**VS2010中的基本功能介绍**

**常用到的快捷键**

F5： 调试 Ctrl+F5: 运行不调试 Shift+F5： 停止调试

Ctrl+Shift+F5: 重启调试

F9: 切换断点 Ctrl+F9: 启用/停止断点 Ctrl+Shift+F9: 删除全部断点

F4： 定位错误 F10：跨过程序执行 Ctrl+F10: 运行到光标处

F11：单步逐句执行 F3：查找中的下一个

Ctrl+Shift+L: 删除当前行

Ctrl+G: 跳转到第多少行处。

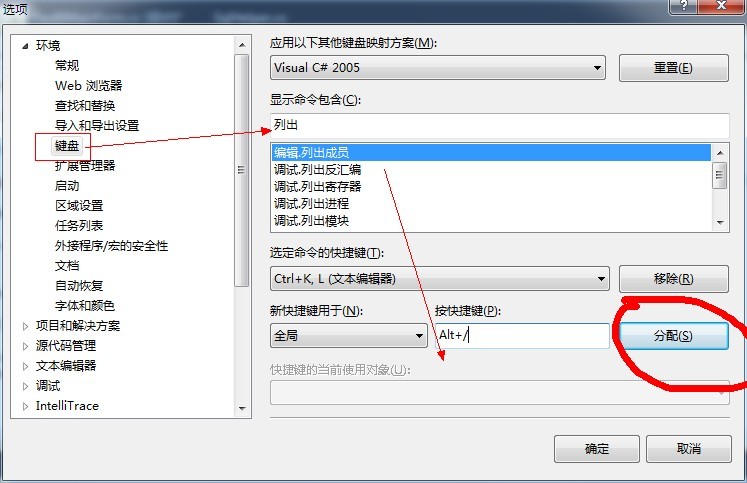
Ctrl+K，Ctrl+C: 注释选定内容

: 取消选定注释内容

**自己修改常用到的快捷键**

还是自己设定的快捷键用起来更舒服一些。

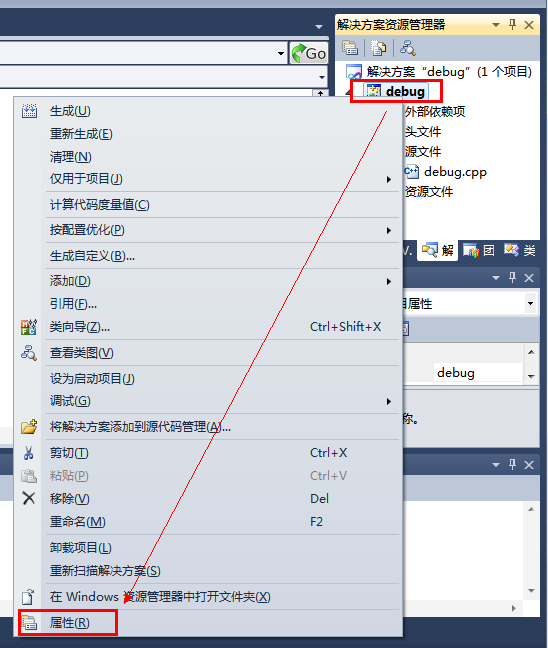
工具 -> 选项 -> 环境 -> 键盘，重点是找到自己想要修改的快捷键很麻烦。



**vs2010按ctrl+f5，调试窗口一闪而过的方法**

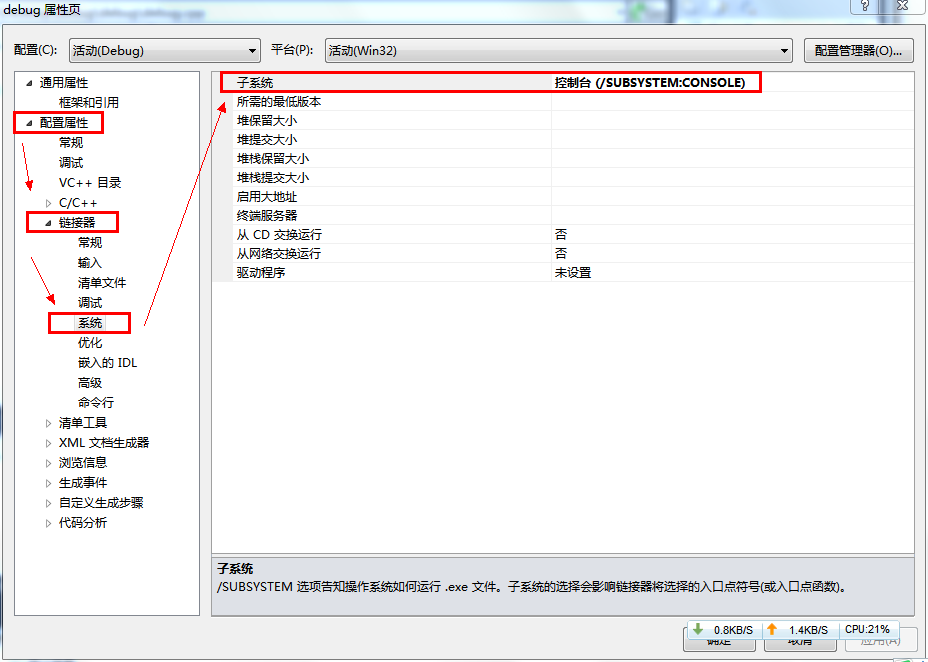
打开一个解决方案

找到【解决方案资源管理器】窗口，右键项目名称-属性



在打开的窗口中选择

配置属性》链接器》系统》子系统中选择【控制台】即可，然后确定保存。



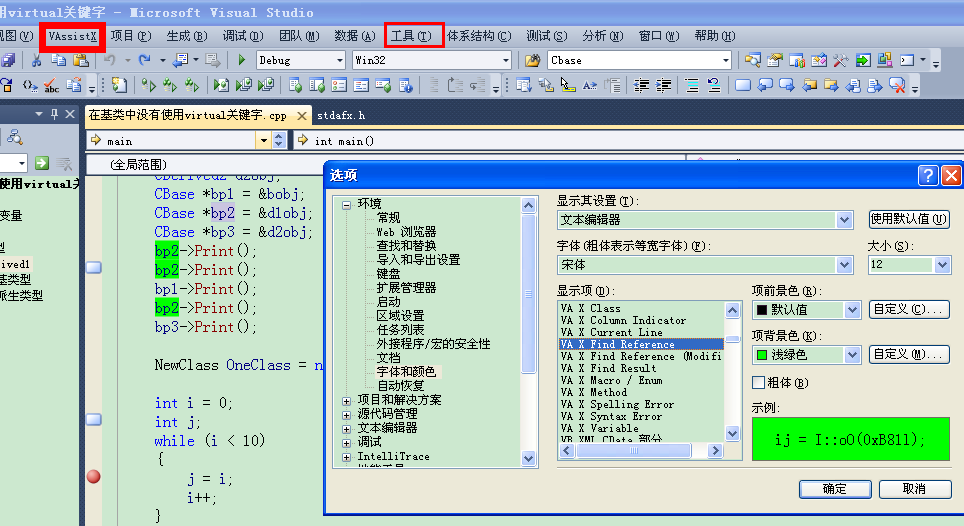
**设置选中的关键字高亮**

选择菜单：工具->选项->环境->字体和颜色.

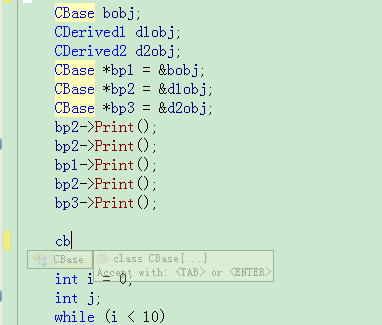
在对话框的display item中下拉，选择VA X Find Reference,设置背景颜色（我设置为绿色）

选择VA X Find Reference（modify) 这表示是被修改过的引用，设置背景颜色（我设置为红色）

需要安装的插件：Visual Assert X(或以上版本），这个插件提供了好几种状态时的颜色变化，提供。

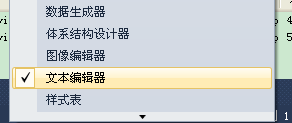
****

Visual Assert X 提供了智能提示

****

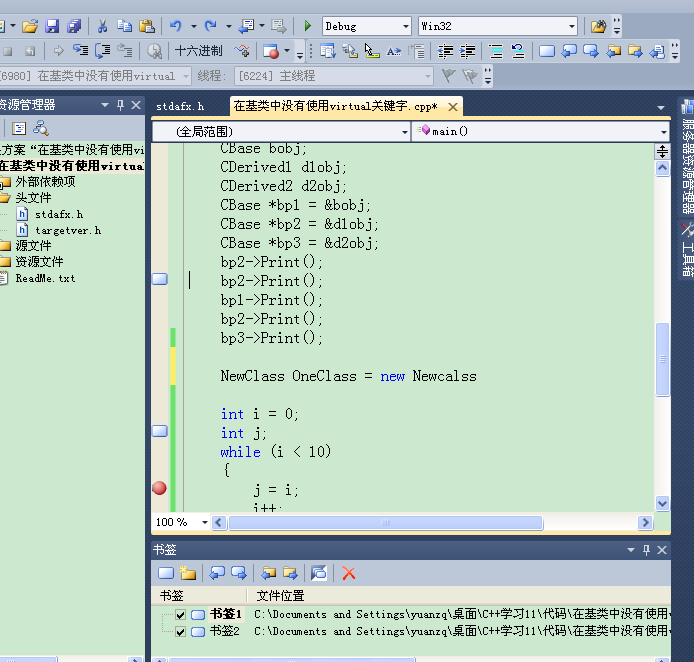
**设置书签**

功能没有delphi强大，但还是可以用作临时标记的，在工具栏空白处右键将工具条弄出来

****

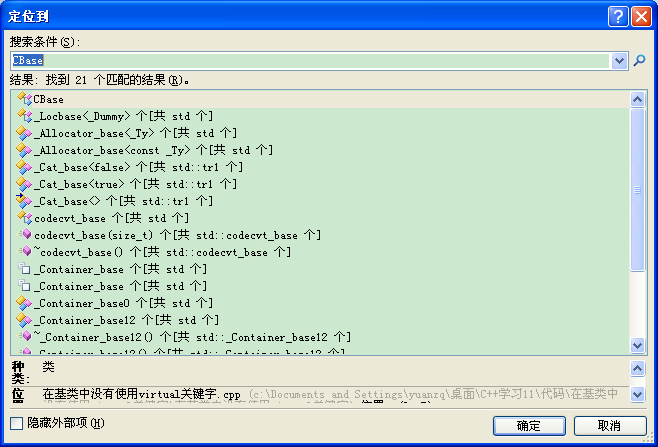
****

书签窗体在 菜单 –> 视图 –> 书签窗口

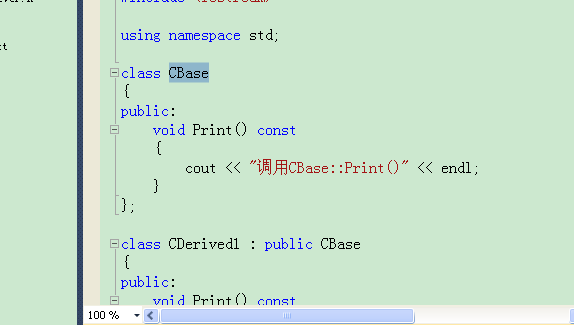
****

**定位搜索（多行批注）**

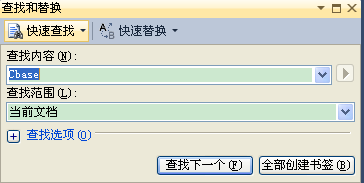
     菜单 –> 编辑 🡪 定位到 或者通过**Ctrl+,** 打开定位搜索窗口。在搜索栏（模糊查询的功能很强）中输入查询内容，VS2010 将列出相关结果信息。



双击搜索结果可以直接转到代码所在位置。

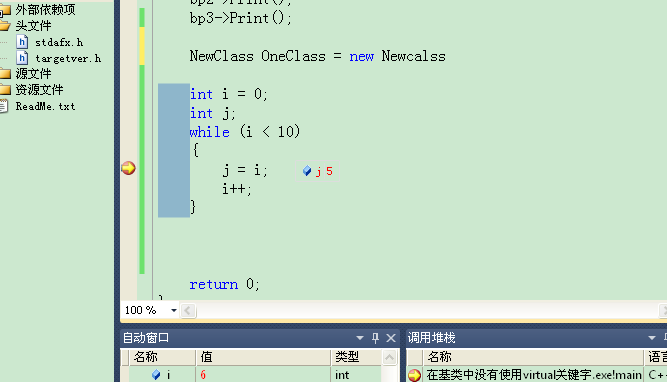


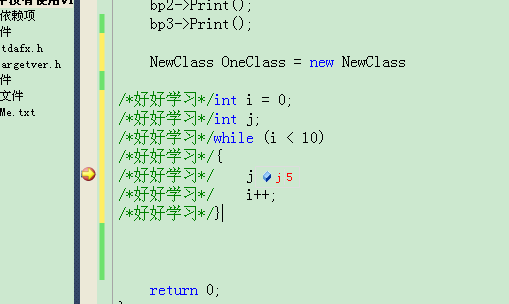
其它的查找方法还有 ctrl + F， ctrl + D；

** **

**矩形选择框**

     在VS2010可以选择矩形框，还可以选择零字符宽的矩形框并在其中输入内容。





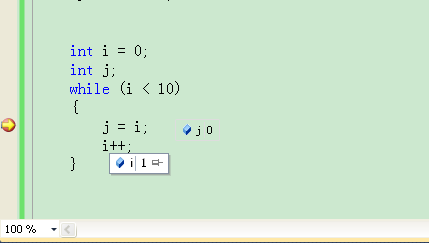
**跳到当前光标处（Ctrl+F10）**

　　我经常看到人们为了到达目标代码位置，而在程序中早早设定了断点，然后反复地按F10/F11，一步步走到目标代码处。当程序员的确需要仔细观察每一步的状态变化时， F10/F11是合理的选择。然而多数情况下，人们只想快速到达他们真正关心的代码处，这时候F10/F11就不是最佳选择了。

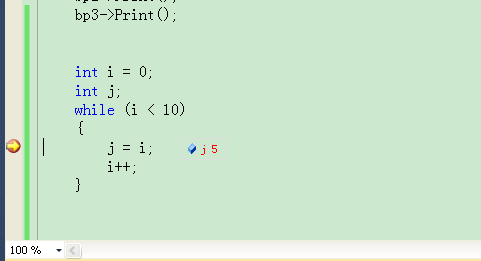
这时，你应该利用“跳到当前光标处”这个功能。先把光标定位在要测的目标代码行上，再同时按Ctrl和F10，被测程序将直接跳到该行停下。你再也不用按许多次F10/F11了。即使目标代码位于独立的类或方法中，你仍然可以从当前正在检查的地方跳过去。

**调试数据跟踪**

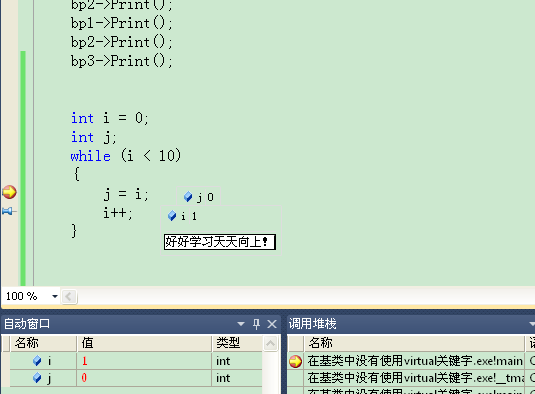
     在程序Debug过程中往往需要实时监控数据变化，VS2010 可以将数据Tip标签Pin在IDE窗口中随时跟踪数据变化。



在下图中随着while循环的进行，j值的变化也会随时体现在Tip标签中。



而且还可以在标签中添加批注，这个标签是随着代码一起保存的。



　下面的断点还可以累加使用。

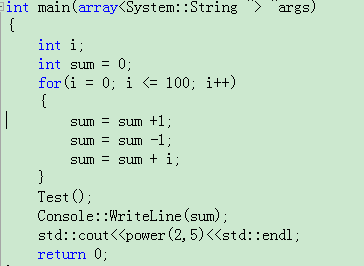
**条件中断**

　　另一种常见的情况是：开发人员设置断点，运行程序，利用不同的输入触发断点，然后在断点处手工检查是否满足某些特定的条件，从而决定是否继续调查。如果当前场景不是他们想要的，按F5继续运行程序，尝试别的输入，手动重复刚才的过程。

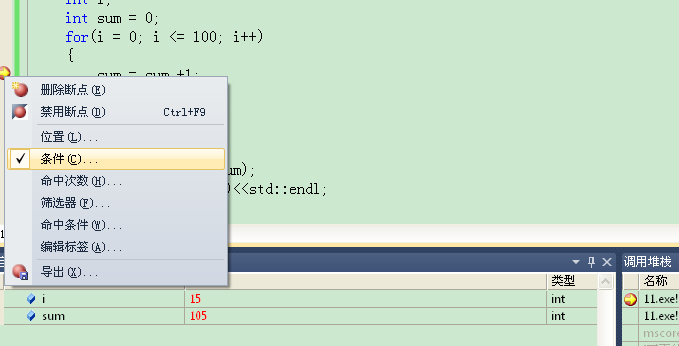
　　针对上述情况，Visual Studio提供了一个方便得多的功能——“条件中断”。只有当程序满足了开发人员预设的条件后，条件断点才会被触发，调试器中断。这将避免频繁地手工检查/恢复程序运行，大量减少调试过程中的手工和烦琐工作。

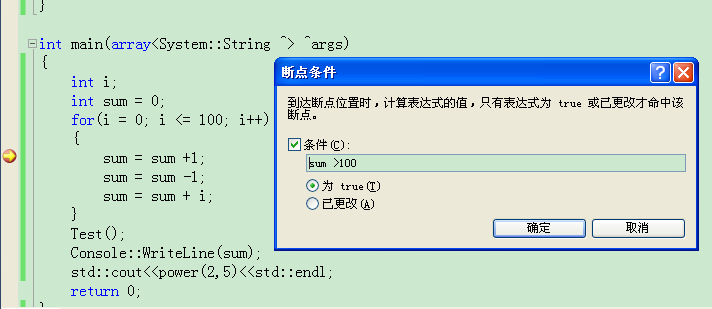
**如何设置条件断点**

　　设置条件断点非常容易。在特定的行上，按F9设置断点。



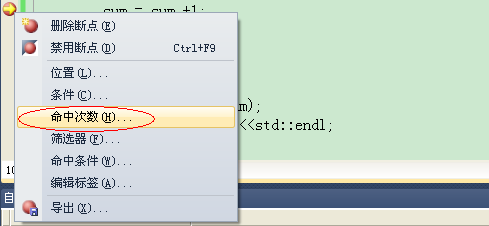
　　然后右击断点–编辑窗口左侧的红点，在上下文菜单上选择“Condition…”。

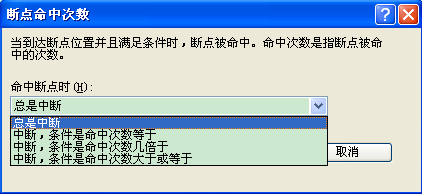
　　这时弹出一个对话框供你设置激活该断点所需的条件。比如：我们希望只有当局部变量paginatedDinners的尺寸小于10时，调试才中断。我们可以写出如下的表达式：

　　现在我再运行这个程序，实现搜索，只有返回值小于10时，程序运行才会被中断。对于大于10的值，该断点将被跳过。

**记录到达断点次数**

　　有时你希望，只有当第N次满足条件的运行到达断点时，才中断程序运行。例如：当第五次返回少于10份晚餐的查询结果时，中断程序运行。  
　　可以通过右击断点，然后在弹出菜单上选择“Hit count…”菜单命令实现。



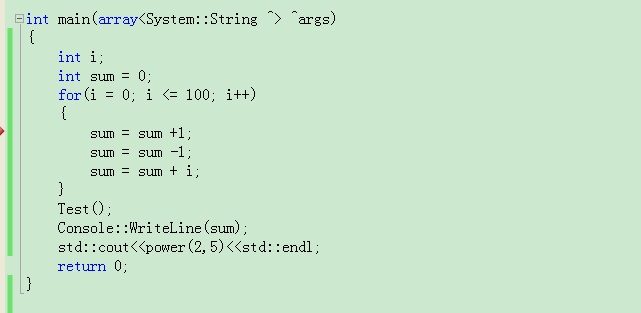
　　这时系统弹出一个对话框，它允许你指定：（1）当满足条件，而且进入断点的累计次数等于N时，断点命中一次。（2）当满足条件，而且进入断点的累计次数是N的倍数时，断点命中一次。（3）当满足条件，而且进入断点的累计次数大于N时，每次命中断点。  


**机器/线程/进程过滤**

　　设置如下：右击断点；在弹出菜单上选择“Filter…”菜单命令;然后指定命中断点的特定条件:在指定的机器上、或指定的进程中、或指定的线程中。

**跟踪点—进入断点时的自定义操作**

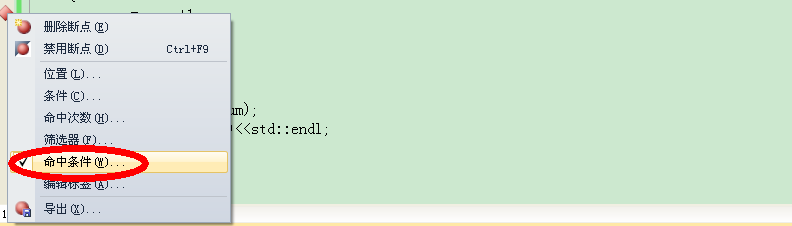
　　许多人不知道“跟踪点(TrackPoints)”这个调试功能。“跟踪点“是种特殊的断点，当它被命中时，它会触发一系列自定义操作。如果你想观察程序的行为，而又不想中断调试的时候，这个功能尤其有用。



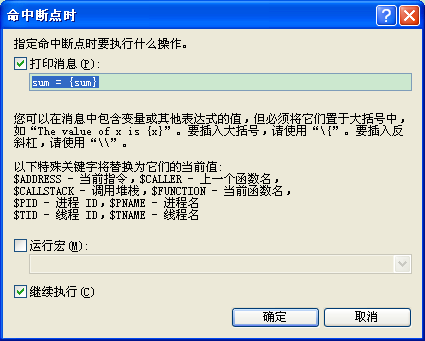
　　以上程序中，我们使用Console.WriteLine() 输出针对特定输入值生成的最终斐波那契数列。如果希望在调试器里观察操作中每一次递归运算后的数列而又不实际中断程序运行，该怎么办呢？“跟踪点”可以轻松实现。

**设置跟踪点**

　　你可以在特定的行上，按F9加跟踪点。然后  
　　右击断点，在上下文菜单中选择“When Hit…”：



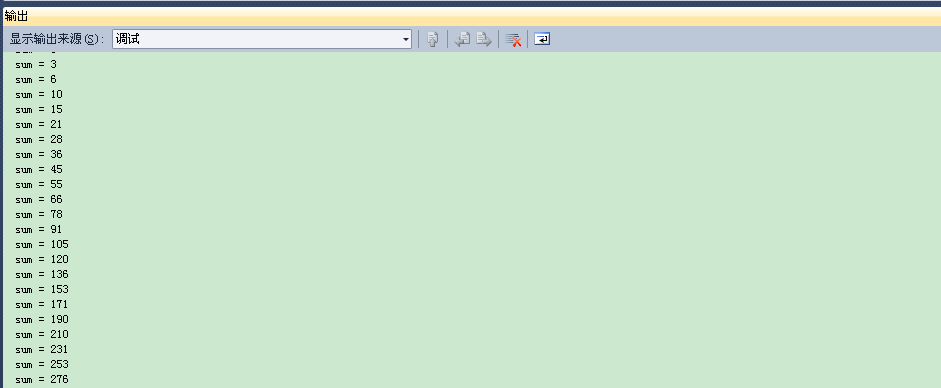
　　在弹出对话框上，你可以设置命中该断点时，所触发的事件。



　　在上面例子中，我们设定一旦命中断点时就打印追踪信息。注意，我们已经把局部变量“x”的值，作为追踪信息的一部分输出。局部变量可以通过{变量名}语法输出。你还可以利用系统内置的命令（$CALLER, $CALLSTACK, $FUNCTION等等），在追踪信息中输出常用的调试值。

　　在上例中，我们同时选中了底端的“continue execution“选项，这说明我们不希望程序中断调试状态，而是继续运行。唯一的不同是：每次断点条件满足时，我们的自定义追踪信息都将被输出。

　　现在当我们运行程序时，会发现自定义追踪信息自动显示在Visual Studio的“输出“窗口里。这让我们很容易看到程序的递归调用过程：



　　你也可以选择往应用程序中添加一个自定义追踪信息的监听器。这时追踪点的输出信息将通过它输出，而不是Visual Studio的“输出“窗口。