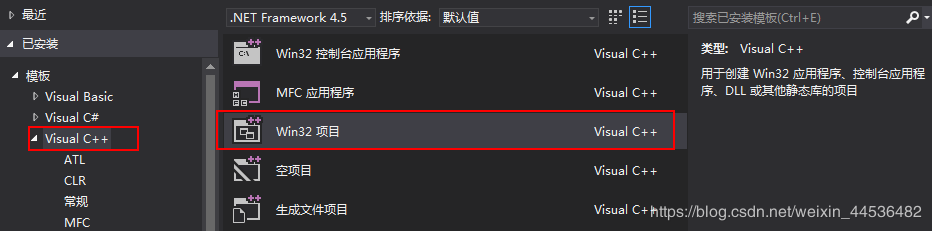
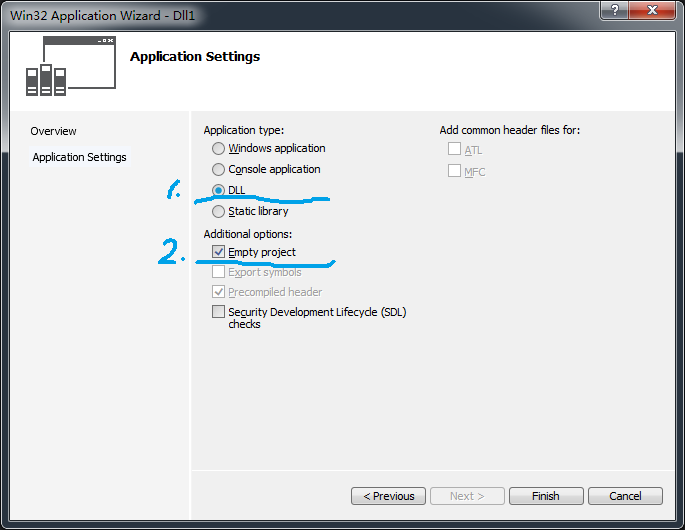
# 1.创建VS DLL工程

（1）、在新建项目时选择"Win32"项目后，点击确定



（2）、在Application Type中选择DLL，在Additional options中选择Empty project，即创建一个空的动态链接库工程。



# 2.创建DLL动态库

dll的创建主要有两种方法：一是使用 \_\_declspec(dllexport) 创建dll，二是使用模块定义（.def）文件创建dll。

2.1通过\_\_declspec(dllexport)方式生成DLL文件

（1）为工程添加一个C++源文件：Dll1.cpp，假设我要实现的是加法和减法运算，则代码如下：

\_\_declspec(dllexport) int add(int a, int b){

return a + b;

}

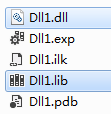
\_\_declspec(dllexport) int subtract(int a, int b){

return a - b;

}

（2）为了让dll导出函数，需要在每一个需要被导出的函数前面加上标识符：\_\_declspec(dllexport)。

（3）利用Build命令生成Dll1动态链接库，这时在Dll1/Debug目录下就会生成.dll文件和.lib文件，这两个文件即为所需的动态链接库的文件。



注意的一点是：应用程序如果想要访问某个dll中的函数，那么这个函数必须是已经被导出的函数。

（4）我们发现导出函数的名称长得很奇怪，add导出函数的名称是“？add@@YAHHH@Z”,subtract导出函数的名称是“？subtrct@@YAHHH@Z”。这是因为在编译链接时，C++会按照自己的规则篡改函数的名称，这一过程称为“名字改编”。这会导致不同的编译器、不同的语言下调用dll发生问题。因此我们希望动态链接库文件在编译时，导出函数的名称不要发生变化。

为了实现这一目的，可以再定义导出函数时加上限定符：extern “C”,如：

extern "C" \_\_declspec(dllexport) int add(int a, int b){

*//...*

}

但是这种方式只能解决C++和C语言之间相互调用时函数命名的问题。为了彻底解决这个问题，可以通过模块定义(.def)文件实现。

　　如果定义了MAKEDLL\_EXPORTS，然后我们再定义MAKEDLL\_API为\_\_declspec(dllexport),否则我们定义\_\_declspec(dllimport)。前者主要为生成DLL的项目所使用，后者为使用DLL的项目所使用。dllexport和dllimport存储类特性是C和C++语言的Microsoft专用扩展。可以使用它们从 DLL 中导出或向其中导入函数、数据和对象。

为什么要使用\_\_declspec(dllexport)与\_\_declspec(dllimport)，及两者的区别：都是DLL内的关键字，即导出与导入。他们是将DLL内部的类与函数以及数据导出与导入时使用的。主要区别在于，dllexport是在这些类、函数以 及数据的申明的时候使用。用过表明这些东西可以被外部函数使用，即（dllexport）是把DLL中的相关代码（类，函数，数据）暴露出来为其他应用程 序使用。使用了（dllexport）关键字，相当于声明了紧接在（dllexport）关键字后面的相关内容是可以为其他程序使用的。而 dllimport关键字是在外部程序需要使用DLL内相关内容时使用的关键字。当一个外部程序要使用DLL内部代码（类，函数，全局变量）时，只需要在 程序内部使用（dllimport）关键字声明需要使用的代码就可以了，即（dllimport）关键字是在外部程序需要使用DLL内部相关内容的时候才使用。（dllimport）作用是把DLL中的相关代码插入到应用程序中。如果没有使用dllexport导出函数，将无法生成lib文件，只能生成dll文件。

2.2. 通过def方式生成DLL（推荐）

（1）def文件创建dll的话就不再需要\_\_declspec(dllexport)，因此将代码写成最原始的样子：

int add(int a, int b){

return a + b;

}

int subtract(int a, int b){

return a - b;

}

（2）同时为工程创建一个后缀名为.def的文件，并添加进工程，编辑其内容为：

LIBRARY Dll1

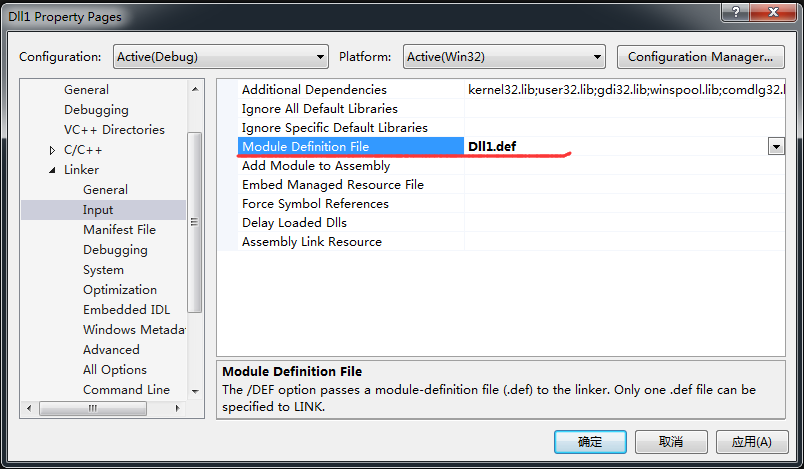
EXPORTS

add

subtract

其中LIBRARY语句用于指定动态链接库的名称，该名称与生成的动态链接库名称一定要匹配。EXPORTS语句用于表明dll将要导出的函数，以及为这些导出函数指定的符号名。

（3）将该模块定义文件链接到工程中，方法为工程属性页面>链接器>输入>模块定义文件中写入“Dll1.def”。



（4）然后重新Build Solution，

# 3.调用DLL

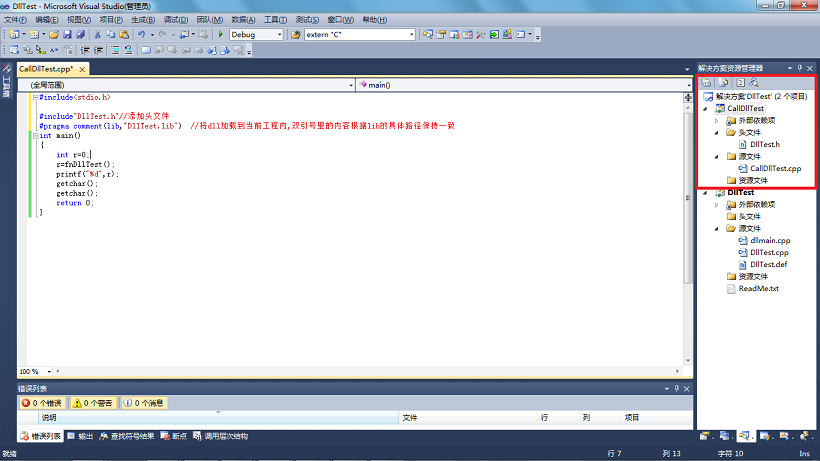
## 3.1. 隐式调用

(1). 格式如下：#pragma comment(lib,"DllTest.lib");

在调用程序内必须写上#include"DllTest.h"

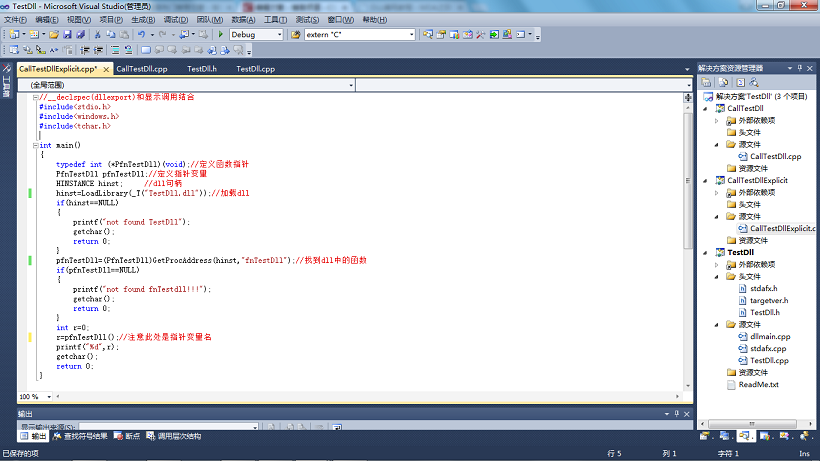
(2). 隐式调用时，调用该dll的程序只能是C或C++

采用隐式调用上面生成的dll文件的代码示例如下图所示：



## 3.2. 显示调用

显示调用需要先用Loadlibrary函数加载dll文件,然后用GetProcAddress找到该dll的地址，采用显示调用dll的示例代码如下图所示：

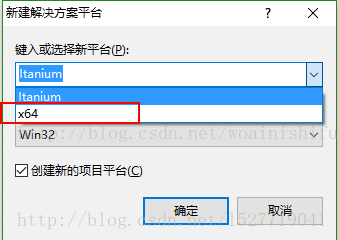


# 4.使用VS生成64位的dll文件

由于现在的jdk一般都是64位版本，所需要调用的dll文件最好应是64位版本的，不然会出现“不是有效的win32应用程序”的错误。

此种情况下，需要根据已有的dll文件的源程序重新生成64位的dll文件。

首先，用vs2010打开之前dll项目，这时候项目的默认是32位的，点击“配置管理器”->”活动解决方案配置”选择debug或者release，“活动解决方案平台”下选择“新建”，在弹出的对话框“键入或选择新平台”下选择“x64”，直接点击“确定”。

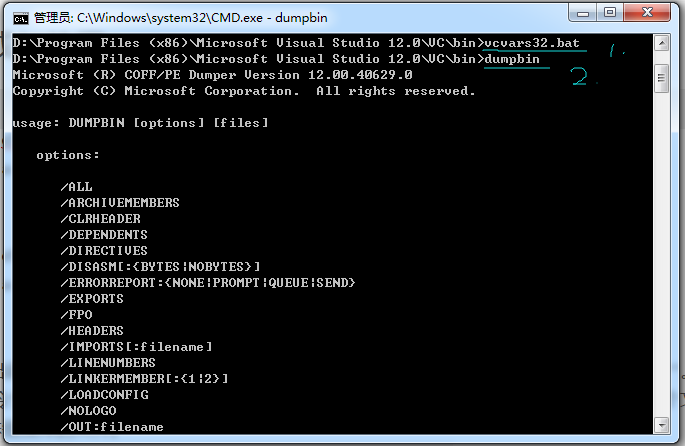


之后重新生成dll文件即可。需要注意生成的dll文件不是在系统目录的Debug或者Release文件夹下，而是在一个x64文件夹下对应的Debug和Release文件夹下面。

# 5. 使用Dumpbin命令确认DLL的导出函数

（1）首先在命令行中进入到VS的安装目录下，运行一个名为VCVARS32.bat的批处理程序（对于VS2013来说，该bat文件位于\VC\bin目录下），该文件的作用是用来创建VC++使用的环境信息。（注意，当在命令行界面执行VCVARS32.bat文件后，该文件设置的环境信息只在当前命令行窗口生效。）

（2）然后输入dumpbin命令，即可列出该命令的使用方法:



（3）那么想要查看一个dll提供的导出函数，在Dll1.dll文件所在目录下，在命令行中输入下述命令：

dumpbin -exports Dll1.dll

