由于C#编写的是一种托管代码，编译生成微软中间语言，而C++代码则编译生成本地机器码，这两种语言进行混合编程就存在一定困难。一般比较常用的方法是使用DllImport的方法，这种方法在网上有很多介绍，这里就不详细叙述了。但是用过这种方法的人都知道这种方法对于导出函数还可以但是却没法导出C++类！非常的要命。

最近我在做一个项目的时候就遇到了需要在C#代码中使用C++类的问题，在对收集的资料学习中还真找到了一种方法能够导出C++本地代码的类给C#调用。现在就做个学习总结，也是为了给遇到相似问题的同道们提供一个参考吧。

以上都说过C++直接生成本地机器码，C#是生成微软中间语言的托管代码，想要直接调用是不可能的。事实上，C系列中还存在一种语言叫做托管C++，这种语言语法上和C++几乎一样，但是却和C#一样编译成为微软中间语言，这样就可以和C#良好地通信，即可以在c#中使用托管C++类。另外，托管C++还有及其重要的两个特性就是：可以调用本地C++的类和函数！一个托管C++的程序集可以嵌套本地C++编译的机器码！好强大的混合体。所以我们的技术路径也就明晰了：C#以托管C++为中介调用本地C++的类和函数。换句话说也就是用托管C++给本地C++代码做一个外壳供C#调用。

我们的例子是将一个将一个函数签名为int Add(int a,int b)的C函数和一个C++本地类class CClassNative导出，最终在C#中使用。

CClassNative类的定义如下：

class CClassNative

{

public:

CClassNative(void);

~CClassNative(void);

int menber;//成员

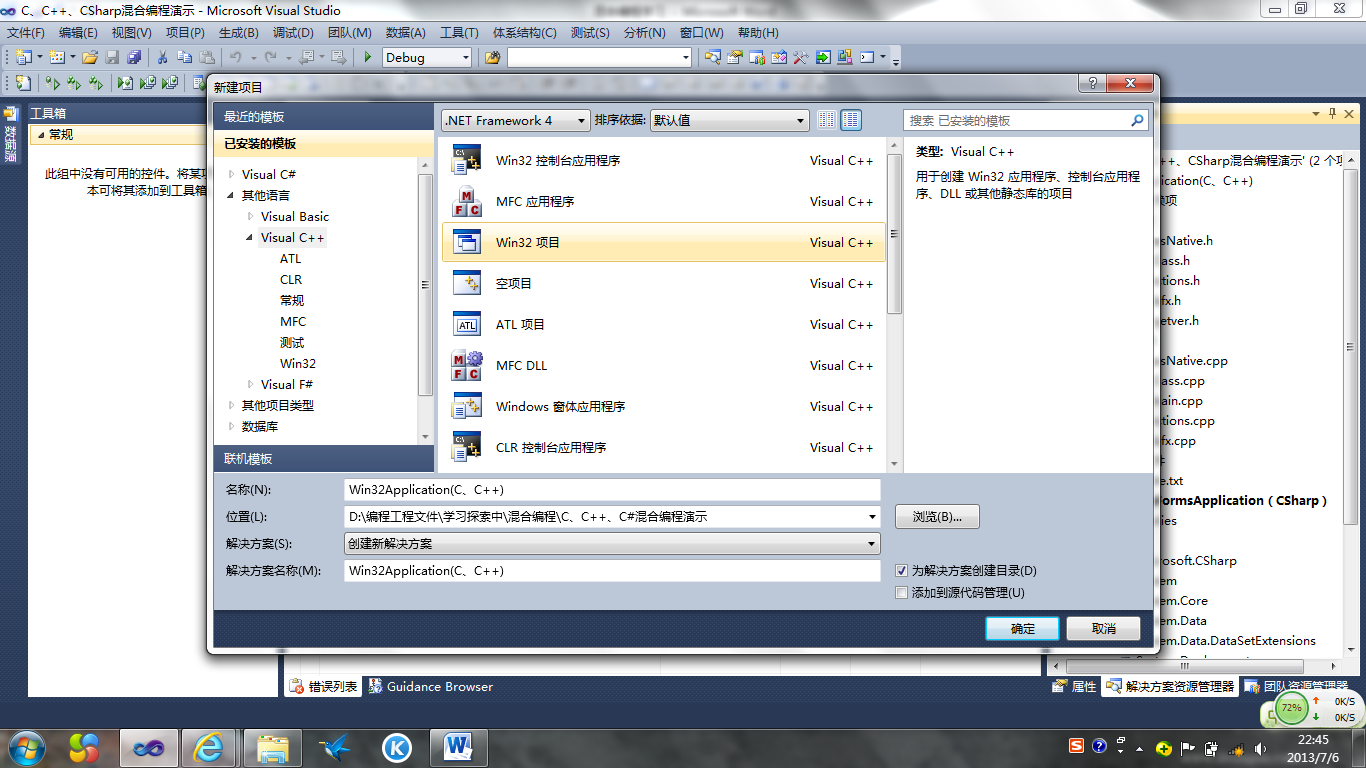
int menderFuncSub(int a,int b);//成员函数

};

为实现这个目的我们建立一个工程来演示。

**一．准备用于导出的类和函数。**

首先，建立一个Win32的C++工程，项目为“Win32Application(C、C++)”，解决方案为“C、C++、CSharp混合编程演示”。



再添加Functions.h文件和Functions.cpp文件，在Functions.h中定义方法 int Add(int a,int b);在Functions.cpp编写实现Add(int a,int b)：

代码：

***Functions.h***文件：

//这个函数是用于导出的C函数，实现算术加法

int Add(int a,int b);

***Functions.h***文件：

#include "stdafx.h"

#include "Functions.h"

//这是C函数实现的算术加法

int Add(int a,int b)

{

return a+b;

}

再新建C++本地代码类CClassNative

在***ClassNative.h***中定义类：

#pragma once

class CClassNative

{

public:

CClassNative(void);

~CClassNative(void);

int menber;//用于导出的成员

int menderFuncSub(int a,int b);// 用于导出的成员函数，实现算术减法

};

在***ClassNative.cpp***中编写代码：

#include "StdAfx.h"

#include "ClassNative.h"

CClassNative::CClassNative(void)

{

//构造

menber=1;

}

CClassNative::~CClassNative(void)

{

}

//这是本地C++类实现的算术减法

int CClassNative::menderFuncSub(int a,int b)

{

return a-b;

}

这样非常简单的本地C++类和函数代码就准备好了，现在我们就需要使用托管C++对这些类和函数进行包装了。

**二．使用托管C++包装本地C++类和C函数。**

为了使用托管C++，需要改编译选项：单击“Win32Application(C、C++) ”项目，在快捷方式里选“属性”，将“配置属性”->“常规”->“公共语言运行时支持”选择为“公共语言运行时支持（/clr）”。

新建类clrClass

在***clrClass.h***文件里定义：

#pragma once

#include "ClassNative.h"

//这是个托管C++类用于对C++本地代码类和函数等进行封装以在C#中使用

public ref class clrClass //必须声明为 **public**，否则类在程序集中不可见，关键词 **ref** 表示类是一个托管类，将编译为中间语言

{

public:

clrClass(void);

int menber;//这个成员访问CClassNative类的公共成员（**事实上只需要包装公共成员和公共成员函数，私有的包装没有意义，也包装不了**）

int menderFuncSub(int a,int b);//这个成员函数包装CClassNative类的公共成员函数

int menberFuncAdd(int a,int b);//这成员函数包装C函数 int Add(int a,int b)

private:

CClassNative \* classNative;//建立一个本地类实例 (在构造函数中实例化，这里可以理解为用于在clrClass 中“继承”CClassNative的公共成员)

};

在***clrClass.cpp***文件里编写：

#include "StdAfx.h"

#include "clrClass.h"

#include "Functions.h"

clrClass::clrClass(void)

{

classNative=new CClassNative();//**这里一定注意要创建对象！** menber=classNative->menber;//这里是简单举例，实际上应该用属性方法来读写CClassNative类的成员，同C#相似，托管C++中有属性函数，用法请自己查

}

//通过调用CClassNative类实现算术减法

int clrClass:: menderFuncSub(int a,int b)

{

return classNative->menderFuncSub(a, b);

}

//通过调用C函数实现算术加法

int clrClass:: menberFuncAdd(int a,int b)

{

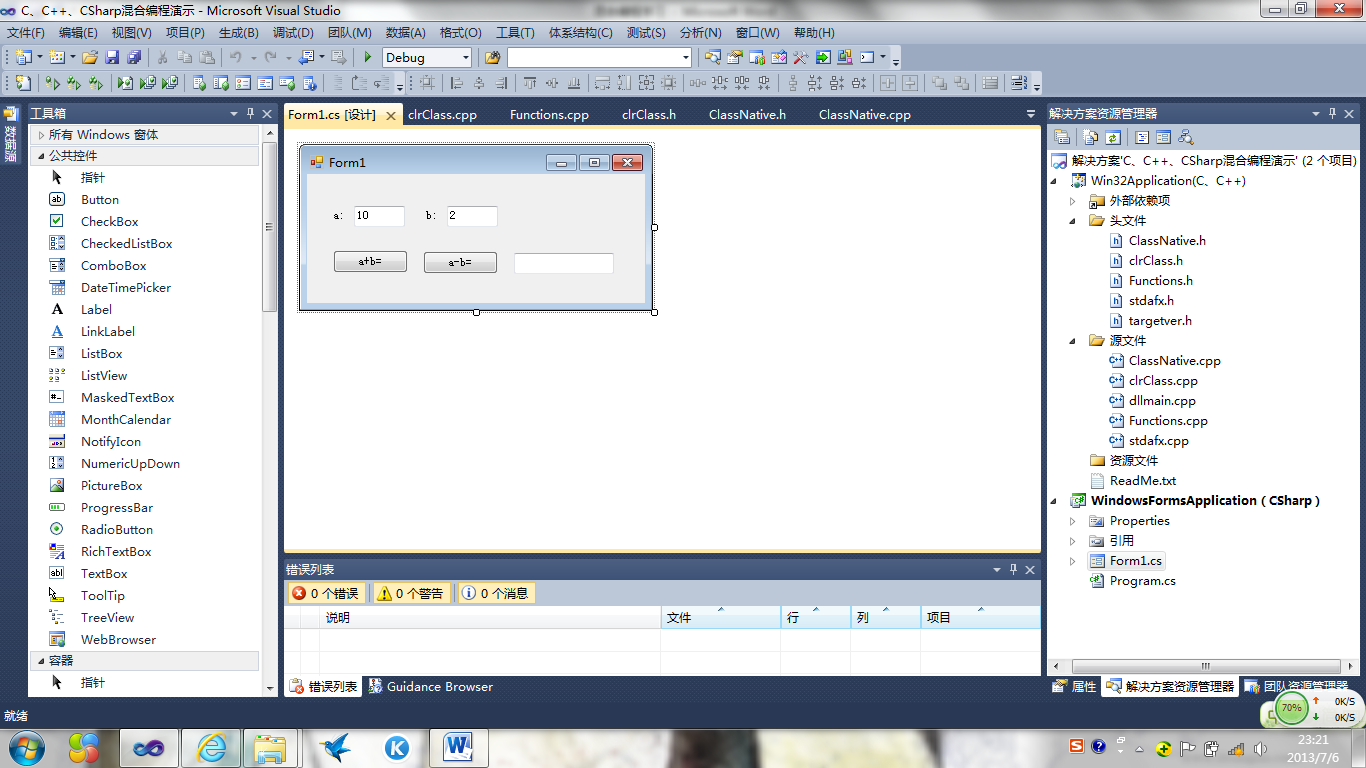
return Add(a,b);

}

这样就实现了clrClass对CClassNative类和C函数int Add(int a,int b)的托管包装。生成的“Win32Application(C、C++).dll”就可以直接在C#中用了。

**三．在C#中包装好的本地C++类和C函数。**

为了试验效果，我们在这个解决方案里建立了一个C#项目“WindowsFormsApplication（CSharp）”来看看效果。这个项目是一个“Windows窗体应用程序”，目的是调用C函数int Add(int a,int b)进行加法运算，CClassNative的int menderFuncSub(int a,int b)成员函数进行减法运算，界面设计如下。



在项目“WindowsFormsApplication（CSharp）”中的“引用”添加对“Win32Application(C、C++).dll”的引用。

由于C#估计大家都比较熟悉了，就只列出创建clrClass类对象和图中两个按钮的事件处理函数，如下：

/// <summary>

/// 这是使用托管C++对C函数和C++类建立的包装，本质上是一个托管类的对象

/// </summary>

clrClass ClrClass = new clrClass();

/// <summary>

/// 加法

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_aADDb\_Click(object sender, EventArgs e)

{

int a = Convert.ToInt32(textBox\_a.Text);

int b = Convert.ToInt32(textBox\_b.Text);

//通过托管C++调用C函数int Add(int a,int b)实现的加法

textBox\_answer.Text = ClrClass.menberFuncAdd(a, b).ToString();

}

/// <summary>

/// 减法

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_aSUBb\_Click(object sender, EventArgs e)

{

int a = Convert.ToInt32(textBox\_a.Text);

int b = Convert.ToInt32(textBox\_b.Text);

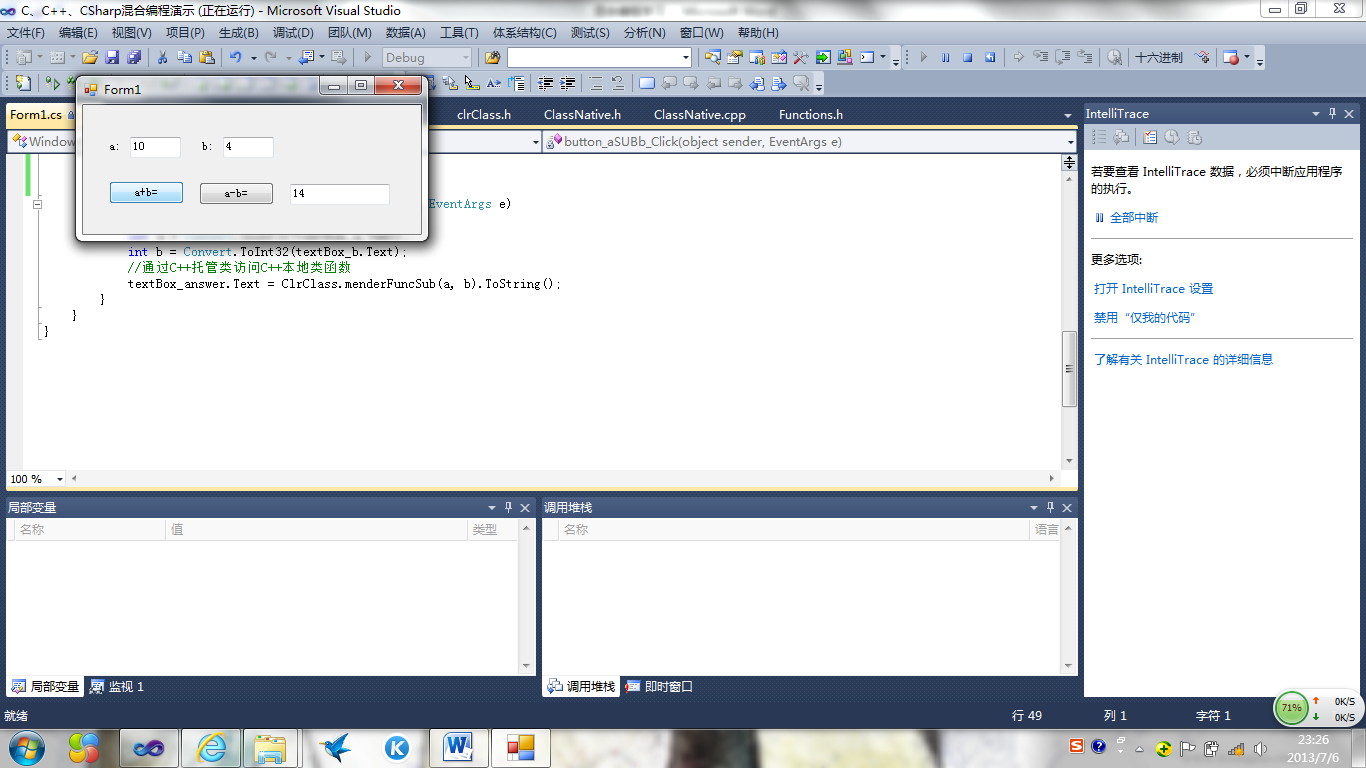
//通过托管C++调用C++的CClassNative类函数实现的减法

textBox\_answer.Text = ClrClass.menderFuncSub(a, b).ToString();

}

}

这是运行效果：



以上只是一个简单的例子，实际上，用托管C++包装的可以是任何本地C++的代码，包括MFC库（这个本人实验过）。

以上示例的开发环境是VS2010，windows版本是windows7旗舰版。

托管调试助手 "LoaderLock"

Message=托管调试助手 "LoaderLock":“正尝试在 OS 加载程序锁内执行托管代码。不要尝试在 DllMain 或映像初始化函数内运行托管代码，这样做会导致应用程序挂起。”

