第3章 程序调试

设计程序的最终目的是得到一个无错误(语法、运行、逻辑)的可执行程序,获得正确的结果。但是在程序设计中,无论规模是大是小,错误总是难免的。程序的设计很少有能够一次完成、没有错误的(这里的程序不是指只有一个输出语句这样的程序,而是要实现一定的功能、具备一定实用价值的程序)。在编程的过程中由于种种原因,总会出现这样或那样的错误,这些程序的错误就是我们常说的“bug”,而检测并修正这些错误的过程就是“debug”(调试)。调试程序是查找、发现和纠正错误的有效途径。主流集成开发环境都提供了强大的程序调试功能,在程序进行编译、连接、运行时,会对程序中的错误进行诊断。能快速查找、发现和纠正错误也是对程序设计人员的基本要求。本章介绍程序调试的一般方法。

3.1 程序调试步骤

程序编写好后,调试程序的基本步骤如图 3.1 所示。

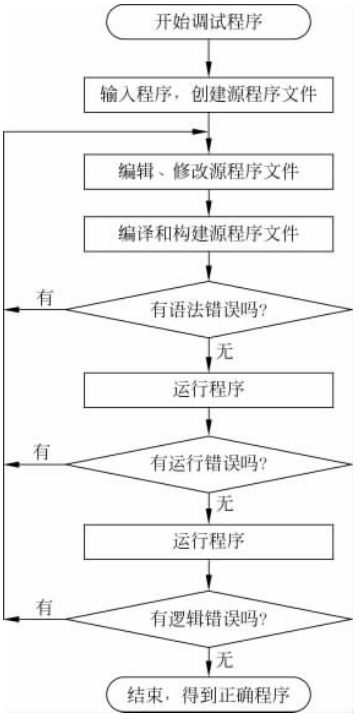

图 3.1 程序调试步骤

3.2 错误类型和查错方法

3.2.1 程序错误类型

程序的错误可以抽象地分为三类：语法错误、运行错误和逻辑错误。

1. 语法错误

语法错误是程序设计初学者出现最多的错误，是在编译过程中发现的、不符合设计语言语法规则而产生的错误。出现语法错误，程序编译或构建就通不过，程序就不能运行。通常，编译器对程序进行编译或构建的过程中，会把检测到的语法错误以提示的方式列举出来，又称为编译错误。

编译器检测的语法错误分为3种：致命错误、错误和警告。

(1) 致命错误：这类错误大多是编译程序内部发生的错误，发生这类错误时，编译被迫中止，只能重新启动编译程序，但是这类错误很少发生，为了安全，编译前最好还是先保存程序。

(2) 错误：这类错误通常是在编译时语法不当所引起的。例如：括号不匹配，变量未声明，表达式不完整，缺少必要的标点符号，关键字输入错误，数据类型不匹配，循环语句或选择语句的关键字不匹配、结构不完整等。产生这类错误时，编译程序会出现报错提示，我们根据提示对源程序进行修改即可。这类错误是出现最多的。

(3) 警告：警告是指怀疑被编译程序有错，但是不确定，有时可强行通过。这些警告中有些会导致错误，有些可以通过。例如调用子程序时，虚参、实参类型不一致等。

语法错误是通过编译和构建来查找、发现和纠正的。此类错误相对简单，调试起来比较容易。一般编译系统会自动提示相应的错误地点和错误原因，比如哪一行代码少了个括号等诸如此类的提示。如图3.2所示，这里块IF结构缺少结束语句ENDIF。对于常见的错误，如能看懂直接改正即可，如果看不懂原因，可以将错误提示信息输入搜索引擎查找，一般都能找到具体的解决办法。

2. 运行错误

运行错误指程序在运行过程中出现的错误。程序通过语法错误检测后，生成可执行文件，但并不说明程序就一定能正确运行，往往还会出现错误，需要继续调试。

运行错误一般可归纳为两类。一类是运行程序时系统给出出错信息，程序被迫终止。例如除法运算时除数为0，数组下标越界，输入数据格式错误，格式编辑符与输出项不匹配，文件打不开，磁盘空间不够等。此类错误发生时，编译平台一般也会提示相应的信息，对于常规的错误会有比较精确的提示，有时提示的错误原因会比较模糊，但因为此类错误一般在程序运行时，只在特定的条件下才会发生，所以根据错误发生的条件，能够大致判断程序出错的代码段，结合错误的原因，能比较方便地调试出错误。

另一类错误表现为运行时系统不正常或结果不正确，如程序不能正常结束，没有任何输出结果或输出结果与预期的不一致等。

3. 逻辑错误

逻辑错误主要表现在程序运行后，得到的结果与预期、设想的不一致，这就有可能是出现了逻辑错误。这种错误在语法上是有效的，但是在逻辑上是错误的。通常出现逻辑错误

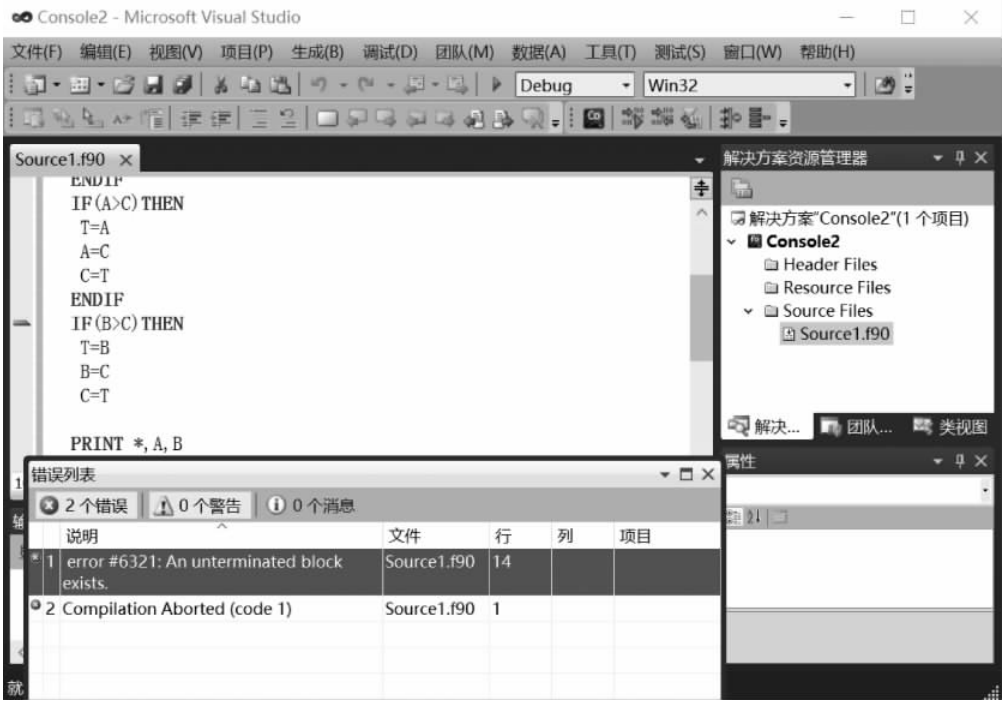

图 3.2　错误信息提示

的程序都能正常运行，系统不会给出提示信息，所以很难发现错误。要发现和改正逻辑错误，需要仔细阅读和分析程序。

程序运行了，也没有出错，但是执行出来的结果不是用户想要的，这分为以下两种情况。

(1) 能够看出错误。例如，查询工资大于 5000 元的人员名单，却出现了 3000 元的工资。

(2) 看不出错误，直到偶然的机会才发现程序肯定出错了，后果就很严重。例如进行一个复合大型运算，某个常数输入错了，最后的结果人工无法判断对错，又以该结果进行其他的运算等，最后发现错误时误差过大，就得从头排查错误，例如使用了不正确的变量，指令的次序错误，循环的条件不正确，程序设计的算法考虑不周全等。

通常，逻辑错误也会附带产生运行错误。在一般情况下，编译器在编译程序时，不能检测到程序中的逻辑错误，也不会产生逻辑错误的提示，因此逻辑错误比较难排除，需要程序员仔细地分析程序，并借助集成开发环境提供的调试工具，才能找到出错的原因并排除错误。

3.2.2　查错的实验方法

1. 利用系统信息

(1) 编译过程中的错误。

用户根据错误信息分析产生错误的原因和性质，并进行相应修改。要注意的是，有时源程序中的一个含糊错误会引起编译程序的连锁反应，产生许多错误信息，在这种情况下，往往只须纠正一个出错信息对应的地方即可。例如，若程序中变量说明语句有错，这时那些与

该变量有关的程序行都会被编译系统检查出错。这种情况下,只要修改了说明语句的错误,其余错误就会同时消失,所以一般改完一处错误后就重新编译一次。

(2) 连接过程中的错误。

在连接过程中要涉及到模块与模块、模块与系统之间的关系。如果程序中有外部调用、存储区设置和各模块间的接口等方面错误,连接程序就会提示错误信息。

(3) 运行过程中的错误信息提示。

2. 插入调试语句

除了利用系统给出的信息进行分析、判断之外,常用的调试方法还有在程序中插入一些调试语句。常用调试语句有以下几种。

(1) 设置状态变量

每个模块中设置一个状态变量,程序进入该模块时,赋给该状态变量一个特殊值,根据各状态变量的值,可以判断程序活动的大致路径。

(2) 设置计数器

在每个模块或基本结构中,设置一个计数器,程序每进入该结构一次,便计数一次。这样,不仅可以判断程序路径,而且当程序中有死循环时,用这种方法能很快发现。

(3) 插入打印语句

打印语句是最常用的一种调试语句,用起来方便,能产生许多有用信息。

① 将打印语句放在靠近读语句(或输入语句)之后、模块入口处或调用语句前后,可以帮助检查数据有没有被正确地输入或数据传递是否正确。

② 将打印语句放置在模块首部或尾部、调用语句前后、循环结构内的第一个和最后一个语句、循环结构后的第一个语句、选择结构前或选择结构每一分支中的第一个语句的位置,用以提供程序执行路径信息。

③ 选择一些适合的点设置打印语句,以便打印有关变量的值,检查是否正确。

3. 借助调试工具

利用编译系统提供的调试工具进行单步、追踪运行。本章将介绍 CVF 编译环境和 IVF 在 VS2010 中的调试工具的一般应用。

以上方法也常常联合起来使用。

3.2.3 错误修改原则

(1) 要勤于思考。程序调试是分析问题、解决问题的过程。培养调试程序的能力,最有效的方法是勤于思考、积极分析、不断总结。

(2) 如果陷入困境,要与别人交流自己的问题。在交流的过程中,有可能突然找到问题所在,别人的提示或许对自己有很大启发。

(3) 如果陷入困境,可适当间隔一定时间后再去考虑,不要一味纠结下去。如果在适当的时间内找不到问题所在(小程序半小时,大程序几小时),就先放下问题,隔一段时间后有可能会灵机一动、解决问题。

(4) 不要在问题没有搞清楚前随意改动程序,这样不利于找出错误,程序越改越乱,以至于面目全非。

3.3 调 试 工 具

3.3.1 CVF6.5 的调试工具

CVF6.5 开发环境提供了功能强大的调试工具 debug，使用 debug 可以快速、方便、高效地检查、发现和纠正错误。debug 功能强大、内容丰富，这里只简单介绍使用 debug 测试工具调试程序的步骤。

(1) 激活 Build 和 Debug 工具栏。在工具栏空白处单击鼠标右键，弹出快捷菜单，选择 Build 和 Debug 工具栏，如图 3.3 所示。

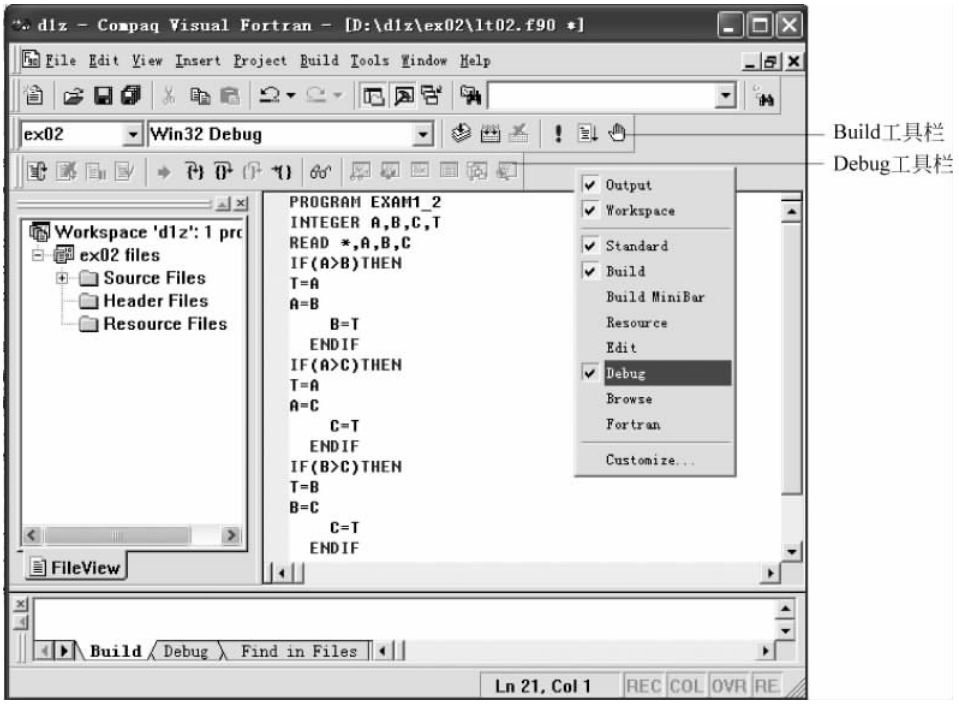

图 3.3 激活 Build 和 Debug 工具栏

(2) 通过 Build 工具栏“断点设置”按钮( )给程序设置断点。断点就是程序在运行中暂停的位置，用于通知调试器何时何地暂停程序的执行。根据需要可设置多个断点。将光标置于需要设置断点的语句位置，单击 Build 工具栏上的“断点设置”按钮，即可在该语句处设置一个断点。“断点设置”按钮是一个开关按钮，再次单击可取消断点，如图 3.4 所示。

(3) 单击 Build 工具栏的“开始调试程序”按钮( )，在运行窗口中输入数据后，运行至第一个断点位置暂停，断点出现黄色箭头，如图 3.5 所示。

(4) 激活显示有关 Debug 调试窗口，通过调试窗口可以观察程序运行过程中的重要参数(变量、内存、寄存器等)。CVF6.5 共提供 6 个 Debug 窗口，常用的有两个：变量窗口 Locals 和观察窗口 Watch，用于了解变量和表达式的取值情况，以判断、分析错误所在。如图 3.5 所示，Locals 窗口中自动显示在范围内的变量，在 Watch1 窗口中可以添加想要观察的变量和表达式。

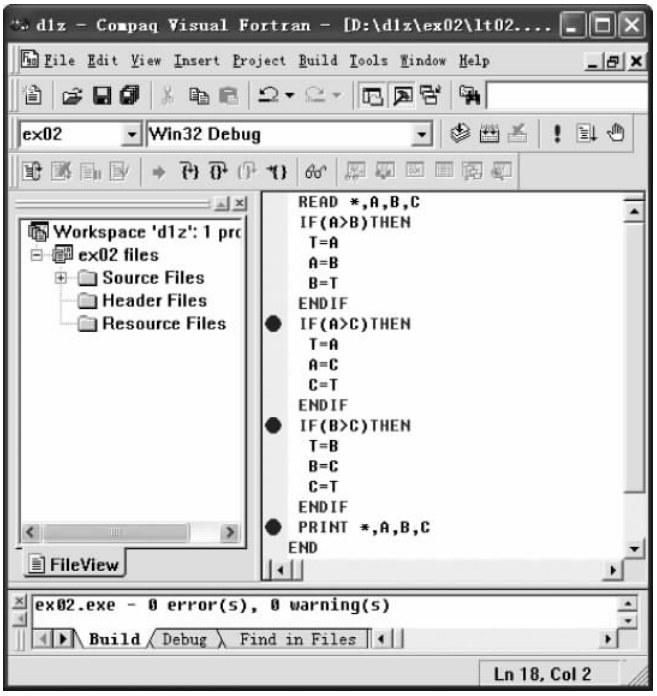

图 3.4　设置断点

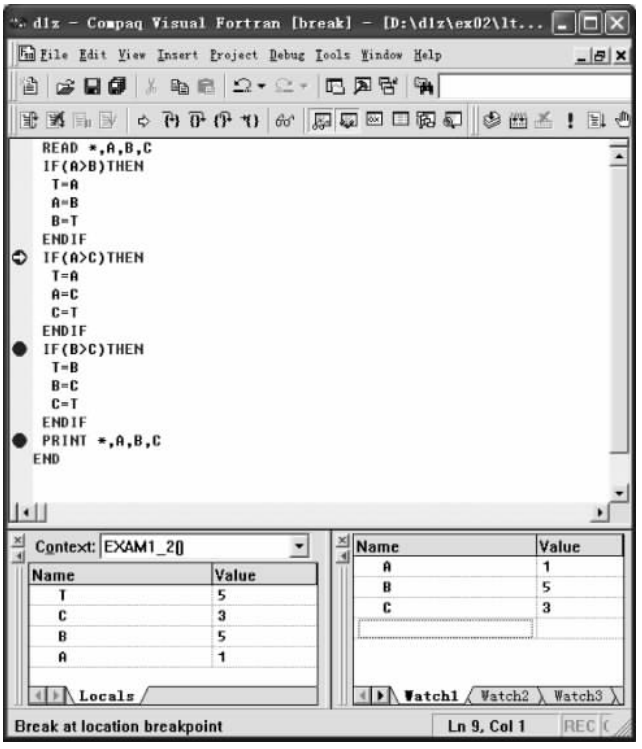

图 3.5　在第 1 个断点处暂停

(5) 单击“开始调试程序”按钮,从暂停断点处继续运行到下一个断点或结束程序运行。运行至第 3 个断点时变量窗口和观察窗口的显示内容进行了刷新,黑色数据为未刷新值,红色为刷新值,如图 3.6 所示,方框内数据为刷新数据。

图 3.6 调试过程中查看刷新数据

3.3.2 VS2010 的调试工具

这里只介绍 VS2010 使用调试工具、通过断点调试程序的过程。

(1) 设置断点。将光标插入点放在要设置断点的位置,单击鼠标右键,在弹出的快捷菜单中选择“断点”|“插入断点”,如图 3.7 所示。如果要删除断点,方法相同,如图 3.8 所示。

如图 3.9 所示,可以通过选择主窗口菜单“调试”|“窗口”|“断点”,打开“断点”窗口,如图 3.10 所示进行断点编辑。

(2) 启动调试。选择菜单“调试”|“启动调试”,或按 F5 键,如图 3.11 所示。

(3) 激活显示有关 Debug 调试窗口。选择菜单“调试”|“窗口”|“监视”|“监视 1”或“调试”|“窗口”|“局部变量”,如图 3.12 所示,打开常用的显示窗口。添加要查看的项,如图 3.13 所示,在第 1 个断点暂停。

(4) 按 F5 键或单击工具栏上的 ▶ 按钮,从暂停断点处继续运行到下一个断点或结束程序运行。运行至第 2 个断点时局部变量窗口和监视 1 窗口的显示内容进行了刷新,黑色数据为未刷新值,红色为刷新值,如图 3.14 所示,方框内数据为刷新数据。运行至第 3 个断点时局部变量窗口和监视 1 窗口的显示内容如图 3.15 所示。

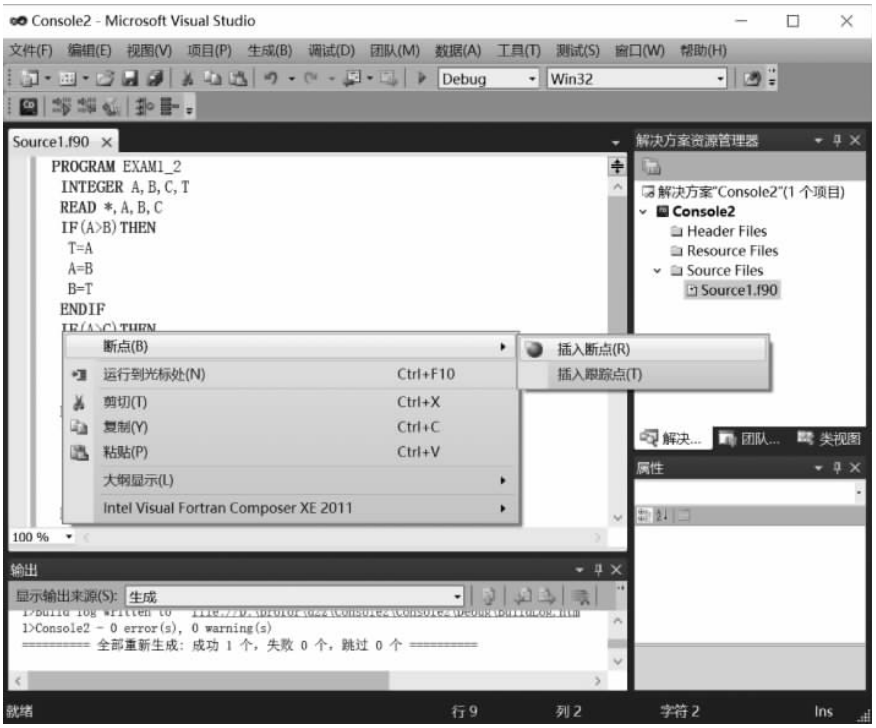

图 3.7　设置断点

图 3.8　删除断点

图 3.9　打开断点窗口菜单项

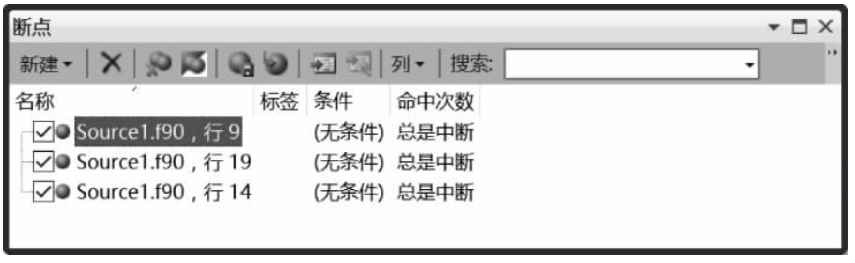

图 3.10　断点窗口

图 3.11　启动调试

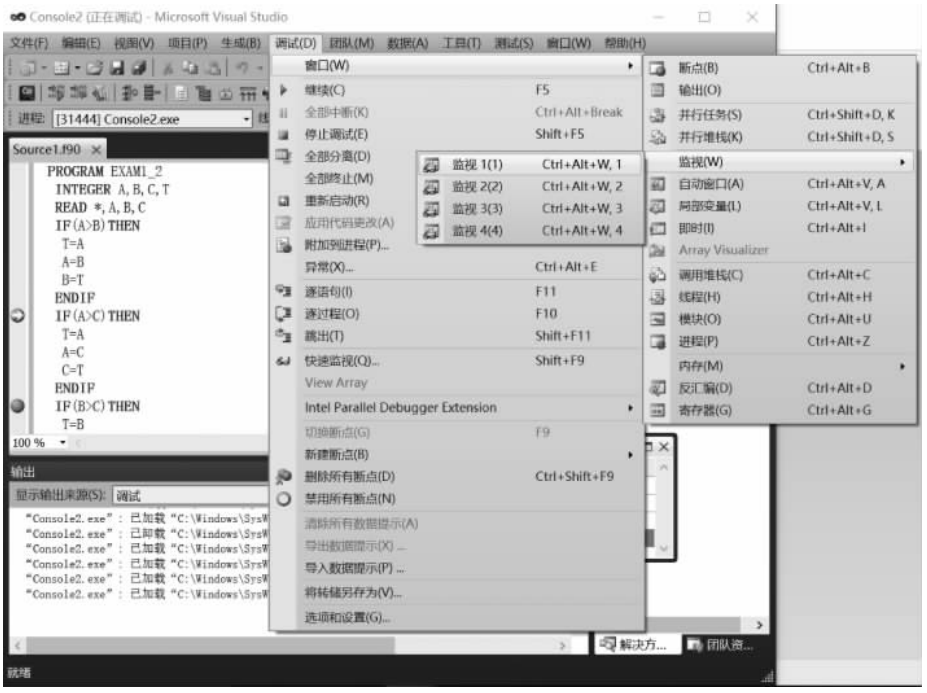

图 3.12　打开调试窗口

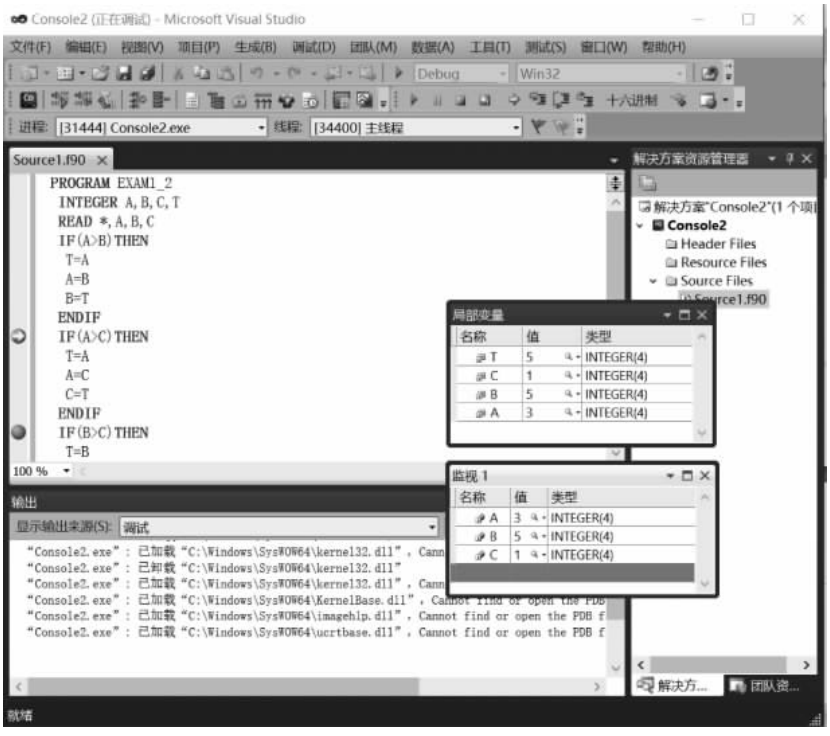

图 3.13　在第 1 个断点暂停窗口查看情况