1：情况1：

#ifdef \_XXXX

...程序段1...

#else

...程序段2...

#endif

这表明如果标识符\_XXXX已被#define命令定义过则对程序段1进行编译；否则对程序段2进行编译。

例：

#define NUM

.............

.............

.............

#ifdef NUM

printf("之前NUM有过定义啦！:) \n");

#else

printf("之前NUM没有过定义！:( \n");

#endif

}

如果程序开头有#define NUM这行，即NUM有定义，碰到下面#ifdef NUM的时候，当然执行第一个printf。否则第二个printf将被执行。

我认为，用这种，可以很方便的开启/关闭整个程序的某项特定功能。

2:情况2：

#ifndef \_XXXX

...程序段1...

#else

...程序段2...

#endif

这里使用了#ifndef，表示的是if not def。当然是和#ifdef相反的状况（如果没有定义了标识符\_XXXX，那么执行程序段1，否则执行程序段2)。

3：情况3：

#if 常量

...程序段1...

#else

...程序段2...

#endif

这里表示，如果常量为真（非0，随便什么数字，只要不是0），就执行程序段1，否则执行程序段2。

我认为，这种方法可以将测试代码加进来。当需要开启测试的时候，只要将常量变1就好了。而不要测试的时候，只要将常量变0。

# ifdef #ifndef 等用法

文件中的#ifndef

头件的中的#ifndef，这是一个很关键的东西。比如你有两个C文件，这两个C文件都include了同一个头文件。而编译时，这两个C文件要一同编译成一个可运行文件，于是问题来了，大量的声明冲突。

还是把头文件的内容都放在#ifndef和#endif中吧。不管你的头文件会不会被多个文件引用，你都要加上这个。一般格式是这样的：

#ifndef <标识>

#define <标识>

......

......

#endif

<标识>在理论上来说可以是自由命名的，但每个头文件的这个“标识”都应该是唯一的。标识的命名规则一般是头文件名全大写，前后加下划线，并把文件名中的“.”也变成下划线，如：stdio.h

#ifndef \_STDIO\_H\_

#define \_STDIO\_H\_

......

#endif

2.在#ifndef中定义变量出现的问题（一般不定义在#ifndef中）。

#ifndef AAA

#define AAA

...

int i;

...

#endif

里面有一个变量定义

在vc中链接时就出现了i重复定义的错误，而在c中成功编译。

结论：

(1).当你第一个使用这个头的.cpp文件生成.obj的时候，int i 在里面定义了当另外一个使用这个的.cpp再次[单独]生成.obj的时候，int i 又被定义然后两个obj被另外一个.cpp也include 这个头的，连接在一起，就会出现重复定义.

(2).把源程序文件扩展名改成.c后，VC按照C语言的语法对源程序进行编译，而不是C++。在c语言中，若是遇到多个int i，则自动认为其中一个是定义，其他的是声明。

(3).C语言和C++语言连接结果不同，可能（猜测）时在进行编译的时候，C++语言将全局

变量默认为强符号，所以连接出错。C语言则依照是否初始化进行强弱的判断的。（参考）

解决方法：

(1).把源程序文件扩展名改成.c。

(2).推荐解决方案：

.h中只声明 extern int i;在.cpp中定义

<x.h>

#ifndef \_\_X\_H\_\_

#define \_\_X\_H\_\_

extern int i;

#endif //\_\_X\_H\_\_

<x.c>

int i;

注意问题：

(1).变量一般不要定义在.h文件中。

ifndef/define/endif的用法与实例分析

用法：

.h文件，如下：

#ifndef XX\_H

#define XX\_H

……

#endif

这样如果有两个地方都包含这个头文件，就不会出现两次包含的情况，因为在第二次包含时XX\_H已经有定义了，所以就不再 include了。

-----------------------------------------------------------------------------------------------------------------------------------

#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 “myheader.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

-----------------------------------------------------------------------------------------------------------------------------------

假设你的工程里面有4个文件，分别是a.cpp,b.h,c.h,d.h

a.cpp的头部是：

#include "b.h "

#include "c.h "

b.h和c.h的头部都是:

#include "d.h "

而d.h里面有class D的定义。

这样一来，

编译器编译a.cpp的时候，先根据#include "b.h "去编译b.h这个问题，再根据b.h里面的#include "d.h "，去编译d.h的这个文件，这样就把d.h里面的class D编译了；然后再根据a.cpp的第二句#include "c.h "，去编译c.h，最终还是会找到的d.h里面的class D，但是class D之前已经编译过了，所以就会报重定义错误。

加上ifndef/define/endif，就可以防止这种重定义错误。

-----------------------------------------------------------------------------------------------------------------------------------

1.比如你有两个C文件，这两个C文件都include了同一个头文件。而编译时，这两个C文件要一同编译成一个可运行文件，于是问题来了，大量的声明冲突。 还是把头文件的内容都放在#ifndef和#endif中吧。

不管你的头文件会不会被多个文件引用，你都要加上这个。

一般格式是这样的：

#ifndef <标识>

#define <标识>

......

......

#endif <标识>

在理论上来说可以是自由命名的，但每个头文件的这个“标识”都应该是唯一的。标识的命名规则一般是头文件名全大写，前后加下划线，并把文件名中的“.”也变成下划线，如：stdio.h

#ifndef \_STDIO\_H\_

#define \_STDIO\_H\_

......

#endif

2.在#ifndef中定义变量出现的问题（一般不定义在#ifndef中）。

#ifndef AAA

#define AAA

...

int i;

...

#endif

里面有一个变量定义在vc中链接时就出现了i重复定义的错误，而在c中成功编译。

原因：

(1).当你第一个使用这个头的.cpp文件生成.obj的时候，int i 在里面定义了当另外一个使用这个的.cpp再次[单独]生成.obj的时候，int i 又被定义然后两个obj被另外一个.cpp也include 这个头的，连接在一起，就会出现重复定义。

(2).把源程序文件扩展名改成.c后，VC按照C语言的语法对源程序进行编译，而不是C++。在C语言中，若是遇到多个int i，则自动认为其中一个是定义，其他的是声明。

(3).C语言和C++语言连接结果不同，可能（猜测）时在进行编译的时候，C++语言将全局变量默认为强符号，所以连接出错。C语言则依照是否初始化进行强弱的判断的。

参考解决方法：

(1).把源程序文件扩展名改成.c。

(2).推荐解决方案：.h中只声明 extern int i;

在.cpp中定义

#ifndef \_\_X\_H\_\_

#define \_\_X\_H\_\_

extern int i;

#endif //\_\_X\_H\_\_ int i;

注意问题：变量一般不要定义在.h文件中。