

Visual Studio 2010 集成开发环境入门使用手册

Visual Studio，以下简称 VS。

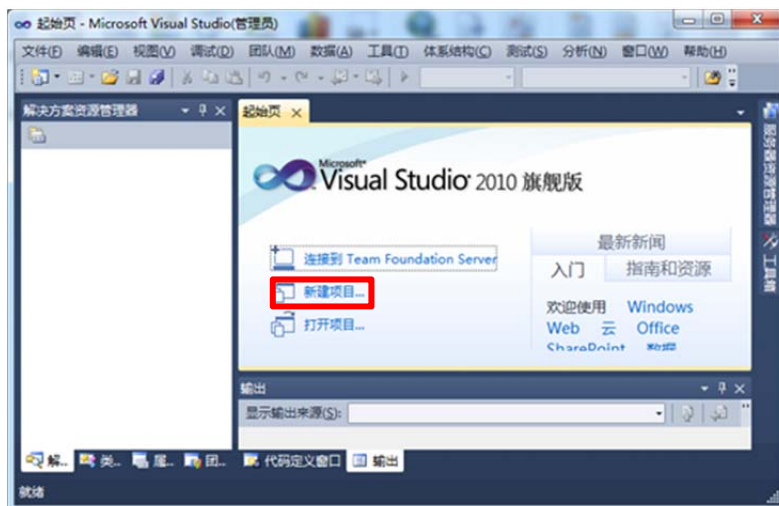
集成开发环境(Integrated Development Environment, 简称 IDE)是用于提供程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面工具。集成了代码编写功能、分析功能、编译功能、调试功能等一体化的开发软件。

一、基本使用

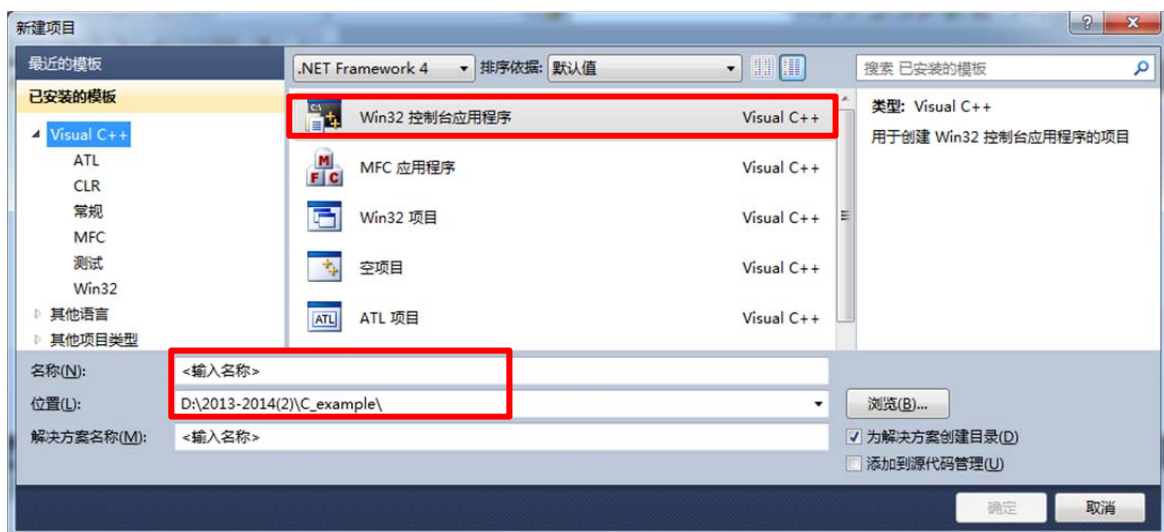
1. 启动：从开始菜单中选择“Microsoft Visual Studio 2010”即可启动。如果是第一次启动，那么可能会让你选择默认的环境设置，我们要使用 VC++，当然选择 VC++ 的配置，如下图所示。



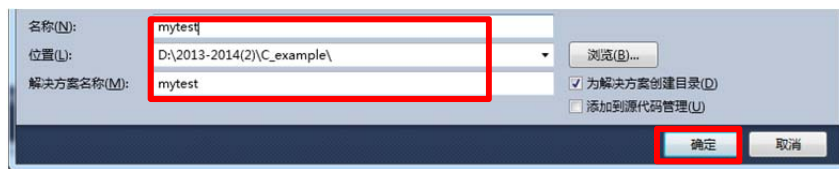
出现下面的界面表示已经成功运行了，这是起始页面，以后你会经常见到它。



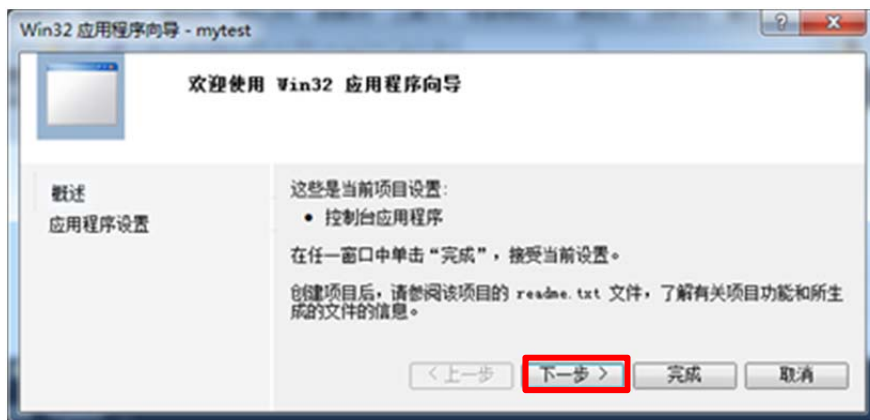
2. 在上图中选择“新建项目...”，或从“文件”菜单中选择“新建”→“项目”，弹出如下窗口：



3. 单击“Win32 控制台应用程序”，并在下面的“位置”处选择保存路径（不要存在 C 盘或桌面），然后在“名称”框中输入名称，例如：mytest，然后单击“确定”，如下图所示。



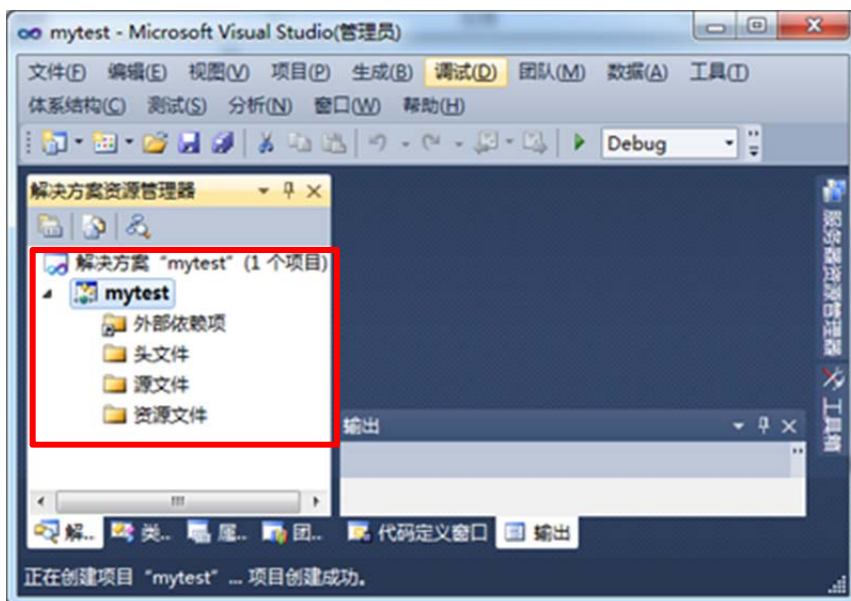
4. 在弹出的窗口中单击“下一步”，如下图所示。



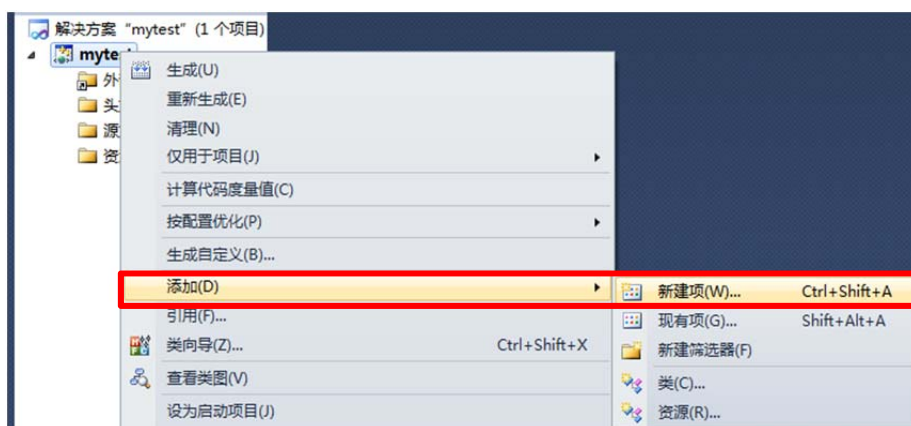
5. 在弹出的窗口中勾选“空项目”，然后单击“完成”，如下图所示。



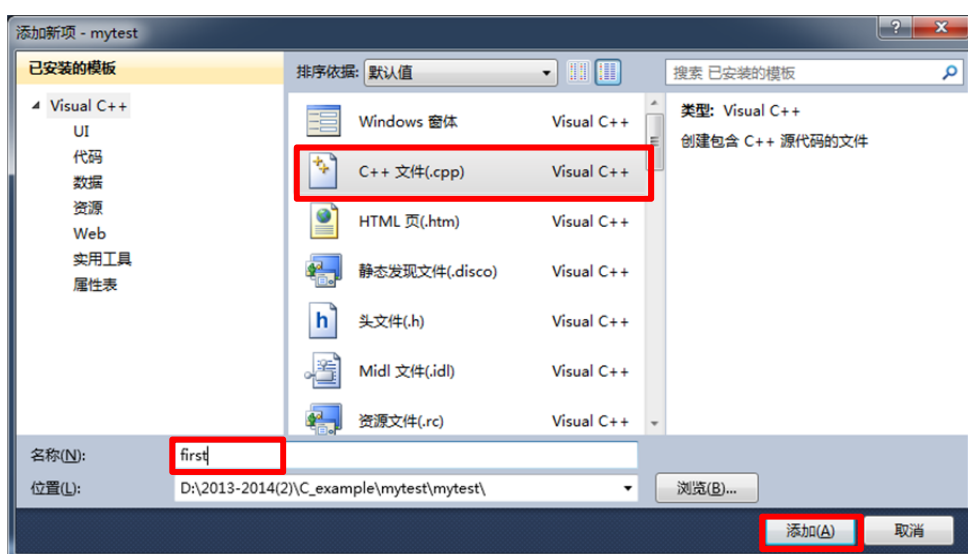
6. 此时，VS 已建好了一个名为 mytest 的项目，如下图所示。



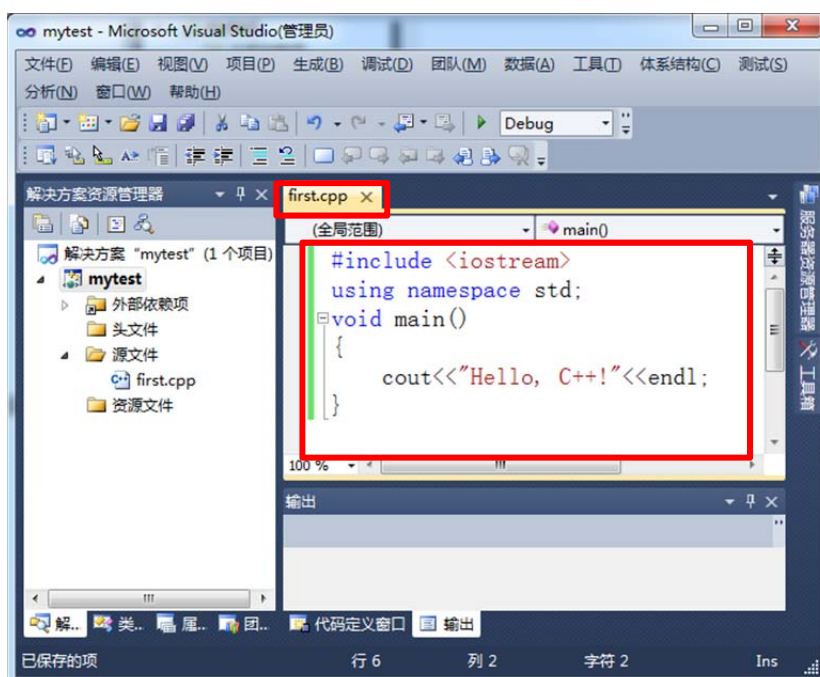
7. 现在，mytest 还是一个空的项目，我们需要向其中添加一个源文件，以便编写 C++ 程序。在 mytest 上单击右键，从弹出菜单中选择“添加”→“新建项”。



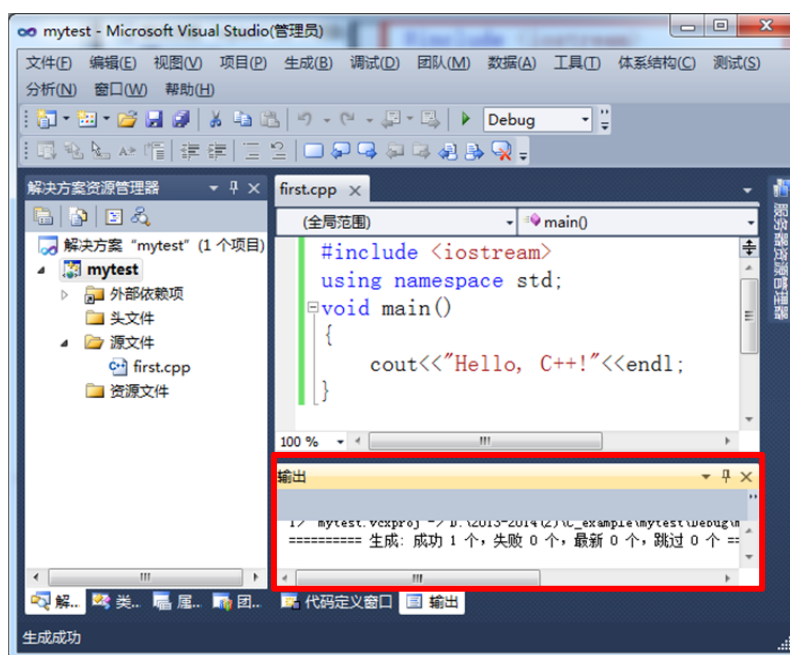
8. 如下图所示，在窗口中单击“C++ 文件(.cpp)”，输入名称，例如 first，然后单击“添加”。



9. 此时，VS 会打开一个编辑窗口，其标签是“first.cpp”，在该编辑窗口中输入源程序，如下图所示。

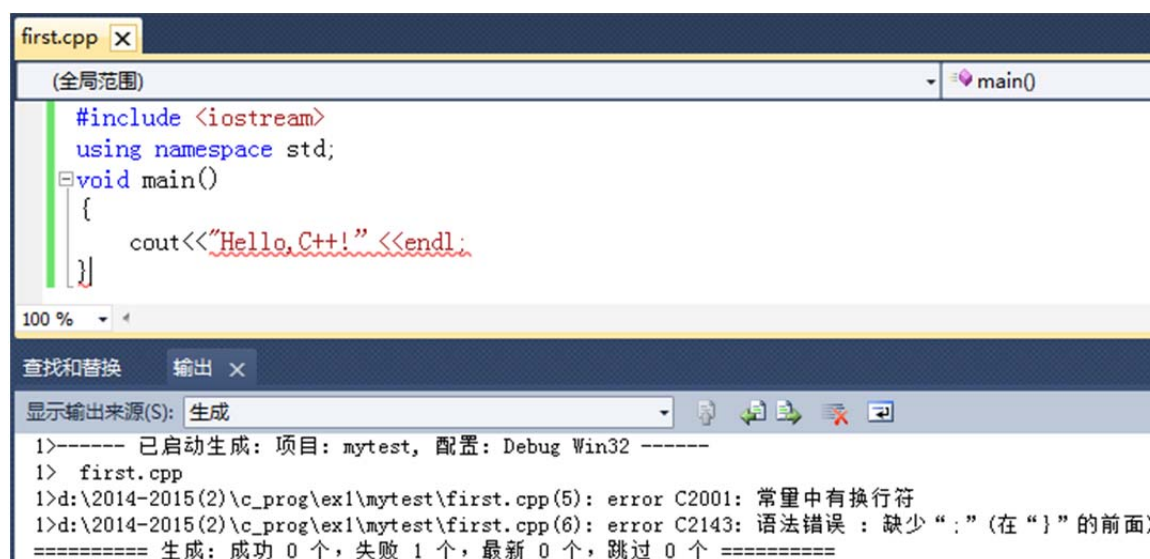


10. 单击“生成”菜单下的“编译”或直接按 **Ctrl+F7** 键，对源程序进行编译。在下面的输出窗口会有相应的提示信息，如下图。



若提示信息显示“成功 1 个，失败 0 个”，说明编译成功，直接进入下一步。

若提示信息显示“成功 0 个，失败 1 个”，则说明源程序中有语法错误，可在输出窗口中查看错误信息，如下图。可双击错误信息，则光标会定义到源程序中错误所在的行。

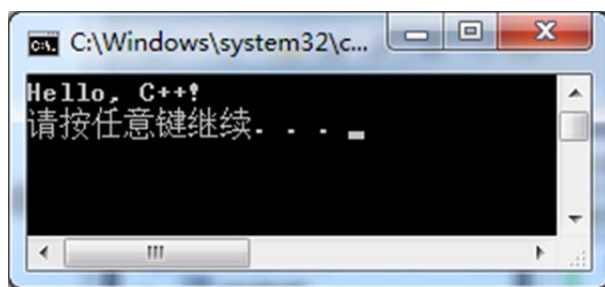


要特别注意，如上图，编译器提示：第 6 行有语法错误，但实际上，错误可能不在第 6 行，很有可能是由于第 6 行前面的某个语法错误而导致的，此时应检查第 6 行及其前面的几行。通过检查发现，错误在第 5 行，字符串 `Hello,C++!` 后面的双引号不小心敲成了中文状态下的双引号，改成英文的双引号后，再次单击“生成”菜单下的“编译”，编译成功。

Tips: 如上图，凡是加了红色波浪线的，都是有语法错误的，在编译前应认真检查。

11. 单击“生成”菜单下的“生成解决方案”或直接按 **F7** 键。在下面的输出窗口会有相应的提示信息，若显示“成功 1 个，失败 0 个”，则可进入下一步。

12. 单击“调试”菜单下的“开始执行（不调试）”或直接按 **Ctrl+F5** 键，运行程序。可看到程序的运行结果，如下图。按任意键后可关闭该窗口。



13. 如需对程序进行修改，在编辑窗口中修改完成后重复第 10、11、12 步即可。

二、项目和解决方案

项目是构成某个程序的全部组件的容器，该程序可能是控制台程序、基于窗口的程序等。程序通常由一个或多个包含用户代码的源文件、可能还有头文件、其他包含辅助数据的文件组成。某个项目的所有文件都存储在相应的项目文件夹中，关于该项目的详细信息存储在一个扩展名为.vcxproj 的 XML 文件中，该文件同样存储在相应的项目文件夹中。

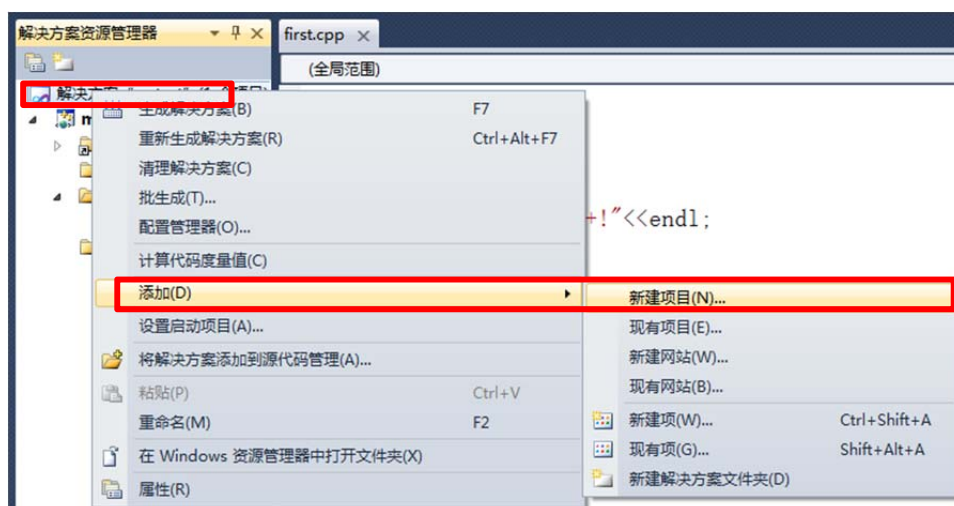
解决方案是一种将所有程序和其他资源聚集到一起的机制。例如，用于企业经营的分布式订单录入系统可能由若干个不同的程序组成，而各个程序可能是作为同一个解决方案内的项目开发的，因此，解决方案就是存储一个或多个项目有关的所有信息的文件夹，这样，就有一个或多个项目文件夹是解决方案文件夹的子文件夹。当我们创建某个项目时，如果没有选择将该项目添加到现有的解决方案中，那么系统将自动创建一个新的解决方案。

三、在一个解决方案中创建多个项目

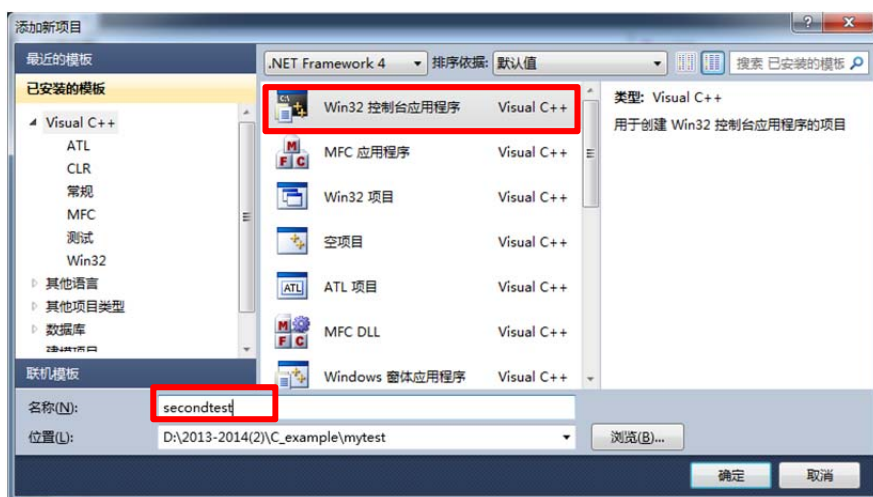
如果一次实验需要完成多个互相独立的源程序的编写与调试，则可以创建一个解决方案，然后在该解决方案中创建多个项目，每个项目对应一个功能独立的源程序。

1. 在一个解决方案中添加一个项目

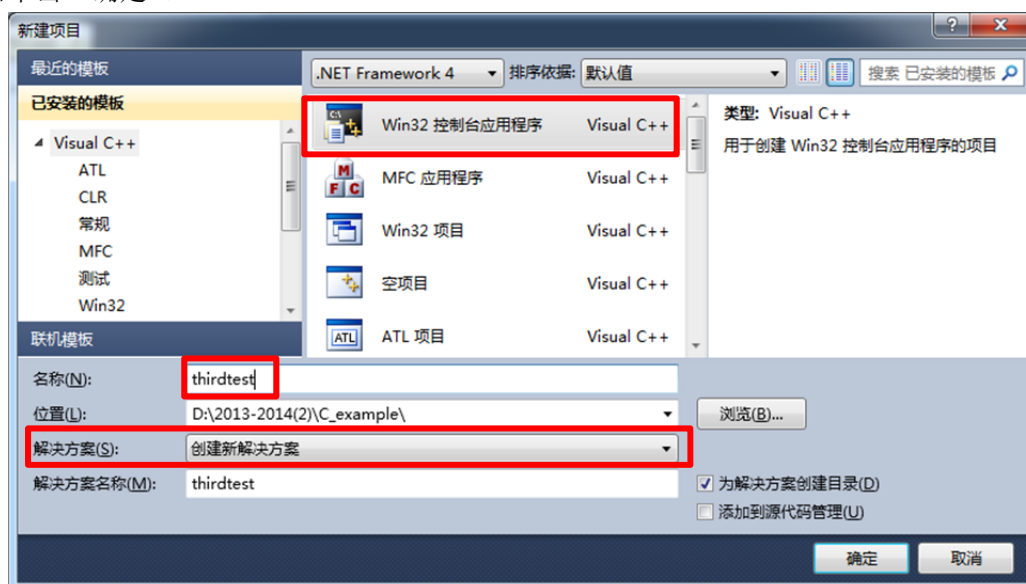
方法一：如下图所示，在“解决方案 mytest”上单击右键，从弹出菜单中选择“添加”→“新建项目”。



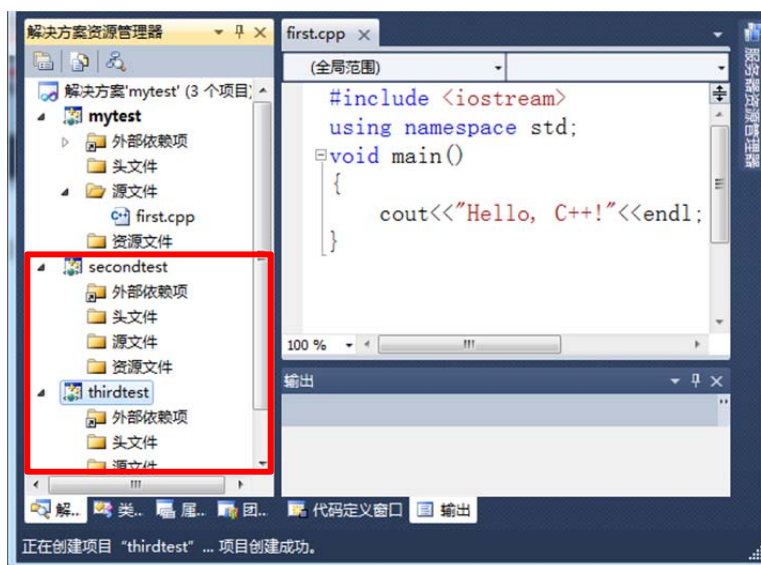
在弹出的窗口中，选择“Win32 控制台应用程序”，输入名称，例如 secondtest，然后，单击“确定”。如下图。



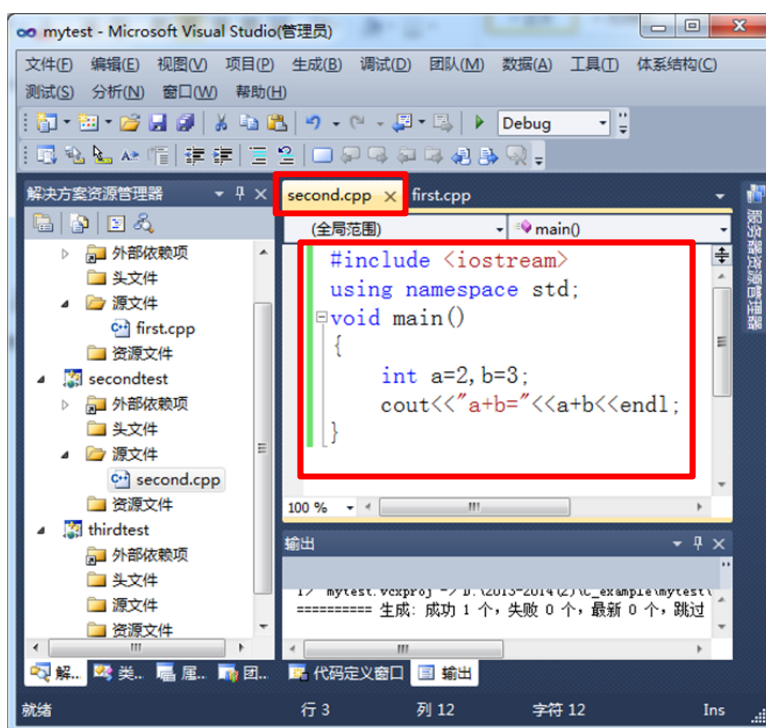
方法二：从“文件”菜单下选择“新建”→“项目”，然后，选择“Win32 控制台应用程序”，输入名称，例如 **thirdtest**，如下图。如果需要将项目 **thirdtest** 包含到现有的解决方案中，应该在解决方案后的下拉列表中选择“添加到解决方案”（如果不选择的话，系统默认项目 **thirdtest** 自动创建一个新的解决方案），然后单击“确定”。



2. 接下来，采用“一、基本使用”中第 4、5 步的方法，即可创建一个新的项目，如下图。



3. 采用“一、基本使用”中第 7、8、9 步的方法向新项目中添加一个.cpp 文件，并在其中编写源程序。例如，向项目 `secondtest` 中添加一个 `second.cpp` 文件。



4. 单击“生成”菜单下的“编译”。

5. 单击“生成”菜单下的“生成解决方案”或“生成 `secondtest`”，其区别在于：

在 VS 中，一个解决方案是可以加入多个项目的，如果当前解决方案中只有一个项目，执行“生成/重新生成/清理解决方案”和“生成/重新生成/清理 `secondtest`”是一样的；当有多个项目时，执行“生成/重新生成/清理解决方案”对解决方案下的所有项目都有效，执行“生成/重新生成/清理 `secondtest`”只对项目 `secondtest` 有效。

相关菜单的含义解释如下：

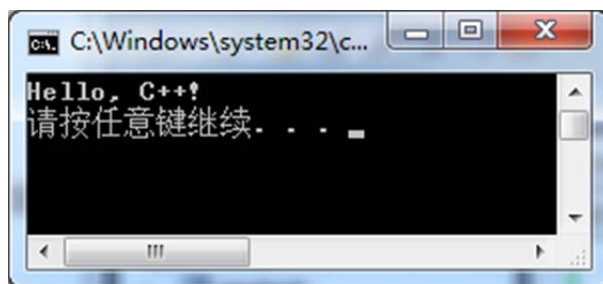
清理解决方案：把编译器编译出来的文件都清理掉，包括可执行文件、链接库等。

重新生成解决方案：顾名思义，就是重新编译每个文件，这样速度要慢些，但可靠度高一些。实际上，就是先清理一次，然后对所有文件进行编译。

生成解决方案：在上次编译的基础上编译那些修改过的文件，而没有修改的文件不编译。

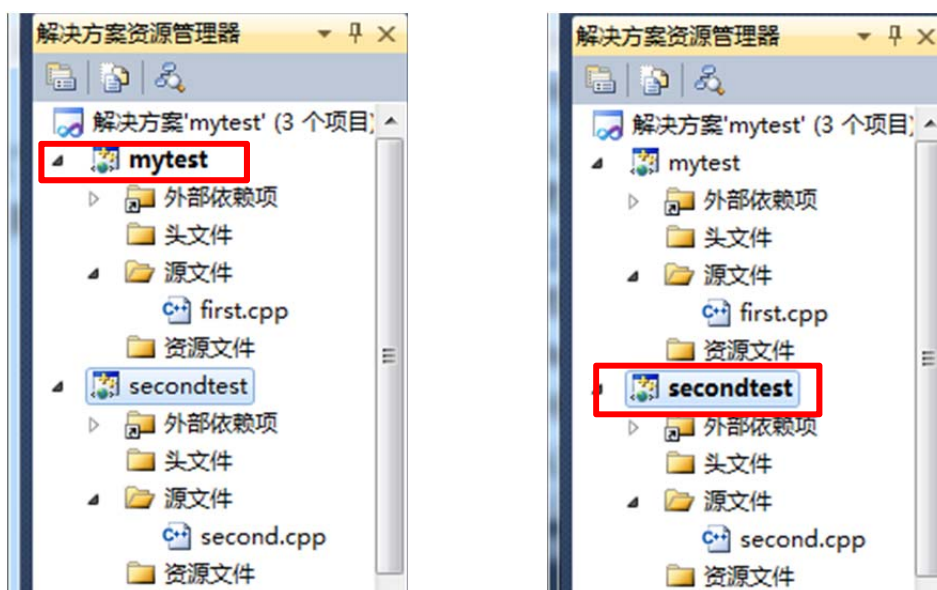
以 `cpp` 为例，当你只改动某些.cpp 之类的文件的时候，可以用“生成”，省了编译没有改动的那些文件的时间；但是如果你改动了某些.h 之类的文件，最好用“重新生成”，因为有可能有些文件包含.h 文件也需要重新编译。

6. 单击“调试”菜单下的“开始执行（不调试）”或直接按 `Ctrl+ F5` 键，运行程序。此时，我们会发现，程序的运行结果如下图所示，这并不是正确的运行结果，而是 `first.cpp` 的运行结果。



出现这种情况的原因是：现在，项目 `mytest` 才是当前启动项目（加粗显示，如下面的左图所示），因此，按 `Ctrl+ F5` 键后执行的是 `mytest` 下的可执行程序。所以，我们需要将项目 `secondtest` 设为启动项目，

方法是：在左侧窗口中的 **secondtest** 上单击右键，从弹出菜单中选择“设为启动项目”，此时，我们会发现 **secondtest** 是加粗显示的，表示它是启动项目，如下面的右图所示。



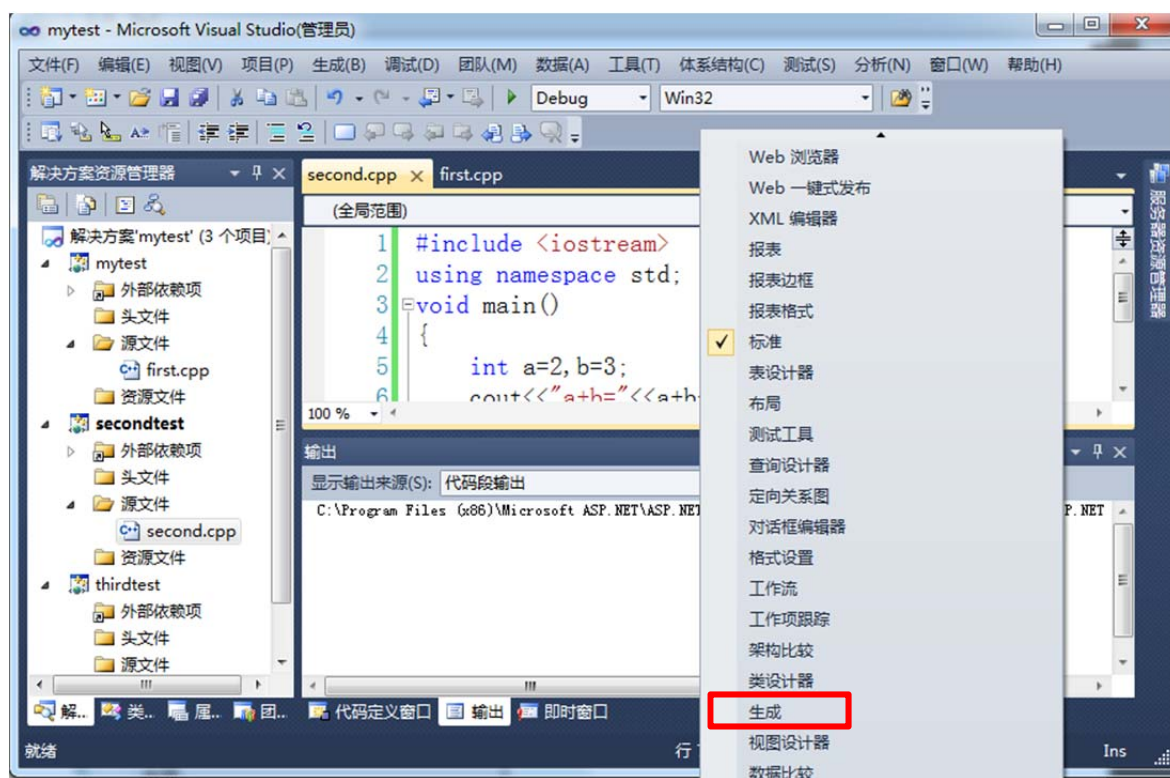
7. 再次单击“调试”菜单下的“开始执行（不调试）”或直接按 **Ctrl+F7** 键，运行程序，即可得到正确的运行结果，如下图。




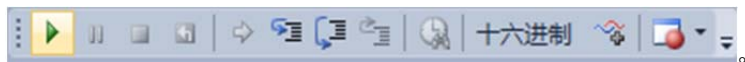
四、一些常用设置

1. 将常用工具栏（例如“生成”、“调试”）显示出来

右键单击工具栏的空白区域，从弹出菜单中选择“生成”，即在“生成”的前面打上勾，如下图。



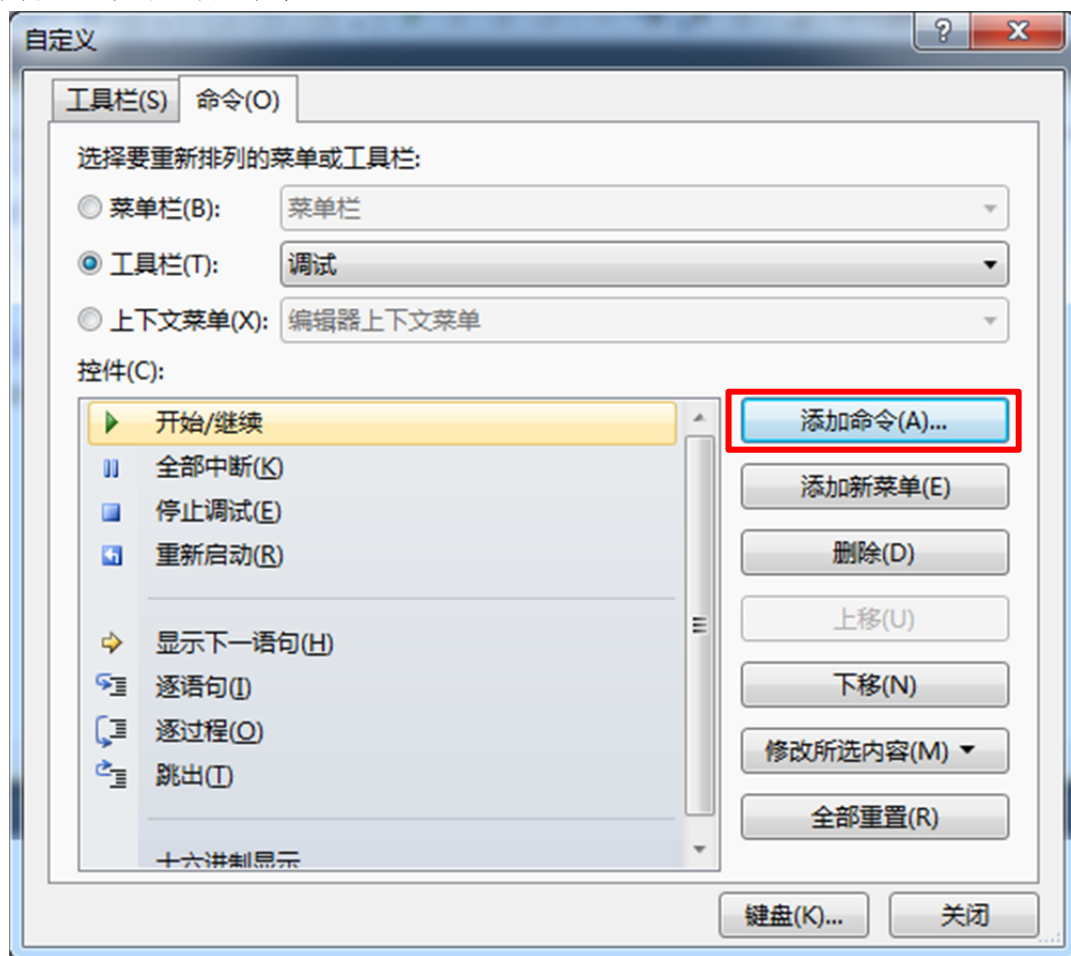
然后,我们会发现“生成”工具栏已显示出来,,鼠标移过去,就会有“生成 secondtest”、“生成解决方案”之类的提示,这样,以后我们就可以直接单击工具栏上的按钮来完成相应的任务了。
同样的方法,将“调试”工具栏也显示出来,如下:



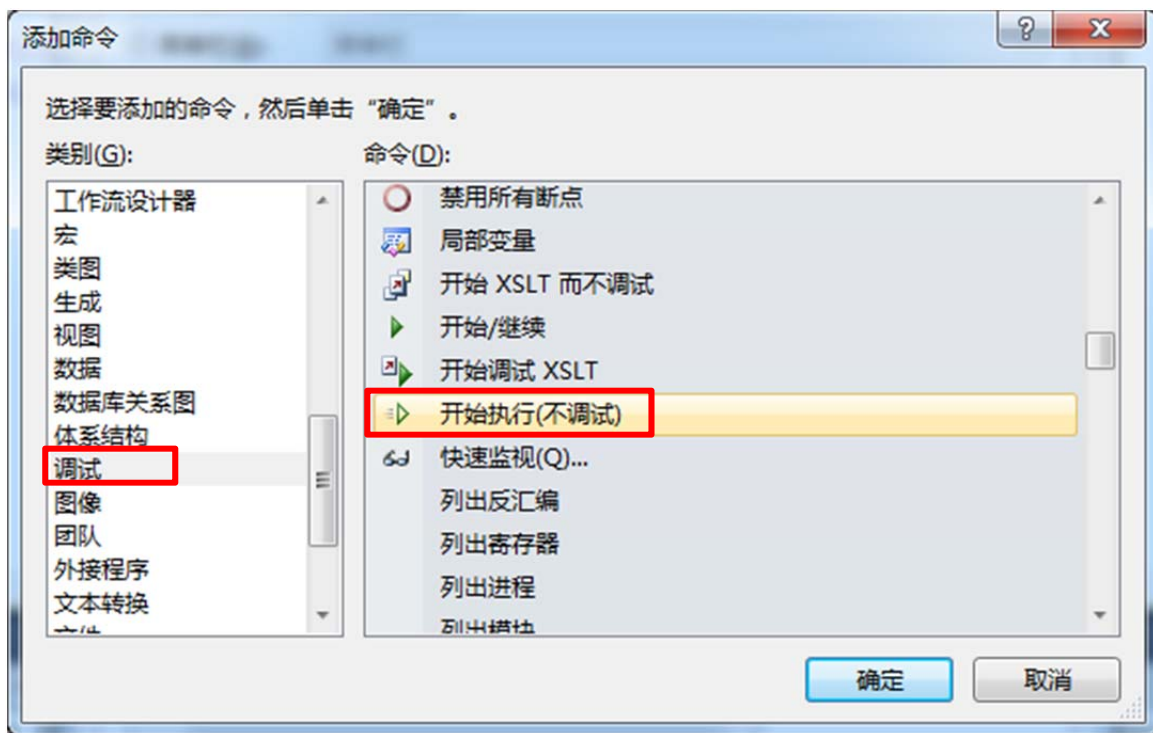
我们会发现,工具栏上默认没有“开始执行(不调试)”的按钮,现在我们要把它调出来。单击“调试”工具栏右侧的箭头,然后选“添加或移除按钮”→“自定义”。




弹出如下窗口,单击“添加命令”。



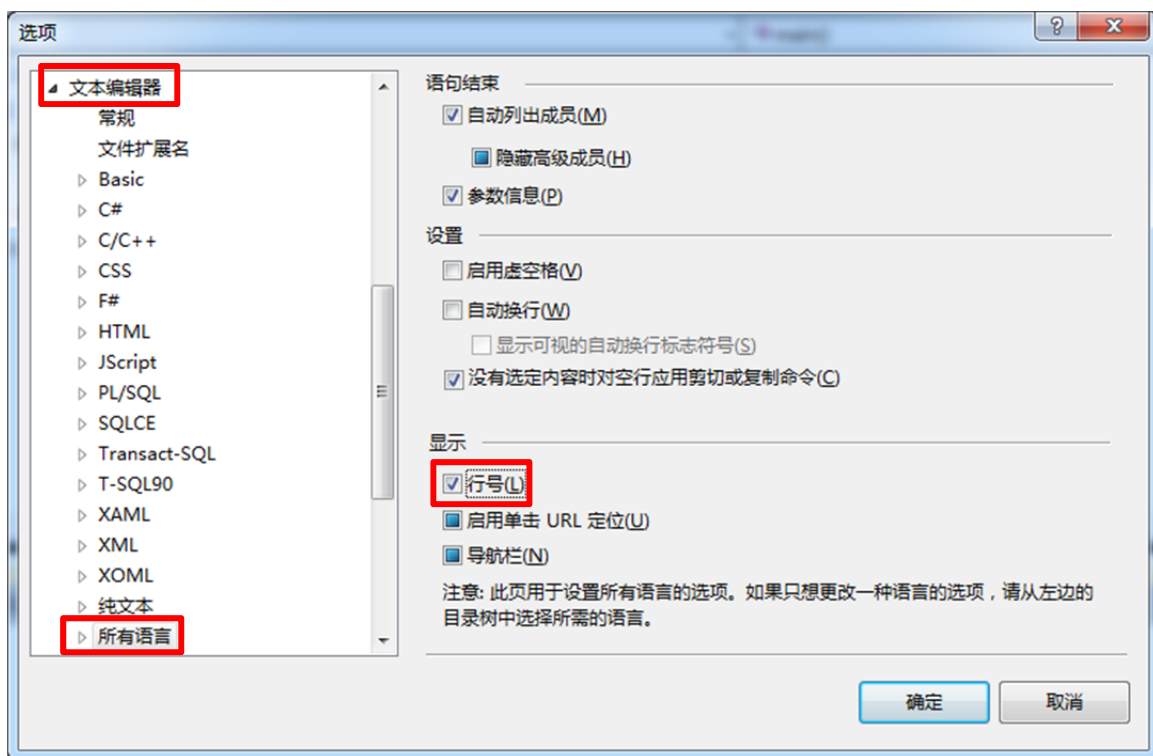
弹出如下窗口,在左侧找到“调试”,然后在右侧找到“开始执行(不调试)”,单击“确定”,会回到上一步的窗口,单击“关闭”即可。



此时，“调试”工具栏上多出了一个空心带尾巴的箭头，它就是“开始执行(不调试)”按钮。

2. 显示行号

单击“工具”菜单下的“选项”，然后选择“文本编辑器”下的“所有语言”，将“行号”前打上勾。如下图：



五、VC 2010 基本调试技术

故障是程序中的错误，而调试就是寻找并消除故障的过程。程序故障的首要来源是编程人员和编程人员所犯的错误。我们可能在代码中犯以下两种错误：

语法错误：这些是因形式错误的语句而引起的错误，比如漏写了语句最后的分号，或者使用了未定义的变量等。我们不必过多担心语法错误，编译器能够识别所有语法错误，并给出关于错误的提示信息，所以，此类错误是很容易改正的。

语义错误：出现这些错误时，代码在语法上正确，但却不能做我们本来想做的事情。编译器无法知道我们想要在程序中达到什么目的，因此不可能检测到语义错误。VC 2010 的调试功能就是为了帮助我们发现语义错误。

1. VC 2010 的调试器

调试器是一个程序，它以下述方式控制着程序的执行过程：我们可以一次一行地单步调试源代码，或者运行到程序中特定的位置。在代码中每个使调试器停下来的位置，我们都可以在继续执行之前检查乃至修改变量的值。我们还可以修改源代码，重新编译并使程序从头开始运行。我们甚至可以在单步执行某个程序的中间修改源代码。当我们修改代码之后移到下一步时，调试器自动在执行下一条语句之前重新编译。

当我们编写的程序未能表现出应有的行为时，调试器允许我们一次一步地检查程序，以找出出现问题的位置和原因。还允许我们在执行过程中在任何时间检查程序中数据的状态。


在调试某程序之前，首先一定要将该程序的编译配置设定为 Win 32 Debug，而非 Win 32 Release。如下图所示：



编译程序时，项目中的 Debug 配置将使可执行程序中包含附加信息，以允许我们使用调试功能。这种额外信息存储在.pdb 文件中，该文件位于项目的 Debug 文件夹中。Release 配置将省略这些信息，因为在经过充分测试的程序中，它们将带来一些不必要的系统开销。另外，编译器在编译 Release 版本时还将对代码进行优化。当编译 Debug 版本时，优化是禁止的，因为优化过程可能涉及到重新安排代码的顺序以提高效率，甚至可能完全省略掉冗余的代码。因为该过程破坏了源代码与对应机器代码块之间的一对一映射关系，所以使得程序的单步调试有可能出现混乱。

调试工具栏如下：



启动调试器的方法：单击调试工具栏上的  按钮，或直接按 F5 键，或单击菜单“调试”→“启动调试”。

调试器有两种主要的工作模式：

(1) 单步调试代码（实质上每次执行一条语句）；

(2) 运行到源代码中指定的位置。源代码中使调试器停止的位置或者由光标所在的位置确定，或者通过指定停止点（称为断点）的方式确定。

2. 断点

断点是程序中使调试器自动暂停执行的位置。我们可以指定多个断点，这样程序在运行过程中就可以在我们指定的位置停止。我们可以在各个断点上查看程序中的变量，并在变量值不符合要求的时候进行修改。

设置断点的方法：单击代码左侧边栏或者移动光标到指定行然后按 F9。此时，在代码左侧的灰色边栏将出现一个红点，表示该行存在一个断点，如下图所示。

```

int sum(int n) {
    int i, s=0;
    for (i=1; i<=n; i++)
    {    s=s+i;    }
    return s;
}

void main() {
    int b=6, he;
    he=sum(6);
    cout<<"1至"<<b<<"的累加和为: "<<he<<endl;
}


```


删除单个断点的方法：单击红点，或将光标定位到断点所在的行然后按 F9。


删除所有断点的方法：按 Ctrl+Shift+F9 组合键，或单击菜单“调试”→“删除所有断点”。

在调试时，我们通常会设置若干个断点，但并非每一行都适合添加断点，选择断点是为了看出我们认为有问题的变量被修改的时间。程序执行时，将在断点指定的语句被执行之前停止。如果我们将断点设置在不包含任何代码的一行，则程序将停止在下一行可执行语句的开始处。

3. 启动调试模式

(1)  F5（启动调试）：使程序执行到第一个断点，然后暂停。再次按下 F5 键，程序将继续执行到下一个断点。以这样的方式，我们可以使程序从一个断点执行到另一个断点，并在每次执行暂停时检查关键的变量，需要时修改它们的值。如果没有设置任何断点，则以这种方式启动调试器将执行整个程序，中间不暂停。

(2)  F10（逐过程）：一次一条语句地执行，若出现函数调用，则将函数一口气执行完。

(3)  F11（逐语句）：一次一条语句地执行，若出现函数调用，则进入函数内部一条语句一条语句地执行。

在调试过程中，经常是以上三种方式结合使用。例如，我们可以在疑似包含错误的位置设置断点，然后按 F5 运行程序，使程序停止在第一个断点上，然后我们可以从该断点开始按 F11 单步执行，若遇到 cout 语句等，机器还要单步执行供流输出使用的库函数的所有代码，而我们实际上对库函数并不感兴趣，这种情况下，我们就可以使用 F10 键。