方法一，也是最直观的方法，就是用DllImport函数将DLL中的方法导入为C#的方法，语法如下

[DllImport("CppDll.dll", EntryPoint = "? GetFoo@@YAXXZ", CharSet = CharSet.Ansi)]  
 extern static void GetFoo(ref Foo foo);

其中，第一个字符串参数的为DLL名称（含路径），EntryPoint为函数入口名称，CharSet为字符集，该函数具体的用法可以参考另一篇笔记《C# DllImport的用法》

dll的EntryPoint获取方法为：

1. 进入VS开发环境，然后Tools -> Visual studio 2010 Command Prompt

2. cd到d:\下

3. 输入命令d:\dumpbin /exports a.dll回车即可看到a.dll的所有导出函数

如果要重定向输出至b.txt文本文件，则命令格式如下：d:\dumpbin /exports a.dll /out:b.txt

结果文件格式类似于

 ordinal hint RVA      name

          1    0 00014C40 ?OpenDQ@@YAXXZ

          2    1 00014C70 ?StartCalc@@YAPAXHNHHHH@Z

          3    2 00012130 ?CloseDQ@@YAXXZ

          4    3 000121A0 ??0?$basic\_string@DU?$char\_traits@D@std@@V?$allocator@D@2@@std@@QAE@$$QAV01@ABV?$allocator@D@1@@Z

在结果文件中，第三列即为EntryPoint名称，

DllImport方法导入的函数，必须为 extern static类型，可以在C#中调用，但必须知道该函数的参数表以及返回值，才能正常使用。

改进

上述方法还有一个不方便的地方，就是DLL名称必须包含路径，而且不能使用变量进行表达，只能使用字符串表达完全路径，比如，

[DllImport(@"C:\dll\CppDll.dll", EntryPoint = "? GetFoo@@YAXXZ", CharSet = CharSet.Ansi)] 是合法的

[DllImport(Application.StartupPath+ "\\CppDll.dll", EntryPoint = "? GetFoo@@YAXXZ", CharSet = CharSet.Ansi)]则是不可以的

这样就给使用造成了较大不便，

可以采用以下方法来进行导入，导入后，变成C#的委托变量，

首先使用这个类，

public class DllInvoke

{

[DllImport("kernel32.dll")]

private extern static IntPtr LoadLibrary(String path);

[DllImport("kernel32.dll")]

private extern static IntPtr GetProcAddress(IntPtr lib, String funcName);

[DllImport("kernel32.dll")]

private extern static bool FreeLibrary(IntPtr lib);

private IntPtr hLib;

public DllInvoke(String DLLPath)

{

hLib = LoadLibrary(DLLPath);

}

~DllInvoke()

{

FreeLibrary(hLib);

}

//将要执行的函数转换为委托

public Delegate Invoke(String APIName, Type t)

{

IntPtr api = GetProcAddress(hLib, APIName);

return (Delegate)Marshal.GetDelegateForFunctionPointer(api, t);

}

}

然后，先申明一个与导入函数格式相同的委托，使用上面类中的，Invoke函数，即可实现导入

public class ImportDll

{

private DllInvoke tmp = new DllInvoke(Application.StartupPath + "\\cppDll.dll");

public delegate void OpenList();

public OpenList tmpdp = null;

public delegate IntPtr StartCal(int id);

public StartCal tmpStartCal = null;

public ImportDll()

{

tmpdp = (OpenList)tmp.Invoke("?OpenList@@YAXXZ", typeof(OpenList));

tmpStartCal = (StartCal)tmp.Invoke("?StartCal@@YAPAXHNHHHH@Z", typeof(StartCal));

}

}

这样就解决了DLL路径使用不方便的问题

**不足**

这种方法的不足之处在于，如果C++代码比较复杂，例如使用了较多的类、对象，或者在函数参数及返回值中使用了C++中定义的类，那么这种方法就不够灵活，甚至于很难实现目标。可以参考本系统第二篇文章中介绍的方法。