防止头文件重复引入

重复引入的问题

- 由于在不知情的原因下，有可能将头文件包含多次，如果在头文件中出现了定义（变量、函数、类），就会出现重复定义的问题。
- 一般头文件中只放声明（变量、函数、类），而声明可以多次。
- C++ 重复包含头文件，编译器拷贝和扫描需要耗费时间，降低效率。
- C++ 中处理头文件重复引入，提供了 3 种解决方案：
  - #ifndef、#define、#endif 宏定义
  - #pragma once 指令
  - _Pragma("once") 操作符

#ifndef、#define、#endif 宏定义

- 格式：

```
#ifndef _NAME_H
#define _NAME_H
//头文件内容
#endif
```

- 当程序中第一次 `#include` 该文件时，由于 `_NAME_H` 尚未定义，所以会定义 `_NAME_H` 并执行“头文件内容”部分的代码；当发生多次 `#include` 时，因为前面已经定义了 `_NAME_H`，所以不会再重复执行“头文件内容”部分的代码。

#pragma once 指令

- 格式：在头文件的最开头的位置加上 `#pragma once`
- `#ifndef` 是通过定义独一无二的宏来避免重复引入的，这意味着每次引入头文件都要进行识别，所以效率不高。
- `C` 和 `C++` 都支持宏定义，所以使用 `#ifndef` 不会影响项目的可移植性。
- 和 `#ifndef` 相比，`#pragma once` 不涉及宏定义，当编译器遇到它时就会立刻知道当前文件只引入一次，所以效率很高。
- `#pragma once` 只能作用于某个具体的文件，而无法向 `#ifndef` 那样仅作用于指定的一段代码。

_Pragma("once") 操作符

- 格式：在头文件的最开头的位置加上 _Pragma("once")
- C99 标准中新增加了一个和 #pragma 指令类似的 _Pragma 操作符，其可以看做是 #pragma 的增强版，不仅可以实现 #pragma 所有的功能，更重要的是，_Pragma 还能和宏搭配使用。
- 总结：
  - #pragma once 和 _Pragma("once")，特点是编译效率高，但可移植性差
  - #ifndef 的特点是可移植性高，编译效率差。
  - 除非对项目的编译效率有严格的要求，强烈推荐采用 #ifndef 的方式。
  - 某些场景中考虑到编译效率和可移植性，#pragma once 和 #ifndef 经常被结合使用。

Thanks