# 防止編譯器重複引入 #ifndef/#define/.../#endif

## io.h 檔案

是配合 io.cpp 檔案的, io.h 檔案裡面主要包含 **io.cpp** 模組想要提供給程式其它部份的工具的宣告, 例如 函式的宣告 (readFile, printArray) 或是 型態的宣告 (struct MyFile), 例如:

```
void readFile(int *dataSize, int data[]);
void printArray(const int[], int);

struct MyFile
{
    FILE *fp;
    int fileLength;
};
```

程式其它部份想要使用 **io.cpp** 裡面工具時, 就可以 引入 io.h 檔案, 例如 :

## main.cpp

```
#include "io.h"
void main()
{
    int data[100];
    int dataSize;
    readFile(&dataSize, data);

}
```

## test.cpp

```
#include "io.h"
int testFunction()
{
    int data[200];
    int size;
    ...
    printArray(data, size);
    ...
}
```

## 編譯器重複引入造成的重複定義錯誤

假如有一個模組 **network.cpp**, 提供一個 sendFile 的函式

```
void sendFile(struct MyFile *fp)
{
    ...
    ...
}
```

在它的界面裡需要使用 io.h 裡面定義的型態 MyFile, 所以 **network.h** 的內容如下:

```
#include "io.h"
void sendFile(struct MyFile *);
```

如果現在有一個模組 Management.cpp, 裡面需要使用 io.cpp 模組提供的 printArray, 又需要使用 network.cpp 模組提供的 sendFile, **Management.cpp** 裡面一定會有下列程式碼:

```
#include "io.h"
#include "network.h"

void manage()
{
...
...
}
```

於是當前處理器處理完這兩個 #include 敘述以後, 交給編譯器的程式敘述如下:

```
//#include "io.h"
void readFile(int *dataSize, int data[]);
void printArray(const int[], int);

struct MyFile
{
    FILE *fp;
    int fileLength;
};
//#include "network.h"
//#include "io.h"
void readFile(int *dataSize, int data[]);
void printArray(const int[], int);

struct MyFile
{
    FILE *fp;
    int fileLength;
};
void sendFile(struct MyFile *);
void manage()
{
...
...
}
```

**編譯器在檢查上面這段程式時會發生 struct MyFile 重複定義的錯誤**

## 防止程式重複定義

為了防止編譯器重複引入 io.h 檔案時發生重複定義的錯誤, 我們在 io.h 裡面運用前處理器的指令 #ifndef / #define / #endif 來防止重複定義, 方法如下:

### io.h

```
#ifndef IO_H
#define IO_H
void readFile(int *dataSize, int data[]);
void printArray(const int[], int);

struct MyFile
{
    FILE *fp;
    int fileLength;
};
#endif
```

這時候我們重新編譯 management.cpp 時, 前處理器處理完兩個 #include 敘述以後, 交給編譯器的程式敘述如下:

```
//#include "io.h"
#ifndef IO_H
```

```
#define IO_H    // 請注意此列
void readFile(int *dataSize, int data[]);
void printArray(const int[], int);

struct MyFile
{
    FILE *fp;
    int fileLength;
};
//#include "network.h"
//#include "io.h"
#endif
```

```
#ifndef IO_H     // 請注意此列
#define IO_H
void readFile(int *dataSize, int data[]);
void printArray(const int[], int);

struct MyFile
{
    FILE *fp;
    int fileLength;
};
void sendFile(struct MyFile *);
#endif
void manage()
{
...
...
}
```

上面第二次的 #ifndef IO_H 敘述會因為前一個 #define IO_H 而失敗, 所以編譯器完全不會看到上面紅字的部份, 因此第二次的引入 io.h 不會造成編譯時重複定義的錯誤

請注意, 每一個 .h 檔案裡用的前處理器符號需要是不一樣的, 否則編譯器會跳過很多不該跳過的敘述, 也就是說你的 io.h, sort.h, statistics.h 應該有如下的內容:

### io.h

```
#ifndef IO_H
#define IO_H
...
#endif
```

### sort.h

```
#ifndef SORT_H
#define SORT_H
...
#endif
```

## statistics.h

```
#ifndef STATISTICS_H
#define STATISTICS_H
...
#endif
```

原則上 IO_H, SORT_H, STYATISTICS_H 這些名字是你自己挑選的, 不同的檔案用的當然要不一樣, 大小寫都可以, 不過不要和你自己程式裡面的變數重複, 否則你會發生很意外的狀況, 也不要用句點

比較新的編譯器裡面可以使用

## #pragma once

來取代, 不過因為你還是會用到比較舊的編譯器 (或是 public domain 的編譯器), 所以你還是應該了解前面的用法


*回 C++ 物件導向程式設計課程 首頁*
*最近更新日期: Sat Feb 22 2014 16:49:08 GMT+0800 (台北標準時間)*
*製作日期: 02/22/2014 by 丁培毅 (Pei-yih Ting)*
*E-mail: pyting@mail.ntou.edu.tw TEL: 02 24622192x6615*
*海洋大學 電機資訊學院 資訊工程學系 Lagoon*