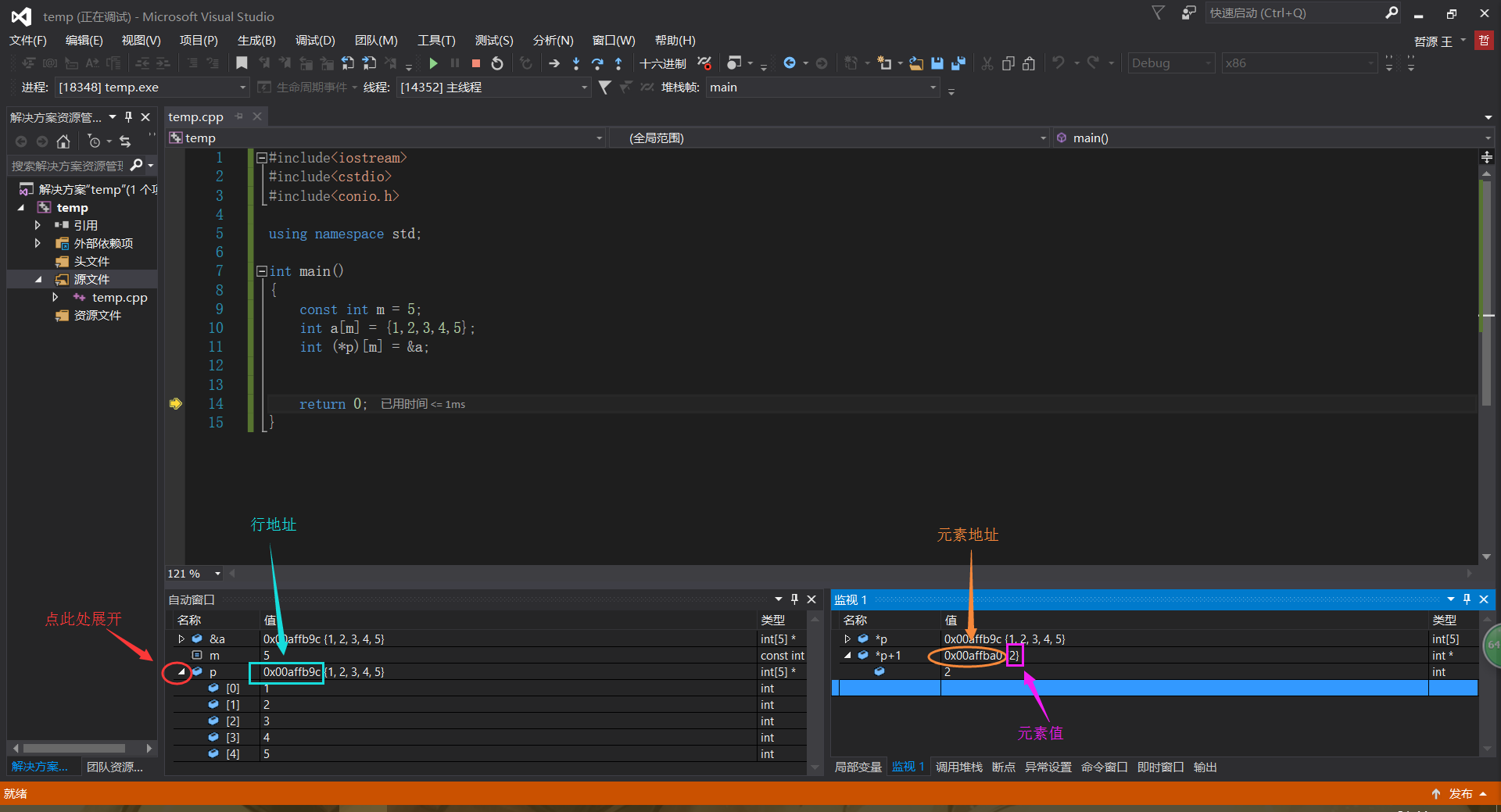
**3.5 二维数组**

同3.1，查看情况如下



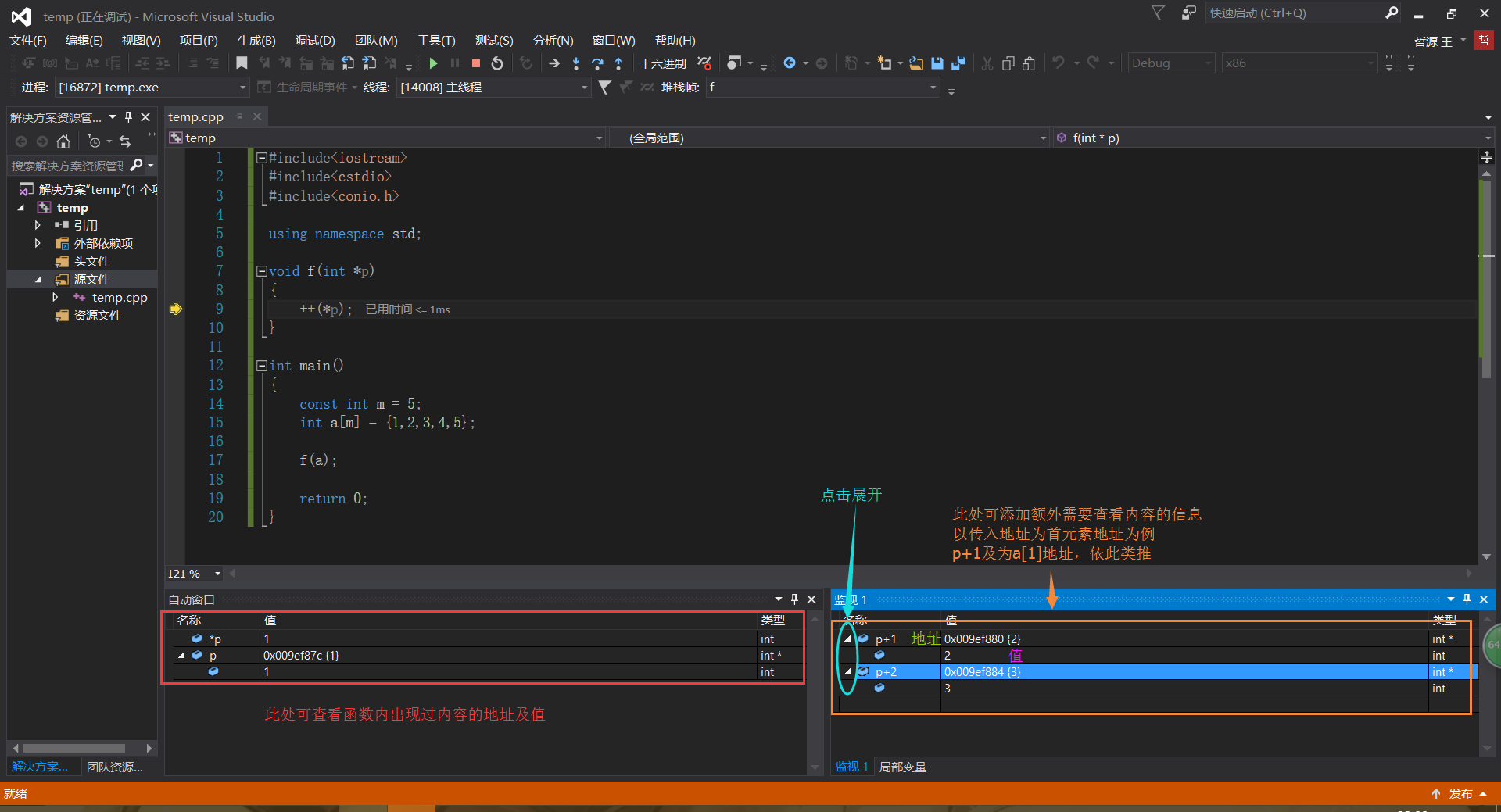
**3.6 指向由 m 个元素组成的一维数组的指针变量**

同3.1，查看情况如下



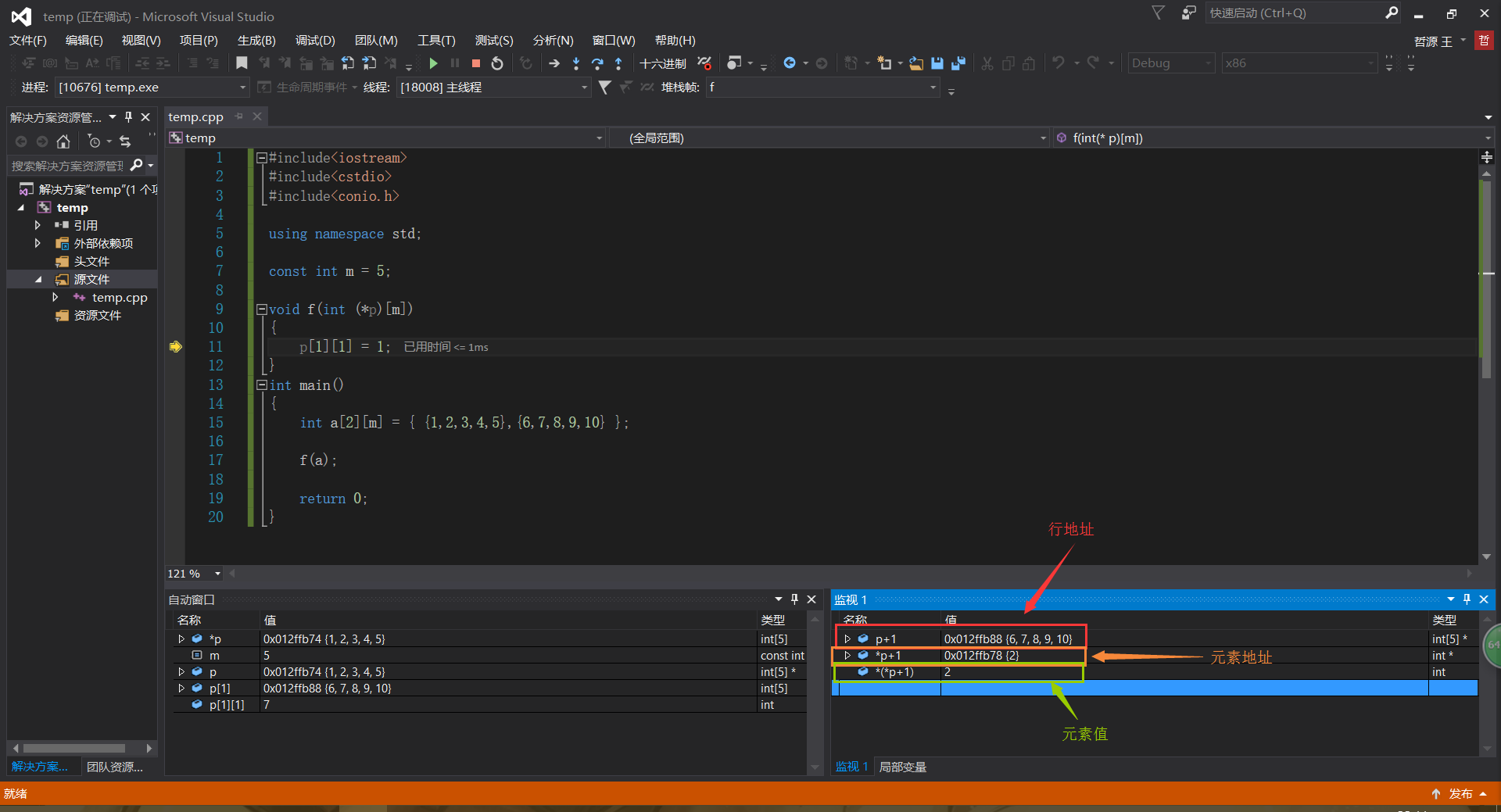
**3.7 实参是一维数组名，形参是指针的情况，在函数中查看实参数组的地址、值**

在监视区输入如下内容即可查看（以传入地址为数组首元素地址为例）



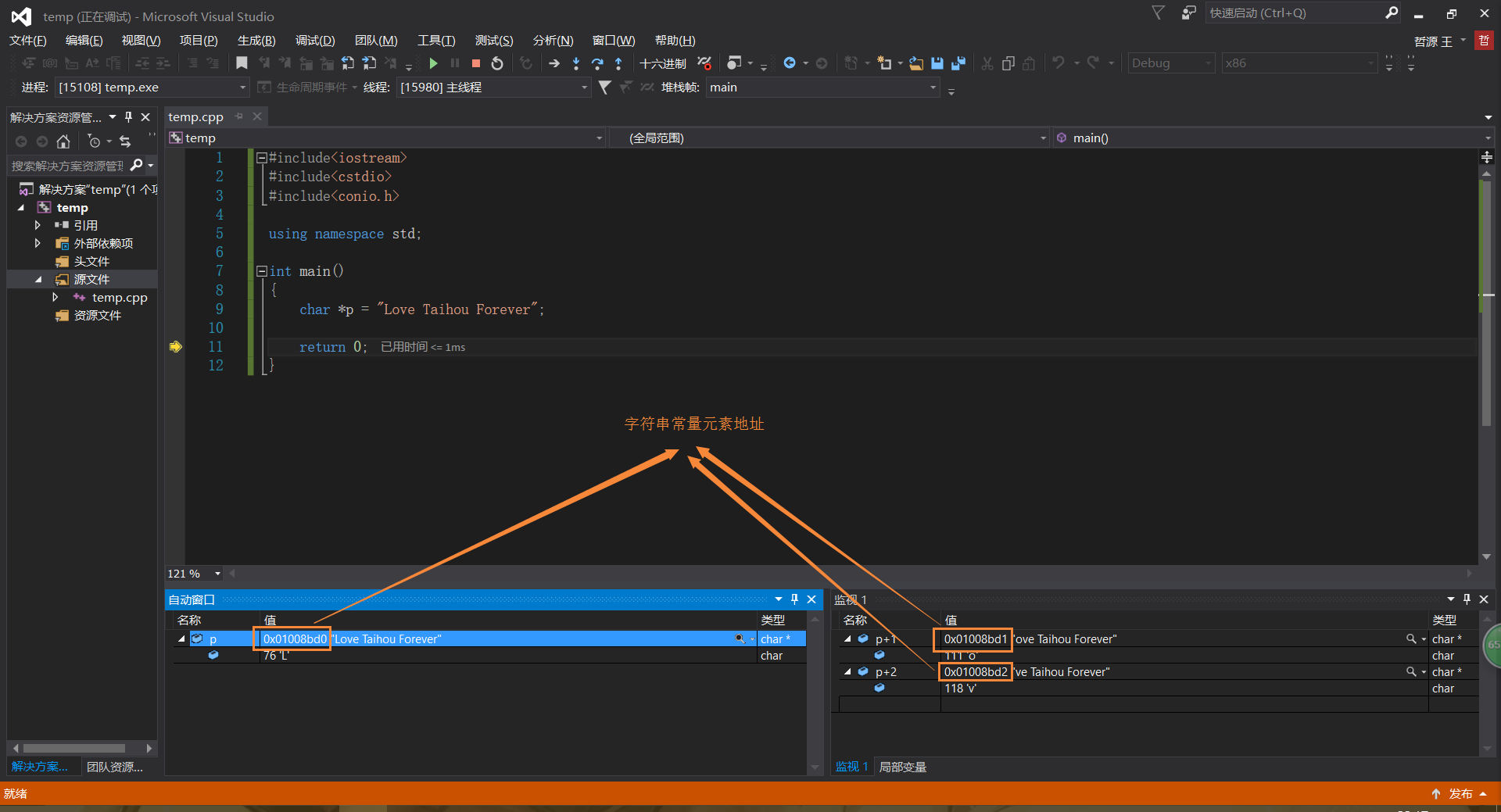
**3.8 实参是二维数组名，形参是行指针的情况，在函数中查看实参数组的行地址、元素地址、值**

添加操作同3.7，此处不再赘述。具体见下图（传入地址以第0个一维数组为例）



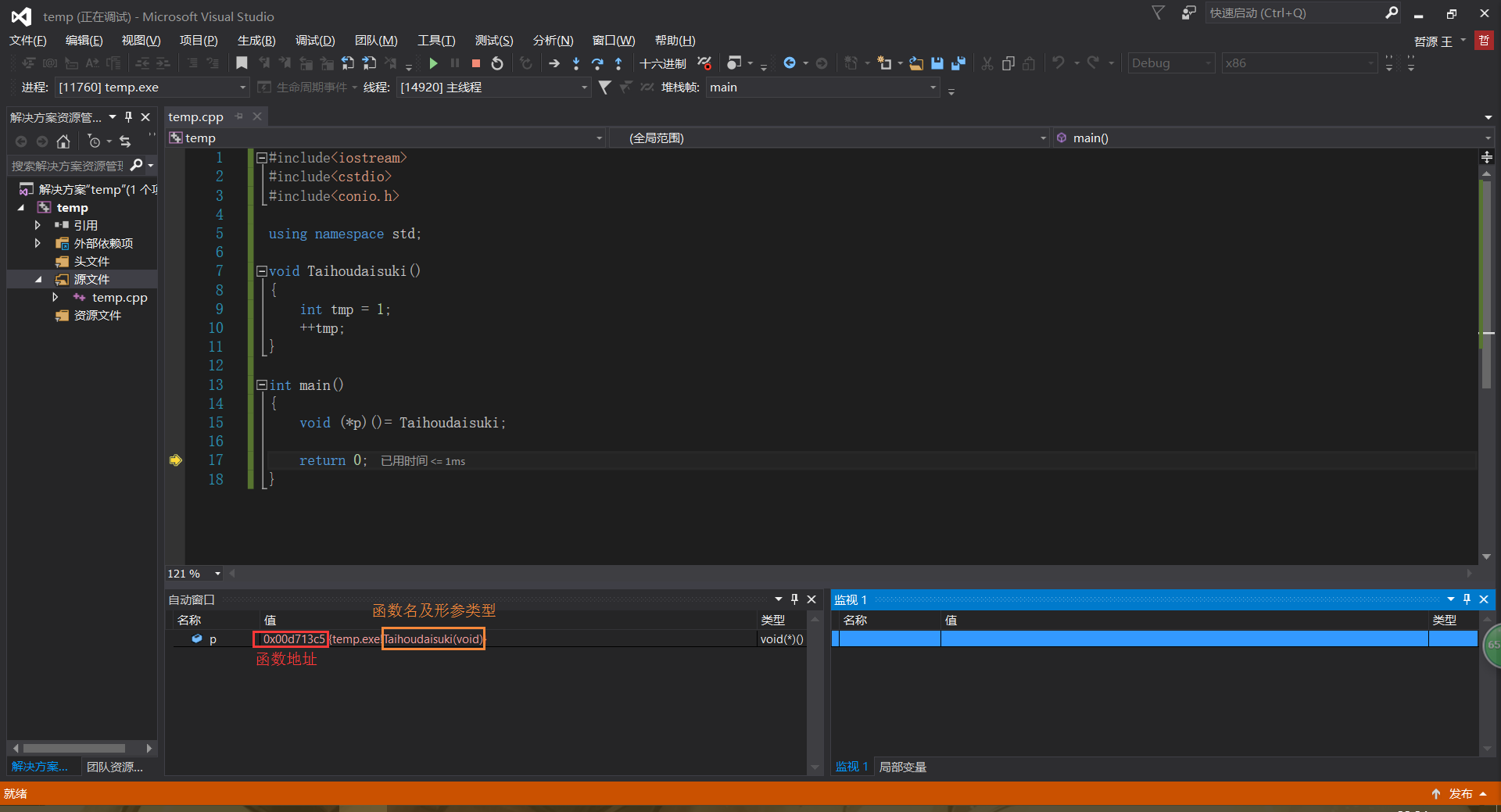
**3.9 指向字符串常量的指针变量**

可以通过监视查看无名字符串常量地址



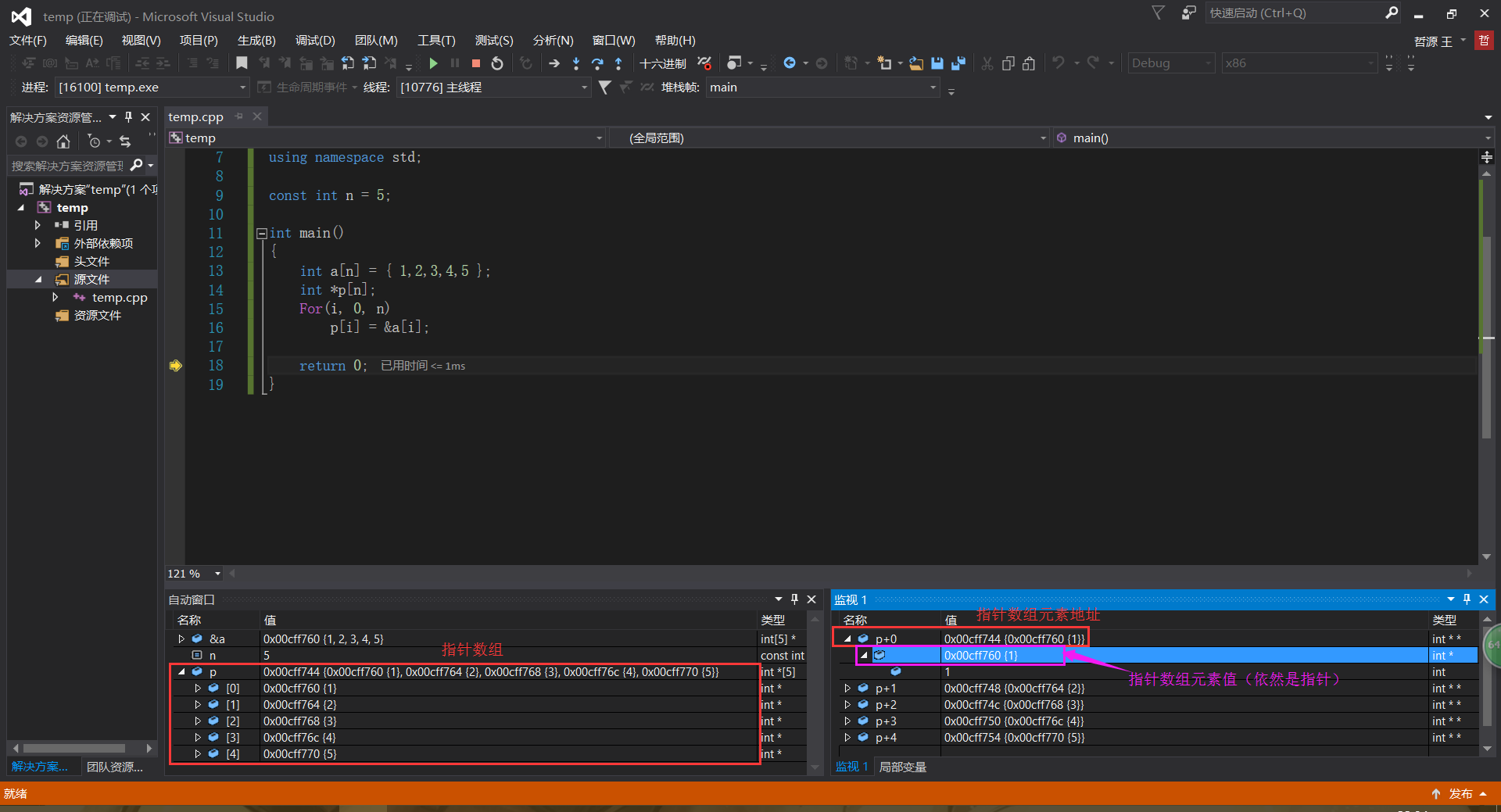
**3.10 指向函数的指针变量**

可以通过监视查看函数地址



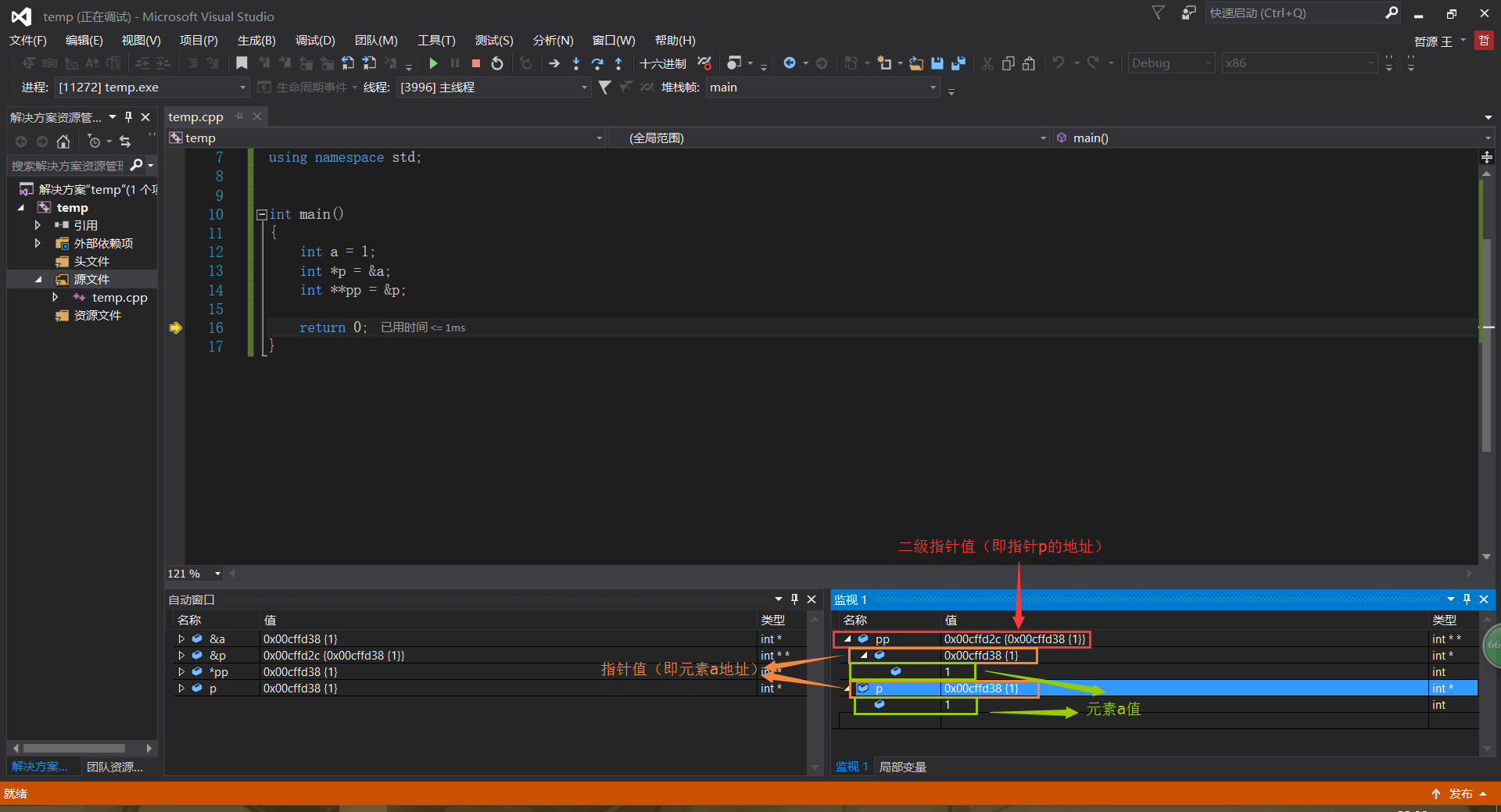
**3.11 指针数组**

添加方式同3.1，具体如下



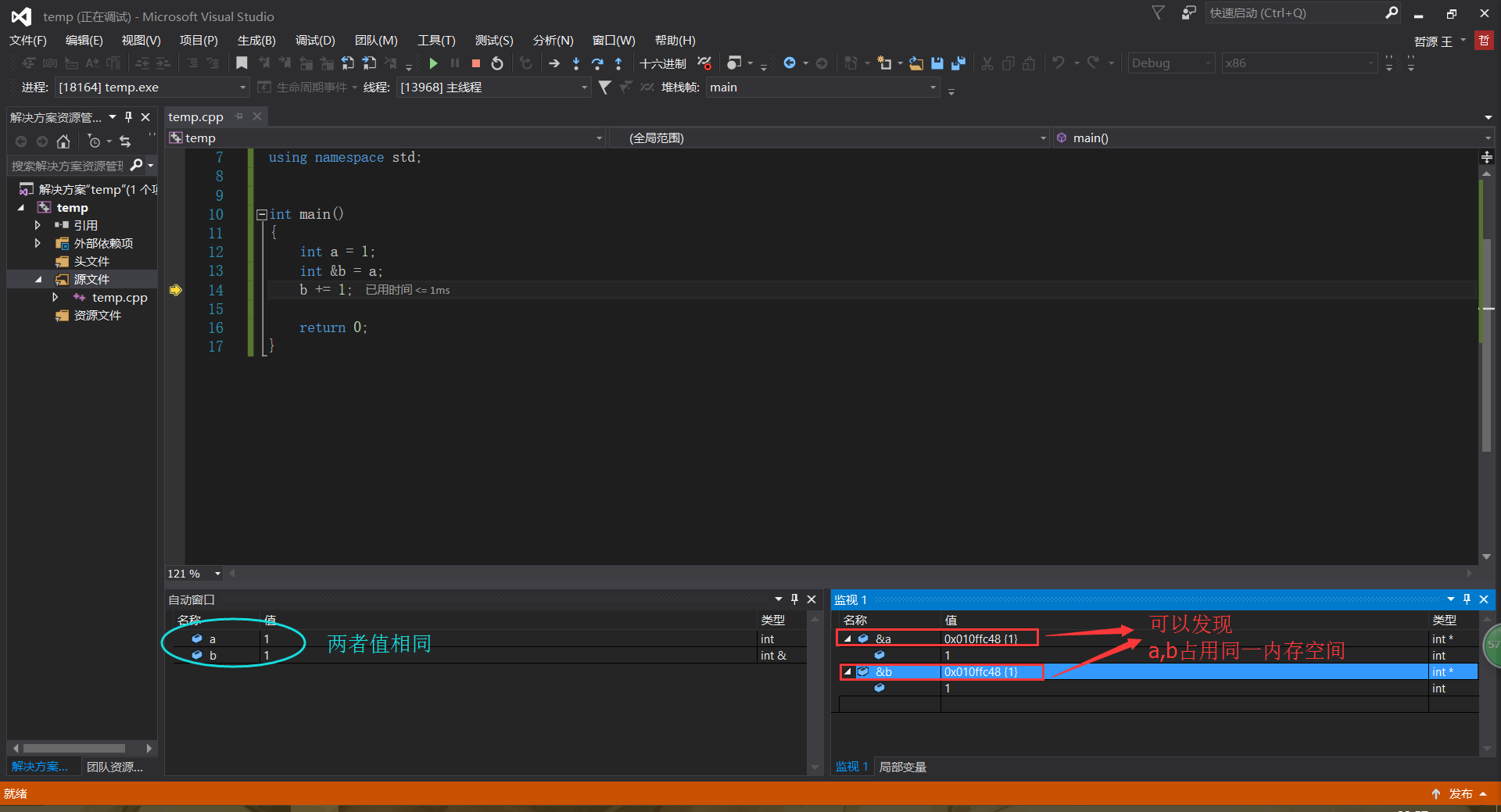
**3.12 二级指针**

查看操作同3.1，具体如下



**3.13 引用**

这里特别查看了引用的内存地址，键入方法同3.1



**3.14 指针越界情况**

调试模式下只要不对非法内存空间进行修改，指针允许在越界的情况下继续运行，且可以显示当前指针所指向的非法内存地址。具体见下图

