
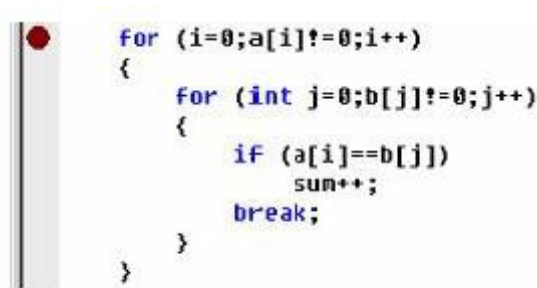


由于引起运行时错误的原因难以被发现，所以我们有时候要利用工具来完成调试工作。Debug 就是 VC++ 提供的一种常用调试工具。它能够让语句一句一句或一段一段执行，并且能够观察程序运行过程中各变量的变化情况。


在介绍如何使用 Debug 工具之前，我们要介绍一下什么是断点（Breakpoint）。当程序运行到断点的时候，它会暂时停止运行后面的语句，供用户观察程序的运行情况，并等待用户发出指令。断点不是语句，而是附在某条语句上的一个标志。

## 如何设置和移除断点


点击需要设置断点的语句，使光标移动到该语句所在的行。按下 F9 键或  按钮就会发现，在该语句之前出现一个红点，这就是断点标志。如下图 11.5.1 所示：



(图 11.5.1)

如果要移除已经设置好的断点，则同样点击断点所在语句，按下 F9 键或  按钮则断点被移除。我们可以给一个程序设置多个断点。

## Go

设置了断点之后，我们就能开始调试程序了。与以前不同，我们不能按执行按钮，而是要按 F5 键或  按钮，或者选择 Build 菜单 Start Debug 中的 Go。一旦按下了 Go，则程序会正常运行直至遇到断点。

我们下面这个程序（程序 11.5）来演示 Debug 功能的使用。该程序主要目的是统计一个不多于 20 项的正整数数列中，有多少对成双倍关系的项，该数列以 0 结尾。比如数列 1 3 4 2 5 6 0 中，成双倍关系的项有 3 对（1 和 2、2 和 4、3 和 6）。

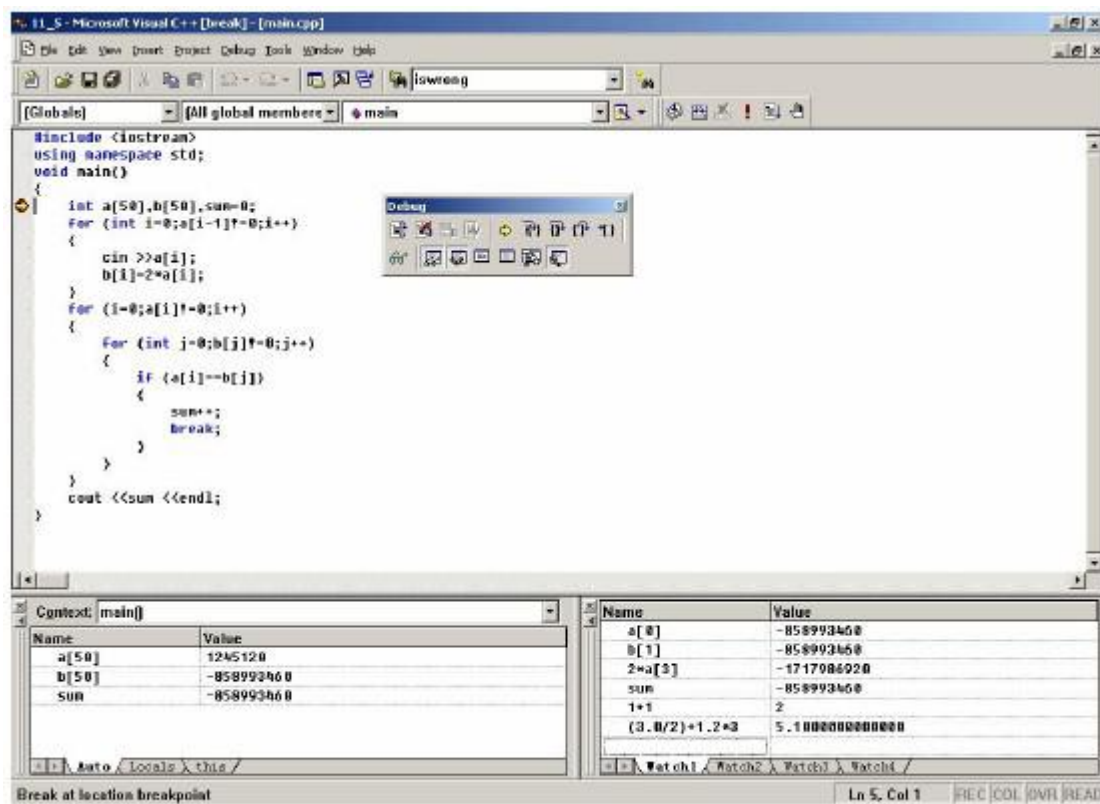
```
#include <iostream>
using namespace std;
int main()
{
    int a[50], b[50], sum=0; //在此设置断点
```

```

for (int i=0;a[i-1]!=0;i++)
{
    cin >>a[i];
    b[i]=2*a[i];
}
for (i=0;a[i]!=0;i++)
{
    for (int j=0;b[j]!=0;j++)
    {
        if (a[i]==b[j])
        {
            sum++;
            break;
        }
    }
}
cout <<sum <<endl;
return 0;
}

```

设置好断点，按下 Go 按钮以后，我们可以看到如下的界面：



(图 11.5.2)

在界面中出现了三个我们不熟悉的窗口。在屏幕中间有着很多按钮的小窗口叫

Debug 窗口，里面的按钮可以控制程序继续运行的方式。在屏幕左下方的窗口称为 Variables（变量）窗口，可以观察每句语句执行后变量变化的情况。在屏幕右下方的窗口称为 Watch（监视）窗口，用户可以监视一些变量或简单表达式的变化情况。

## Debug 窗口



Debug 窗口中，第一行按钮是我们常用的。它们依次是：

Restart——重新开始调试。

Stop Debugging——停止当前调试。

Break Execution——停止程序的执行并转回调试状态。

Apply Code Changes——使调试过程中修改的程序代码生效。

Show Next Statement——显示将要执行的下一条语句的位置。在语句之前用黄箭头表示。

Step Into——进入语句调用的函数，并进行调试。

Step Over——不调试语句调用的函数。

Step Out——从当前调试的位置回到调用该函数的位置。

Run to Cursor——正常运行直到光标所在的语句。

我们在调试的时候，不要总是按“Step Into”，因为它对于一些系统提供的函数也是有效的。也就是说我们能够用它详细地看到系统是如何实现一个输出功能的，甚至可以看到这些语句的汇编语言形式。但是，这却并不是我们调试的主要目标。如果不小心进入了系统函数里，我们要及时按“Step Out”以退回到我们所编写的程序中。

在调试过程中，对于大多数语句应该按“Step Over”。如果要调试自己编写的函数，则在调用该函数的语句处按“Step Into”。

## Watch 窗口

在 Watch 窗口中分为两列，一列为 Name，一列为 Value。其中 Name 是可以被编辑的，我们可以在里面输入变量名或简单表达式。如果改变量或表达式是可以被计算的，则会在 Value 中显示它们的值，如下图 11.5.3 所示：

Name	Value
a[0]	1
b[1]	6
2*a[0]	4
sum	3
1+1	2
(3.0/2)+1.2*3	5.10000000000000
Watch1 Watch2 Watch3 Watch4	

(图 11.5.3)

## 如何用 Debug 找到错误

在 Debug 中，我们可以让语句一句句地执行。如果执行到某一句语句时发生了运行时错误，那么这个错误一般就是由这个语句引起的。

在 Debug 中，我们可以观察每一句语句执行的顺序和执行后变量变化的情况。如果发现程序无法实现既定的功能，我们可以将期望的结果和实际的结果作比对，并分析可能引起这些不同的原因。这样一来，大大加快了我們找到问题和解决问题的速度。