**想必很多人都看过“头文件中的 #ifndef/#define/#endif 防止该头文件被重复引用”。但是是否能理解“被重复引用”是什么意思？是不能在不同的两个文件中使用include来包含这个头文件吗？如果头文件被重复引用了，会产生什么后果？是不是所有的头文件中都要加入#ifndef/#define/#endif 这些代码？**

**其实“被重复引用”是指一个头文件在同一个cpp文件中被include了多次，这种错误常常是由于include嵌套造成的。比如：存在a.h文件#include "c.h"而此时b.cpp文件导入了#include "a.h" 和#include "c.h"此时就会造成c.h重复引用。**

**头文件被重复引用引起的后果：**

**有些头文件重复引用只是增加了编译工作的工作量，不会引起太大的问题，仅仅是编译效率低一些，但是对于大工程而言编译效率低下那将是一件多么痛苦的事情。  
有些头文件重复包含，会引起错误，比如在头文件中定义了全局变量(虽然这种方式不被推荐，但确实是C规范允许的)这种会引起重复定义。**

    是不是所有的头文件中都要加入#ifndef/#define/#endif 这些代码？

    答案：不是一定要加，但是不管怎样，用#ifnde xxx #define xxx #endif或者其他方式避免头文件重复包含，只有好处没有坏处。个人觉得培养一个好的编程习惯是学习编程的一个重要分支。

**下面给一个**#ifndef/#define/#endif的格式：

    #ifndef A\_H意思是"if not define a.h"  如果不存在a.h

    接着的语句应该#define A\_H  就引入a.h

    最后一句应该写#endif   否则不需要引入

**--------------------------------------------------------------------------------------------------  
#ifndef GRAPHICS\_H // 防止graphics.h被重复引用   
#define GRAPHICS\_H   
  
#include <math.h> // 引用标准库的头文件   
…   
#include “header.h” // 引用非标准库的头文件   
…   
void Function1(…); // 全局函数声明   
…   
class Box // 类结构声明   
{   
…   
};   
#endif  
--------------------------------------------------------------------------------------------------**

**#if defined的使用**

#if后面接的是一个宏。

#if defined (x)

...code...

#endif

这个#if defined它不管里面的“x”的逻辑是“真”还是“假”它只管这个程序的前面的宏定义里面有没有定义“x”这个宏，如果定义了x这个宏，那么，编译器会编译中间的…code…否则不直接忽视中间的…code…代码。另外 #if defined(x)也可以取反，也就用 #if !defined(x)

**#ifdef的使用**

#ifdef的使用和#if defined()的用法一致   
#ifndef又和#if !defined()的用法一致。

最后强调两点：   
第一：这几个宏定义只是决定代码块是否被编译！   
第二：别忘了#endif

#if defined还可以串连多个条件：  
#if defined( \_\_MMI\_MYAPP1\_SUPPORT\_\_ ) || defined( \_\_MMI\_MYAPP2\_SUPPORT\_\_ )  
＃i nclude "ConnectivityResDef.h "  
#endif

defined是预处理运算符。   
defined NAME是用来判断NAME是否被定义了（被用define定义了）。   
#ifdef NAME == #if defined(NAME)   
#ifndef NAME == #if !defined(NAME)   
但是在处理双重和多重判断时，defined的优势就显示出来了   
#ifdef NAME1   
#ifdef NAME2   
......   
#endif   
#endif   
可以写成   
#if defined(NAME1) && defined(NAME2)   
.......   
#endif