1. **静态库和动态库**

反汇编printf。想一想；printf的代码来自哪里？DLL里面的函数可以不被调用者发现，此外建议被外部调用的函数前面加上 extern “C”。



**创建dll，见项目Dll\_Export**

//下面8行要拷贝进去

#define DATAEXCHANGEDLL\_EXPORTS

#ifdef DATAEXCHANGEDLL\_EXPORTS

#define DATAEXCHANGEDLL\_API \_\_declspec(dllexport)

#else

#define DATAEXCHANGEDLL\_API \_\_declspec(dllimport)

#endif

#define PREFIX extern

//自己加的

extern "C" DATAEXCHANGEDLL\_API int Add(int a,int b);

**静态调用dll，该方式调用和直接编译cpp没有区别。遵循4步：见项目Dll\_Test**

//1、如下必须的1

#include "Dll\_Core.h"

//2、这个建议操作2

#pragma comment(lib,"Dll\_Export.lib")

//3、拷贝编译好的头文件，lib到该项目目录

//4、拷贝dll到能找见的目录

//5、调用dll中被export的函数。如果输出dll中函数名混乱则该dll是c++规范，需要动态调用dll

**动态调用dll，该方式不需要和DLL配套的LIB文件，甚至不需要头文件（你要明白dll函数参数规则）。遵循4步：见项目Dll\_Test**

HINSTANCE h = LoadLibraryA("Dll\_Export.dll");

typedef int (\*FunPtr)(int a, int b);//定义函数指针

if (h == NULL)

{

FreeLibrary(h);

printf("load lib error\n");

}

else

{

FunPtr funPtr = (FunPtr)GetProcAddress(h, "Add");

if (funPtr != NULL)

{

int result = funPtr(3, 3);

printf("3 + 3 = %d \n", result);

}

else

{

printf("get process error\n");

printf("%d", GetLastError());

}

FreeLibrary(h);

}

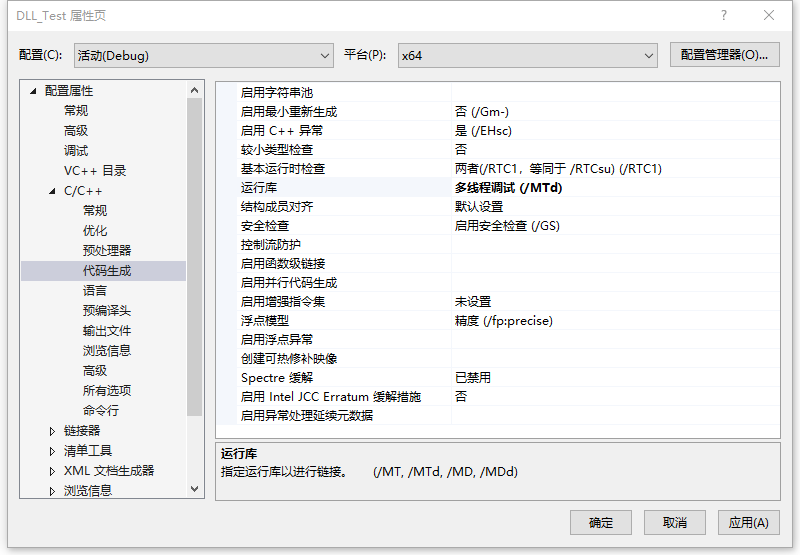
**注意：LIB有两种，一种配合DLL存在，只是函数的描述信息，另外一个是静态库，包含代码。用指令 lib如下（进入native命令行）:**

lib /list C:\Program Files (x86)\Microsoft Visual Studio\2019\Professional\VC\Tools\MSVC\14.16.27023\lib\x64\libcmt.lib 如果有XX.DLL输出那该lib是动态的，如果什么都没有输出这个lib就是静态的。

1. **静态编译和动态编译**

观察exe文件大小，看exe的imports项，有什么区别

3.5日结束：



1. **代码、EXE、加载**

**程序运行流程——链接、装载及执行.docx**

**PE文件格式介绍.docx**

1. **全局和局部变量**

见项目：Variable\_Memory

变量在c文件中共享

