静态库

静态库——即多个**obj**文件的集合，**Windows**上后缀为`.lib`，**Linux**上为`.a`

编译链接的过程

1. 预处理
2. 编译（CPP -> OBJ）
3. 链接（OBJ -> EXE）

使用静态库

1. 拷贝、包含头文件
2. 把`lib`添加入工程中
    1. 项目属性页 ——> 链接器 ——> 输入 ——> 附加依赖项
    2. `#pragma comment(lib, "xxx.lib")`
    3. 把`lib`拖入工程中

为了使一个头文件同时在CPP与C文件中使用，可以使用

```
#ifdef __cplusplus
extern "C" {
#endif
    // 这里填入需要的东西
#ifdef __cplusplus
}
#endif
```

静态库的缺点

1. 每次使用都需要重新编译
2. 多个程序使用同一个静态库，每个程序中都会有一份静态库代码的拷贝，会造成资源的浪费

动态库

静态库是在编译阶段将代码加入到**可执行文件**中，而动态库则是在运行时将代码加入到**进程**中

默认情况下，动态库中的函数禁止外部项目使用。如果要使用，需要动态库指明此函数可以被外部项目使用。可以被外部使用的函数称为**导出函数**，外部项目使用动态库中的函数，需要指明此函数是**某库的导出函数**（即**导入函数**）

`__declspec(dllexport)`指明此为**导出函数**

`__declspec(dllimport)`指明此为**导入函数**

```
#ifdef DLL_EXPORTS
#define DLL_API __declspec(dllexport)
#else
#define DLL_API __declspec(dllimport)
#endif
```

使用动态库

静态导入

1. 包含头文件
2. 将`dllexport`改为`dllimport`
3. 添加库`xxx.lib`

此`lib`文件只是保存了`dll`文件的相关信息，并没有可执行的代码。在运行时需要加载`dll`，而静态库在运行时并不需要`lib`

注

导出类的语法有点区别`class __declspec(dllexport) TestClass { ... }`

动态库的查找顺序

- 启用"安全DLL查找模式"时
    1. 应用程序所在目录
    2. 系统目录
    3. Windows目录
    4. 当前目录
    5. 环境变量PATH中所有目录
- 禁用"安全DLL查找模式"时
    1. 应用程序所在目录
    2. 当前目录
    3. 系统目录
    4. Windows目录
    5. 环境变量PATH中所有目录

"安全DLL查找模式"默认是启用，将`HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\SafeDllSearchMode`设为**0**可禁用