IDE，Integrated Development Environment, 集成开发环境

Char\*, string, CString: char\*是一个指向字符的指针，是一个内置类型，可以指向一个字符，也可以表示字符数组的首地址; string是C++标准库（STL）中的类型，他是定义的一个类，定义在<string>头文件中，包含了对字符串的各种常用操作，相比较于char\*，它的优势在于可以动态拓展以及对字符串操作的方便快捷; CString常用于MFC编程之中，属于MFC（Microsoft Foundation Classes， 微软基础类库）的类，比如从对话框中利用GetWindowText得到的字符串就是CString类型，使用CString时要包含afx.h文件

COM（Component Object Model）组件对象模型是microsoft制定的一个组件软件标准，凡是遵循COM标准开发出来的组件成为COM组件，在windows平台上COM的实现形式有DLL（Dynamic Link Library，动态链接库文件）（进程内组件）和EXE（executable program，可执行程序）（进程外组件）两种

ATL（Active Template Library，活动模板库），常用于开发COM程序和ActiveX程序。ATL 可以说是把 COM 封装了一下，象 MFC 一样做成一个库，并有一个向导，使COM应用开发尽可能地自动化、可视化，这就决定了 ATL 只面向 COM 开发提供支持。

ATL 因其采用了特定的基本实现技术，摆脱了大量冗余代码，使用 ATL 开发出来的 COM 应用的代码简练高效。 在 ATL 产生以前，开发 COM 组件的方法主要有两种：一是使用 COM SDK ( COM 软件开发包)直接开发 COM 组件，另一种方式是通过 MFC 提供的 COM 支持来实现。而现在 ATL 已经成为 Microsoft 支持 COM 应用开发的主要开发工具。

MFC（Microsoft Foundation Classes，微软基础类库），以C++类的形式封装了Windows的API（Application Programming Interface，应用程序编程接口），并且包含了一个应用程序的框架以减少应用程序开发人员的工作量。

静态库：会将编译的目标文件.obj与引用到的库.lib一起链接打包到可执行文件exe（也称为目标代码中），程序运行时将不再需要该静态库，因此最终链接成的可执行文件（.exe）体积较大，在windows中一般以.lib为后缀名，在Linux中一般以.a为后缀名。

动态库：在链接阶段，动态库.dll并没有真正被连接到目标代码中，只是将这个动态库的声明链接到目标代码中，动态库.dll依然是独立存在的，只有在程序运行时才会将.dll载入到内存中被程序调用，程序运行时必须要有这个动态库并且放在正确的路径之中。因此最终链接成的可执行文件（.exe）体积较小。在windows中一般以.dll为后缀名。

编译静态库：要编译成静态库，我们可以这样设置我们的工程： 右键工程->Properties->

Target Extension& Configuration Type，然后右键Build就可以了，你可以在解决方案下的Debug(实际的情况中一般要编译成Release版本，设置的方法一样，这里的内容后一章中再讲)目录下就能看到Utils.lib，这就是编译出的库。要将这个库给别人使用，只要提供这个Utils.lib和这个工程的头文件就可以。将Utils.h拷贝到D:\ReleaseLibs\StaticLib\Includes，将Utils.lib拷贝到D:\ReleaseLibs\StaticLib\Libs，把D:\ReleaseLibs\StaticLib这个文件提供出去就可以了。

编译动态库：要编译成动态库，我们可以这样设置我们的工程： 右键工程->Properties，然后右键Build就可以了，你可以在解决方案下的Debug(实际的情况中一般要编译成Release版本，设置的方法一样，这里的内容后一章中再讲)目录下就能看到Utils.dll和Utils.lib，这就是编译出的库。要将这个库给别人使用，只要提供这个Utils.dll、Utils.lib和这个工程的头文件就可以。将Utils.h拷贝到D:\ReleaseLibs\DynamicLib\Includes，将Utils.dll和Utils.lib拷贝到D:\ReleaseLibs\DynamicLib\Libs，把D:\ReleaseLibs\DynamicLib这个文件提供出去就可以了。

Debug通常称为调试版本，它包含调试信息，并且不做任何优化，便于程序员调试程序

Release 称为发布版本，它往往是进行了各种优化，使得程序在代码大小和运行速度上都是最优的，以使编译出的可执行文件或库的性能最高，以便用户很好地使用。

Debug 和 Release 本质上就是两组不同的编译选择配制，各自的默认设置一般不用更改。它们最大的区别在于 Release 进行了优化，去掉了调试信息，因此 Release 常用无法进行正确的调试；而 Debug 没有进行优化，包含了调试信息，以方便程序员进行调试。

每一个源文件的第一行代码必须包含预编译头文件。如果你的工程选用了预编译头文件的方式，每一个.cpp 文件的第一行代码必须包含预编译头文件(#include “stdafx.h”)，否则会编译出错。

在 Windows 程序开发时，经常要在各个文件中包含 windows.h、afx.h 等标准头文件，而这些文件非常的大，在编译时就非常的慢，非常耗时。为解决这个问题，已是就有了预编译头文件的技术。

所谓头文件预编译技术，就是把一个工程 ( Project ) 中常用的一些头文件(如标准头文件 Windows.h、Afxwin.h 等，也可以是自己定义的头文件)包含在 stdafx.h 中，并对 stdafx.h 预先编译(在所有的.cpp 文件编译之前进行编译)，得到编译结果 .pch 文件(默认名称为 ProjectName.pch )，后期该工程在编译其它.cpp 文件时不再编译 stdafx.h 中的内容(即使 include 了它)，仅仅使用预编译的结果。 其中 stdafx.h 叫做预编译头文件，stdafx 名称的英文全称为：Standard Application Framework Extensions，当然你也可以自己定义预编译头文件的名称，手动重命名 stdafx.h，同时将上面图 2 和图 3 中对应的名称也得改过来。ProjectName.pch 叫做预编译头。

你编写的任何.cpp 文件都必须首先包含 stdafx.h 。

如果你有工程文件里的大多数.cpp 文件需要的.h 文件，顺便将它们加在 stdafx.h (后部)上，然后预编译 stdafx.cpp。

由于.pch 文件具有大量的符号信息，它是你的工程文件里最大的文件。

Runtime Library运行时库，Runtime Library 就是运行时库，也简称 CRT(C Run Time Library)

我们知道 C++ 基本数据类型中表示字符的有两种：char、wchar\_t。 char 叫多字节字符，一个 char 占一个字节，之所以叫多字节字符是因为它表示一个字时可能是一个字节也可能是多个字节。一个英文字符(如’s’)用一个 char (一个字节)表示，一个中文汉字(如’中’)用 3 个 char (三个字节)表示

字符数组可以表示一个字符串，但它是一个定长的字符串，我们在使用之前必须知道这个数组的长度。为方便字符串的操作，STL 为我们定义好了字符串的类 string 和 wstring 。大家对 string 肯定不陌生，但 wstring 可能就用的少了。

string 是普通的多字节版本，是基于 char 的，对 char 数组进行的一种封装。

wstring 是 Unicode 版本，是基于 wchar\_t 的，对 wchar\_t 数组进行的一种封装。

字符集（Charcater Set 或 Charset）：是一个系统支持的所有抽象字符的集合，也就是一系列字符的集合。字符是各种文字和符号的总称，包括各国家文字、标点符号、图形符号、数字等。常见的字符集有： ASCII 字符集、GB2312 字符集(主要用于处理中文汉字)、GBK 字符集(主要用于处理中文汉字)、Unicode 字符集等

字符编码（Character Encoding）：是一套法则，使用该法则能够对自然语言的字符的一个字符集（如字母表或音节表），与计算机能识别的二进制数字进行配对。即它能在符号集合与数字系统之间建立对应关系，是信息处理的一项基本技术。通常人们用符号集合（一般情况下就是文字）来表达信息，而计算机的信息处理系统则是以二进制的数字来存储和处理信息的。字符编码就是将符号转换为计算机能识别的二进制编码。

一般一个字符集等同于一个编码方式，ANSI 体系( ANSI 是一种字符代码，为使计算机支持更多语言，通常使用 0x80~0xFF 范围的 2 个字节来表示 1 个字符)的字符集如 ASCII、ISO 8859-1、GB2312、 GBK 等等都是如此。一般我们说一种编码都是针对某一特定的字符集。

一个字符集上也可以有多种编码方式，例如 UCS 字符集(也是 Unicode 使用的字符集)上有 UTF-8、UTF-16、UTF-32 等编码方式。