

写在前面

在我还在上学的时候，我选择了 C++，最初我用 VC6 作为我的 IDE，我看过很多本 C++ 的教材，有的适合我，有的不适合我，其中有一本叫《Visual C++ 2005 入门经典》的书帮了我不少的忙。因为通常的 C++ 教材都只会介绍 C++ 的语法什么的，很少会告诉我们如何去编译、运行，告诉我们什么是控制台程序，什么事 Win 程序，什么是 GUI 程序，C++ 能干什么，VC 和 C++ 的区别是什么。现在有很多的朋友应该也有这些问题吧？

学 C++ 用 C++ 也有几年了，算不上熟悉，算是初窥门径吧，我想我应该做点什么帮助一下那些和曾经的我一样困惑的朋友，特别是学生朋友，告诉他们他们所困惑的问题的答案。记得我学 C++ 的时候，没有人教，有的时候也走了不少弯路，甚至连调试也不会，也不知道可以通过看调用堆栈看调用次序，还自己慢慢地去搜索，好傻啊。

接下来我会做一个《Visual C++ 2010 入门教程》系列，用来帮助初学者。刚开始学的时候是很痛苦的，这个我深有体会，特别是身边还没有人能够指导一二。内容主要涵盖在 Windows 下面使用 C++ 进行开发的常见内容，Visual Studio 2010 的使用，如何创建新项目，如何调试，如果配置项目属性等等，另外还会介绍 Visual C++ 2010 中新加如的一些内容，包括一些新的 STL 组建，一些新的语法支持等等。

由于本人水平有限，其中难免有错误，希望大家谅解，如果大家有发现问题还请务必及时指出来，否则误导了他人我就罪不容恕了。

注意，本教程非 C++ 教程，不会教你 C++，只会教你如何使用 Visual C++ 2010 去练习去学习其它 C++ 教材上面的程序。因此建议大家认真的去看其他的 C++ 教程，在使用 Visual C++ 2010 实践的时候如果遇到问题可以到这里来参考。推荐《C++ Primer》、《C++ 程序设计语言》《Visual C++ 2008 入门经典》。

第一章： 关于 Visual Studio、VC 和 C++的那些事

首先，这三个东西有什么区别呢？VC 和 C++是相同的吗，有什么区别呢？我刚开始学 C++的时候也有这样的问题，在这里我来替大家解释一下他们到底是什么。

Visual Studio，我们一般都简称为 VS，VC 全称是 Visual C++，C++就不用解释了吧？Visual Studio 其实是微软开发的一套工具集，它由各种各样的工具组成，这就好像 Office 2007 是由 Word 2007、Excel 2007、Access 2007 等等组成的一个道理。其中 Visual C++就是 Visual Studio 的一个重要的组成部分。Visual Studio 可以用于生成 Web 应用程序，也可以生成桌面应用程序，在 Visual Studio 下面，除了 VC，还有 Visual C#，Visual Basic，过去还有 Visual J#，现在还有 Visual F#等等组件工具，使用这些工具你可以使用 C++语言、C#语言或者 Basic 语言(微软改进版)进行开发。这就好比桌上放着刀、叉、筷子，你可以用它们来吃饭，无论用什么工具都可以，因为它们都是餐具的一种。

VC，全称是 Visual C++，它只是一个工具而已。

C++是一门和 C、Basic、C#、Java 一个概念的东西，它是一门语言，这个概念就同汉语、英语、法语之间的关系一样。语言的可以不同，他们有的复杂有的精简，比如汉语，我个人认为这是这个星球上最复杂的语言；又比如电影《阿凡达》中外星人使用的语言，不过 100 多个单词而已。当然，语言并没有高低贵贱之分，而对语言掌握的好坏，说的好坏是有高下之别的。

一个语言都有哪些要素呢？语法就是其中一个重要的方面，比如你要对你所喜欢的人表达爱慕之情，那么你应该说“我爱你”，你不能说“你爱我”，因为这搞错了主谓宾关系，因此我们通常都需要按照一定的规矩和原则表达才会让别人正确理解我们真正的含义。在我们的汉语中有一些成语，它们通常都比较简单短小，但是却能表达非常强烈的感情，而且效果非常好。比如你想骂一个坏人“你实在是太变态了，这种事情都干得出来。”你可以这样说：“你也太禽兽不如了！”对于编程的语言来说，其实也是有成语的，在我看来，这就是库函数。通常我们在编程的时候都推荐选择库函数，因为它通常会比较快一些。

一直以来都有不少人混淆 Visual C++和 Visual Studio，其实最初 Visual C++发布的时候还没有 Visual Studio 这个东西，Visual C++是一个独立的开发工具，与 Visual Basic 等并列，最后微软将它们整合在一起组成了 Visual Studio。

Visual C++从发布起到现在已经有 10 个大版本了，我们这里介绍的 Visual C++ 2010 就是 Visual C++ 10，简称 VC10。上朔 10 多年，Visual C++ 6.0 发布了，这个被称为史上最经典的 VC，现在有很多企业还在用它，大量的教材基于这个版本的 VC 来写的。然而实际上 VC6 并没有想象中的那么美好，孱弱的 IDE，不完善的 STL 等等让它越来越不适应时代的发展，同时由于后来的版本和

它的差别越来越大，让很多习惯于 VC6 教材的人在新版本上无所适从。VC6 走向历史的终结点其实是必然的，只在于时间问题罢了，况且大部分人都有追求最新的“嗜好”，所以这里也选择最新版的 VC 来做入门教程。

下面来介绍下这个 VC 版本和 VS 版本的对应关系：

Visual Studio .net (2002) --> Visual C++ .net (2002) --> Visual C++ 7.0 -> _MSC_VER 值 1300

这个是微软推出 .net 战略之后的最早的一个版本的 VC，这个版本相对 VC6 来说变化并不是很大，一些不标准的语法依然得到支持，不过 IDE 界面已经出现了 VS 时代的雏形。

Visual Studio .net 2003 -> Visual C++ .net 2003 -> Visual C++ 7.1 -> _MSC_VER 值 1310

这个版本我觉得只能算是一个对 .net 2002 的一个 Patch 版本，解决了一些 Bug，不规范的语法依然得到支持，比如 for 循环变量作用域的问题。

Visual Studio 2005 -> Visual C++ 2005 -> Visual C++ 8.0 -> _MSC_VER 1400

这个版本算是一个比较符合 C++ 标准的 VC 版本了，微软为这个版本的 VC 加入了大量的增强版的 C 库函数 (CRT 函数)，如 strcpy_s 之类，当然，这部分并不是 C 和 C++ 标准所要求的。

Visual Studio 2008 -> Visual C++ 2008 -> Visual C++ 9.0 -> _MSC_VER 1500

这个版本应该是在 Vista 发布之后出的，如果你是在 Vista 或者 Win7 下面使用，那么应该选择这个版本。这个版本与 VC2005 变化并不大，不过它的 SP1 为 C++ 带来一些新的 STL 库组件 (tr1 部分)，以及 MFC 下面的 Ribbon 界面。

Visual Studio 2010 -> Visual C++ 2010 -> Visual C++ 10.0 -> _MSC_VER 值 1600.

这个就是我们即将要介绍的版本，那么让我们来看一下它的启动画面吧。



这个是 Visual Studio 2010 中文旗舰版的启动画面。下一章中，我将教大家安装、配置 VS2010/VC2010，以及如何使用它来创建第一个 C++ 程序。

如果你没有下载过 Visual Studio 2010 的话，那么请你提前下载好它吧：

<http://hi.baidu.com/%E2%B7%B3%DE%B2%C2%D2/blog/item/bb0975dd801291d58c1029f5.html>

第二章 安装、配置和首次使用 VS2010

本章将帮助大家安装 Visual C++ 2010，帮助大家做一些常见的配置，以及第一次使用它来写 HelloWorld 程序。

安装

Visual C++ 2010 是属于 Visual Studio 2010 的一部分，这个在前一章中已经讲解过，实际上 Visual C++ 2010 也有更多的子版本，正如 Windows7 有旗舰版、家庭高级版、家庭初级版一样。在上一章的下载地址中有两个版本可以选择，我选择的时候旗舰版，这个功能最全面。

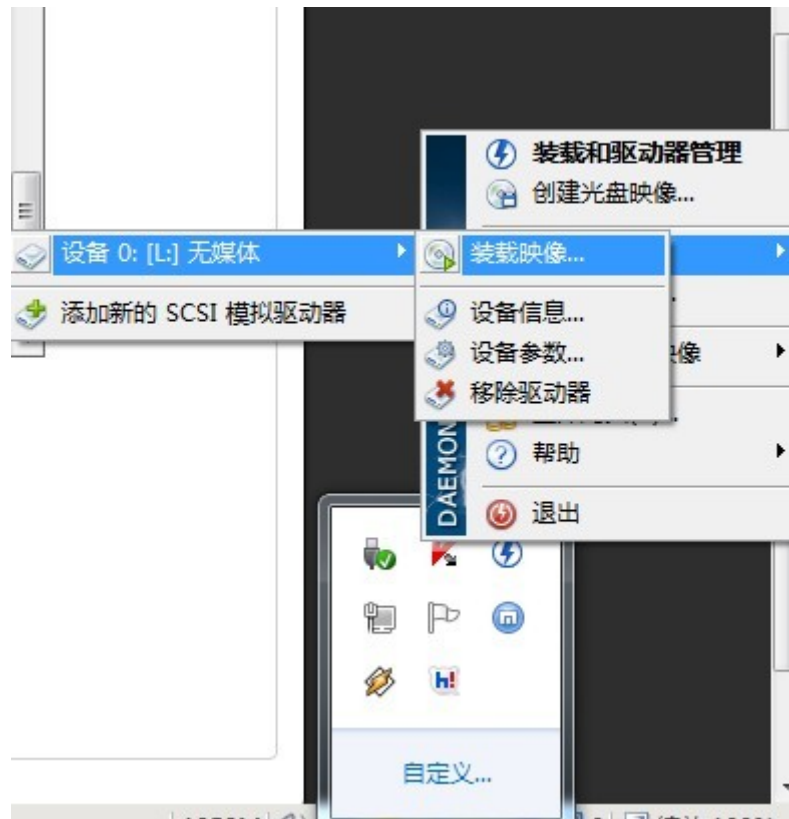
如果你下载完毕了，应该有一个 ISO 镜像文件，如果你是从我给的地址下载的，那么那个序列号已经被 替换成正版序列号，直接安装即可。下面是 ISO 文件截图：



这就是光盘镜像文件，我们有多种方式可以安装它，你可以把它烧录成光盘，也可以用虚拟光驱软件安装它，由于这个 ISO 已经被修改过了，你甚至可以直接解压它，然后运行里面的 **setup.exe** 进行安装。

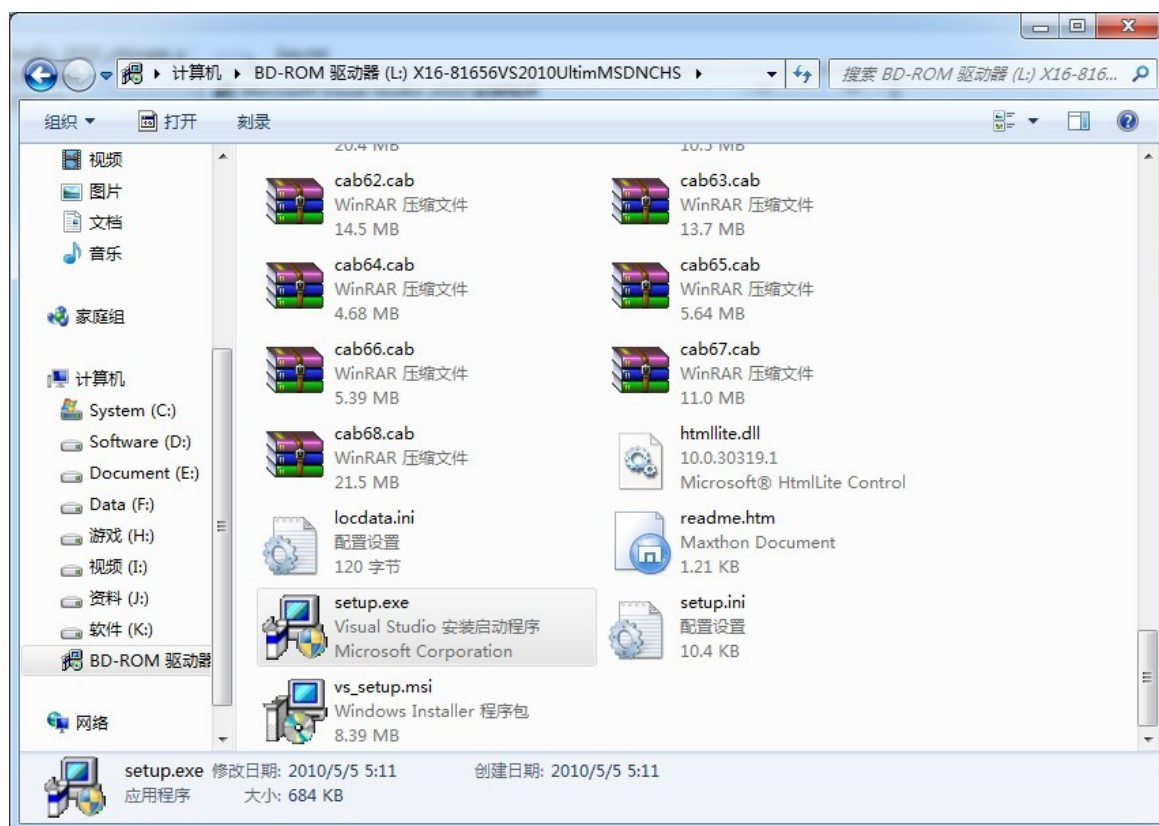
这里我使用虚拟光驱进行说明，首先我假设你已经安装好了虚拟光驱软件 **DAEMON Tools Lite**，如果你还没有安装，请去游侠补丁网下载，因为在那里你总是能找到最新的免费版本。

接下来我们通过虚拟光驱加载刚才下载好的 ISO 镜像：





在弹出的对话框中选择我们下载好的 ISO 文件。这个时候，虚拟光驱里面已经替我们打开了 ISO 文件，有的机器可能会自动弹出安装提示，有的不会。如果自动弹出安装提示，那么直接通过即可，如果没有弹出，那么用资源管理器方式进入虚拟光驱点击 **Setup.exe** 即可开始安装。



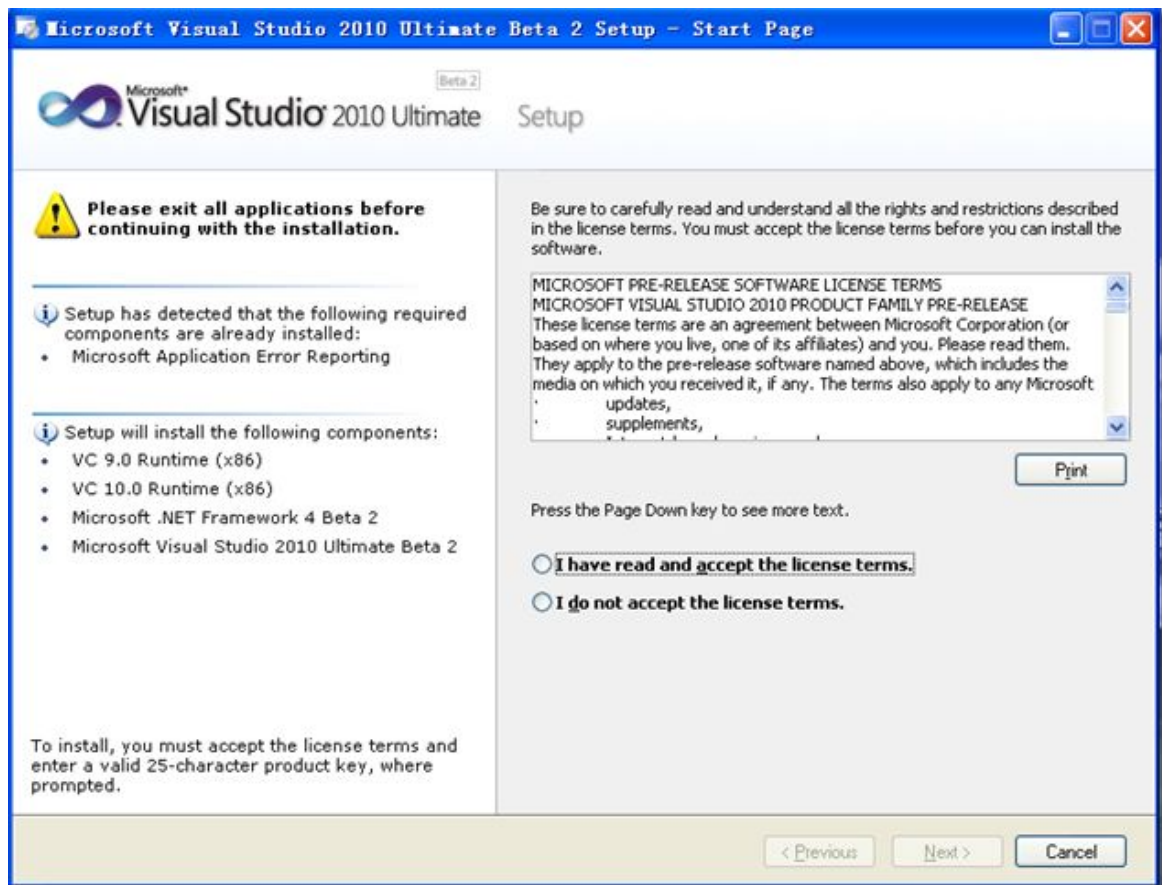
由于我已经安装好了 VS2010，所以接下来无法给大家截图了，我就直接用文字描述吧。

当弹出 VS2010 安装程序之后点下一步，然后我们可能会看到 完全 最小 自定义 这样的选项，我推荐大家选择自定义，因为只有这样我们才可以自定我们安装的目录。接下来会选择安装的组件。这个时候我们会看到大量的 VS2010 的组件，这里我们可以选择是否

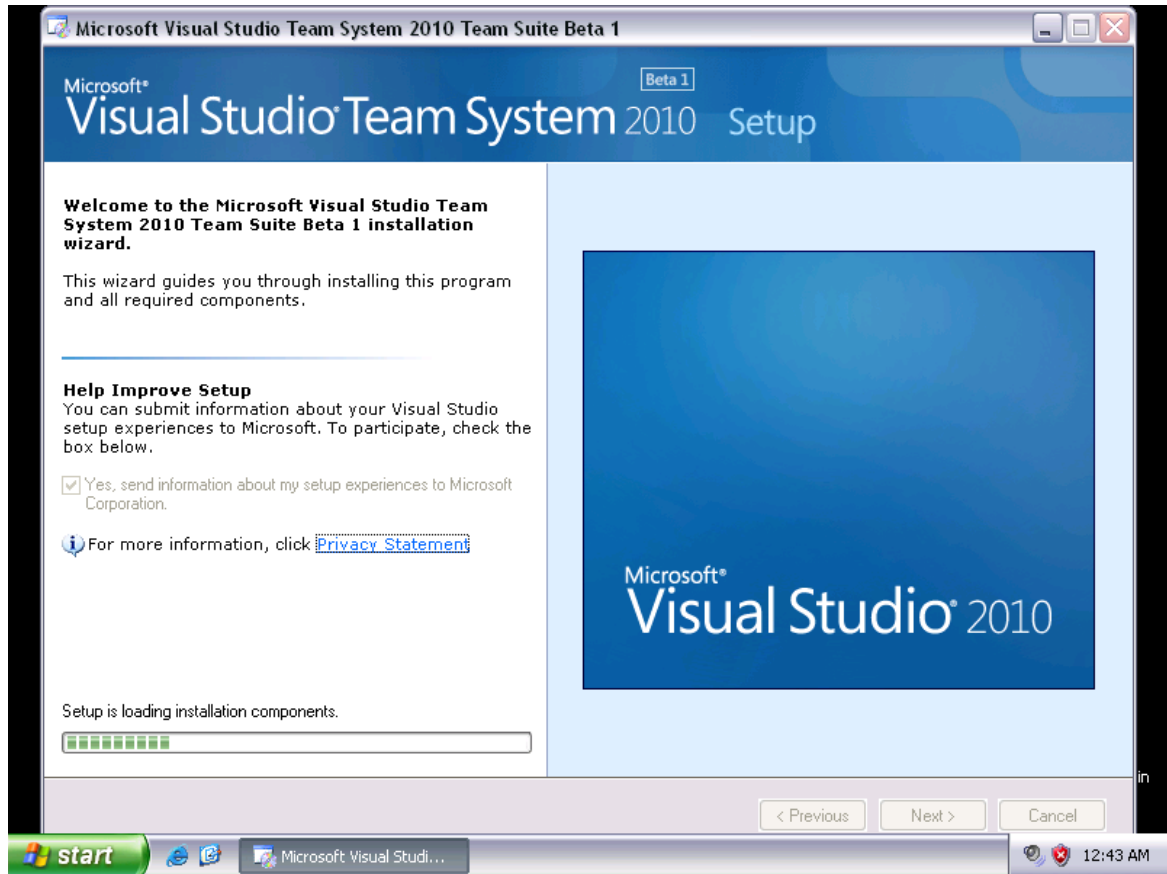
安装 Basic 是否安装 C#等等，我在网上找到一些图片，虽然跟中文版 2010 不一样，大家大概的样子是一样的，大家就凑合着看看吧。

下面是开始的界面，我们应该选择安装 VS2010.

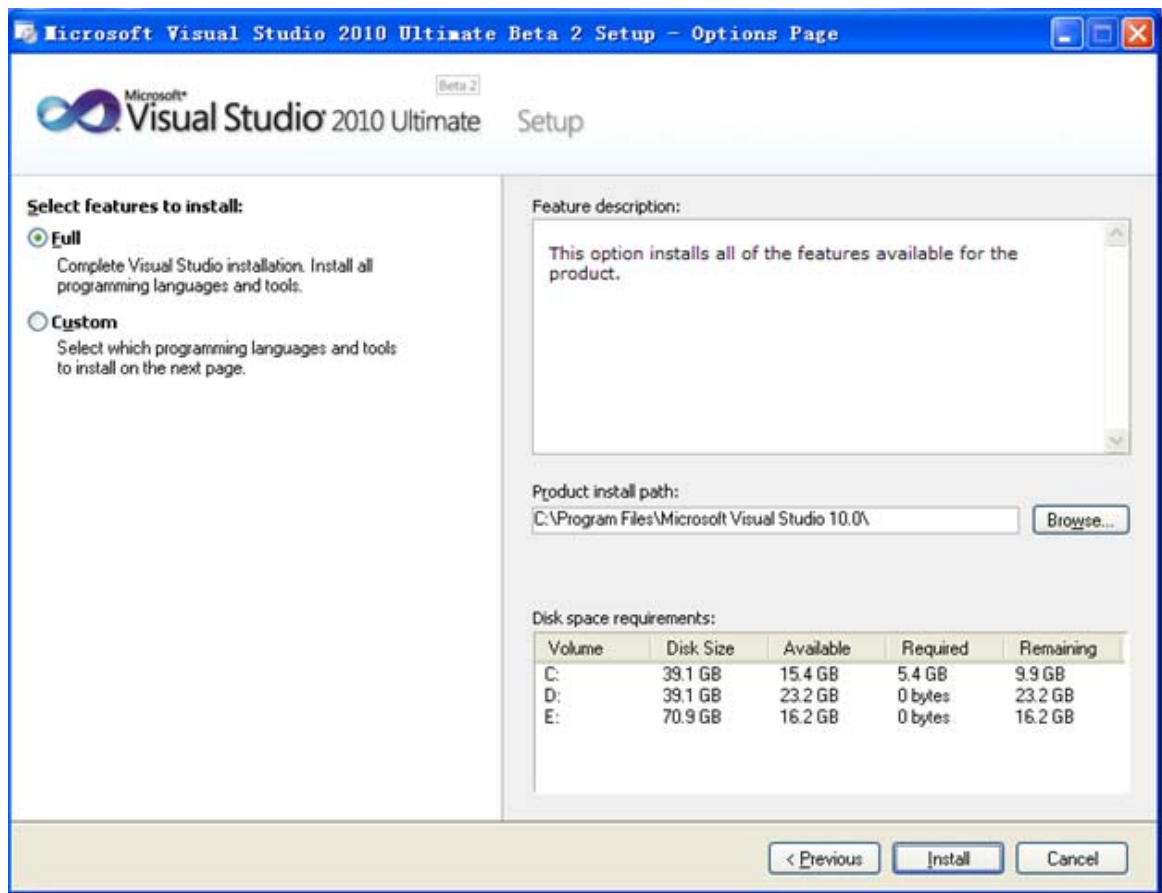




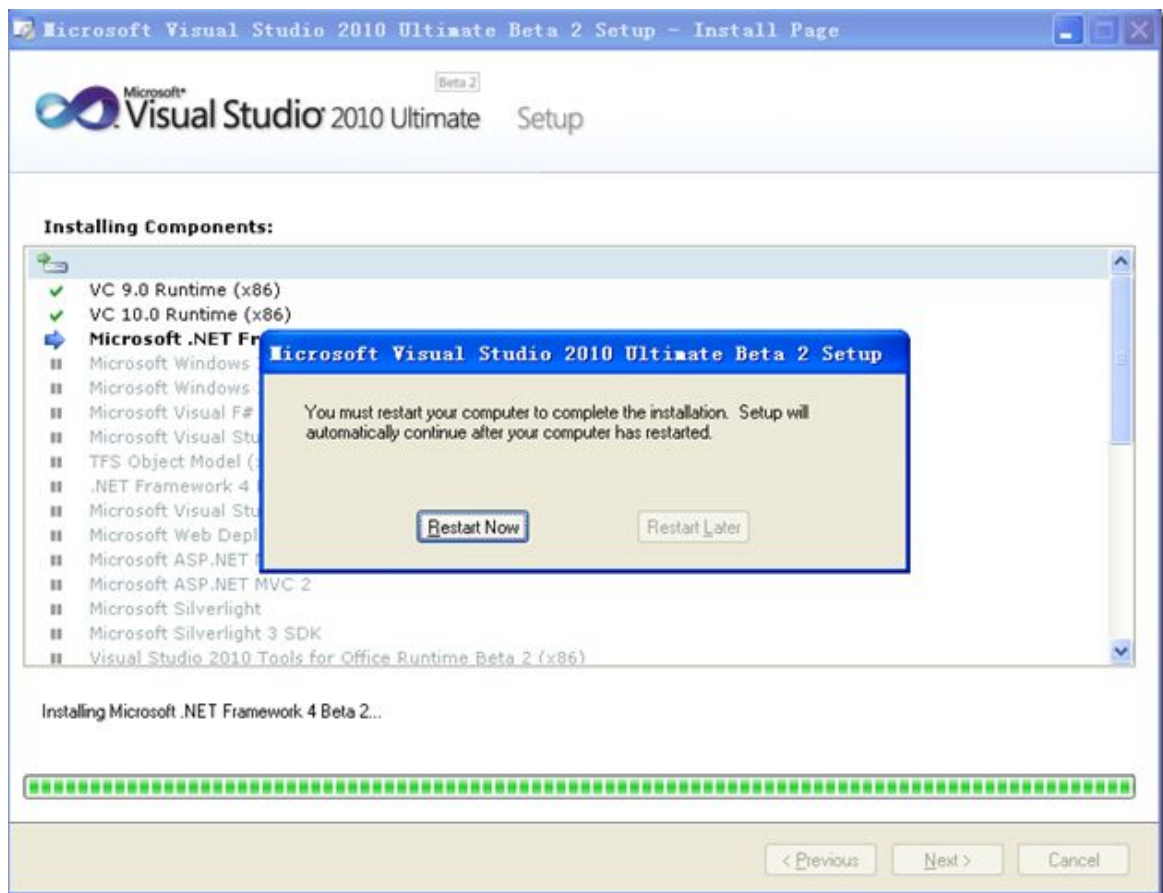
上面这个时候选择同意。然后安装包会开始搜集信息：



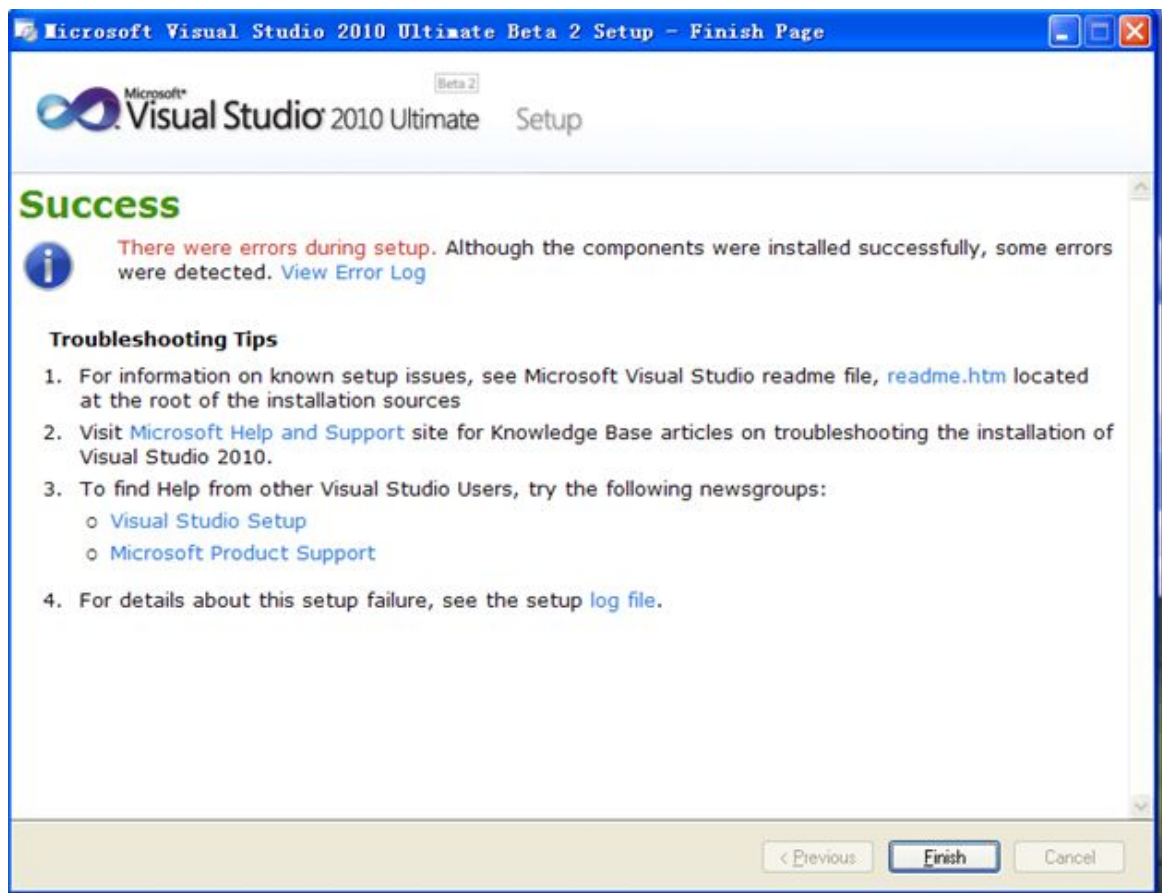
选择目录:



然后就是开始安装了，注意安装的时候可能需要重启一两次。



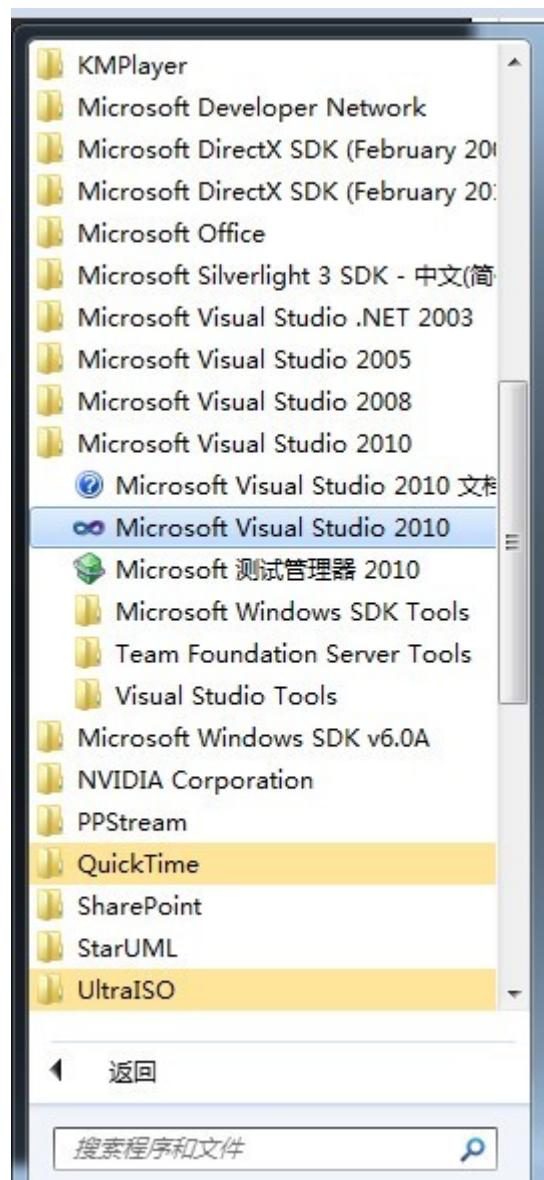
最后安装成功:



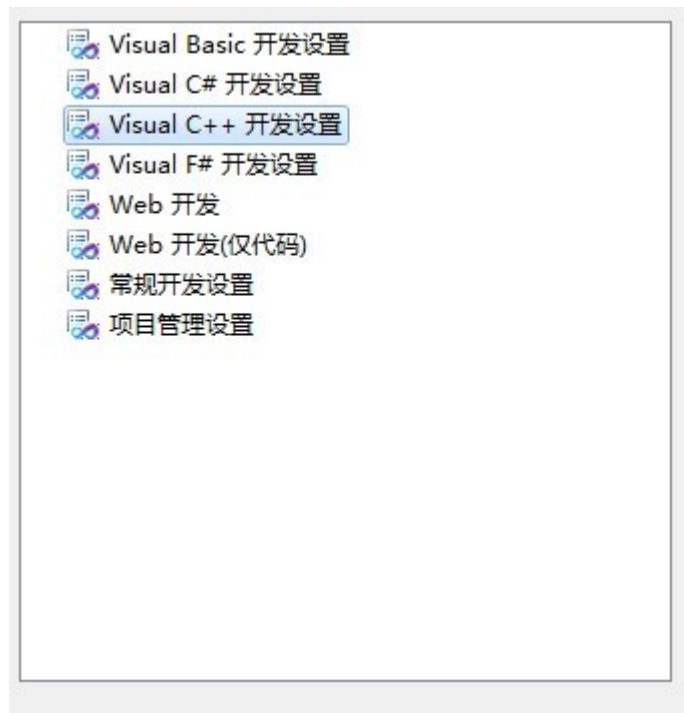
配置

当安装成功之后，我们就可以开始使用了，这里我先教大家做一些常见的配置，当然大家也可以直接用默认的设置，我这样配置主要是为了方便。

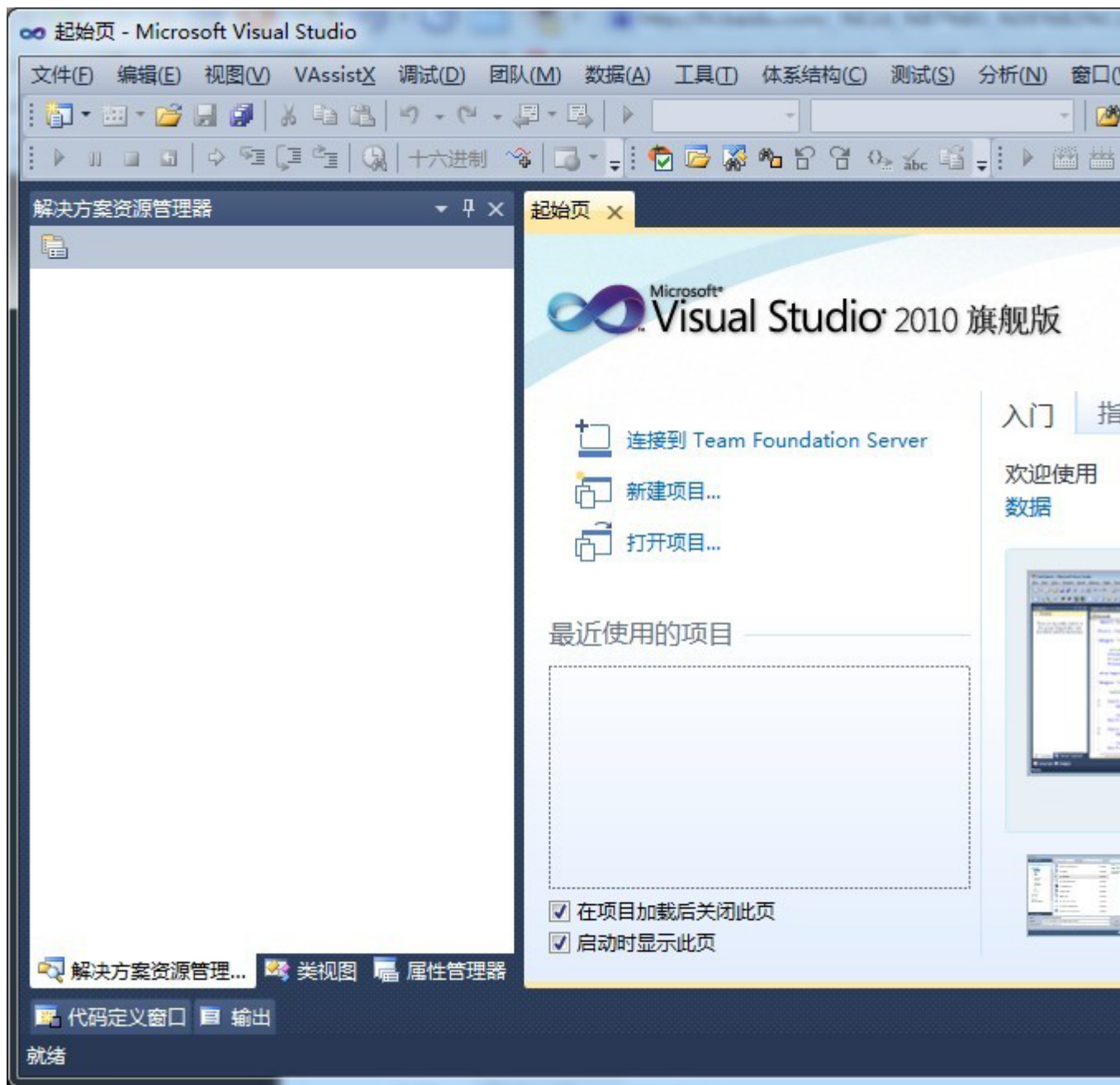
下面通过开始菜单来启动 VS2010 吧：



如果你是第一次开始，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：

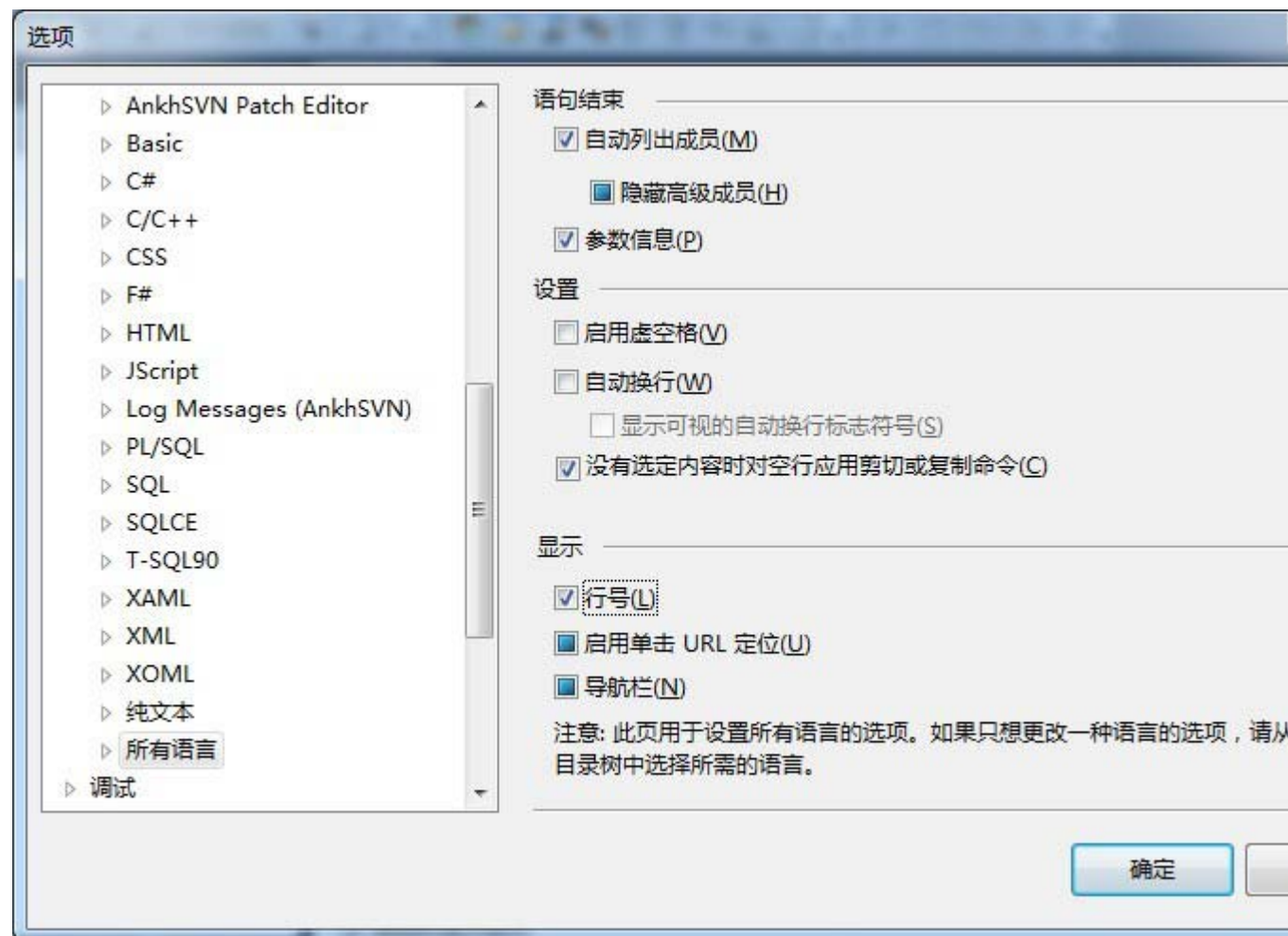


出现下面的画面表示已经成功安装和运行了。这是起始页面，以后你会经常见到它。

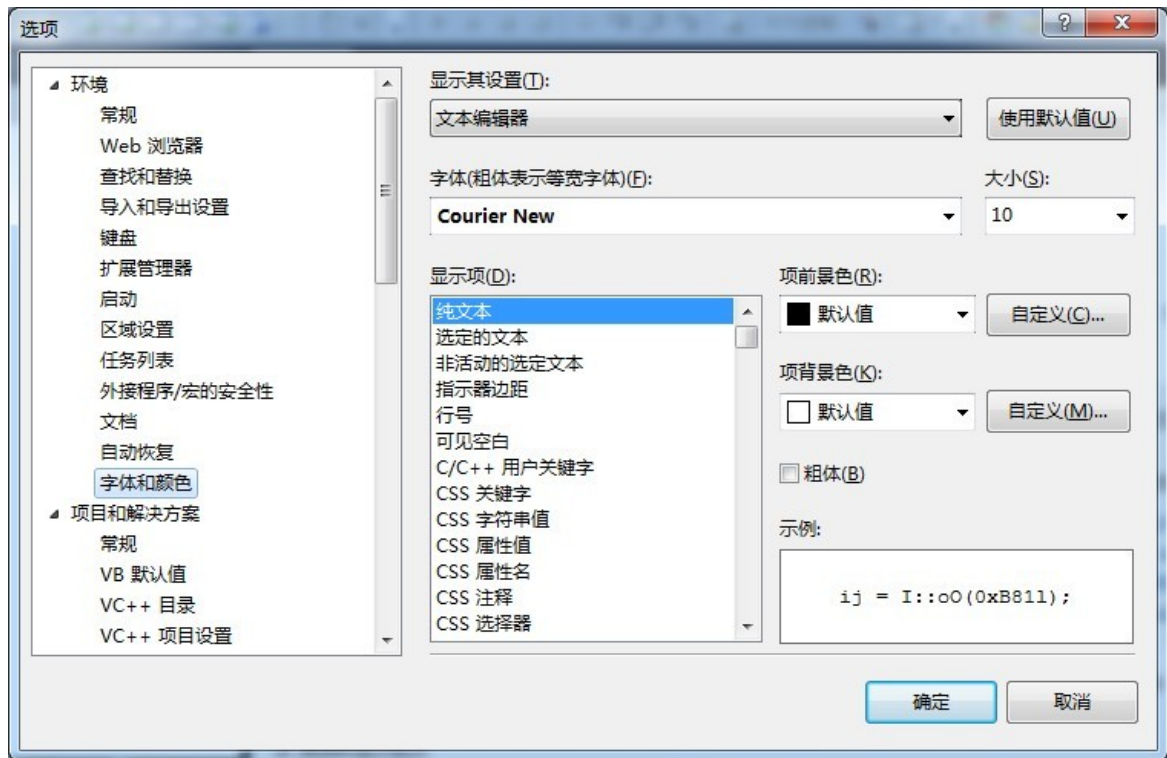


注意，你可能会没有 VAssistX 这个菜单，因为这个是个外部工具，以后给大家解释它的用处，现在暂时忽略。接下来让我们来做一些常见的设置。通过菜单工具》选项调出配置对话框，下面是一些常见的设置：

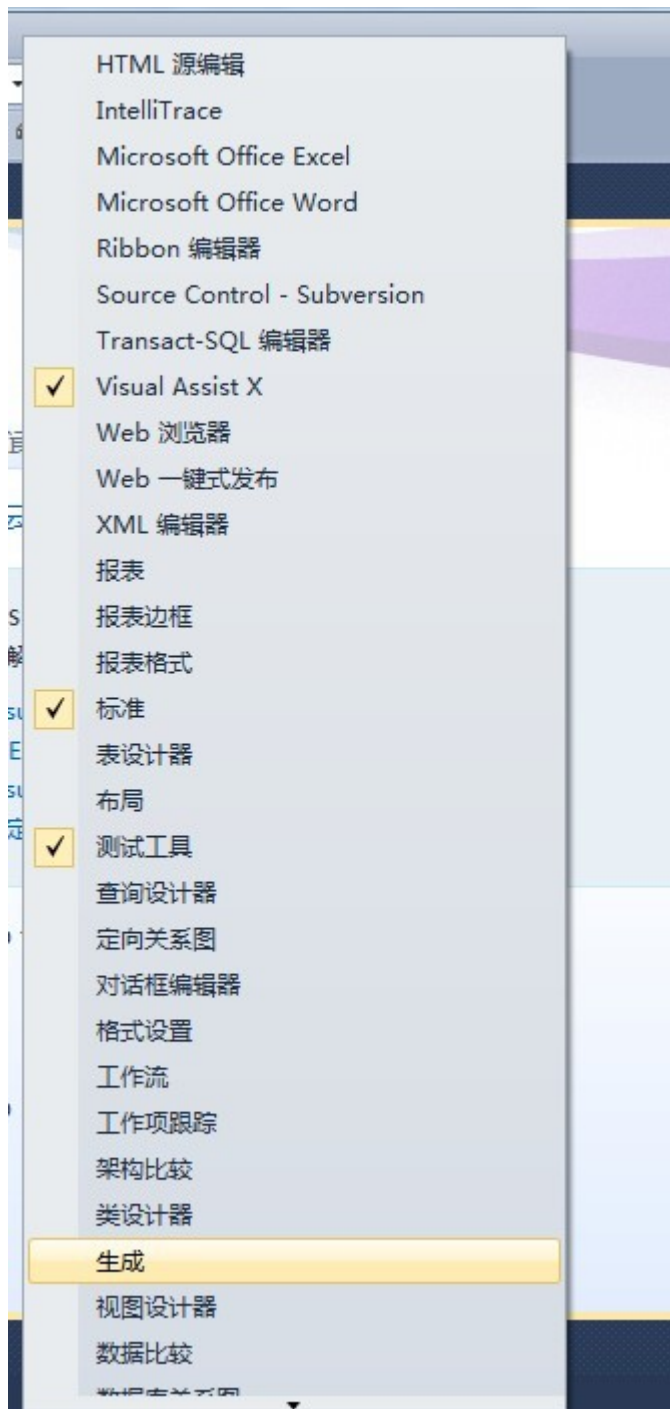
调处行号，选择文本编辑器，所有语言，把行号打成勾。



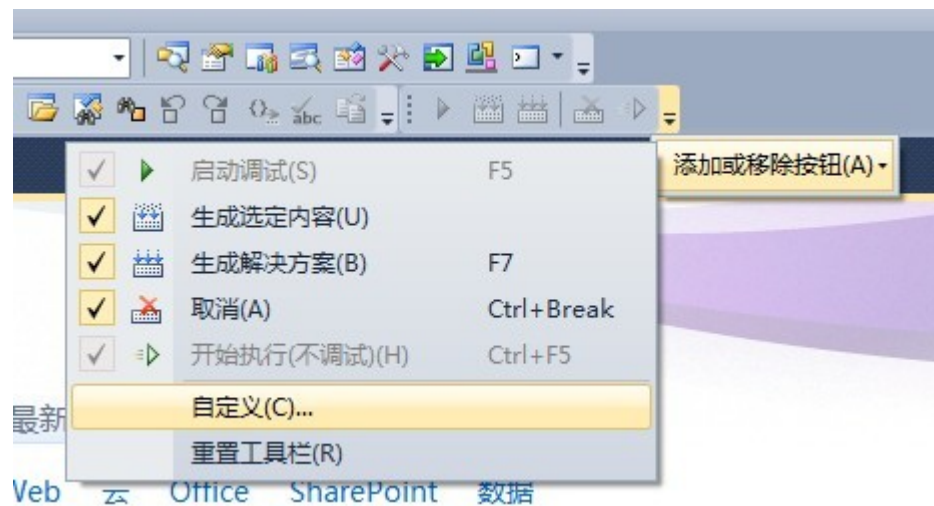
如果你想调整字体或者颜色，那么可以在这里选择：



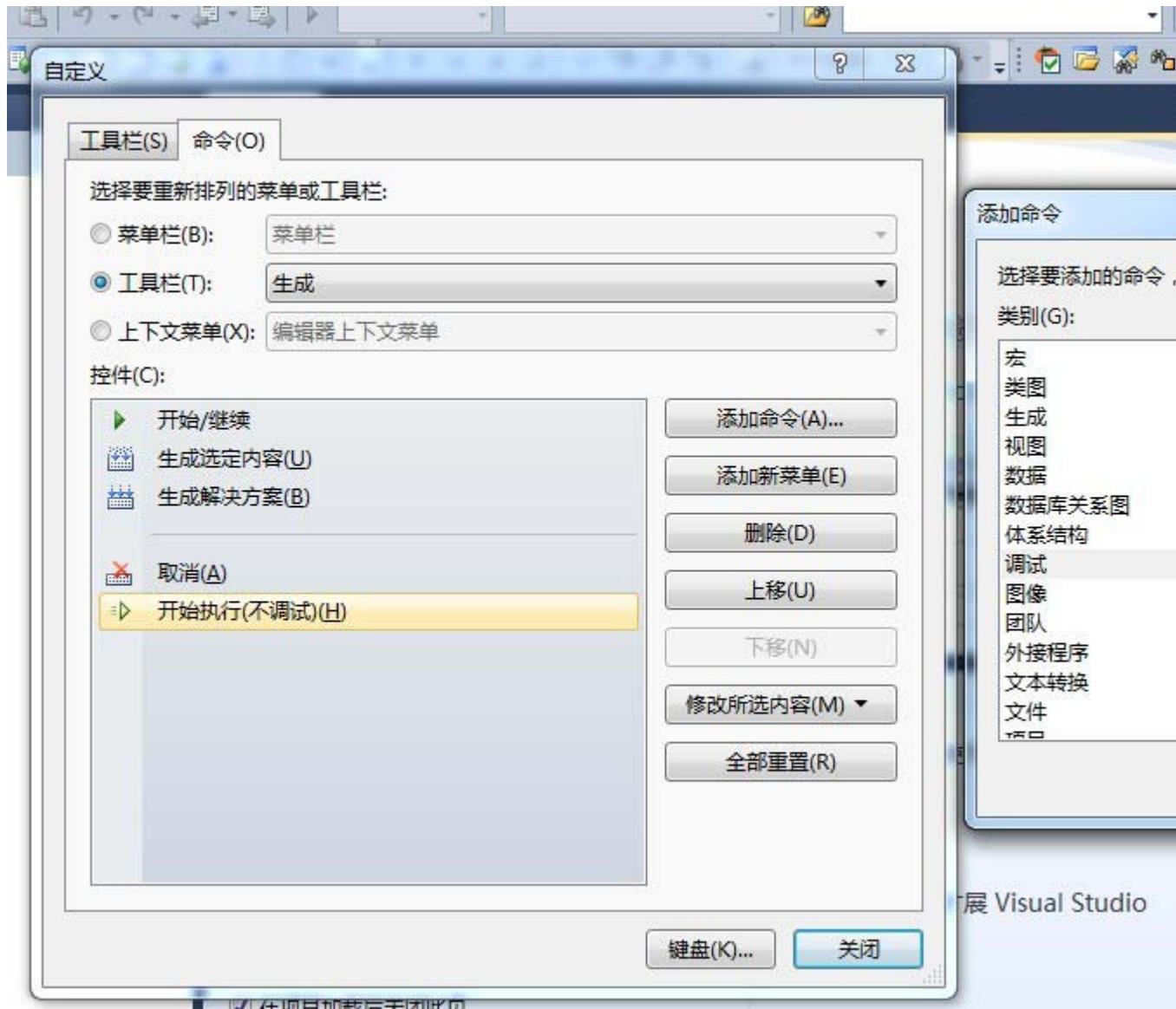
单击工具栏的空白区域，让我们把生成工具栏调出来：



这样我们就可以直接在工具栏上面选择编译项目、编译整个解决方案、运行程序和调试程序了。选择生成工具栏旁边的小三角形可以自定，我们还可以添加一些按钮：



单击自定义界面下面的添加按钮就可以添加新的按钮了，现在我们把开始执行（不调试）添加工具栏中：



以前经常有人问我为什么我的程序闪一下就没了，这就是因为它们把调试（F5，那个实心的三角形）当成了运行（不调试，Ctrl+F5，空心带尾巴的三角形）了。当然是一闪而过啦。关于调试以后说到。

HelloWorld.

下面让我们用 VC++ 2010 也就是 VC10 来做一个控制台的 HelloWorld 程序吧。

VC2010 里面不能单独编译一个.cpp 或者一个.c 文件，这些文件必须依赖于某一个项目，因此我们必须创建一个项目。有很多种方法都可以创建项目，可以通过菜单：文件，新建，项目；也可以通过工具栏点击新建项目进行创建。这里我们点击起始页面上的新建项目：



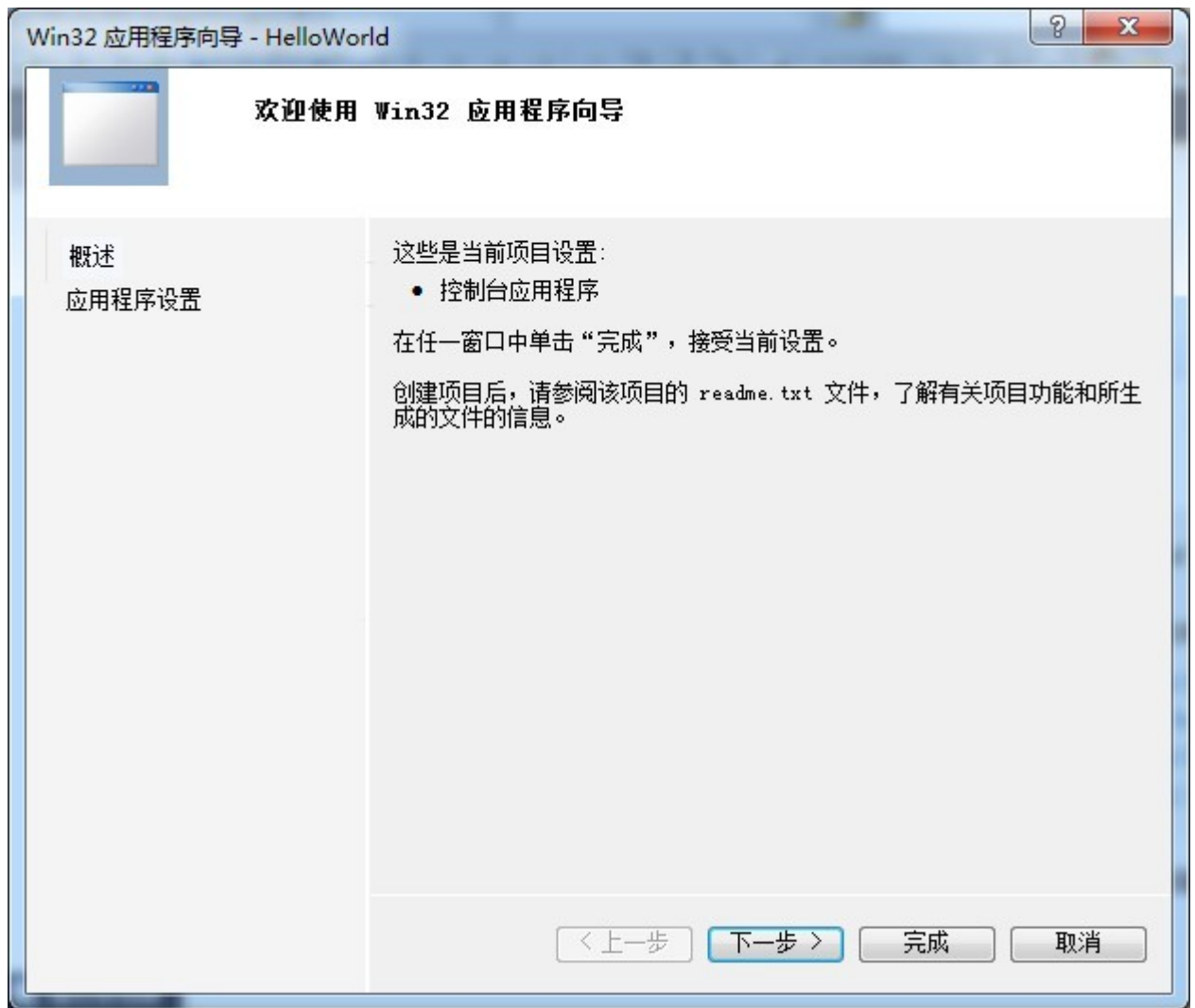
点击之后进入新建项目向导：



上面选择 Win32 控制台应用程序，名称中输入 HelloWorld 点确定，至于是否为解决方案创建目录我们暂时不管，那主要区别在于解决方案是否和项目文件在同一目录。



接下来进入创建页面，在 Win32 应用程序向导的第一个页面直接点下一步即可：



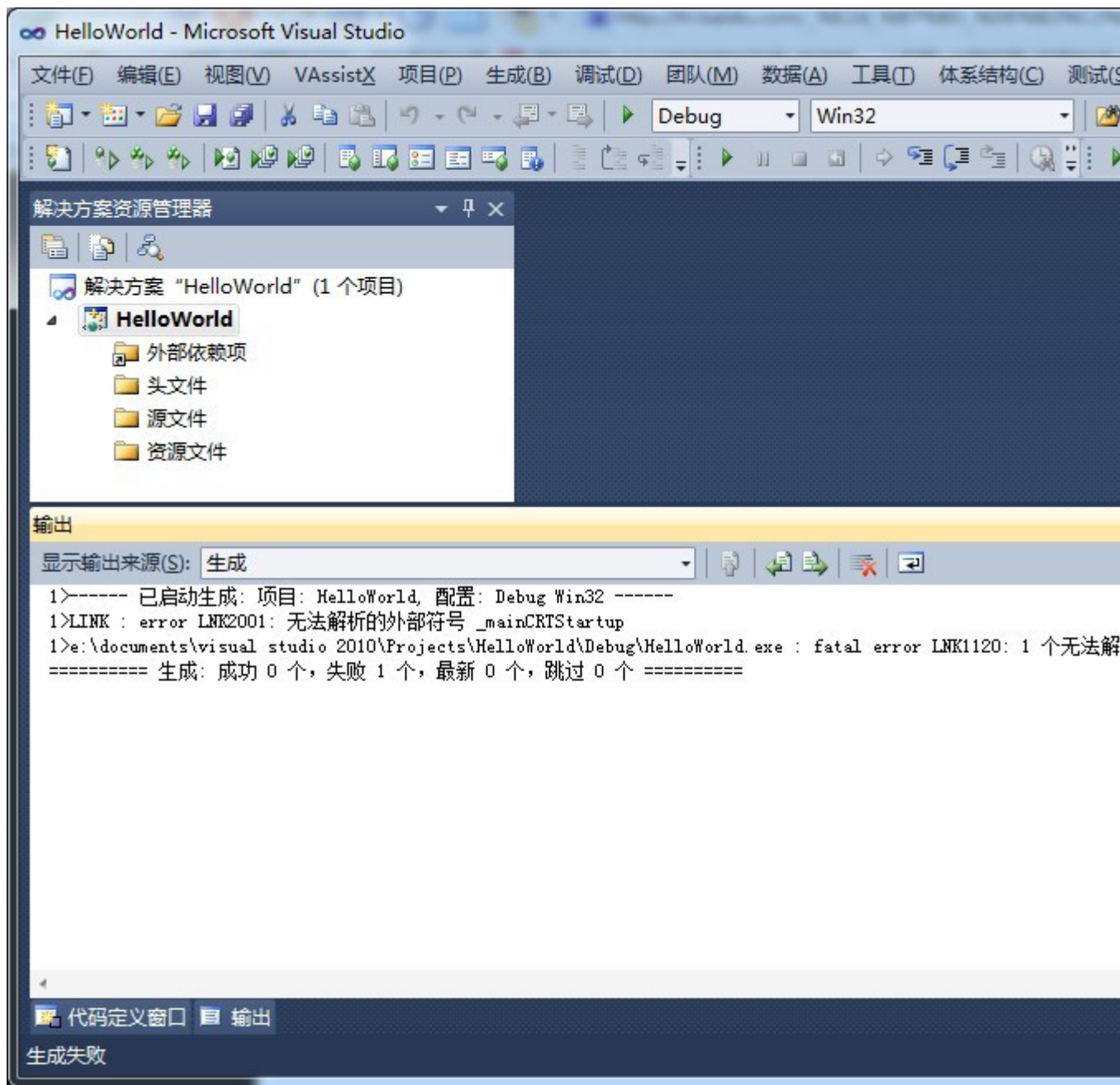
下个页面记得选择空项目，我们不需要预编译头：



点击完成。

这时候一个空的项目编译成功了，我们不妨编译一些试试。点击刚才添加的生成工具栏的生成按钮：



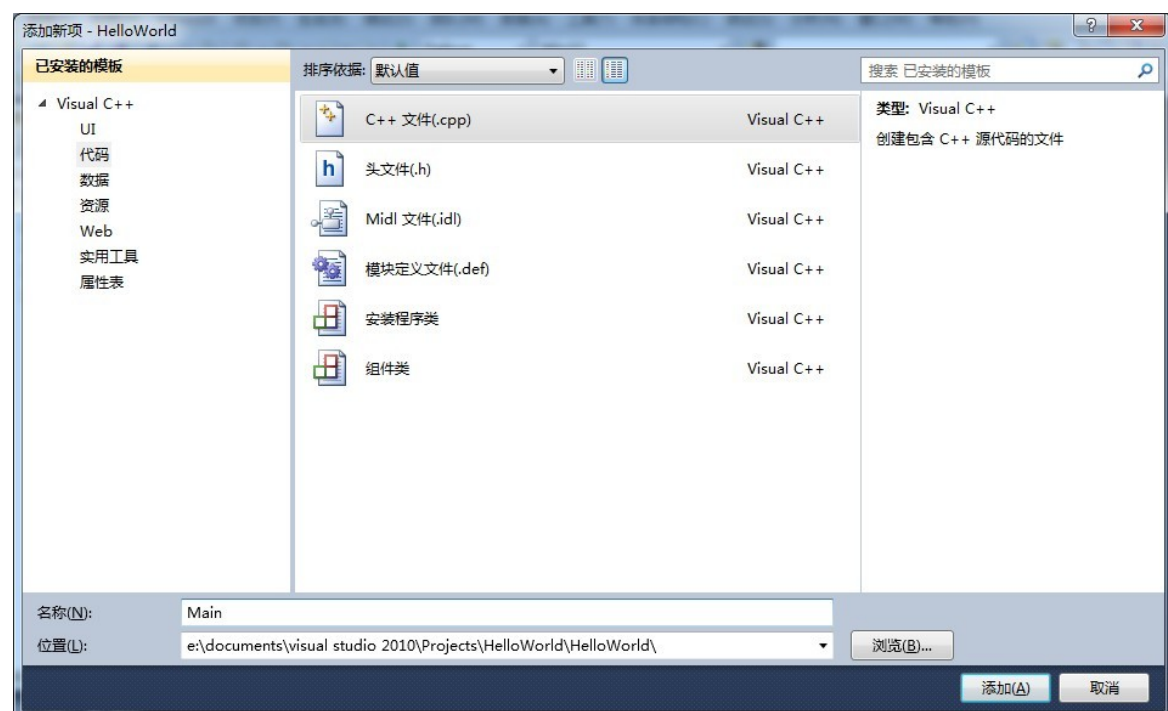


这时候我们会遇到编译错误，为什么呢？因为我们还没有 **Main** 函数，对于一个 **C++** 项目来说，一定要有一个且仅有一个 **main** 函数(Windows 程序需要 **WinMain**，区别以后再讲)，可以是隐式提供也可以是显式提供，至于区别以后会讲到。现在让我们记住这个错误，以后遇到这个错误的时候一定要想想，我们项目中是否有 **Main** 函数。注意这时候即使你托一个有 **Main** 函数的文件到 **VC10** 中进行编译也是没有意义的，因为那个文件并不是我们项目的一部分。

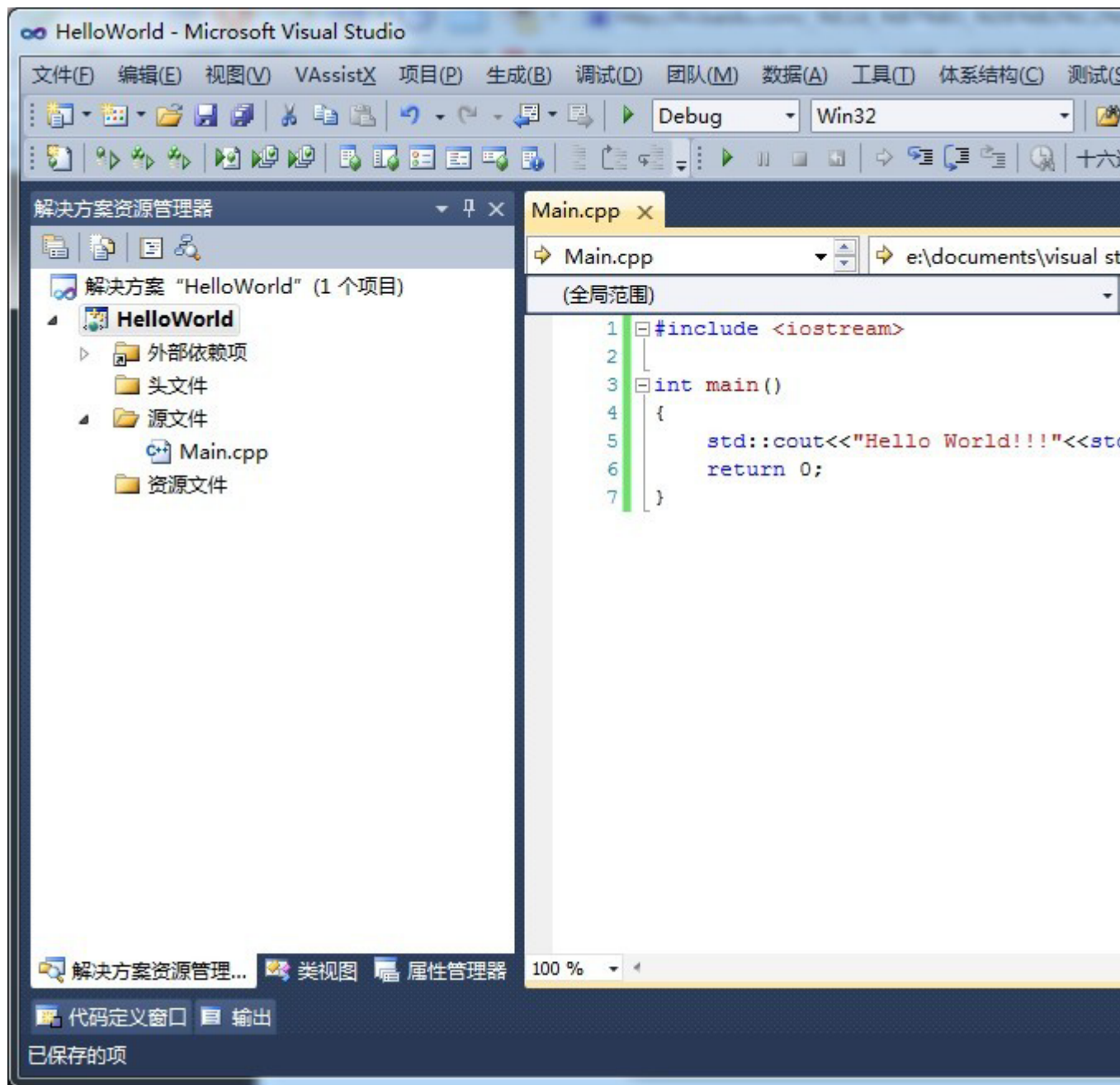
现在我们可以添加一个代码文件进来了，这个代码文件可以是已经存在的也可以是新建的，这里新建一个。右键单击项目名称，选择添加，新建项：



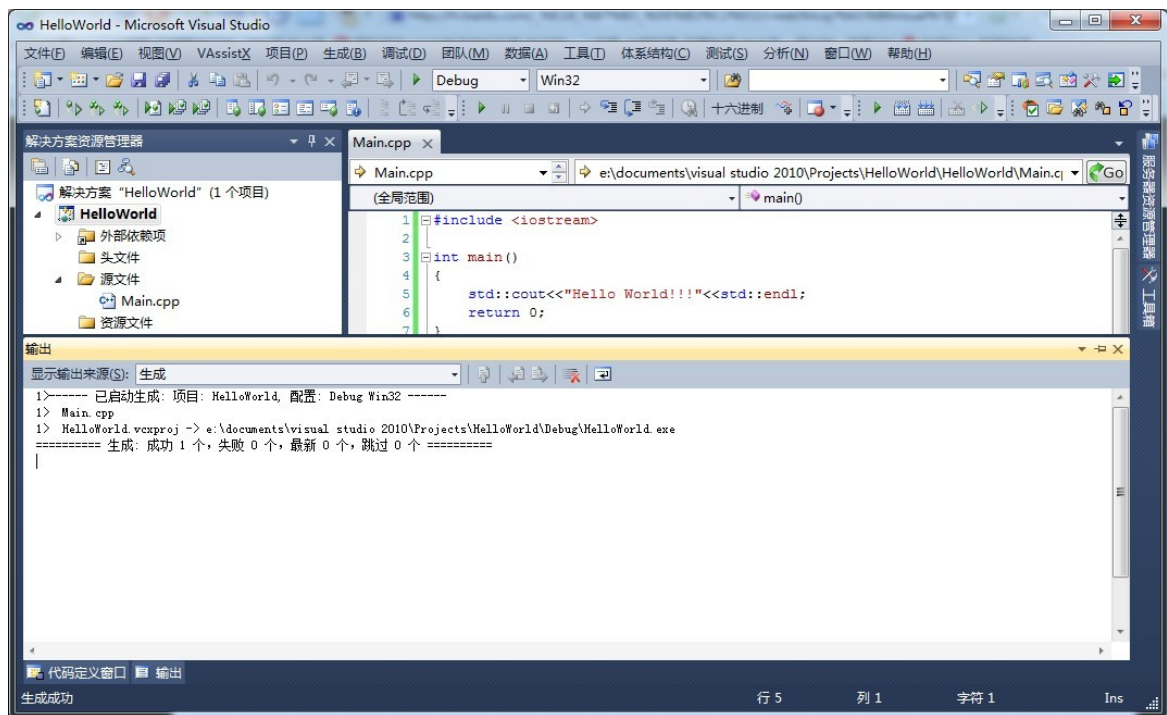
在向导中选择代码、C++文件(.cpp)，名称输入 Main，确定。



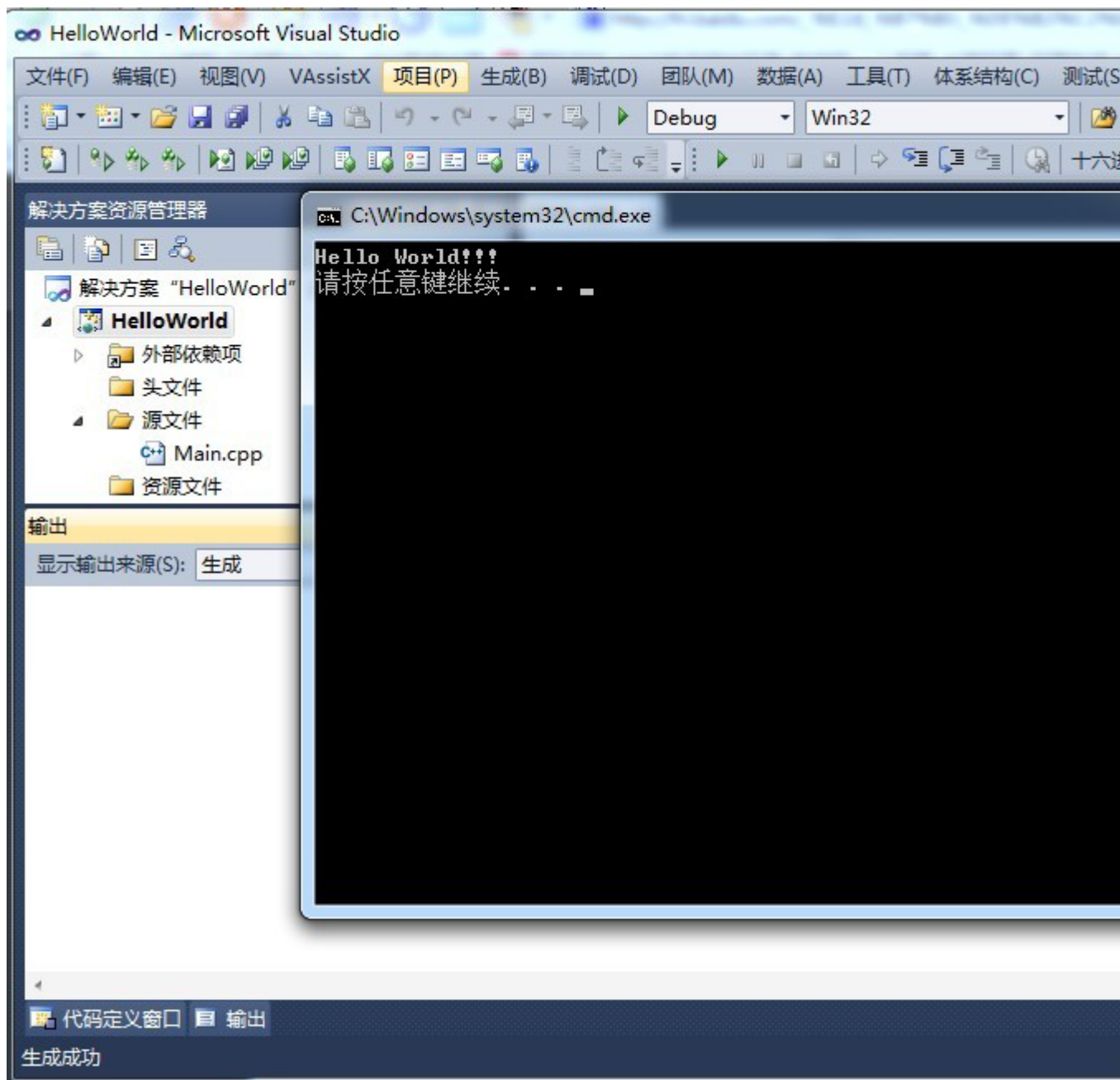
这时候已经成功添加了一个 Main 文件，注意添加新文件的时候要防止重名：



然后我们输入最简单的几行代码，然后编译它吧，编译方法和上面一样。



如果编译成功你会看到上面的画面，如果失败会有错误提示，那么你可以根据提示去修改项目配置或者代码。然后让我们用 **Ctrl+F5** 或者点那个空心三角形运行一下吧：



OK 了，接下来你可以通过更改这个程序去练习 C++教材上面的程序了。

这一章的内容就是这么多了，下一章中我们将讨论以下内容：什么是 C++？什么是编程等等。

第三章关于 C++的一些问题

这一回我自己都不知道应该写点什么好，或许今天的篇幅会比往常短很多。我说过，这不是 C++的教程，因为我还没有那个能力来教大家 C++，我能做的，是分享一些学习的经验，以及教新手如何使用 VC2010 这个工具去练习和实践其它 C++教材上面的程序，如《C++Primer》等。

今天说的是我学习和使用 C++这些年来对 C++的一些理解，这样的日志其实是最难写的，因为众口难调，为什么这么说呢？同样的一段音乐不同的人去倾听都会有不同的感受，不同的心境去听也会有不同的感觉。记得当年我在听雅尼的《If I could tell you》听到的是无尽的伤感和无奈，而我的朋友却怎么也听不出来，呵，你去听听之后会有什么感觉？

好，回归正题吧。

什么是编程

下面这段话我是从[百度百科“编程”](#)词条复制过来的，不敢掠人之美：

编程就是让计算机为解决某个问题而使用某种程序设计语言编写程序代码，并最终得到结果的过程。为了使计算机能够理解人的意图，人类就必须将需解决的问题的思路、方法、和手段通过计算机能够理解的形式告诉计算机，使得计算机能够根据人的指令一步一步去工作，完成某种特定的任务。这种人和计算机之间交流的过程就是编程。

这段话说的比较明白，我们如果想让计算机按照我们想要的方式工作，那么我们必须给它命令，有的常见的任务已经由操作系统帮助我们提供了，操作系统将一些简单的指令封装成一个简单的操作，使得我们的操作更容易更简单，然而操作系统提供的操作始终是有限的，如果要扩展计算机的软件系统，就需要编程了。

什么是编程语言

如上所说，编程语言就是人类与计算机交流的桥梁。首先编程语言必须是计算机可以理解的或者是可以间接转换成计算机可以理解的指令的东西。打个比方，如果你在一台普通电脑面前站着，大叫道：“电脑，给哥把 C 盘根目录下面的 A 文件拷贝到 D 盘根目录下面去”。我猜想你就算叫上一天也没用，因为普通的电脑无法识别你的指令，这就算是对“机”弹琴吧。

什么是编译器

如上所说，编程语言是计算机可以直接理解的如机器码或者可以间接转换成计算机可以理解的指令的东西。那么 C++就属于后者，那么是什么把 C++翻译成计算机可以识别的指令的呢？这就是编译器。

什么是 C++

我不想说 C++的历史了，我想说，它是一门编程语言，它可以通过编译器的翻译成为

计算机可以识别的指令。比如我们常见的 C++ 代码：

```
int a = 3 + 1;

cout<< a << endl;
```

这翻译成机器码或者计算机可以理解的指令大概可以是这样的：

现在请为我准备一个变量 **a**，请用 **3+1** 的和去初始化它，然后请加它的值输出到控制台上并追加上一个空格。

关于 C++ 的更多东西，建议大家去看专门的 C++ 教材或者维基百科、百度百科等相关词条。

学 C++ 学的是什么？

学 C++ 学的是什么？可以说 C++ 是很复杂的，因为要实现一些在其它语言中很简单的任务，在 C++ 中通常都需要更多的代码和时间，以创建一个窗口为例，其它语言由于有内置的支持，甚至当我们创建一个 **HelloWorld** 程序时候就已经创建了一个窗口。而在纯粹的 C++ 环境中，这是做不到的，因为 C++ 并没有提供 GUI 的内置支持，如果要用 C++ 创建窗口，我们需要使用到系统 API 或者使用封装了这些 API 的一些库，前者如 Win32 程序设计后者如 MFC、wxWidgets。

但是这些并不能代表 C++ 不好，反而 C++ 是很强大的，我们学习 C++ 学得不仅仅是 C++ 的语法，因为仅仅懂得 C++ 的一些语法又能怎么样呢？演示算法么？其实我认为学习 C++ 的初期当然是熟悉 C++ 的语法，中期当然是学习 STL 等库、系统 API 以及熟悉 C++ 的面向对象的思想 and 一般的设计方法；后期是伴随着你 C++ 生涯的一生的，因为它没有终点的，这时候除了要继续加深对 C++ 的理解之外，还需要去学习各种各样的基于 C++ 的库，因为你需要在这些库的帮助下使用 C++ 去实现更具体的东西，比如一些 GUI 库如 QT、wxWidgets，一些游戏引擎如 Ogre、Unreal、Bigworld 等，一些网络库如 ASIO、RakNet、ACE 等。这些库大多有一个特点就是他们是基于 C++，封装了底层的 API 使得我们可以不必每次多去使用繁杂的 API 来实现我们想要的功能。当然，你也可以用 C++ 去封装这些 API，让自己成为一个库作者，方便他人。

学习 C++，学习 C 和 C++ 的库，学习其它各领域的 C++ 库，好好的使用这些库或者自己写一些库，其它一些你能想到的与 C++ 相关的事情。

怎样才能用 C++ 来做事情

通常我认为如果只使用最基本的 C++ 以及其内置库来做事情，那么能做的事情始终是有局限的，因为 C++ 的库毕竟是有限的，我们能用它来写一些核心代码，因为这部分代码通常都不需要与具体的应用关联起来。使用 C++ 以及 C++ 继承的 C 库我们能做一些简单的文件操作，因此还可以写一些与此相关的处理程序。其实不仅仅是 C++，比如 C#，加入你在应用中不引入 .net Framework 下面提供的大量的其它的类（指除了最基本的如 System.Console 等之外的类），你又能用 C# 做什么呢？我们在做 C# 的时候通常要引入一

些新的类，引用一些新的命名空间，其实 C++ 也是这样的，要做更具体的事情，就需要除 C++ 本身之外的更多东西才行。

因此，要用 C++ 来做事情，除了 C++ 的基本语法之外，还需要学习一些其它一些内容如 Windows API、MFC、DirectX 3D API 等。

C++、C#、Java 到底谁好？

对于这个问题，我想说的是：语言无贵贱，技术有高低。其实无论什么语言，它们在这里都是编程语言，也许它们看起来不一样，它们用起来也不一样，它们的“翻译”编译器不一样，但是它们的本质作用就是要用它们自己的方式将我们人类的解决问题的方法、思路和流程告知给计算机。这也就是为什么 C++、C#、Java 可以相互交互的原因。

每个语言就像江湖上的各门派的武功一样，其实真的没有高下之分，只有个人的修为，少林武功就一定强于武当么？《碧血剑》中袁承志不也用普普通通的五行拳击溃了荣彩么？只是不同的武功在不同的条件下表现不同，他们各自占据了自己有利的地位罢了，以杨家枪法为例，或许二人比武它不厉害，但是征战沙场估计是再适合不过了。

因此不要再纠结什么语言好，什么语言流行了，选择一个自己喜欢的方向，然后调查好那个方向最常用最适合的语言，然后集中注意力坚持的去学习它，这就好了。

今天的内容就是这些，希望对大家有帮助。请大家务必好好学习 C++ 教材，并使用 VC2010 好好演练，最好能够举一反三自己创造问题自己解决！多看看百度知道 C/C++ 下面的那些问题，每解决一个那样的问题，你自己也会进步。

第四章 VC2010 中初学者常见错误、警告和问题

这一章将帮助大家解释一些常见的错误、警告和问题，帮助大家去理解和解决一些常见问题，并了解它的根本原因。

iostream.h 与 <iostream>

下面的代码为什么在 VC2010 下面编译不过去？

```
#include <iostream.h>

int main()
{
    cout<<"Hello World."<<endl;

    return 0;
}
```

错误信息: **fatal error C1083: 无法打开包括文件:"iostream.h": No such file or directory**

造成这个错误的原因在于历史原因,在过去 **C++98** 标准尚未订立的时候, **C++** 的标准输入输出流确实是定义在这个文件里面的,这是 **C** 风格的定义方法,随着 **C++98** 标准的确定, **iostream.h** 已经被取消,至少在 **VC2010** 下面是这样的,取而代之的是我们要用 **<iostream>** 头文件来代替,你甚至可以认为 **<iostream>** 是这样定义的:

```
namespace std

{

    #include "iostream.h"

}
```

因此我们可以简单的修改我们的 **Hello World**。

```
#include <iostream>
using namespace std;

int main()
{
    cout<<"Hello World."<<endl;

    return 0;
}
```

iostream.h 是属于 **C++** 的头文件,而非 **C** 的,因此标准订立的时候被改成了 **<iostream>**。而 **C** 的头文件 **stdio.h** 等依然可以继续使用,这是为了兼容 **C** 代码。但是它们依然有对应的 **C++** 版本,如 **<cstdio>** **<cstdlib>** 等。记住,在 **VC2010** 上面采用 **C++** 风格的头文件而不是 **C** 风格的头文件,除非你是在用 **C**。

warning C4996

这是一个警告,请看下面的代码:

```
#include <iostream>
using namespace std;

int main()
{
    char sz[128] = {0};
    strcpy( sz, "Hello World!" );
    cout<< sz << endl;

    return 0;
}
```

上面的 `strcpy` 会产生这个警告:

warning C4996: 'strcpy': This function or variable may be unsafe. Consider using `strcpy_s` instead. To disable deprecation, use `_CRT_SECURE_NO_WARNINGS`. See online help for details.

这是因为 VC 从 2005 版本开始, 微软引入了一系列的安全加强的函数来增强 CRT (C 运行时), 这里对应的是 `strcpy_s`。_s 意为 **safe** 的意思, 同样的道理, `strcat` 也是同样。因此要解决这个问题, 我们可以用 `strcpy_s` 来替换 `strcpy`, 但是注意 `strcpy_s` 并非所有编译器都提供, 因此如果要跨编译器, 请采用错误信息中所提示的方式, 定义 `_CRT_SECURE_NO_WARNINGS` 宏来掩耳盗铃吧。另外注意并非所有的加强函数都是在屁股后面加_s, 比如 `stricmp` 这个字符串比较函数的增强版名字是 `_stricmp`。下面, 用 `strcpy_s` 来更改程序:

```
int main()
{
    char sz[128] = {0};
    strcpy_s( sz, "Hello World!" );
    cout<< sz << endl;

    char* pSz2 = new char[128];

    strcpy_s( pSz2, 128, "hello");
    cout<< pSz2 << endl;
    delete pSz2;

    return 0;
}
```

注意, `strcpy_s` 有两个版本, 一个可以帮助我们自动推断缓冲区的大小, 而另外一个不能帮助我们推断, 因此在编译器不能推断缓冲区大小的时候, 我们需要自己指定缓冲区的大小, 如上面的程序所演示的那样, 关于增强版的函数请参考我写的 [《深入学习 C++ String2.1 版》](#)。

TCHAR、wchar_t、char

请大家看下面这个程序:

```
#include <iostream>
#include <Windows.h>
#include <tchar.h>
using namespace std;

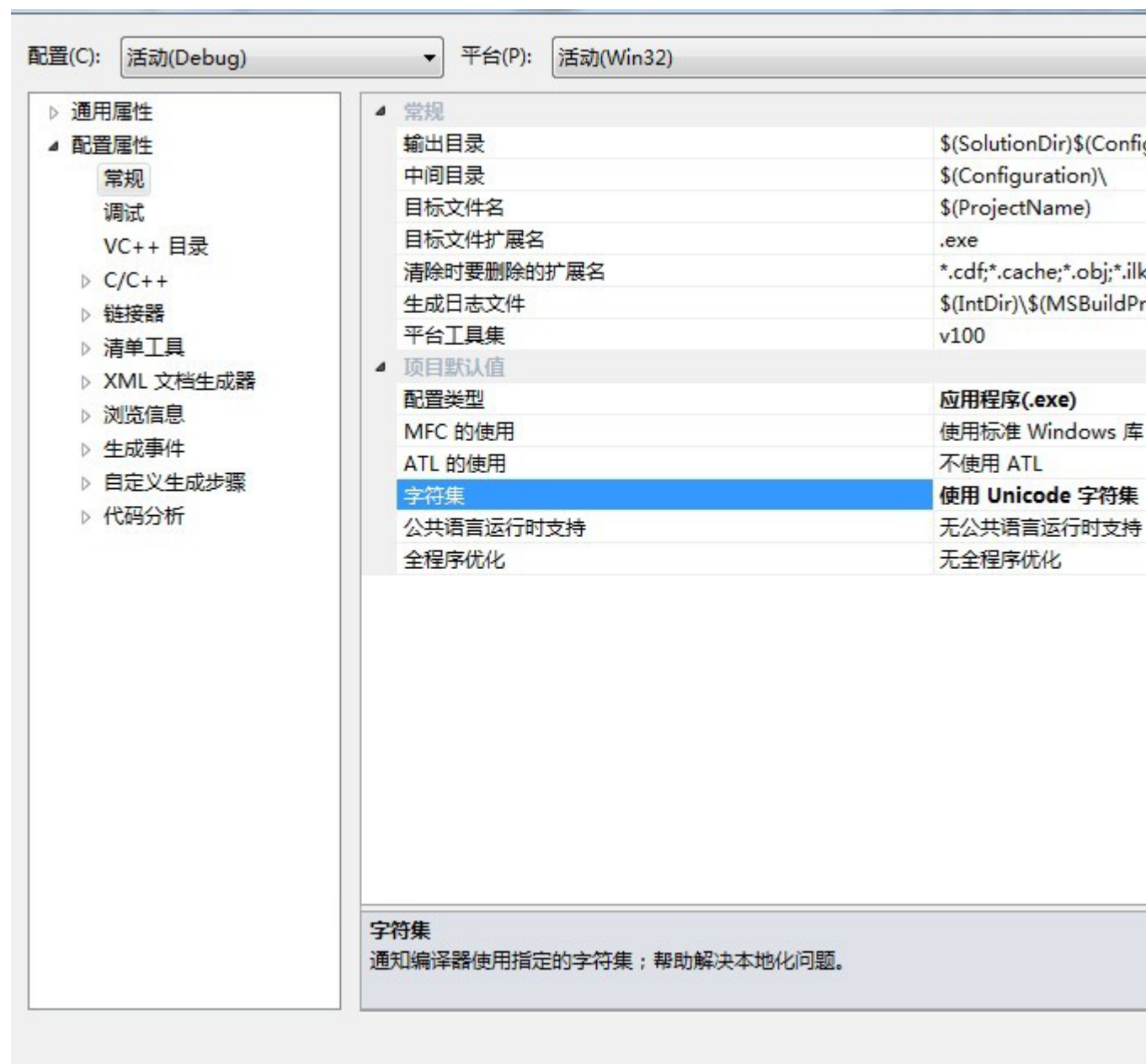
int main()
{
    MessageBox( NULL, "你好 HelloWorld! ", "Information", 0 );
}
```

```
return 0;
}
```

貌似没什么问题吧？错了，如果你是按照我教你的方法创建的控制台空工程的话，那么会有编译错误：

error C2664: “MessageBoxW”: 不能将参数 2 从“const char [17]”转换为“LPCWSTR”

这个问题太普遍了，几乎所有的初学者都会遇到而且感到难以应付，因为按照提示使用(LPCWSTR)强制转型貌似并不能帮助我们解决问题，而且这个程序在 VC6 下面应该是没有任何问题的，那问题出现在哪里呢？问题在这里，请右键单击解决方案浏览器下面的项目，属性，



问题的根本就是字符集问题，在 VC6 中，我们默认使用的是多字节字符集，而现在我们默认需要的是 UNICODE 字符集，简单的，我们把这个字符集改成多字节字符集这个问题就解决了：

项目默认值	
配置类型	应用程序(.exe)
MFC 的使用	使用标准 Windows 库
ATL 的使用	不使用 ATL
字符集	使用多字节字符集
公共语言运行时支持	无公共语言运行时支持
全程序优化	无全程序优化

再试试应该就可以了吧？但是我并不推荐大家这么做，因为让自己的程序适应各种字符集是我们写代码的人义不容辞的义务。

我们把程序改成下面这样：

```
#include <iostream>
#include <Windows.h>
#include <tchar.h>
using namespace std;

int main()
{
    MessageBox( NULL, TEXT("你好 HelloWorld! "), TEXT("Information"), 0 );
    MessageBox( NULL, _T("你好 HelloWorld! "), _T("Information"), 0 );

    return 0;
}
```

用两个宏 TEXT 或者 _T 都可以解决这个问题，它们两个并没有太大区别，也许区别在于前者是通过 windows.h 头文件引入的，而 _T 是通过 tchar.h 引入的，我推荐大家使用 _T 和 tchar.h，因为 tchar.h 还帮助我们引入了其它一些很有用的宏，比如 _tcscpy_s，这个宏在使用 UNICODE 字符集的时候被替换成 wcscpy_s，在使用多字节字符集的使用被替换成 strcpy_s。关于这部分的内容，请大家不要错过《Windows 核心编程》的第二章(第四版或第五版都可以)，以及《深入学习 C++ String2.1 版》。它们都有提到。

有人听说 _T 可以把多字节字符串转换成 UNICODE，因此他写了如下的代码：

```
const char* pStr = "haha 哈哈";
MessageBox( NULL, _T(pStr), _T("Information"), 0 );
```

当然，除非你运气好的抓狂，否则你是编译不过去的，为什么呢？我们现在应该知道对于 "Hello" 这样的字符串，VC2010 会默认的将它视为 const char*，即多字节字符串，而

L"Hello"前面有个 L 前缀的被视为 UNICODE 字符串，这和 C#是有区别的，因为 C#的字符串总是被视为 UNICODE，C++/CLI 下面编译器也会帮助我们做到这件事情，所以它们不需要 L(C++/CLI 兼容 L 这种写法)。

让我们看看_T 的定义吧：

```
#define wxCONCAT_HELPER(text, line) text ## line

/* could already be defined by tchar.h (it's quasi standard) */
#ifndef _T
    #if !wxUSE_UNICODE
        #define _T(x) x
    #else /* Unicode */
        /* use wxCONCAT_HELPER so that x could be expanded if it's a macro */
        #define _T(x) wxCONCAT_HELPER(L, x)
    #endif /* ASCII/Unicode */
#endif /* !defined(_T) */
```

_T 在 UNICODE 下面最终会被替换成 L ## x。##是一个编译预处理指令，意味着让 L 和 x 贴在一起，比如 L ## "Hello"最终就是 L"Hello"，因此它可以把"Hello"转换成 UNICODE 字符串。那为什么上面的程序不行呢？让我们看看_T("pStr")会被替换成什么：L ## pStr -> LpStr，哦，LpStr 是一个新的标识符，如果你没有定义过它，你当然不能通过编译啦。

因此我们可以了解到_T 这样的宏只能处理直接的常量字符串，不能处理其它的情况。而我们上面演示的那种情况需要我们动态的去转换编码，Windows 有 API 可以帮助我们做到，C 库也有函数可以帮助我们。恰好我曾经写过这样的代码，欢迎大家参考：

ASCII/UNICODE/UTF8 字符串互相转换的 C++代码

对于_T 宏，再说一点东西，或许你会感到奇怪为什么_T 不直接定义成#define _T(x) L ## x，而要绕个圈子去调用 wxCONCAT_HELPER 呢？这实际上涉及到宏展开顺序和截断的问题。在这里，我们需要说一个宏参数的概念，这很函数的参数是类似的，这里_T(x)的 x 就是宏参数，好，记住下面一句话：

如果你定义的宏中使用了#或者是##的话，宏参数将不会被展开，也就是说_T(x)如果直接定义成 L##x 那么在下面这种情况就会出错(PS: #是给参数加引号的意思)：

_T(__FUNCTION__)，__FUNCTION__是一个预定义的宏，它代表了当前函数的名字，这个展开会是什么呢？L__FUNCTION__。为什么间接调用 wxCONCAT_HELPER 就能得到正确的结果呢？因为当我们调用 wxCONCAT_HELPER 的时候，__FUNCTION__已经被 _T 展开成了函数名。

说多了说多了，如果你觉得复杂可以暂时跳过这些东西，我只是顺便说说。

重定义的编译错误和链接错误

让我们在项目里面再添加一个 **Test.h** 头文件，方法是右击解决方案中的项目，添加，新建项，**C++**头文件，名称输入 **test.h**。然后我们在 **test.h** 中输入：

```
/*#pragma once*/

void print()
{
}
```

回到 **main.cpp** 中：

```
#include <iostream>

using namespace std;

#include "Test.h"
#include "Test.h"

int main()
{

return 0;
}
```

编译一下我们会得到重定义的编译错误：

error C2084: 函数“void print(void)”已有主体

或许你会说，你引用(**#include**)了两次，我没你那么傻，我只引用一次不就好了么？是的。你聪明，但是是小聪明哈，因为你不能保证每个人都不去引用它。

这个问题演示的是**#pragma once** 的用处，让我们解开它的注释。编译成功！**#pragma once** 的作用就在于防止头文件被多次引用。你或许见过

```
#ifndef __TEST_H__

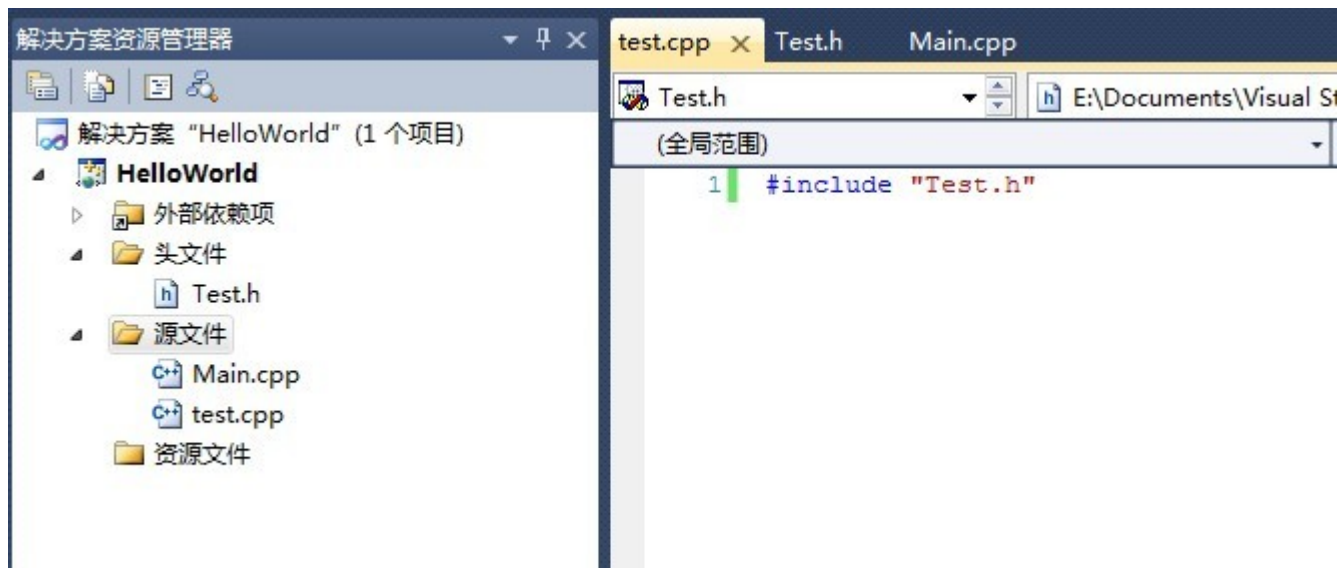
#define __TEST_H__

代码

#endif
```

这样的代码，它们的作用是一样的，如果你跟我一样懒，那么就用**#pragma once**，如果你打算去没有这个指令的编译器上编译代码，那么还是用后面一种方式吧。

现在让我们来见识一个对初学者稍微复杂一点的链接错误，用创建 **main.cpp** 的方法再添加一个 **test.h** 头文件，输入**#include "Test.h"**即可。



让我们再编译一次。

```
1>test.obj : error LNK2005: "void __cdecl print(void)" (?print@@YAXXZ) 已经在
Main.obj 中定义
1>e:\documents\visual studio 2010\Projects\HelloWorld\Debug\HelloWorld.exe : fatal
error LNK1169: 找到一个或多个多重定义的符号
```

如果说编译错误好找的话，链接错误对于初学者来说就有点麻烦了，聪明的初学者会去 Google、百度寻找答案，笨的初学者就会找所谓的高手、前辈问，而这些高手 Or 前辈未必有心情为你解释。要解决这个错误有无数种方法。

1.内联，把 print 声明为内联函数。

```
inline void print()
{
}
```

这个方法的好处是简单，坏处是局限性太强，意味着你总是需要公开 print 的实现，因为内联函数必须在编译时就知道实现才行。

2.static，把 print 声明为 static 函数：

```
static void print()。
```

这便告诉编译器，哥是唯一的，而且哥只能被本编译单元的代码调用，这和 extern 是对应的。简单来说，想要哥帮你做事，请先 include 哥声明的头文件，也就是#include "test.h"。

3.h 头文件中只放声明，实现放到.cpp 中去。

现在 test.h 中只有 void print();，而实现在 test.cpp 中：

```
#include "Test.h"
```

```
void print()
{
    int a = 1;

    cout<< a++ << endl;
}
```

这个时候有意思的是我们在 `main.cpp` 无需包含 `test.h` 头文件也可以引用 `print` 函数，因为 `print` 并非 `static` 的函数：

```
void print();
```

```
int main()
{
    print();
    print();

    return 0;
}
```

但是声明一下是必须的。

由于百度空间的帖子的篇幅是有限制的，因此今天只好就说这么几点了。新的内容请大家等候下一章。

合理组织项目、使用外部工具让工作更...

这一章跟大家分享一些与 `c++` 项目管理、`VAX`、`SVN`、`VS` 快捷键等方面的东西。

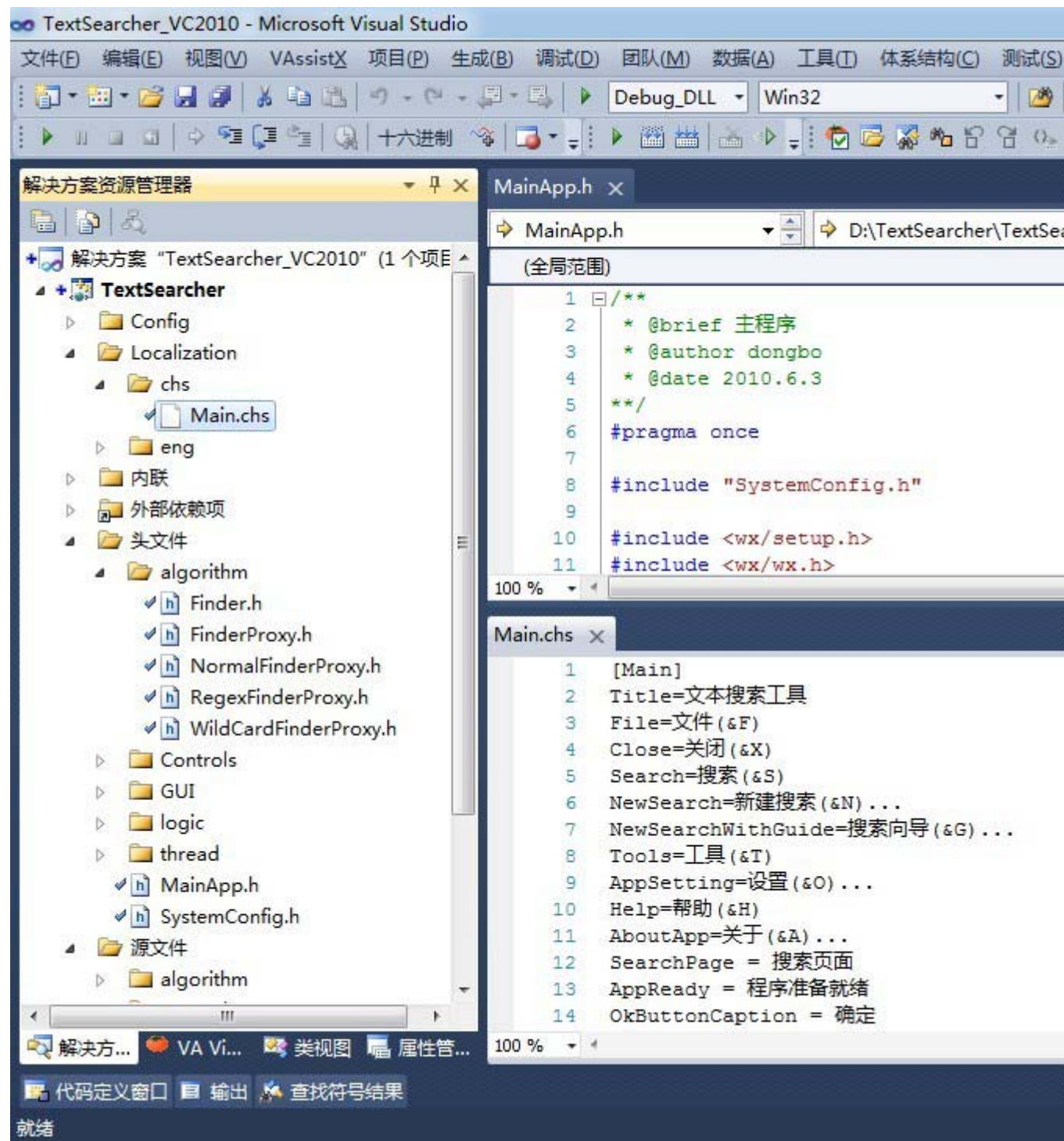
有效的在项目中组织 `C++` 文件，分配各种文件的目录对以后的维护会有好处的，至少不会出现不知道什么东西在什么地方，特别是大的项目，这里用 `TextSearcher` 来做例子。

使用 `SVN` 来管理项目会让我们的工作更轻松，工作也会更简单容易。

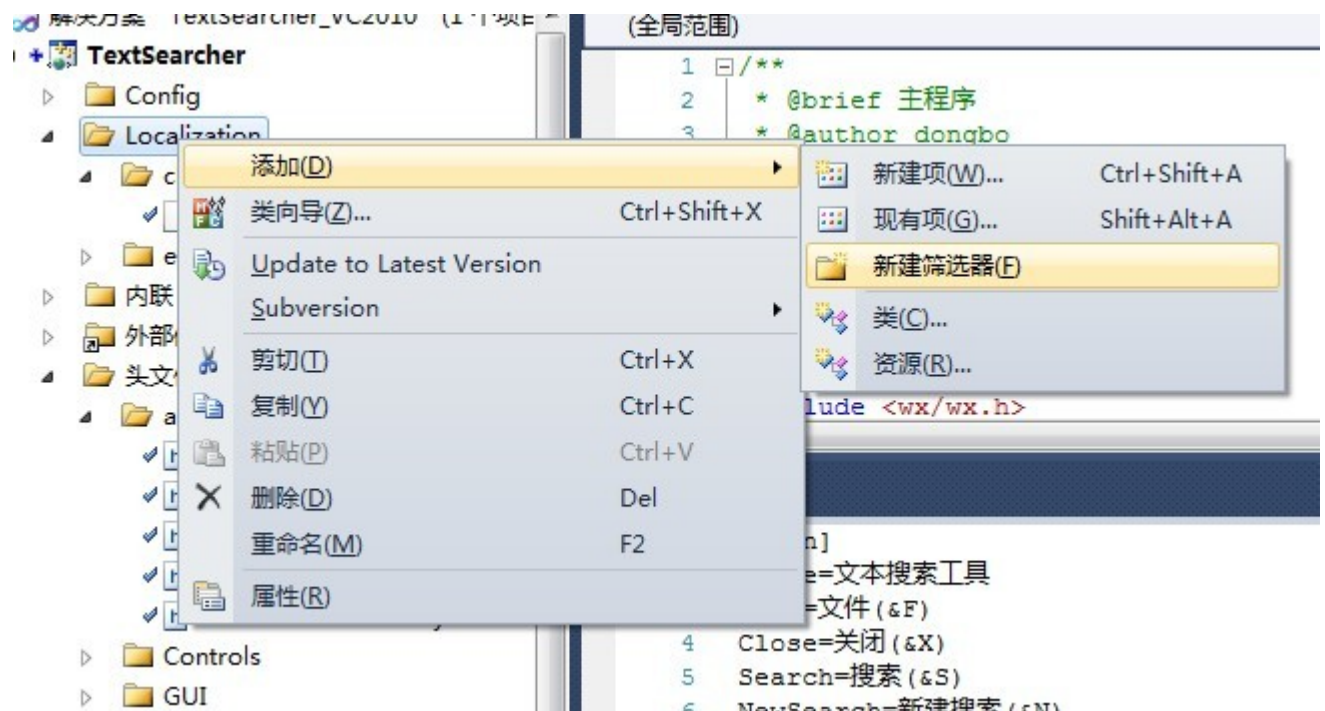
掌握常用的快捷键和常用的 `VS` 功能让我们的工作更有效。

合理的组织文件体系

首先说在 `IDE` 中为我们的文件分类组织，如下图所示：

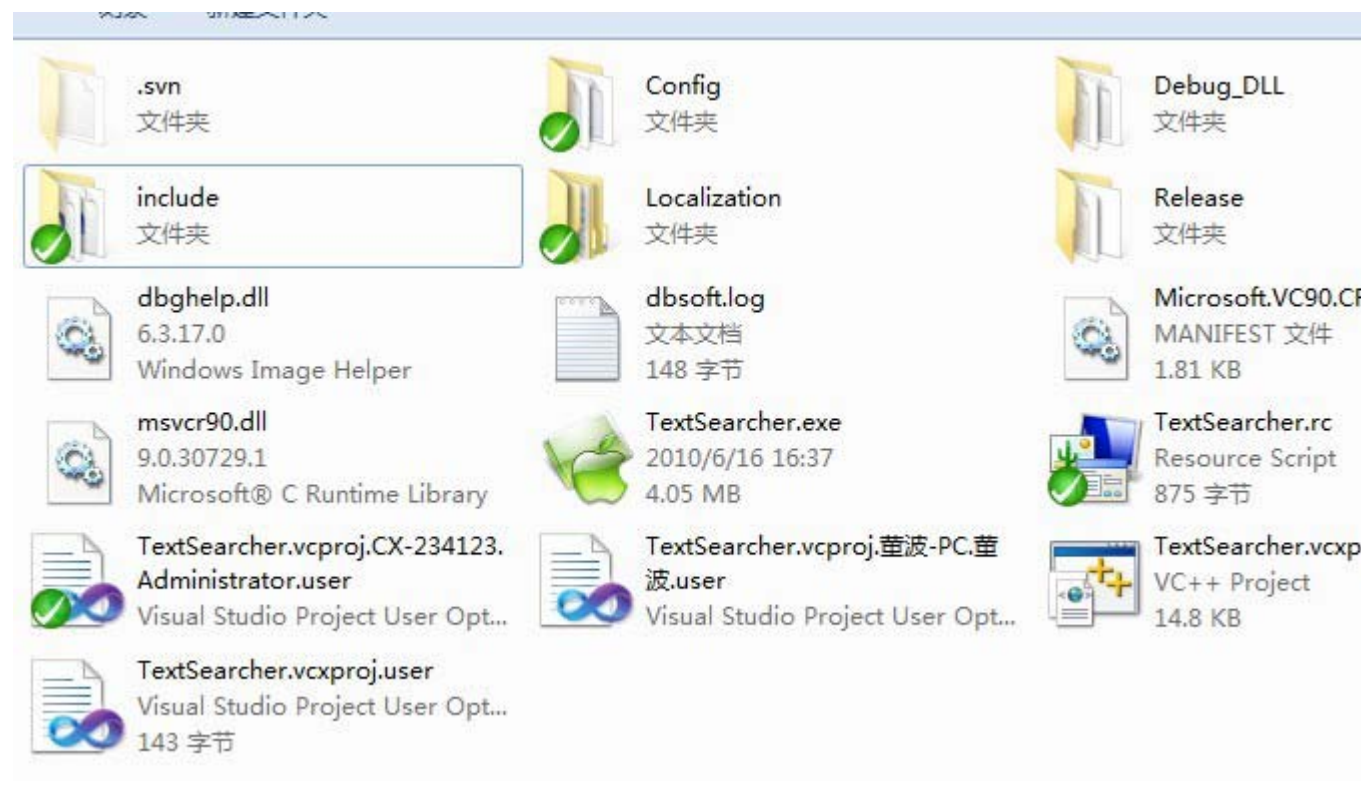


我把不同功能的代码和文件放在不同的 Filter 下面，如何添加这样的 Filter 呢？



这样就可以添加筛选器了，默认情况下 VS 为我们创建三个筛选器：头文件、源文件和资源文件，实际上我们可以再增加很多。这样区分开的好处就是各个功能的代码被分开了，在文件很多的情况下不会造成混乱。如 **TextSearcher**，它的搜索算法、软件控制逻辑、自定义控件、GUI 模块、线程化操作都是分开的，这样我可以很容易找到我想找的文件，而且还可以检视自己的模块划分是否合理等等。

接下来推荐大家在项目资源浏览器中为不同的文件划分目录。

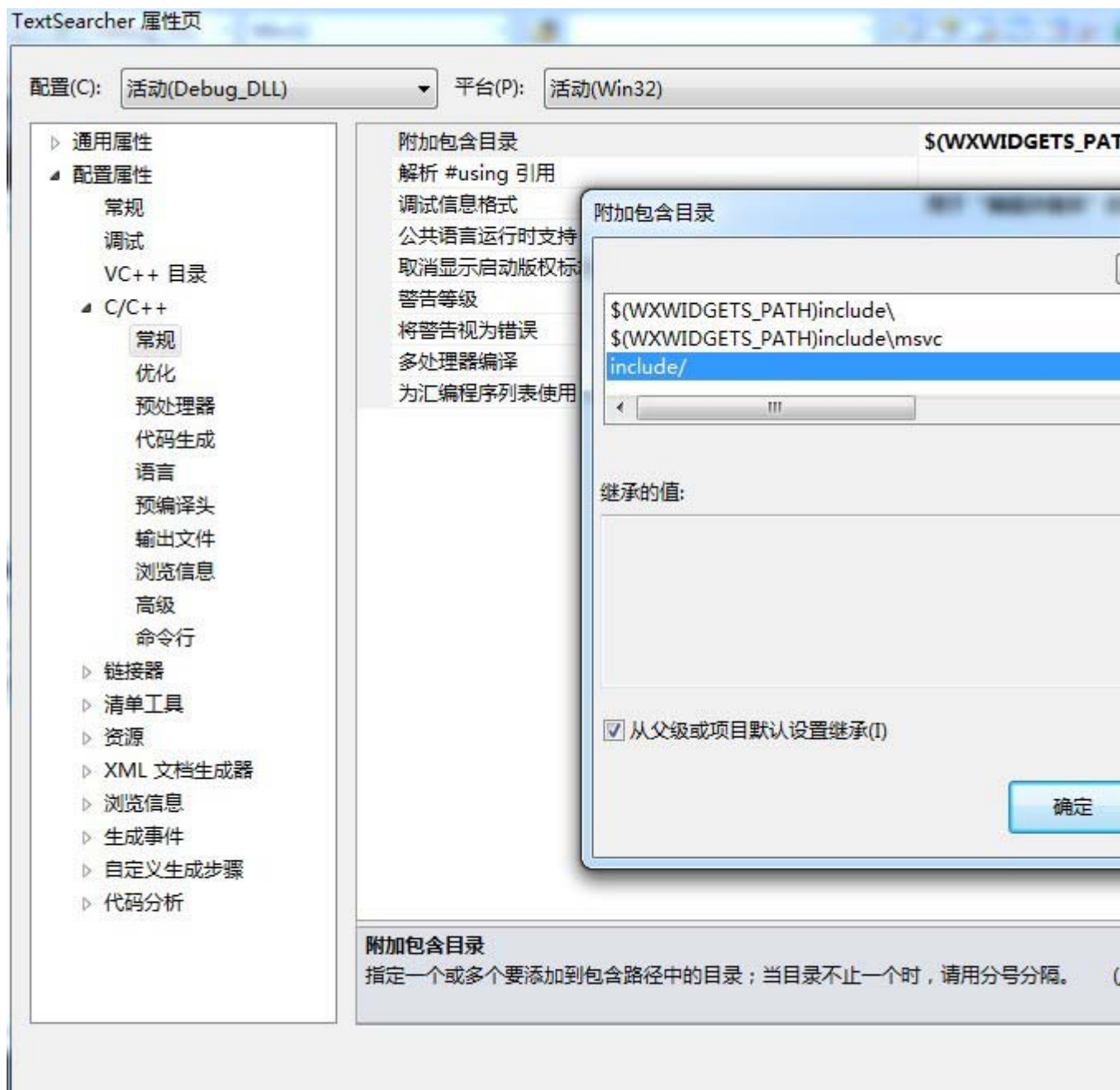


比如上图将头文件和源文件、资源图标文件、本地化文件和配置文件分开组织，这样也是为了防止混乱。值得注意的是当我们把文件用文件分开的时候，需要在项目属性设置里面包含我们的子目录，否则我们无法在源文件中直接用`#include` 指令包含我们的头文件。如下图所示这样的情况，如果不添加，无法找到头文件。

```
SearchJobCacheWorkerThreadManager.h x MainApp.h
D:\TextSearcher\TextSearcher\include\SearchJobCacheWorkerThreadManager.h
(全局范围)
10 #include "SearchJobCacheOption.h"
11 #include <wx/wx.h>
12 #include "SearchJobCacheWorkThread.h"
13
14 class CSearchJobCacheWorkerThreadManager : public CWorkThreadManager
15 {
16 public:
17     CSearchJobCacheWorkerThreadManager( CSearchPanel* pParent );
18     virtual ~CSearchJobCacheWorkerThreadManager();
19
20 public:
100 %

SearchJobCacheWorkerThreadManager.cpp
D:\TextSearcher\TextSearcher\source\SearchJobCacheWorkerThreadManager.cpp
(全局范围)
1 /**
2  * @brief Search Job cache worker
3  * @author dongbo
4  * @date 2010.6.4
5  */
6
7 #include "SearchJobCacheWorkerThreadManager.h"
8 #include "SearchPanel.h"
9 #include "SearchJobCacheWorkThread.h"
10 #include <algorithm>
```

如下图所示，找到项目属性中，C++，常规中把我们的子目录作为附加路径添加到“附加包含目录”中。

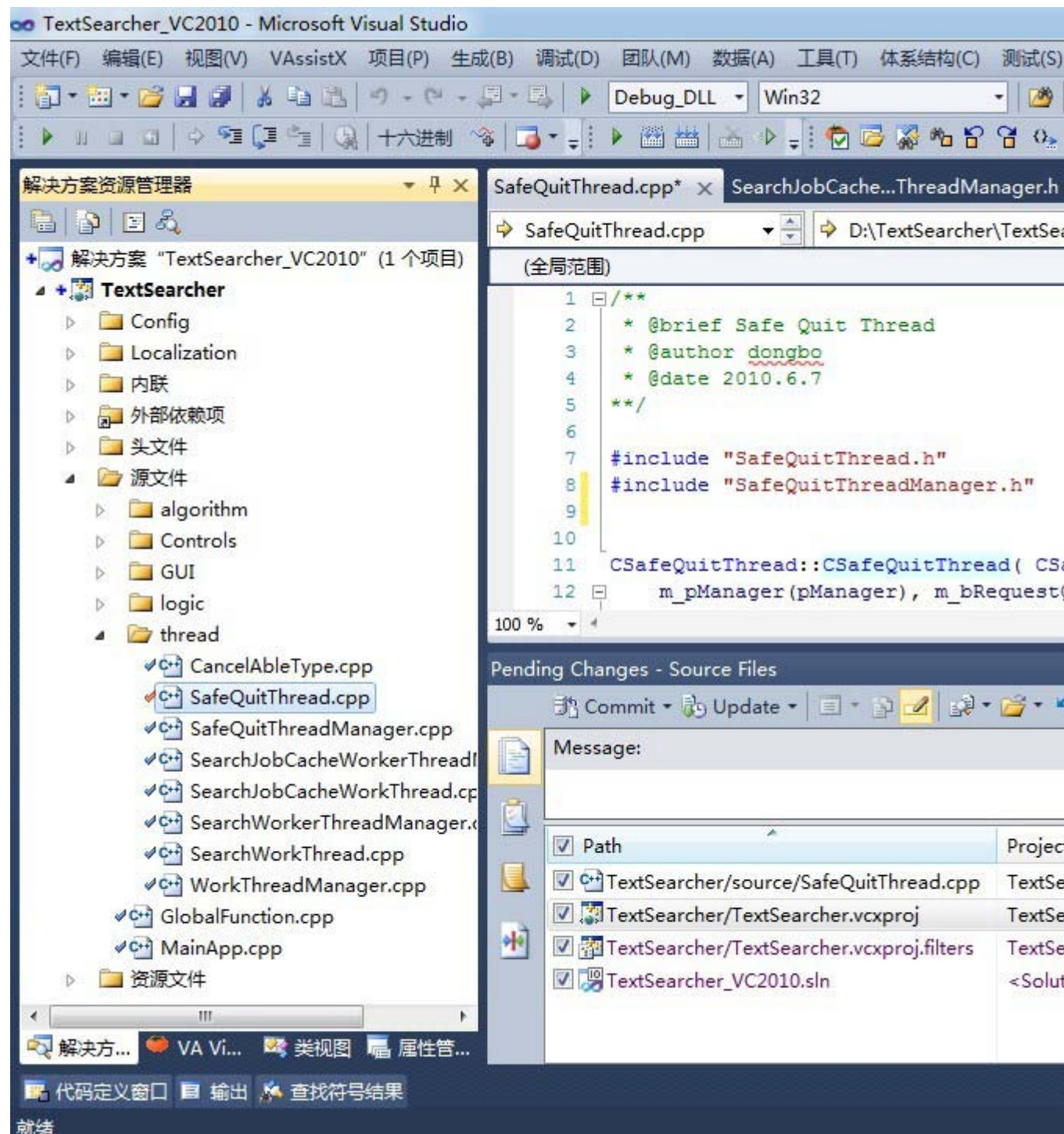


使用 **SVN** 或其它源代码管理工具管理我们的项目

如果你打算写一个比较大一点的项目，我推荐你使用源代码管理工具来管理你的 **C++** 项目，你可以选择 **SVN**，也可以选择其它的，我推荐 **SVN**，因为简单容易上手。

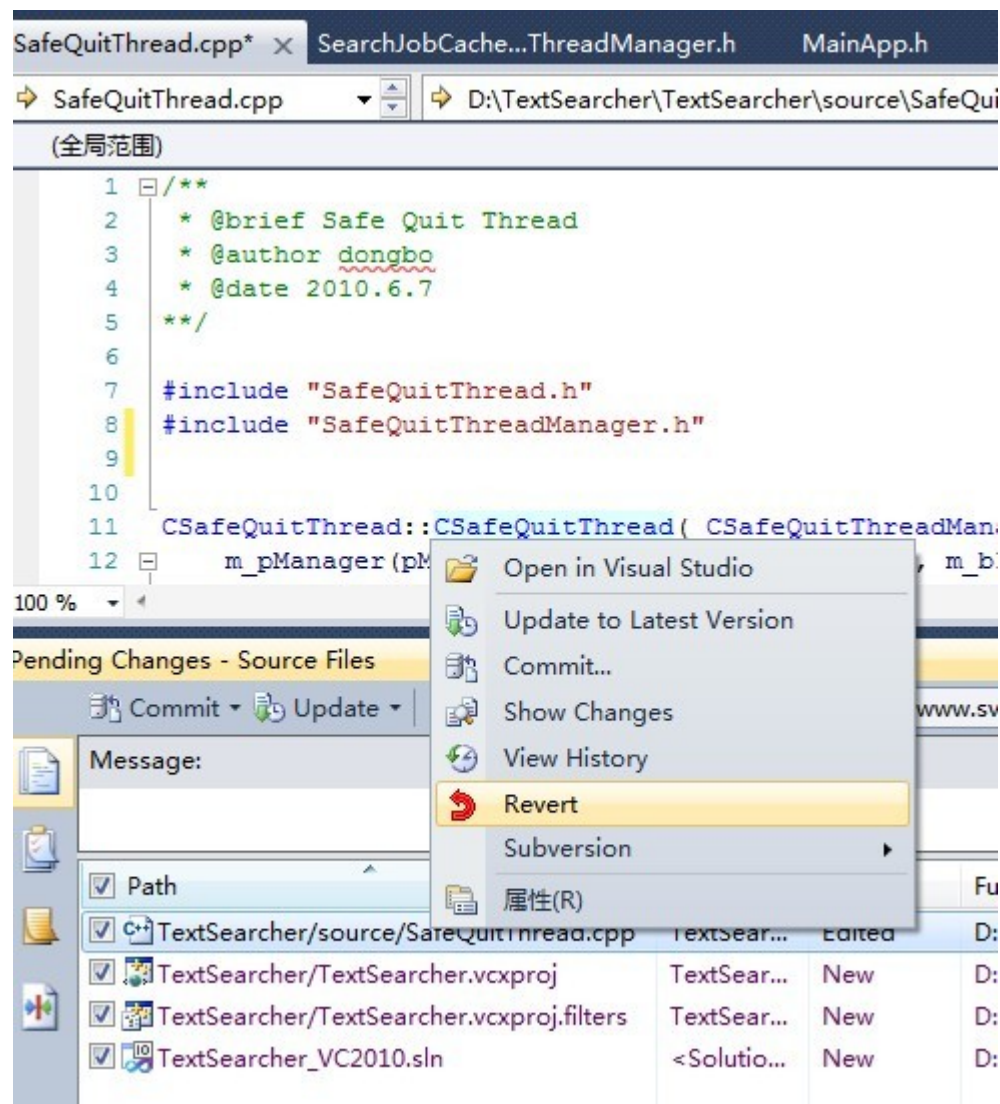
当你在做一个很复杂的东西的时候，花了两三天的时间去做修改，不过后来发现这个修改并不合适，想还原到三天以前，如果你没有用源代码管理工具管理自己的项目也没有自己手动的备份，那恭喜你，你得开始人肉还原了，这是多么悲剧的一件事情啊，然而如果你使用了 **SVN** 管理的话，只需要在三天前开始修改前的最后一次稳定版本 **Commit** 一次，三天之后如果要还原，只需要使用工具 **Revert** 就好了，而且不但可以回到三天前的版本，你甚至可以回到以前每一次 **Commit** 的版本，(◕ ◡ ◕)哇，这是多么好的工具啊！

要使用 SVN，首先需要 SVN 客户端，SVN 服务器是可选的。我推荐大家使用 TortoiseSVN 这个 SVN 客户端，因为它简单易用、免费，支持 Windows32Bit、64Bit，你可以去他们的官方网站下载，地址点我。有了这个工具当然还不够，为了让我们的 SVN 跟 VS2010 结合的更紧密，我们需要再下载一个 SVN For VS 的插件，我强烈推荐你使用它，因为它也非常简单易用，下载地址点我。最新版是支持 VS2010 的。当你装了这个插件的时候，如果你的项目是在 SVN 的管理之下，那么你的项目看起来会一些不同：

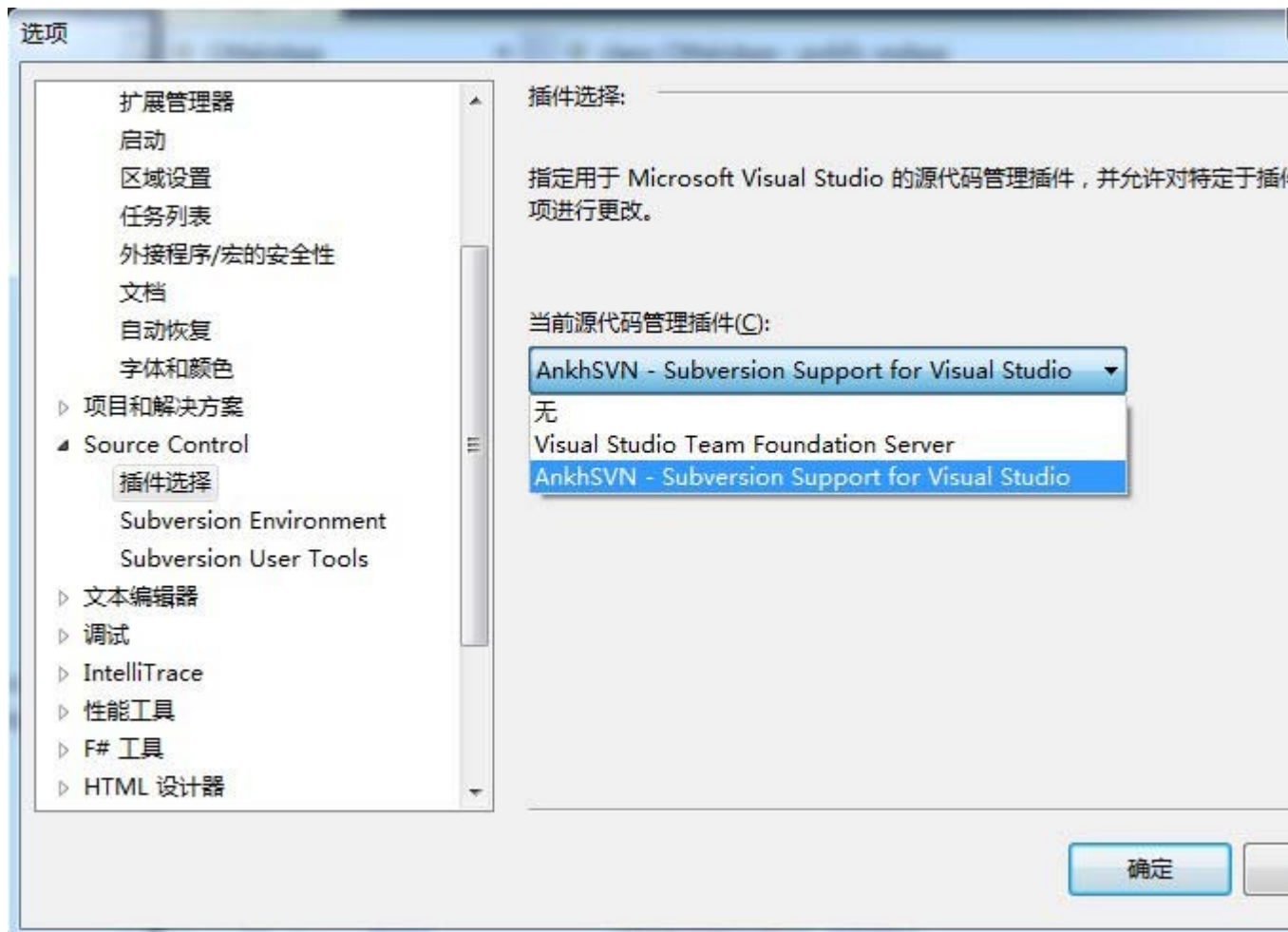


看到文件左边的勾了吗？灰色的勾表示正常，而橙色的勾表示已经更改了，而 PendingChanges 则告诉我们哪些文件是新加的，哪些文件被改过了，如果要还原的话，只

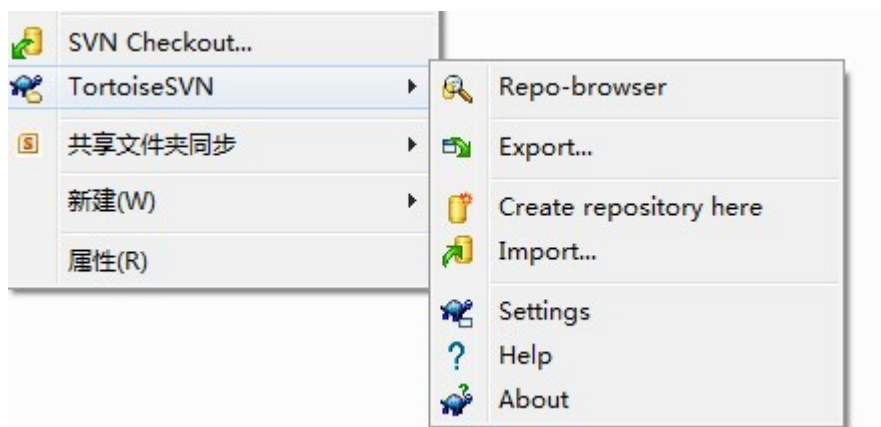
需要选择该文件，右键点击，**Revert** 就好了：



如果你已经安装好了 AnkhSVN 插件但是又看不到的话，那么请检查下系统选项卡里面是否选择了它作为默认的源代码管理工具：



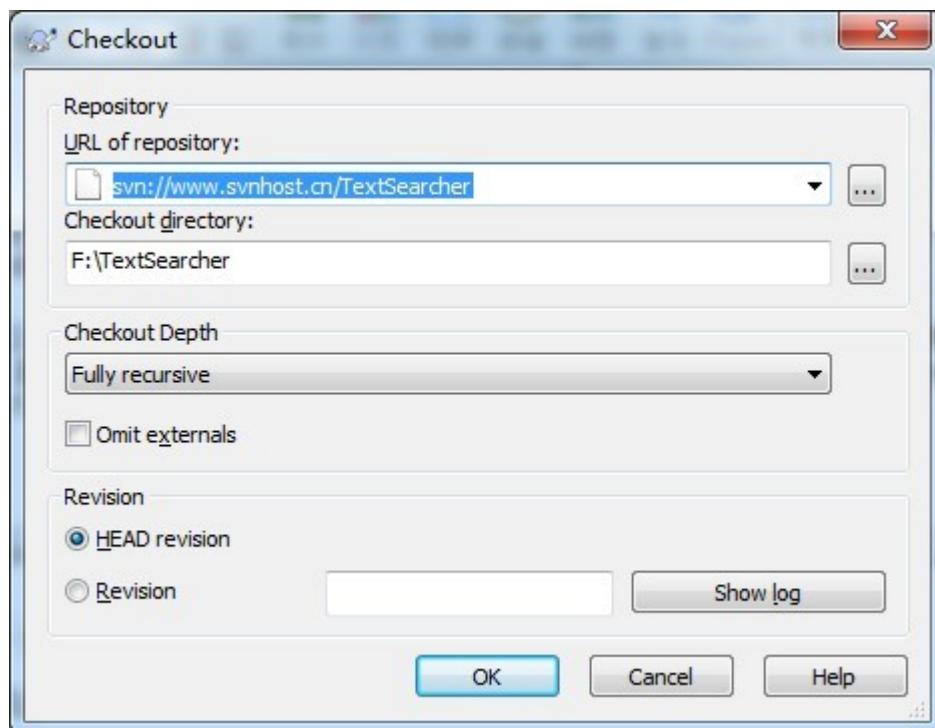
回过头去继续说 SVN，当我们安装好 TortoiseSVN 之后（安装后可能需要重启），我们在资源管理器中点击右键的时候就能看到它的菜单了：



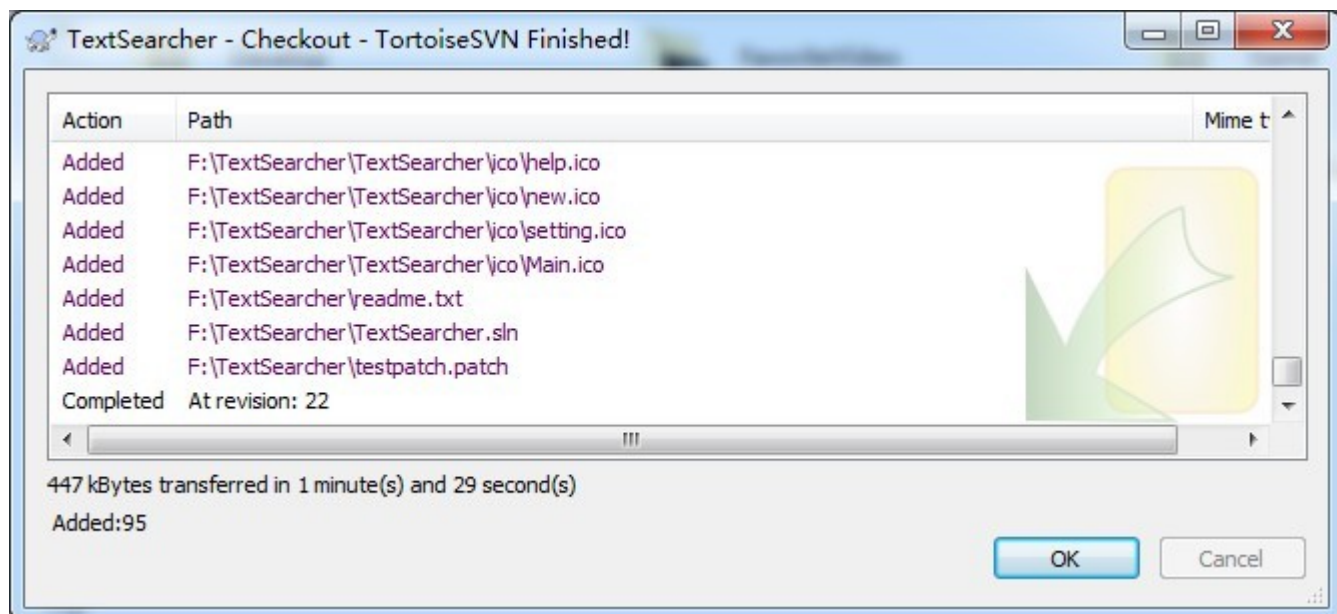
虽然 TortoiseSVN 有中文语言包，但是我推荐大家使用英文的，就像古诗一定要用中文来表达一样，没有比英语单词表达 SVN 项目管理更恰当的词了。

SVN CheckOut 可以让你获取其它地方 SVN 服务器上面的某个项目的源码，当然，前

提是你要有权限才行, 现在让我们试试。随便找个盘符如 F 盘, 点右键, 选择 SVN CheckOut, 然后 Url of Repository 中输入 `svn://www.svnhost.cn/TextSearcher` 这个地址, 其它的不变, 点击 OK。如图所示:



如果不出意外你能看到:

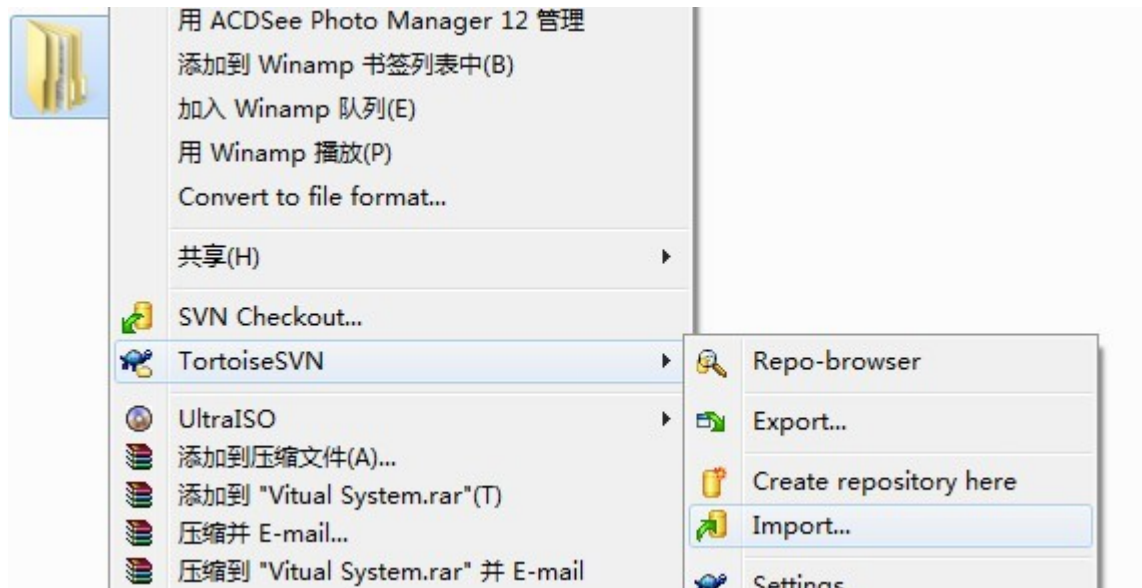


如果你看到这个画面说明你已经 **CheckOut** 成功了, 那么恭喜你, 你已经取到了 **TextSearcher** 的源码, 当然要说一点的是, 这个源码可能你取到之后编译不了, 因为还有

另外一些依赖项如 **dbsoft**、**boost**、**wxWidgets** 并不在这个 SVN 上面。

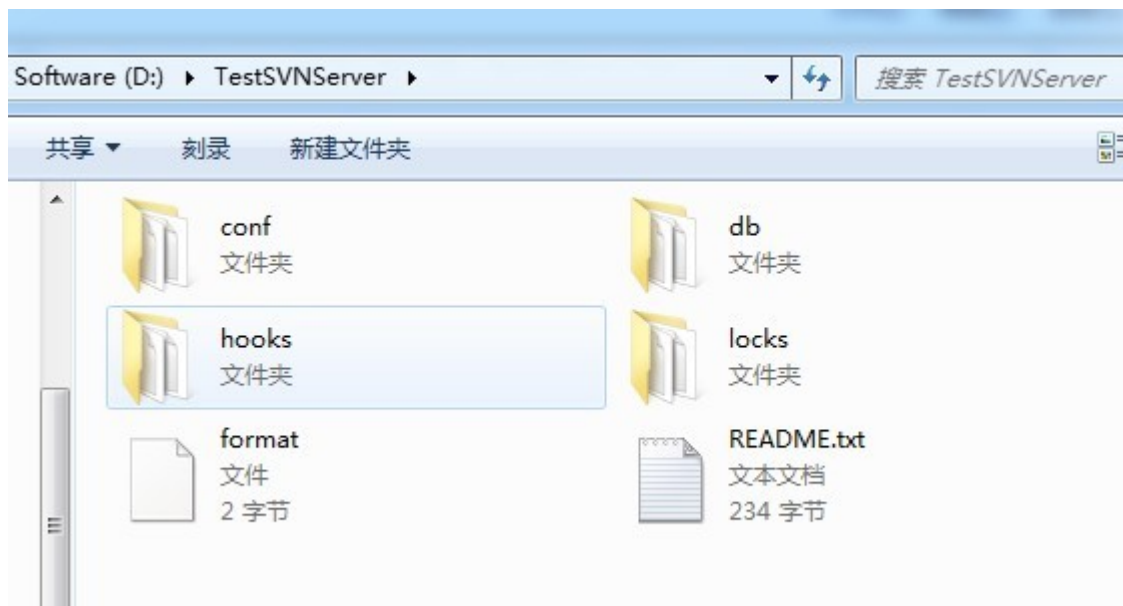
这样大家就可以去一些开源的网站上面 **Check** 你想要的东西了，哈哈。推荐大家去这个网站找自己感兴趣的代码 **CheckOut**: <http://sourceforge.net/>

现在我们说如何管理自己的项目，大家可以像我一样去一些提供免费 SVN 服务器服务的网站上面注册一个账号建立项目就好了，比如说我使用的这个 www.svnhost.cn，这个毕竟是国内的，另外上面的 sourceforge.net 也可以。当你注册建立项目成功之后就可以通过菜单 **Import** 把需要导入的东西导入到 SVN 服务器了：

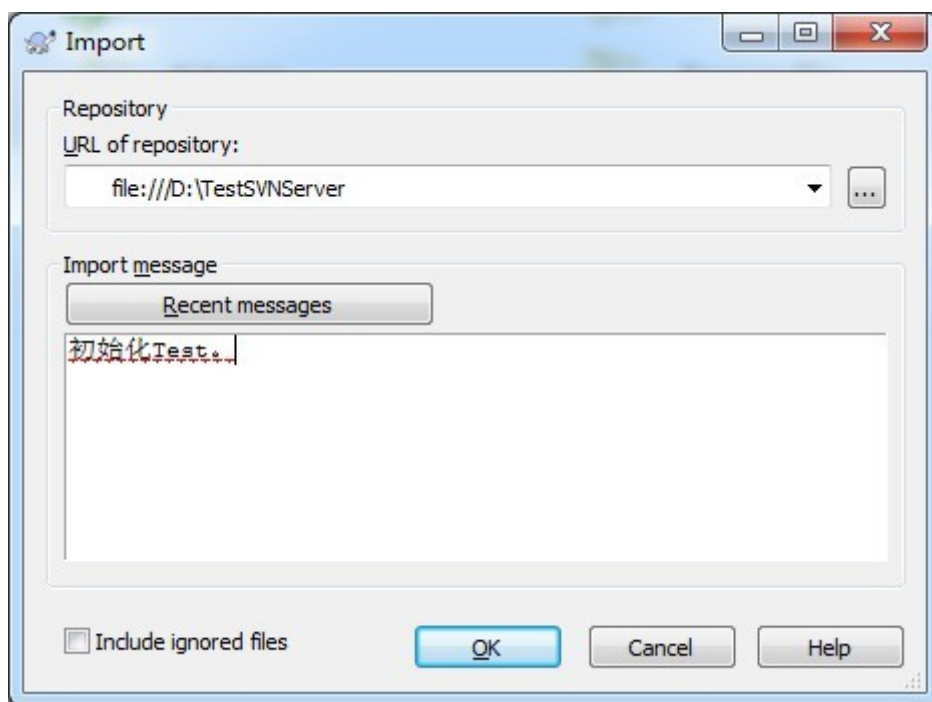


导入界面的地址栏输入我们的 SVN 服务器地址即可，类似于：
`svn://www.svnhost.cn/TextSearcher`。导出的时候记得输入日志。除了可以导入 SVN 服务器之外，我们还可以使用文件协议在自己的电脑上管理我们的代码，如果你不需要在多台电脑上面共同维护这个项目的話。

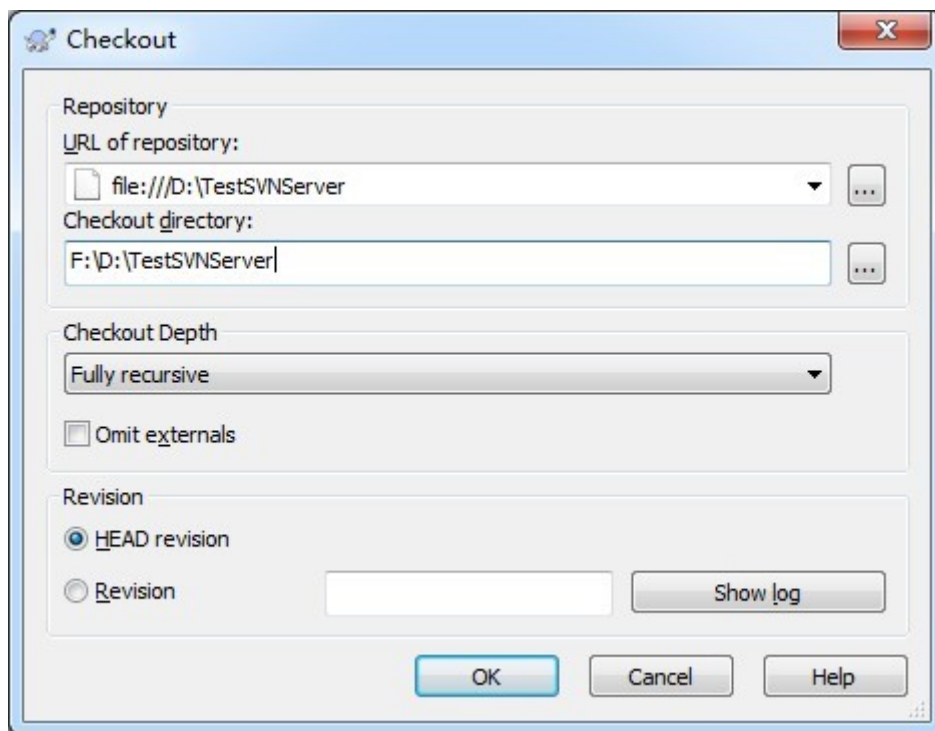
让我们随便找个地方新建一个文件夹，如 `D:\TestSVNServer` 这个文件夹，然后对着这个新建的文件夹点右键，选择 **SVN>Create repository here**，OK，你的本地 SVN 服务器已经建好了。如果创建成功，你会发现这个文件夹里面多了好多东西：



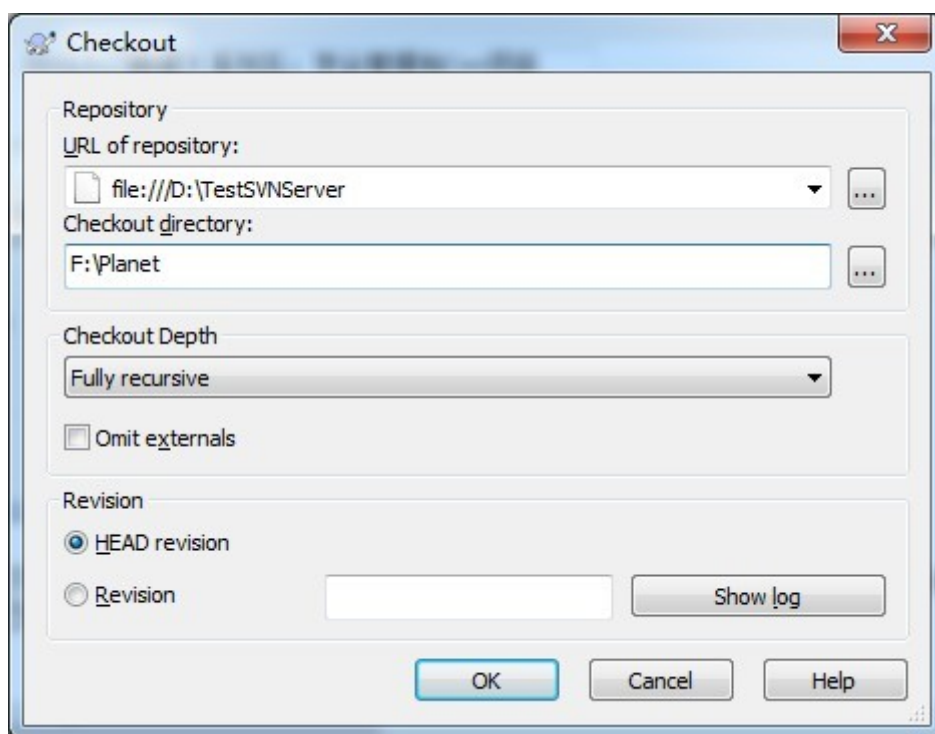
没关系，这是 SVN 服务必须要的一些东西，现在让我们使用文件协议导入我们的项目到该 SVN 服务器中，找到我们要导入的项目，跟导入网络上的 SVN 服务器一样，右击项目文件夹，SVN>Import。在 Import 界面中这样填：



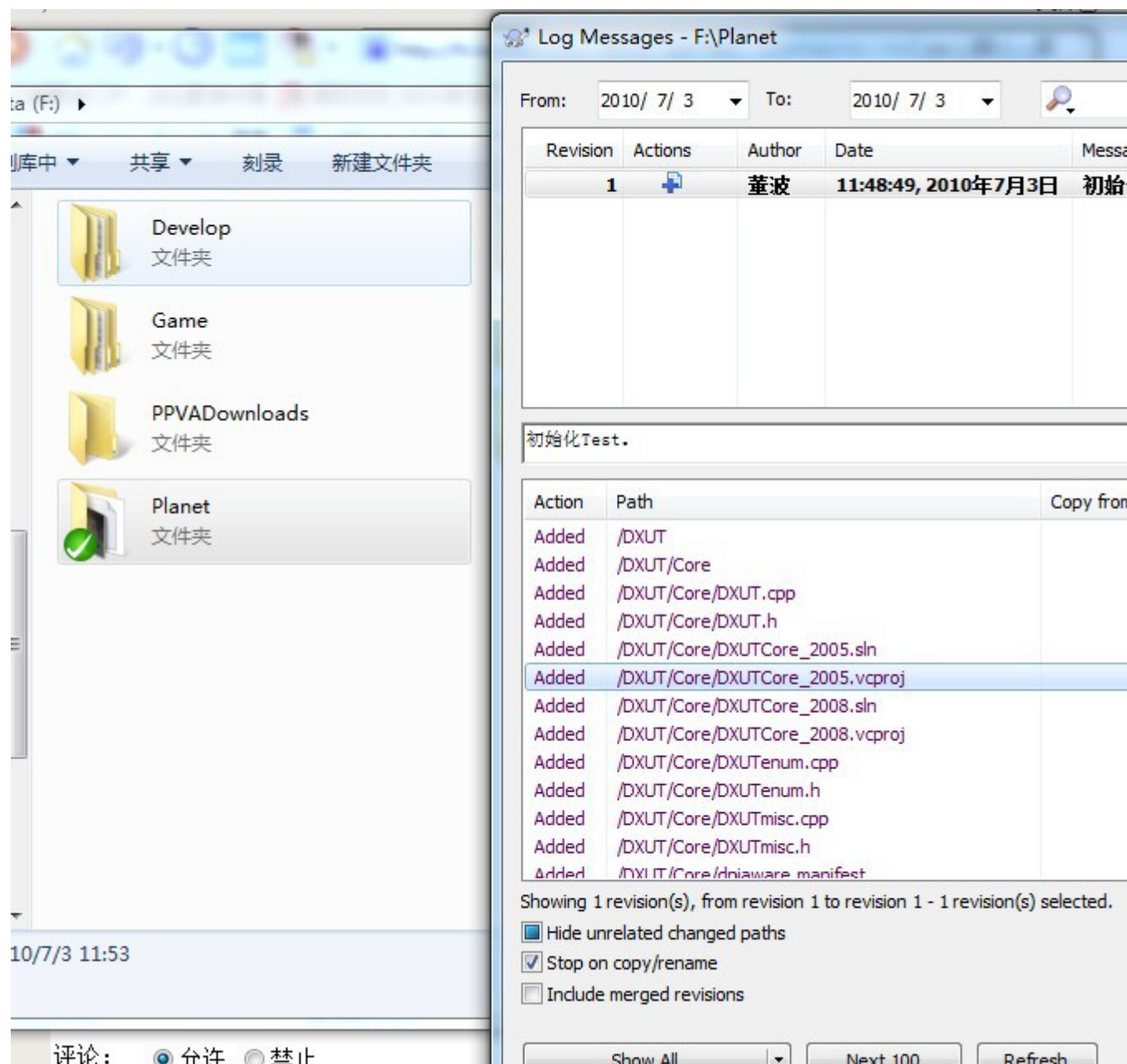
不出意外你会导入成功，那么现在我们需要去其它地方重新获取这些东西了，换个地方，比如 F 盘根目录，选择 SVN CheckOut，然后地址输入刚才导入的地址：



注意如上图这样是不行的，因为我们没办法在 F 盘根目录下面创建一个 D:\TestSVNServer 文件夹，把 D:\这个 SVN 帮我们自己填充的路径删掉然后点确定。



现在我们已经 CheckOut 了刚才导入的项目了，Planet 文件夹已经带上了一个绿色的勾。点击这个 Planet 目录，选择 SVN>Show Log，你会看到：



在这个界面我们还可以检查代码的改变都是什么，这里由于是新的项目，所以没有更改，我们可以找到 **TextSearcher** 的更改来比较它们的改动都是什么：

Log Messages - D:\TextSearcher

From: 2010/ 6/13 To: 2010/ 7/ 3
Messages, authors and paths

Revision	Actions	Author	Date	Message
23		db123	12:03:47, 2010年7月3日	为TextSearcher添加VS2010项目编译支持。
22		db123	20:00:17, 2010年6月25日	添加部分信息，用于取消对源代码修改的依赖。
21		db123	19:19:45, 2010年6月21日	更新。
20		db123	11:14:25, 2010年6月21日	修复提示信息错误的Bug。
19		db123	10:33:38, 2010年6月21日	增加搜索通知。
18		db123	18:10:30, 2010年6月17日	更新设置对话框，重命名部分内容。
17		db123	16:36:58, 2010年6月16日	添加UML。
16		db123	15:35:24, 2010年6月16日	自动滚动更新。

添加部分信息，用于取消对源代码修改的依赖。

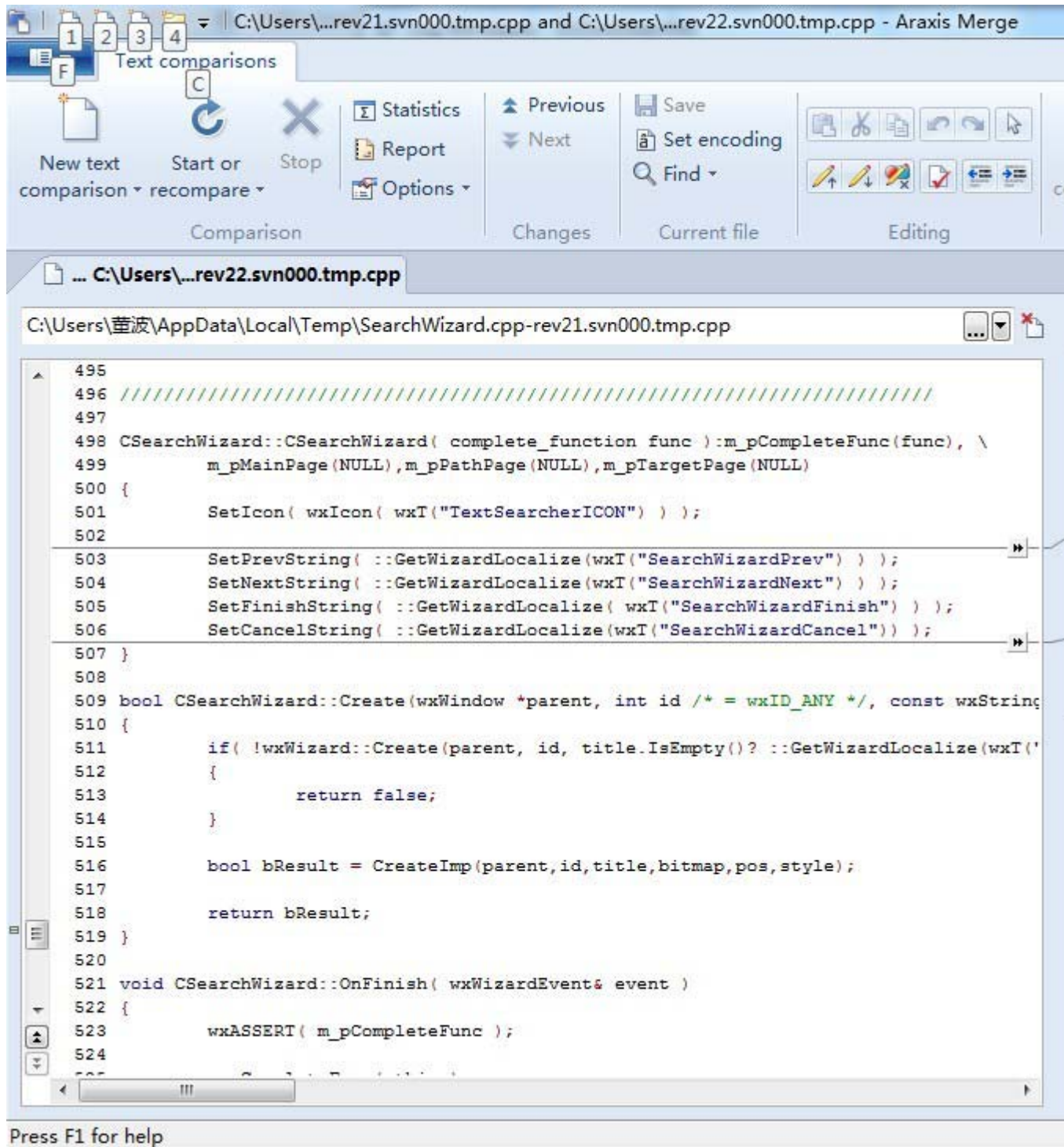
Action	Path	Copy from path	Revision
Modified	/TextSearcher/Config/system.ini		
Modified	/TextSearcher/include/SearchWizard.h		
Modified	/TextSearcher/source/S		
Modified	/readme.txt		

Showing 23 revision(s), from revision
☒ Hide unrelated changed paths
☒ Stop on copy/rename
☐ Include merged revisions

Show changes
 Blame changes
 Show changes as unified diff
 Open
 Open with...
 Blame...
 Revert changes from this revision
 Show properties
 Show log
 Get merge logs
 Save revision to...

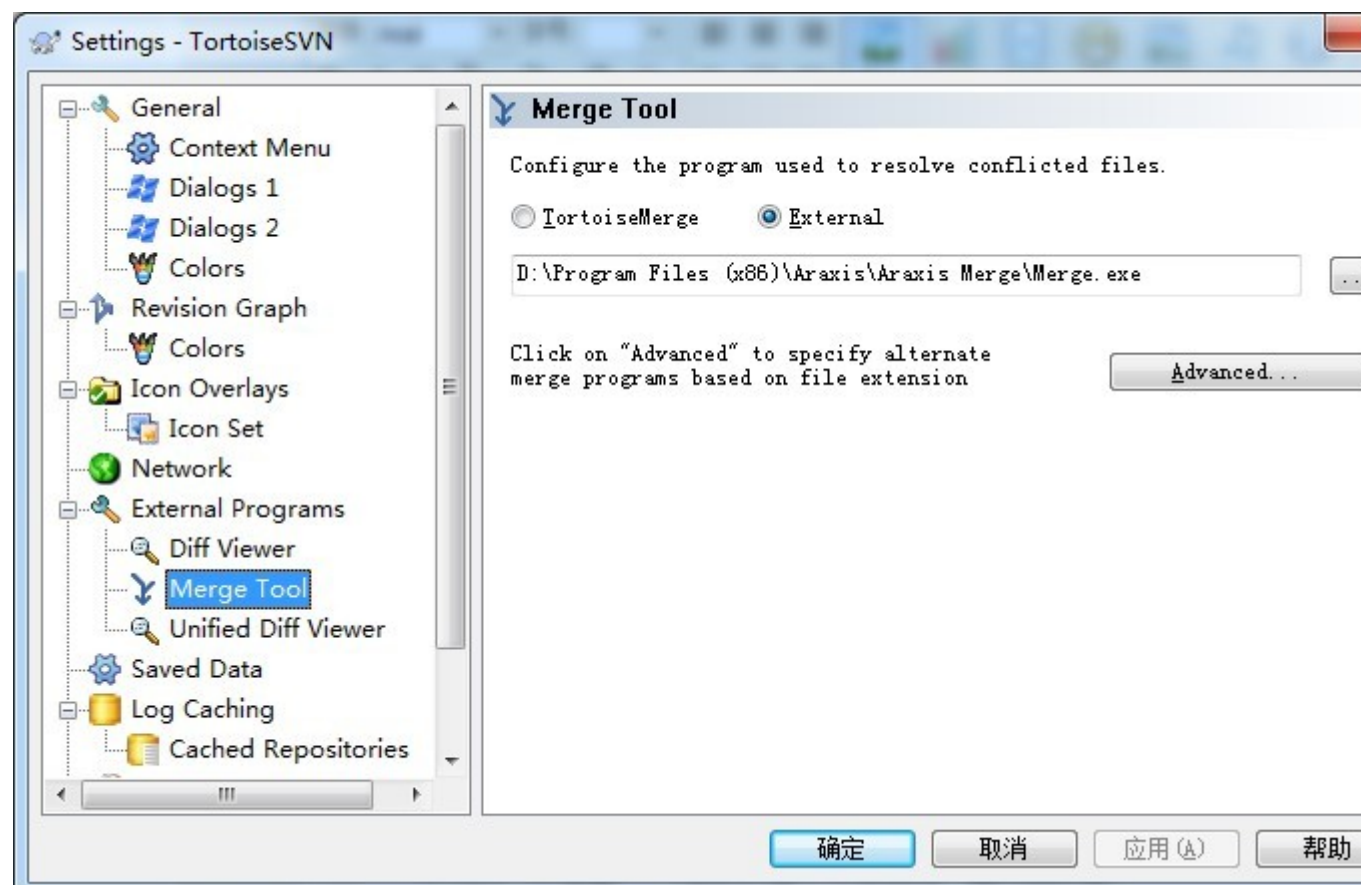
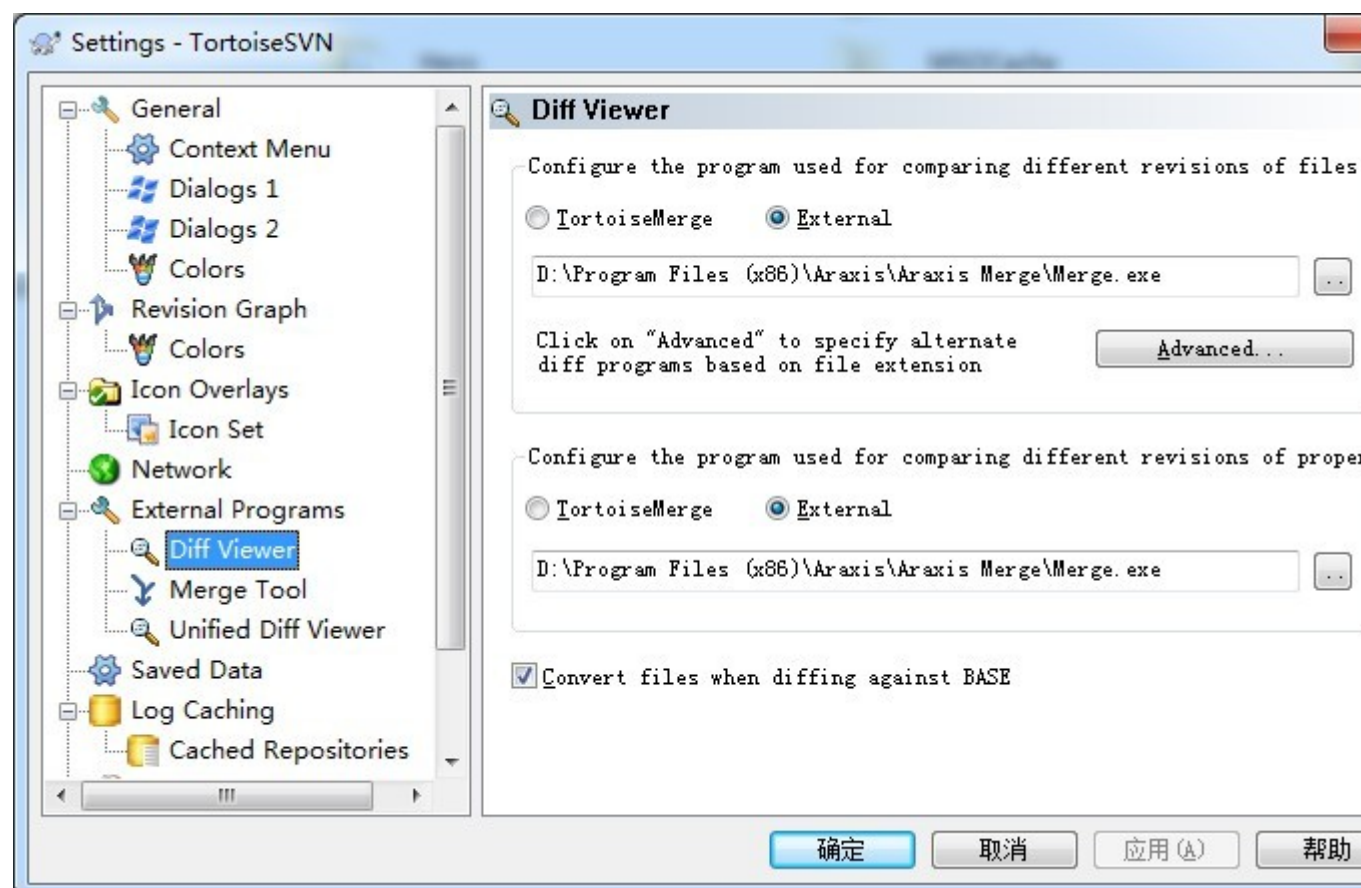
Show All
Next 100
Refresh

Show Changes:



可能你弹出的界面跟我的不一样，因为我的比较工具是外部工具而不是默认自带的，因为默认自带的有问题，在合并的时候总是出错（公司项目）。所以我对默认的不感兴趣。这个软件的名称是：Araxis.Merge.Professional.2010，你可以去搜索来下载，[这里有一份可以试试](#)，不行就自己去搜索吧。

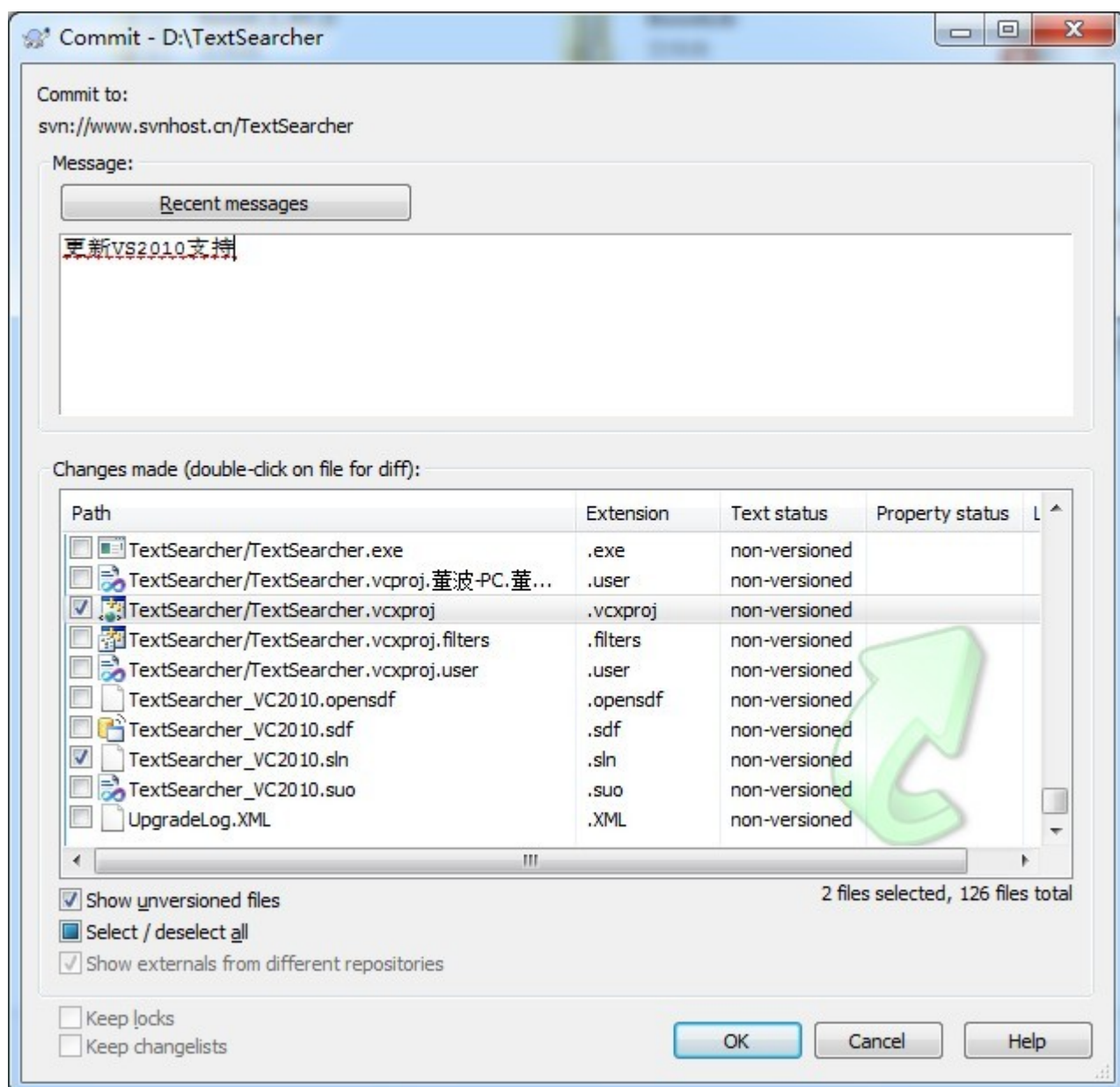
装好这个软件之后需要去 SVN 里面设置一下，SVN》Setting，设置如下：



这样就可以使用外部工具了。

下面再说明一下如何提交代码(Commit).

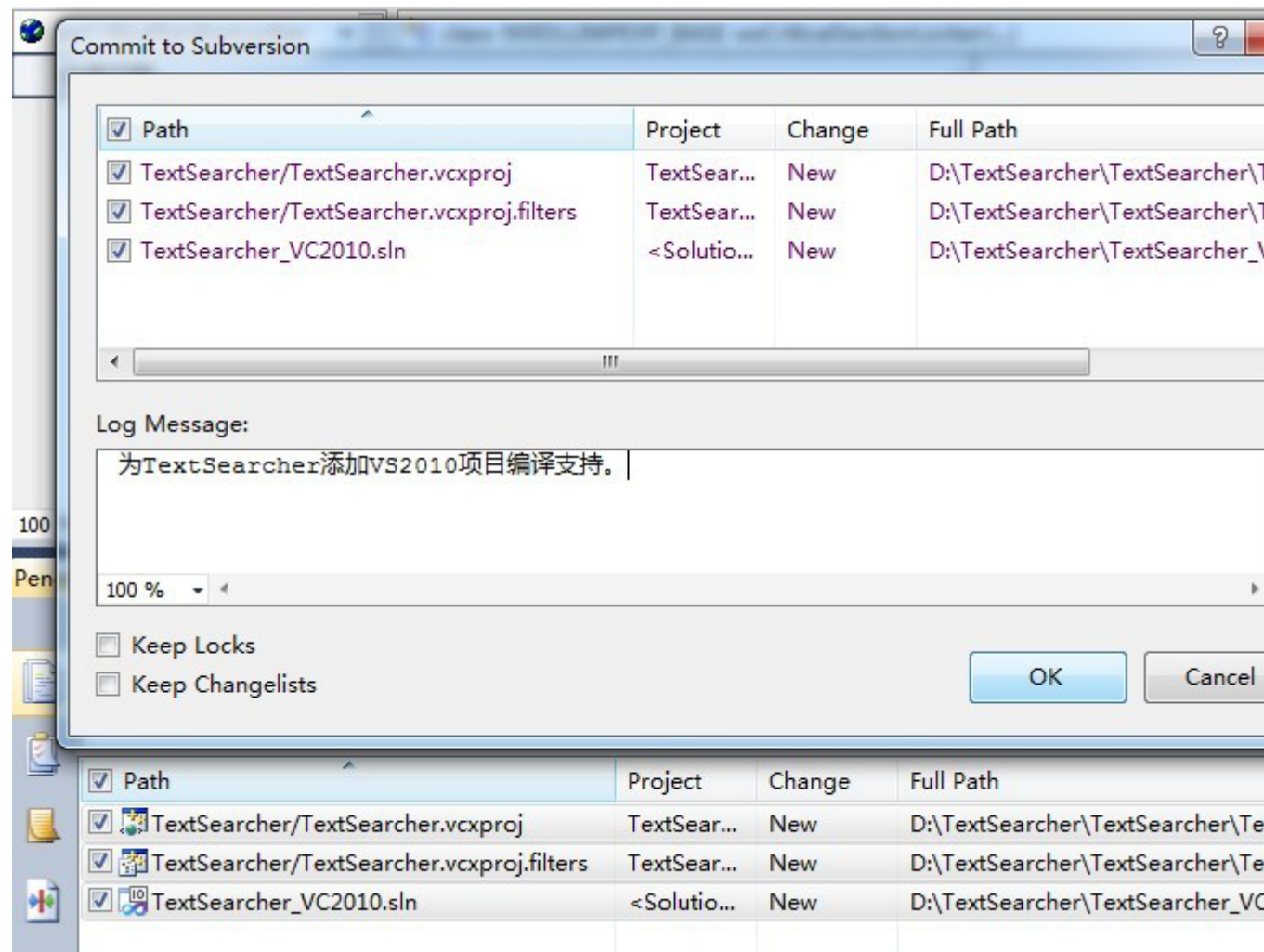
我们可以通过右击被 SVN 管理的项目, 选择 SVN Commit 输入相关信息之后就可以提交了, 通常提交都需要你提供用户名和密码验证等等, 特别是网络上的 SVN 服务器, 本地 SVN 服务器可以不管。



注意, SVN 不会把你新添加的文件默认选中, 它只会默认选中已经在 SVN 中添加了的文件。所以当新加了文件的时候注意勾上, 防止漏传, 这个在现在的 SVN 管理中漏上传时非常普遍且经常发生的事情。那么现在用 VS 的 SVN 插件也可以上传啊, 对于添加到

VS2010 的 IDE 中的文件我推荐大家使用这种方式，因为这样可以有效的防止漏上传新加的代码，注意，有的二进制文件如资源文件 MP3、PNG 或者其他资源打包文件等可能不会添加到 IDE 中，因此这时候一定要注意不要漏传，去资源管理器中右击项目，SVN》ADD，将新加的有用的文件添加进去，一些临时文件和垃圾文件不要添加。

VS 的 SVN 插件要提交更简单，在 Pending Changes 中选择要上传的，点击右键，Commit 即可：



如果你的这个选项卡没有调出来可以通过视图》Pending Changes 把它调出来。

如果项目中的其它人更新了代码，版本增加，那么请用 Update 获取最新的代码。

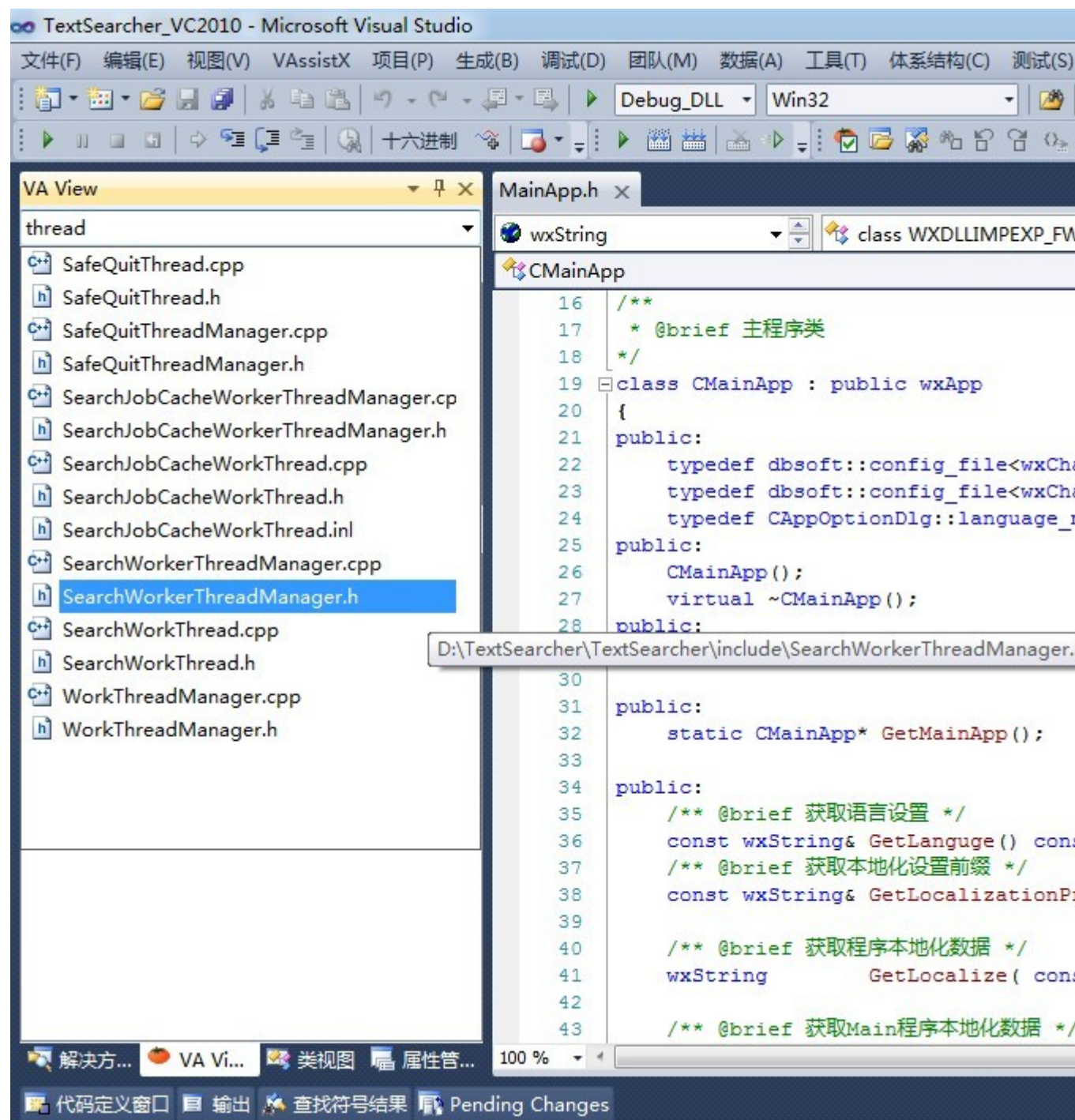
对于 SVN，有很多丰富的内容，不是这里短短的时间可以说完的，建议大家去申请一个 SVN 服务器自己多使用一下，多摆弄下就熟悉了，能够熟练掌握 SVN 这样的源代码管理工具在找工作的时候是一个加分的不错的砝码。

使用 Visual AssistX 让我们的工作更有效

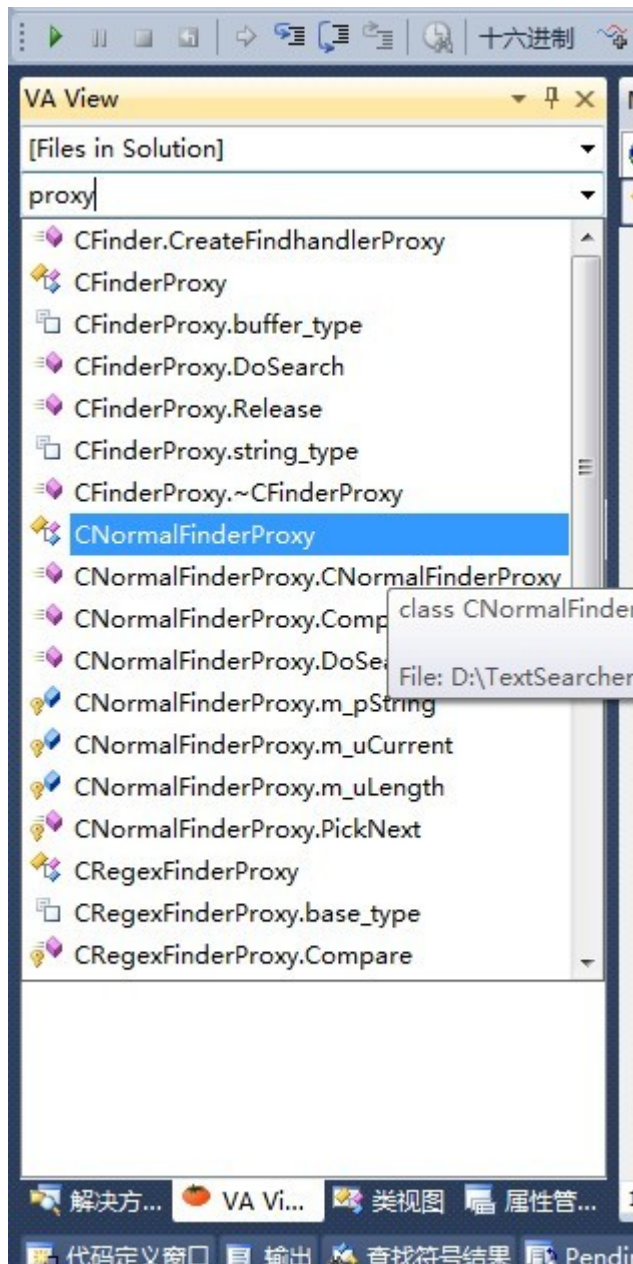
Visual AssistX(简称 VAX)是一款优秀的 VS 辅助插件，最新版支持 VS2010。如果你有

钱而且生活宽裕，建议你购买正版，你可以去他们的官方网站购买；如果你和我一样是个不折不扣的穷鬼，那么你可以[点这里](#)去下载破解版，当然请大家在心里感谢 VAX 的工程师们吧。

通常我们安装好 VAX 之后它会默认的帮我们开启它的大部分功能，而且这通常都是足够的，使用 VAX 的 View 选项卡可以轻松的在大型项目中找到我们想要的文件，如下图所示：

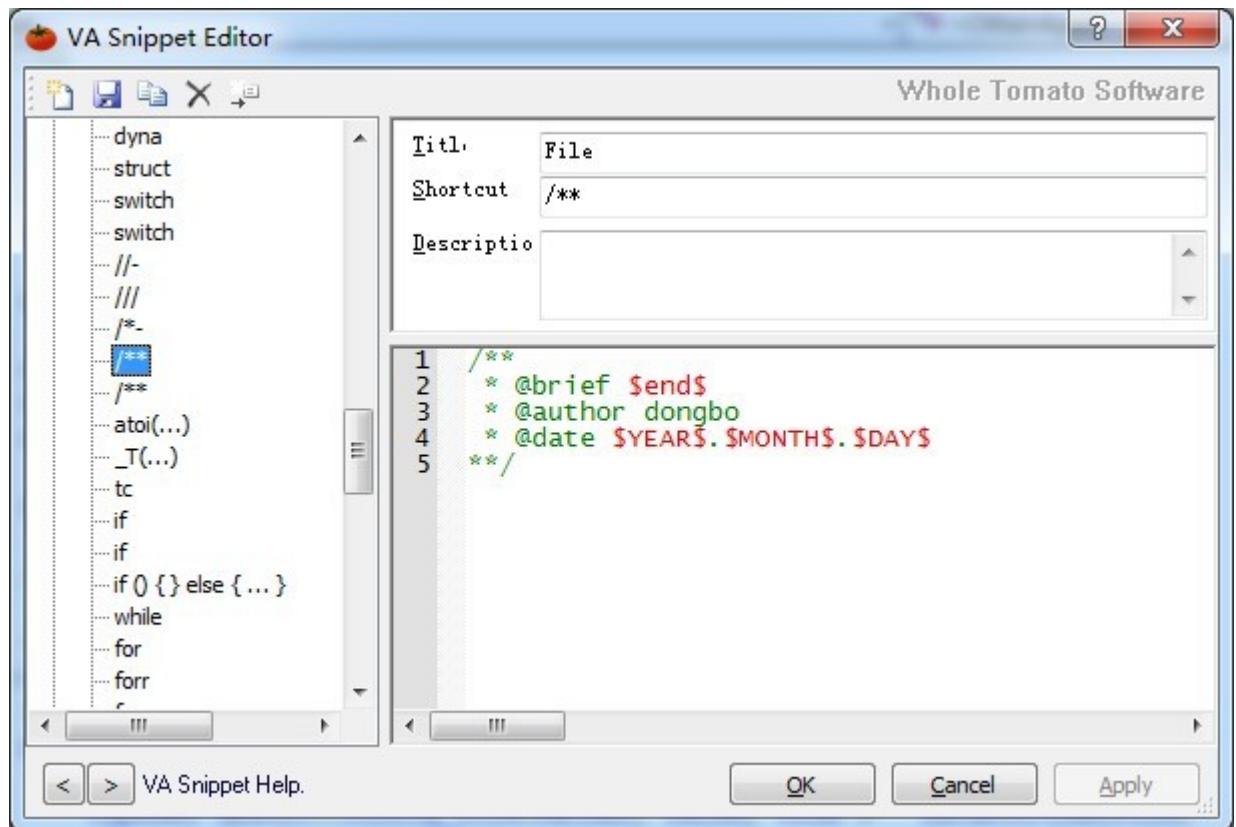


在符号表中还能轻松的找到我们的类在什么地方：

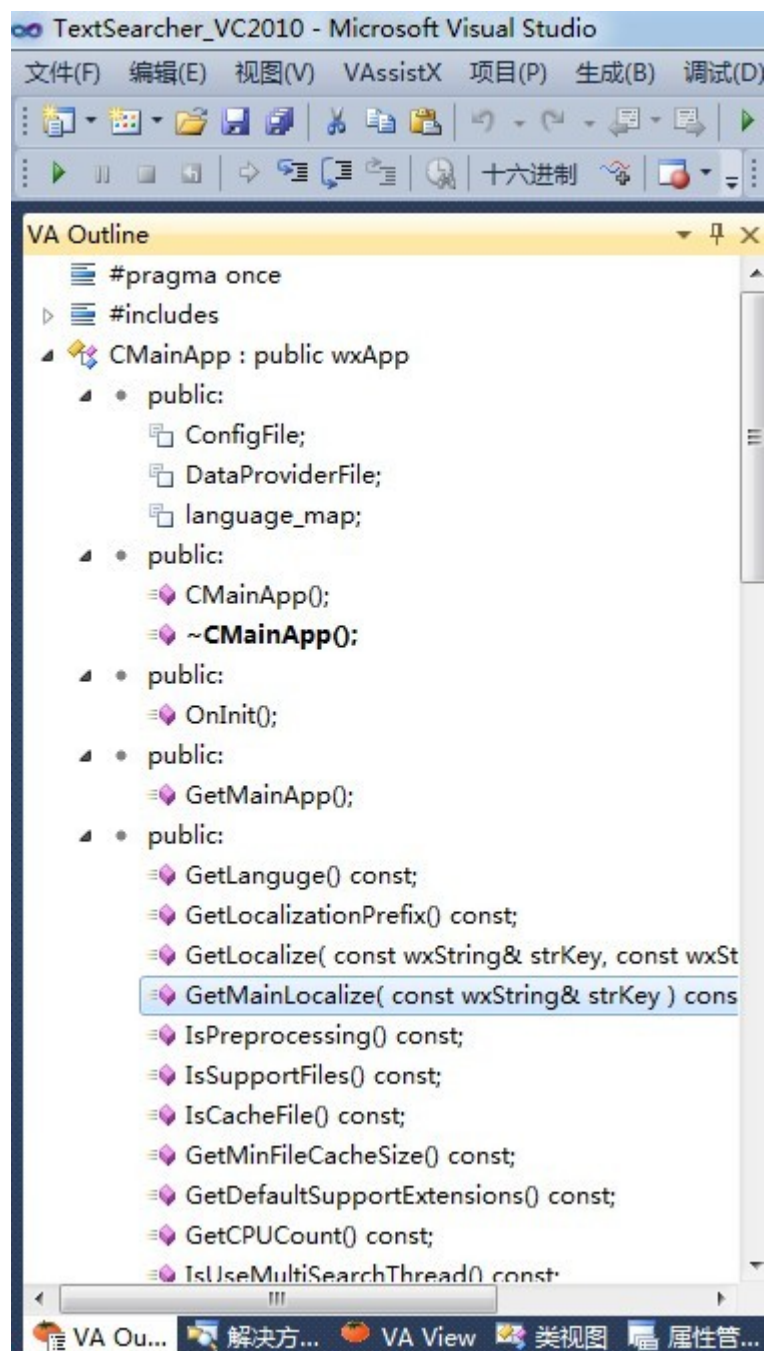


自动着色、自动添加括号，自动换行等功能可以给我们莫大的帮助。

通过编辑 Snippet 可以让我们轻松添加 Doxygen 风格的注释帮助：



通过 VAX 的 OutLine 选项卡让我们阅读代码变得轻松：



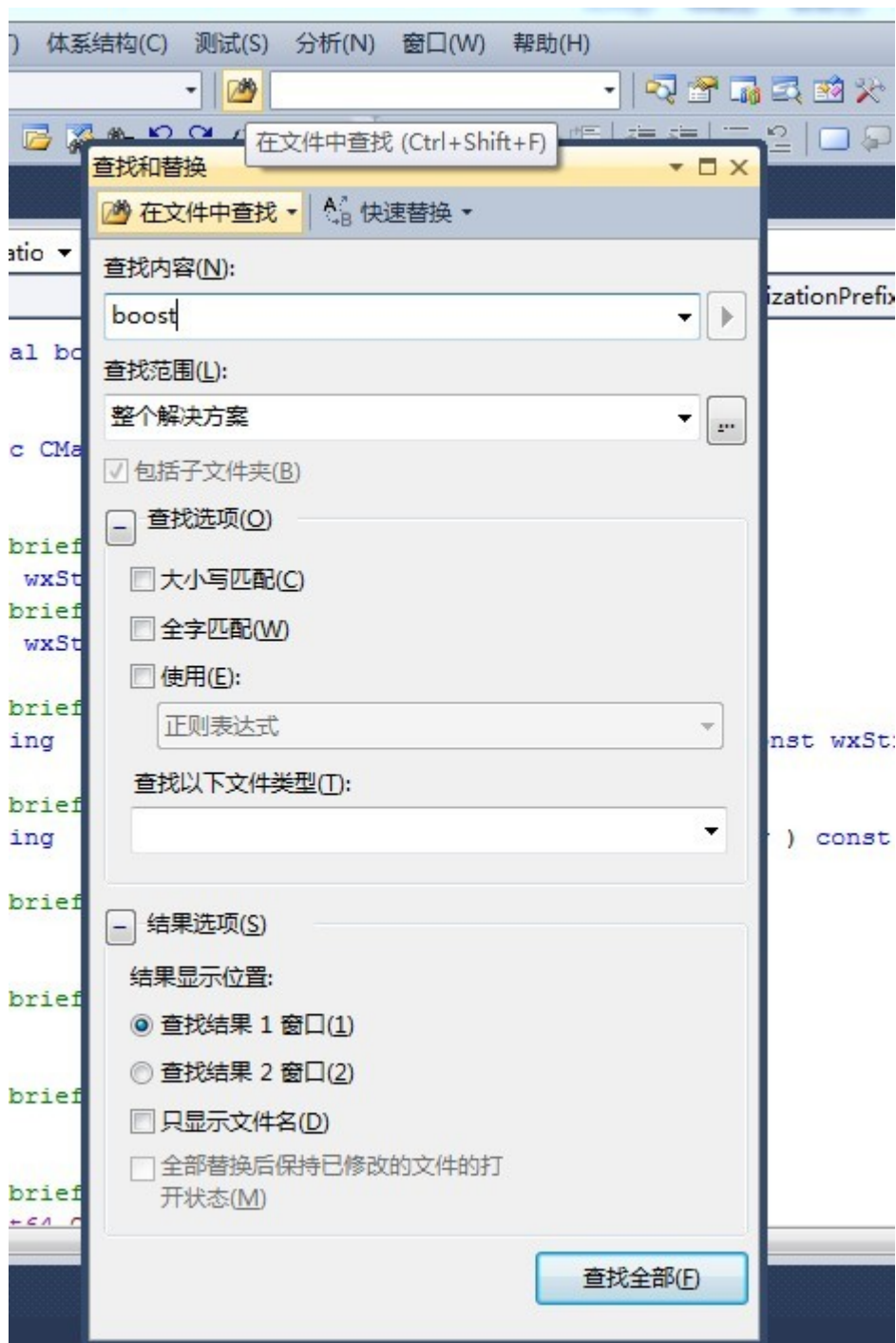
没有这两个选项卡可以通过 VAX 的菜单，Tools 把它们调出来。更多功能请自己去体会。

让我们熟悉 VS2010 的快捷键和其它一些有用的功能

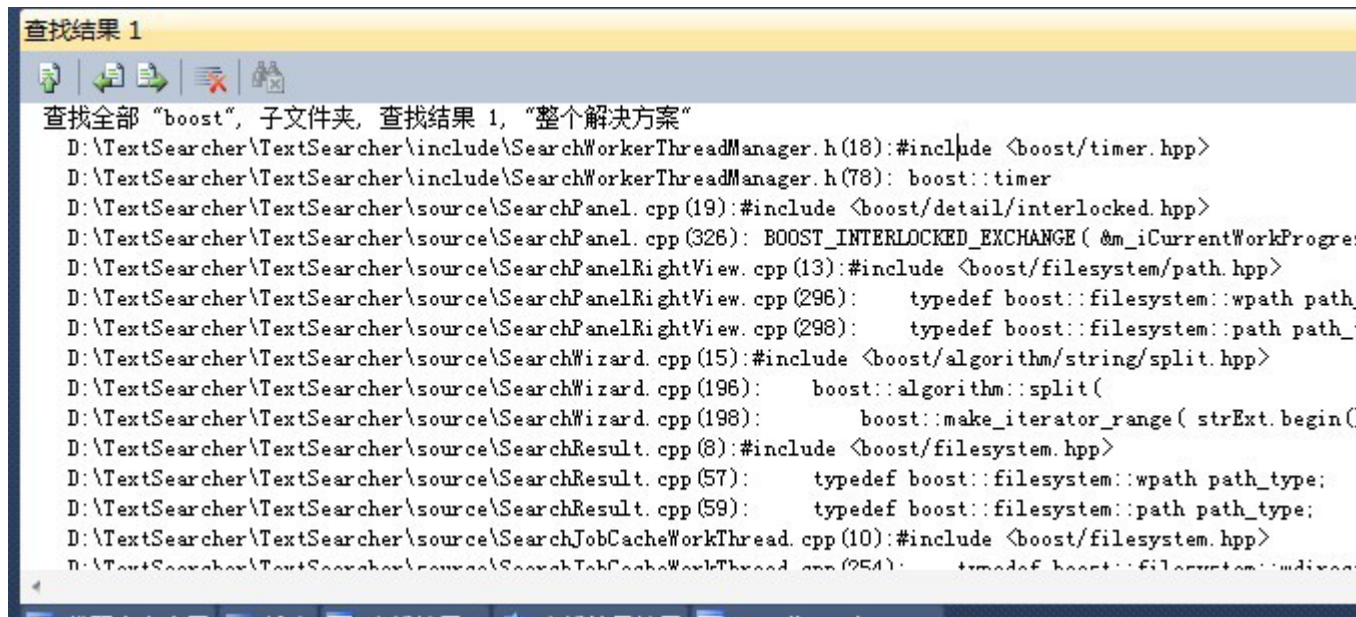
VS2010 是一个强大的工具，下面介绍一些常用的功能和快捷键。

1. 搜索所有文件

大部分朋友都知道用 **Ctrl+F** 来搜索和替换文件，但是并非每个朋友都知道 **Ctrl+Shift+F** 可以搜索所有文件：



我们还可以通过点击工具栏上面的那个按钮来弹出这个界面，这个界面可以搜索整个解决方案（一个解决方案可以有多个项目）、整个项目、打开的文件、单个文件和选定内容，还支持普通搜索、正则表达式和通配符搜索，如上图所示，我们搜索所有的 **boost**。查找结果 1 会显示出项目文件中所有出现 **boost** 的地方，不区分大小写，不完整匹配：



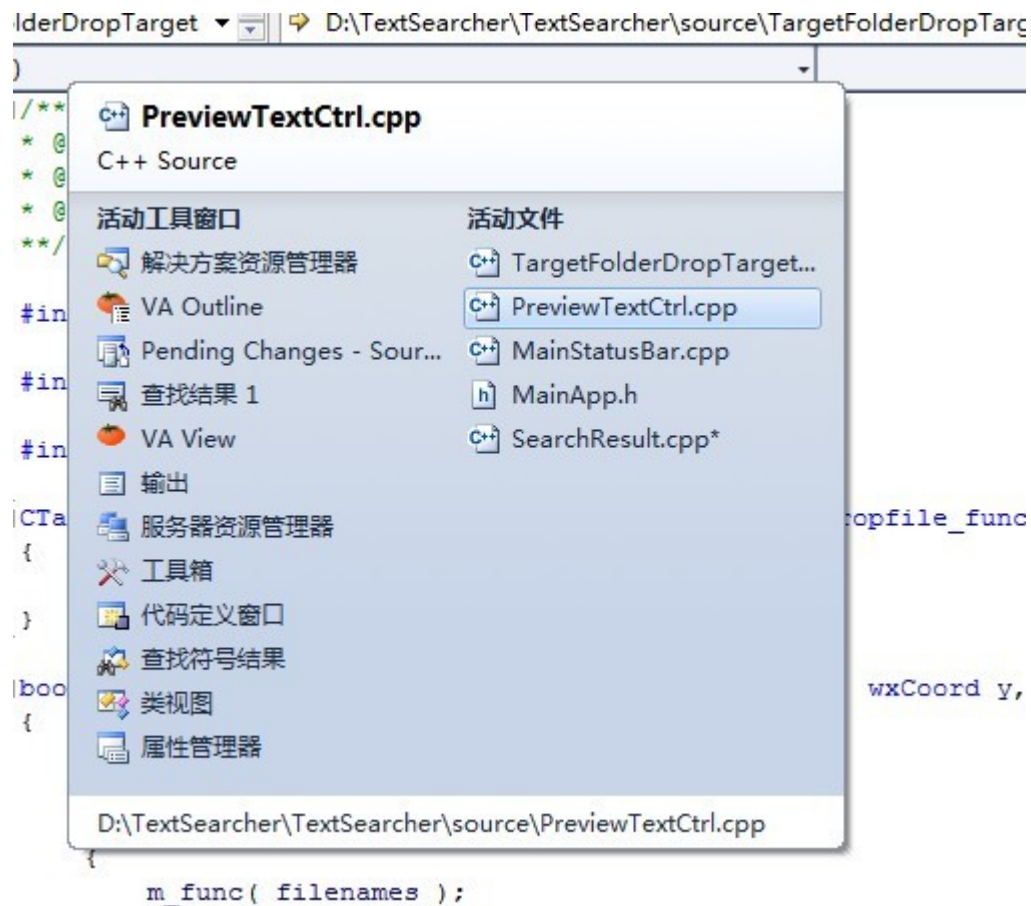
双击就可以定位到行。

2. 格式化代码 先按 **Ctrl+K** 接着按 **Ctrl+F**

通常我们都应该保持良好的编码风格，除非你打算参加“国际混乱程序大赛”。这个功能实际上在菜单《编辑》高级中就有，至少大部分人都不喜欢去挖掘这些功能。通常我们都是要格式化当前文件的所有代码，所以这个快捷键通常和 **Ctrl+A** 组合(全选)使用。 先全选，然后再全部格式化。

3. **Ctrl + Tab** 在前一个文件和后一个文件之间切换

首先我们打开多个文件，然后要切来切去用哪个，就用这个 **Ctrl + Tab**：



4. VAX 快捷键 Alt+G

这个快捷键超有用，当你把光标定位到头文件，**Alt+G** 会打开那个头文件，当你把光标定位到类、变量、函数等的时候，**Alt+G** 都会跳到它的定义或者实现。它等于装了 VAX 之后右上角的那个 **GO!** 按钮。为什么要用这个，VS 不是也有智能感知和识别吗？当你用 VS 多了的时候你就知道 VS 的智能感知是多么弱智的了。



```
1  /**
2   * @brief 预览窗口的控件
3   * @author dongbo
4   * @date 2010.6.14
5   **/
6
7   #pragma once
8
9   #include "SystemConfig.h"
10  #include <wx/wx.h>
11  #include <wx/textctrl.h>
12
13  class CPreviewTextCtrl : public wxTextCtrl
14  {
15  public:
16      CPreviewTextCtrl(
17          wxWindow *parent, wxWindowID id, const wxString &value=wxEmptyStrin
18      );
19
20  public:
```

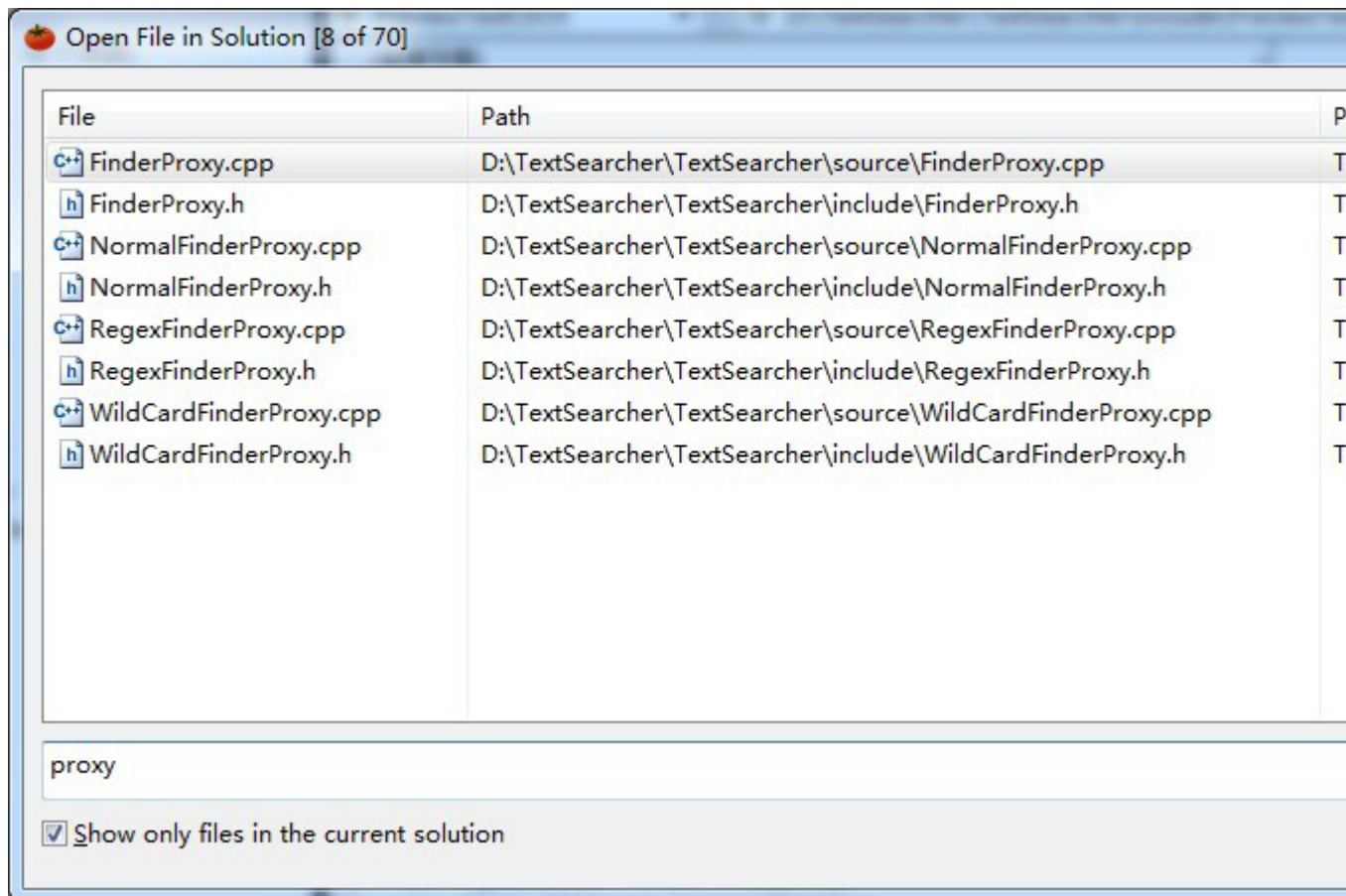
5.编译生成调试相关

F5 调试 Ctrl+F5 直接运行 F7 生成解决方案即生成所有项目 Ctrl+Alt+F7 强行重新生成所有项目

F9 当前光标处设置断点 F10 逐语句不进入子过程 F11 逐语句并进入子过程

6. VAX 快捷键 Shift+Alt+O 打开文件

这个和上面说的 VAX View 中搜索并打开文件的功能差不多，这个会更友好一些。



7. 其它

编辑器相关的 Ctrl+S 保存 Ctrl + Z 撤销 Ctrl+Y 重做 ...

这一章的内容到此结束了，希望对大家有帮助，另外说一下，以上内容不仅适用于 VC2010，包括 VC2005、VC2008 都是适用的。下一章再见。

系列六：VC2010 常见调试技术

犹豫了好久，最终还是决定开始这一章，因为我不清楚到底有没有必要写这样的一章，是应该在这里说明一些简单的调试方法，还是干脆直接让大家去看《Visual C++ 2005 入门经典》的第 10 章，因为那里已经说出了我们几乎所有的常见调试方法。

另外一点就是这一章也许会是《Visual C++ 2010 入门教程》系列的最后一章了，因为在入门的这方面，我已经找不到值得和大家分享的经验了，算是黔驴技穷了吧。回头看看这个系列，最初的目的就是为了解决一些初学者常见的问题，教会初学者如何使用 VS2010 这个工具，因为我也经历过那些阶段，我希望我能帮助那些“曾经的我”少走弯路。

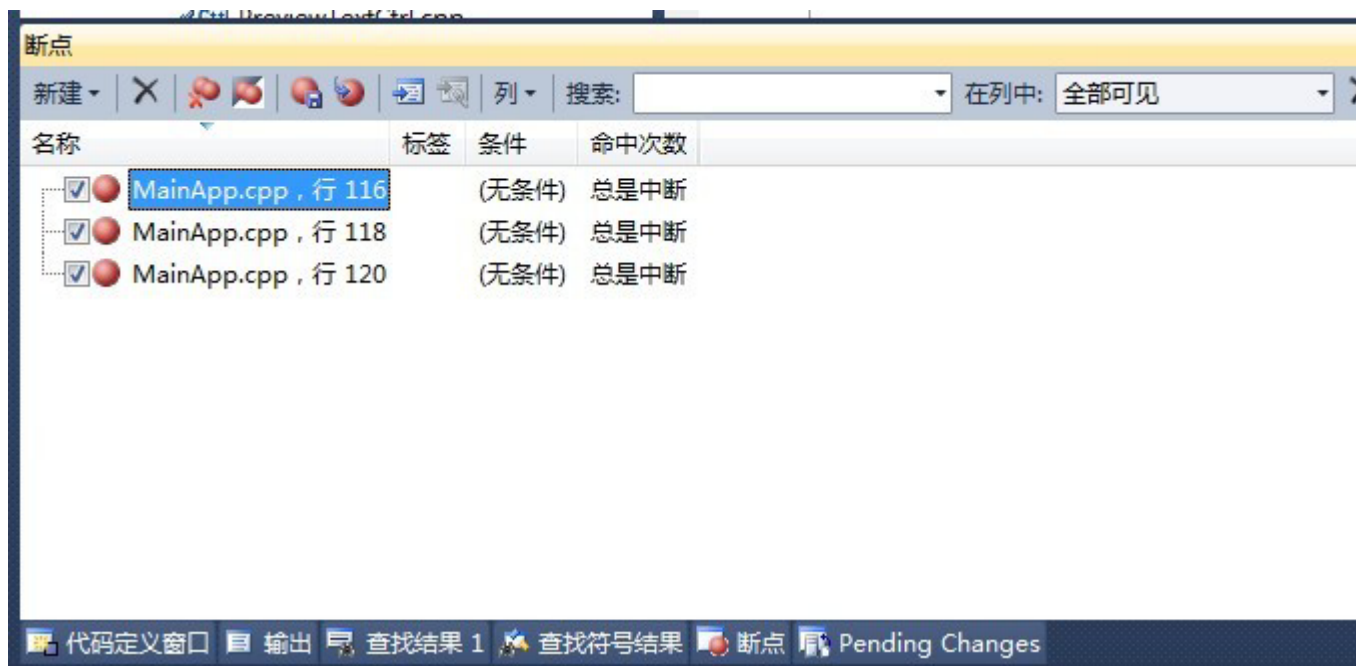
过去，我们讨论了一些诸如 C++ 和 VC 有什么区别、怎么用 C++ 做项目这样的问题，介绍了 SVN 的使用，常见功能的快捷方式，VC 配置等等。接下来就是最后的一些与调试相关的东西与大家分享。另外强烈推荐对基本 VC 调试技术不熟悉的朋友去看看《Visual C++ 2005 入门经典》的第十章。

9.11	练习	528
第 10 章	调试技术	531
10.1	理解调试	531
10.1.1	程序故障	532
10.1.2	常见故障	533
10.2	基本的调试操作	534
10.2.1	设置断点	536
10.2.2	设置跟踪点	538
10.2.3	启动调试模式	538
10.2.4	修改变量的值	542
10.3	添加调试代码	542
10.3.1	使用断言	543
10.3.2	添加自己的调试代码	544
10.4	调试程序	549
10.4.1	调用堆栈	549
10.4.2	单步执行到出错位置	551

断点

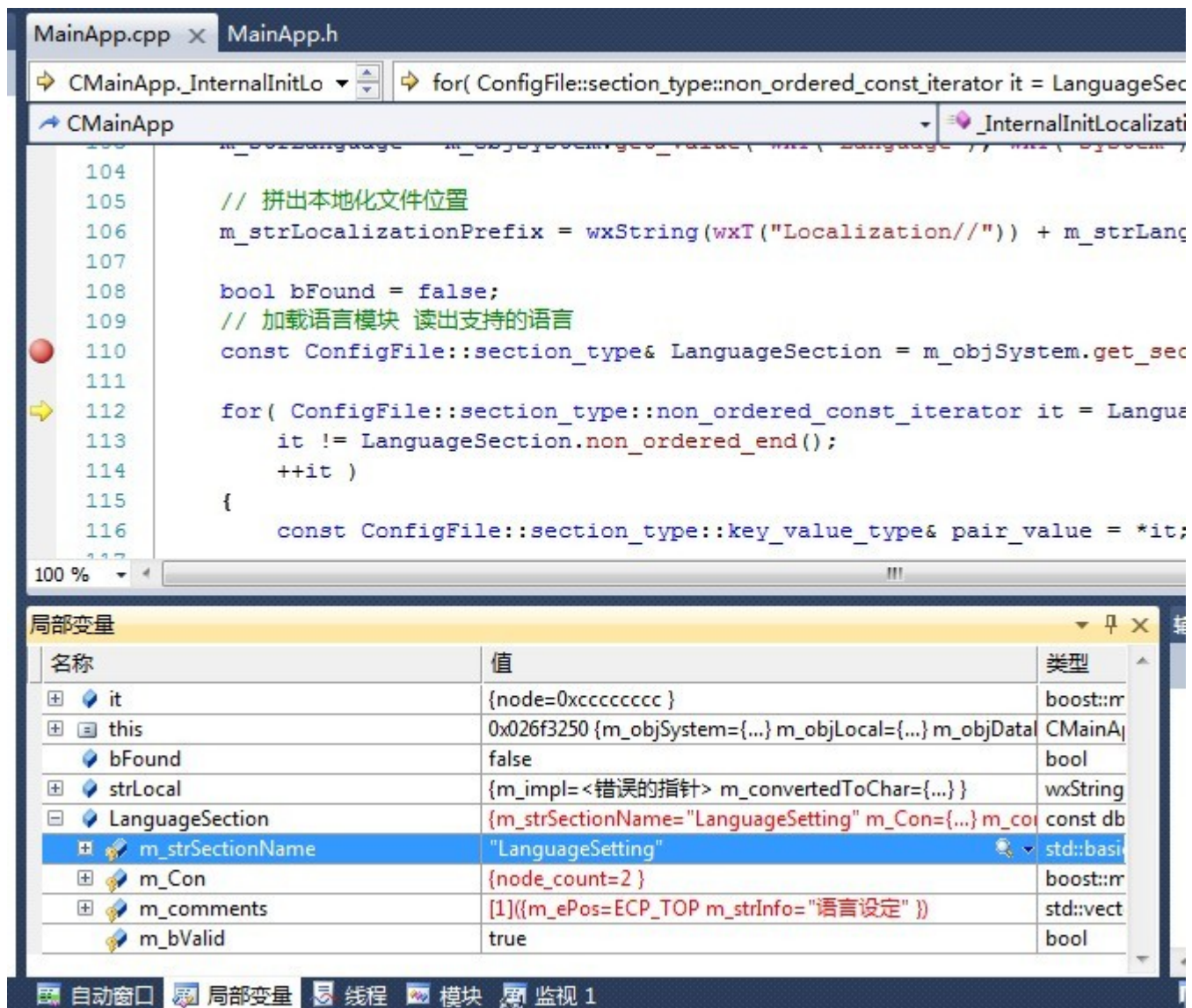
没有比断点更常用的了，通过点击代码左边边栏或者移动光标到指定行按 **F9** 等都可以添加断点。值得注意的是并非每一行都可以添加断点，这个就留给大家去实践中体会吧。

通过菜单《调试》窗口《断点》或者直接按 **Alt+F9** 可以调出断点选项卡，通过选项卡我们可以设置条件断点、数据断点等。



单步和监控

调试中除了 F5 之外,另外更常用的估计是 F10、F11 了,前者是一次一个语句的执行,或者可以看出一行;而后者如果出现能进入的子过程,那么就会进入子过程。这个请大家找个程序,至少要有函数调用的,当断点触发的时候,请自行体验一下 F10 和 F11 的效果你就明白了。说白了,实践才是最好的老师,我负责告诉你有这么个东西。



两个选项卡，局部变量和自动变量，它们都负责显示一些当前断住状态下的变量的值，注意，这些只有在程序中断的时候才有意义。自动变量选项卡并非指 **auto** 变量，而是指 VS 帮我们猜想我们可能感兴趣的一些变量的值，或者函数返回值，而局部变量基本上就是本过程的一些变量的值了。

注意，这些选项卡不仅仅可以用于查看，甚至可以用于你临时修改它们的值，方法就是双击值就可以了，如下图所示：

自动窗口		
名称	值	类
dbsoft::config_info<wchar_t,0,0>::get_section 返回	{m_strSectionName="LanguageSetting" }	coi
LanguageSection	{m_strSectionName="LanguageSetting" m_	coi
it	{node=0xcccccccc }	bo
m_objSystem	{m_bValid=true m_strFile="config//system.i	db
dbsoft::config_info<wchar_t,0,0>	{m_container={...} }	db
m_bValid	true	bo
m_strFile	"config//system.ini"	std
this	0x026f3250 {m_objSystem={...} m_objLocal=	CM

这时候你可以把它临时改成 **false** 都行哈。

大家看到监视 1 了吗？这个选项卡是留给用户的，如果前面的变量太多你不想用滚动条滚来滚去的看就可以在这里输入要监视的变量了：

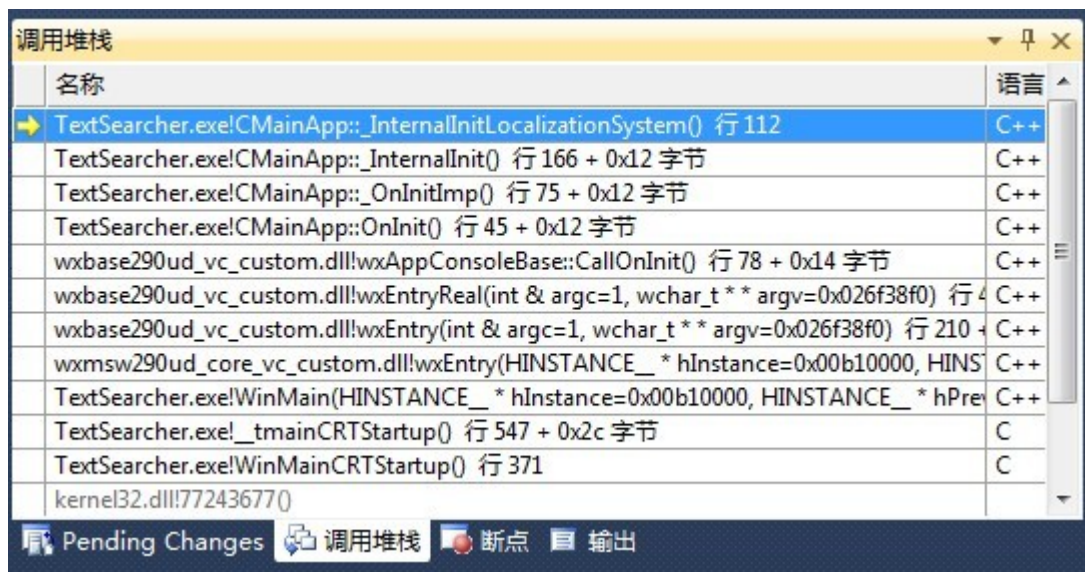
监视 1		
名称	值	类型
m_strLanguage	{m_impl="chs" m_convertedToChar={...} }	wxString
m_impl	"chs"	std::bas...
m_convertedToChar	{m_str=0x00000000 <错误的指针> m_len=3452816845 }	wxString
\$err,hr	S_OK	unsigned

这里我检查了当前语言字符串的设置。注意并非什么变量都可以检视，必须是调试器可以为我们推断出来的才行，即调试器知道它的地址是什么。如上图所示有一个特殊的用法 **\$err,hr** 这个是 VS 特别的，它的意义相当于让调试器帮你获取 **GetLastError** 的值，这在 Windows 编程的时候非常有用。

对于一些指针类型的变量我们还可以在监视里面对它做强制转型，比如你的函数传递一个 **void* p** 进来，但是你知道这次你传递的是一个 **Data** 结构体的指针，而调试器是无法知道这个 **p** 指向的是 **Data**，所以你可以在监视中输入 **(Data*)p**。这样调试器会自动帮我们把他当做 **Data** 结构体的指针来识别。

调用堆栈

调试过程中调用堆栈实在是太重要了，因为它指出了你的程序是正在处于什么状态，是谁调用了谁：

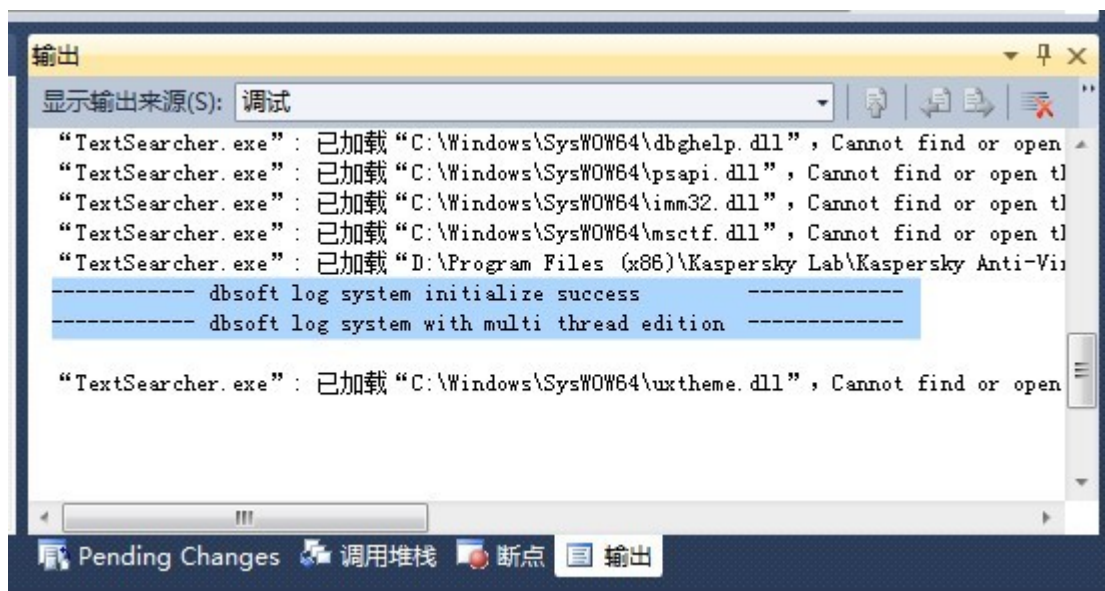


如果你没有这个选项卡可以通过 **Alt+7** 或者调试》窗口》调用堆栈把它调出来。

运行时也可以获取调用堆栈的,这个需要 Windows API 的帮助,[这个请看我写的这个](#)。

日志

日志有很多种,你可以写一个专门的日志系统来处理日常的日志工作,但是这里我只说把信息输出到 VC 的输出窗口,像这样:

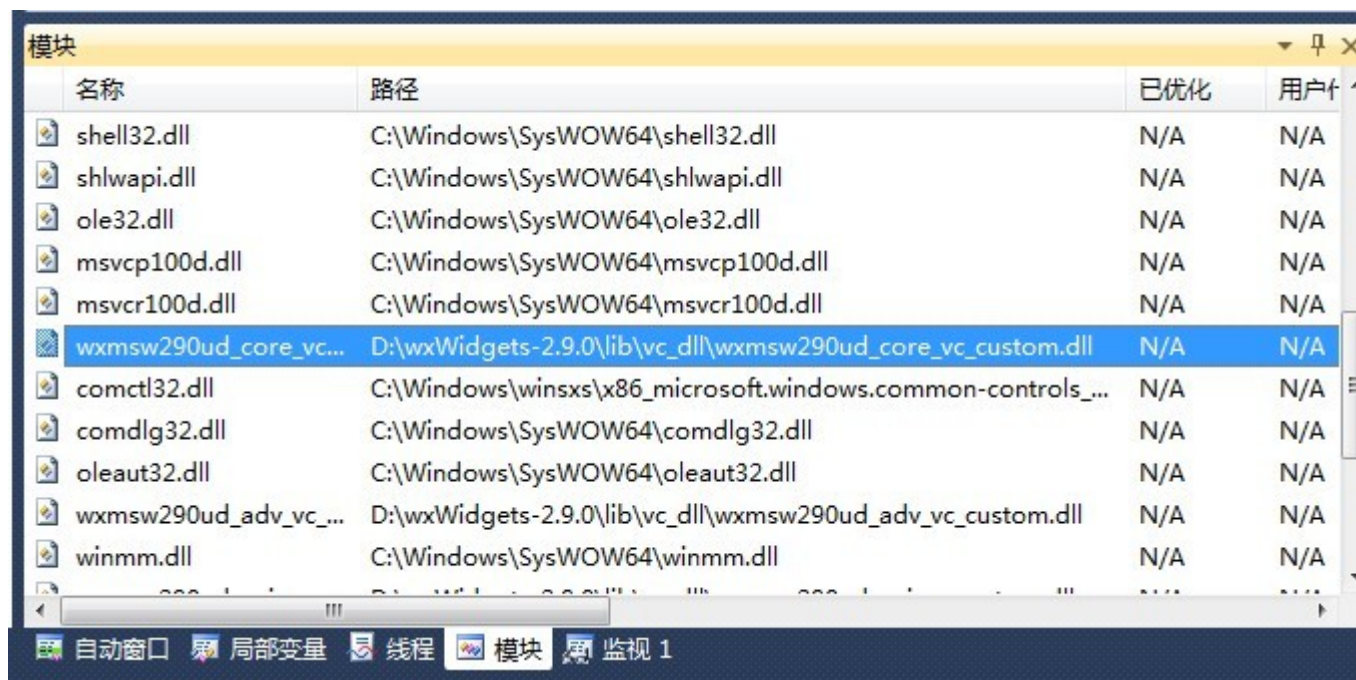


使用 Windows API `OutputDebugString` 来实现,当然你也可以对他做一些封装,在程序中重要的代码部分记录下日志,这对调试很有帮助,你这样会一眼知道哪里出了问题,甚至你可以把调用对战嵌入到这个包装中去:


```
131
132     if( !bFound )
133     {
134         TS_LOG_ERROR( "Error, Can't Get language: ", m_strLanguage.t_str()
135         return false;
136     }
137
138     // 加载本地化文件
139     wxString strLocal = GetLocalizationPrefix() + wxT("Main.") + m_strLangu
140
141     if( !m_objLocal.create_from_file( strLocal.t_str() ) )
142     {
143         TS_LOG_ERROR( "Can't Get Localization File : ", strLocal.t_str() );
144         return false;
145     }
146
```

100 %

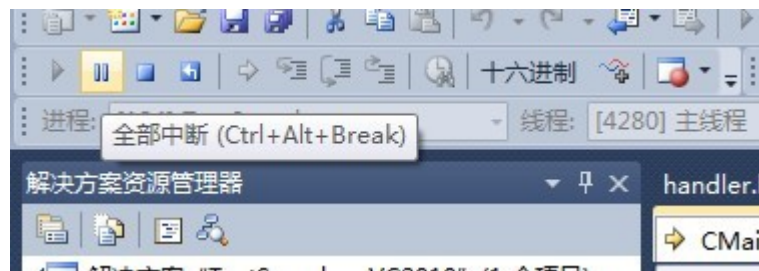
通过模块选项卡发现外部模块错误



假设我们依赖于一个外部库 Test.dll，这个 DLL 在系统目录下面有一个，而在 Path 路径下面还有一个，而他们的版本不同，甚至只是名字相同而内容完全不同。或者其他一系列的类似的问题，都可以通过模块选项卡来察觉，另外这个模块选项卡还告诉了我们我们依赖了那些外部 DLL，这在发布的时候很有用，使得我们可以漏掉需要的 DLL。

通过暂停按钮发觉死锁和死循环

当我们的程序失去响应的时候我们不妨尝试点击调试窗口上面的暂停按钮：



如果中断（暂停）成功那么我们会看到死锁或者死循环的调用堆栈了。

断言(assert)

assert 大家应该很熟悉了吧，这是最直接提供错误信息的方法了。特别的，当我们在调试的时候，调试器会帮助我们定位到断言触发的地方。

暂时就想到这么多，如果您还有其它好 **Case**，一定不要忘了要同大家分享。

转眼间毕业一年了，感触良多。每当我情绪低落的时候，我就看苏珊大妈的视频，因为苏珊大妈是我的偶像，因为她告诉我有梦想就一定要坚持，当你具备了成功的基础的时候，梦想就会慢慢的实现。