**模板不支持分离编译**

我们来分析一下模板为什么不支持分离编译呢，所谓的分离编译就是我们在编写程序的时候可能会出现如下的一种情况就是，（我下面就是举具体的例子了）

**代码**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*template.h\*\*\*\*\*\*\*\*\*\*\*//

#include<iostream>

using namespace std;

template<class T>

class A

{

public:

void Show();

private:

T a;

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*template.cpp\*\*\*\*\*\*\*\*\*\*\*//

#include"template.h"

template<class T>

void A<T>::Show()

{

T m;

cout << "hello" << endl;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*test.cpp\*\*\*\*\*\*\*\*\*\*\*//

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include"template.h"

int main()

{

A<int> a;

a.Show();

return 0;

}

**现象一**

上面的程序中，我们调用的时候会出现如下的报错

1>test.obj : error LNK2019: 无法解析的外部符号 "public: void \_\_thiscall A<int>::Show(void)" (?Show@?$A@H@@QAEXXZ)，该符号在函数 \_main 中被引用

1>W:\Code\C++\TemptlateSeparateCompilation\Debug\TemptlateSeparateCompilation.exe : fatal error LNK1120: 1 个无法解析的外部命令

**现象二**

我们改一下代码再来看一个现象，如果我们在主函数中把a.Show给注释掉，然后在template.cpp把cout<<"hello"<<endl;改写成cout<<"hello"

在进行编译的时候，发现竟然没有报错，这就奇怪了吧

从预处理，编译，汇编，链接的角度来分析问题

我们知道模板的一个特性，只有当调用的时候才会实例化它，而我们的程序从一开始的文本到最后的可执行程序经历了四个过程，分别是预处理，编译，汇编，链接，最后是在链接的时候出现了错误

**一.预处理**

在预处理的时候，会进行头文件展开，这个时候会把头文件中的函数声明展开到test.cpp和template.cpp中

**二.编译**

在编译test.cpp时由于只能看到模板声明而看不到实现，因此不会实例化模板函数，但此时不会报错，因为编译器认为模板定义在其它文件中，就把问题留给链接程序处理。

在编译template.cpp的时候，编译的时候会进行一些检查，比如一些语法的检查，然后生成汇编代码，因为我们的模板函数是在调用的时候才会进行实例化，即只有在调用模板函数的时候才会生成代码，但是在我们的编译的时候，在template.cpp这个文件中是没有对函数进行实例化，所以并没有生成汇编的代码，这也是上面的第二种情况为什么我们的代码错误从却没有被检查出来的原因，因为我们的根本就没有生成汇编代码

**三.汇编**

再来接着分析，在汇编的时候就是生成二进制代码，然后是一个.obj文件，在linux下面是.o文件，同时生成一个符号表，符号表中放置的就是我们所有函数的地址。

**四.链接.**

当我们进行最后一步链接的时候，执行到a.Show();会去.ob文件中查找我们的函数地址，这里有一个符号表，但是因为我们没有生成一个函数代码，所以这个函数表是查找不到的，所以这个时候就会报上面的错误。

还需要分析的一个问题就是，为什么我们把函数的声明和定义放置在一个文件中的时候就可以呢，因为如果我们把函数声明和定义放置在一个头文件中的话，我们在编译的时候，就会头文件展开了，这个时候调用了我们的函数，然后又是在一个文件内如就会直接直接生成代码，然后在查找的时候就会很容易的查找到这个内容。

**解决办法**

方法一：

使用上面说的内容，放模板函数的声明和定义放在一个头文件下面

但是这样会出现一些问题

（1）放置在头文件中，暴露了模板函数的 内容

（2）不符合分离编译的原则

方法二

我们可以在函数声明的时候加上一个export，就是下面的这种形式

export void Show();

但是很多的编译器是不支持的

方法三

我们可以显示的实例化，就是我们可以在头文件中对我们需要的内容 进行一个显式的实例化