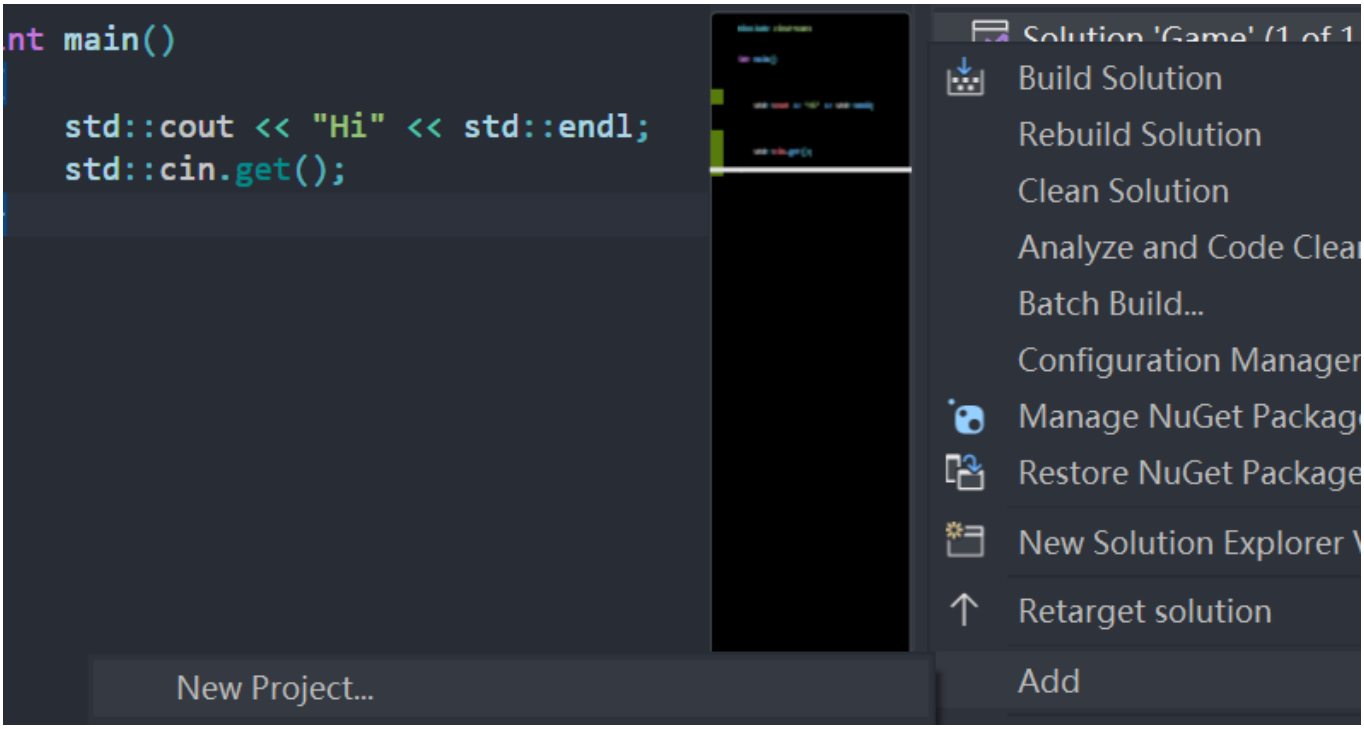
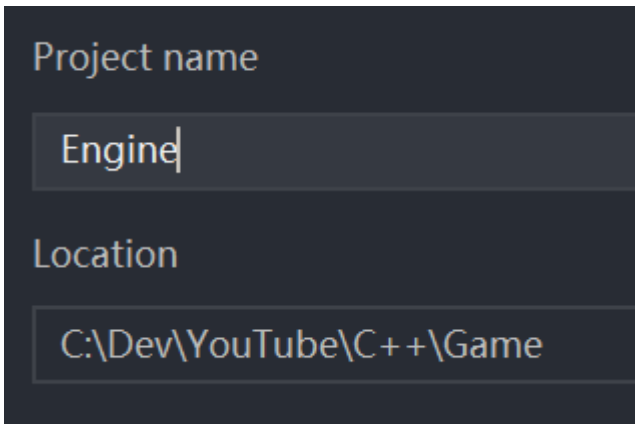
51 Making and Working with Libraries in C++ (Multiple Projects in VS)

C++中创建与使用库
本节课主要讲的是如何在Visual Studio中建立多个项目，以及如何创建一个库让所有项目都能使用。这是非常重要的一点，如果你的项目规模非常大的话，它不仅可以帮助你用代码创建模块或库，并多次重用这些代码，还允许你混合语言。

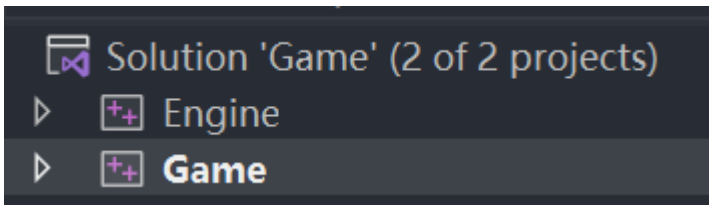
1. 添加项目

13 BEST Visual Studio Setup for C++ Projects!请见本课配置好第一个项目“Game”
然后右键Solution，新建第二个项目：

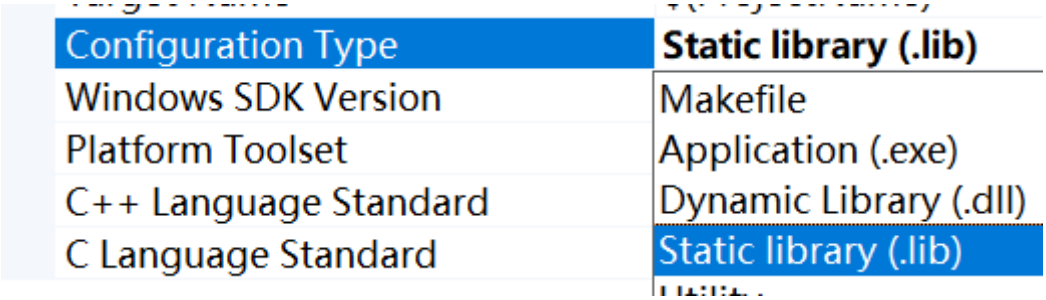




新建后目录应为如下样式：



确保其中Game项目的配置类型为可执行文件：
而因为我们要静态链接，所以Engine的配置类型要设为静态库：



如果我想使用这个命名空间，可以通过两种方法：

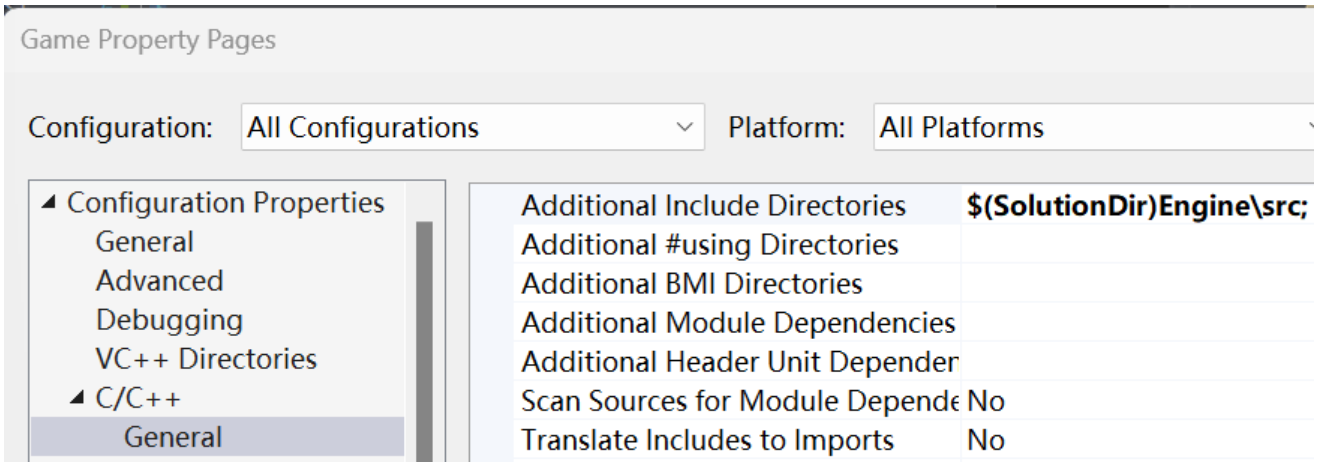
- 相对路径：

```cpp
#include "../../Engine/src/Engine.h"

int main()
{
    engine::PrintMessage();
}
```

但如果挪动文件位置则会发生致命问题，因此我们应该使用绝对路径，特别是使用编译器的包含路径。

- 绝对路径：




```cpp
#include "Engine.h" //这样就够了
```

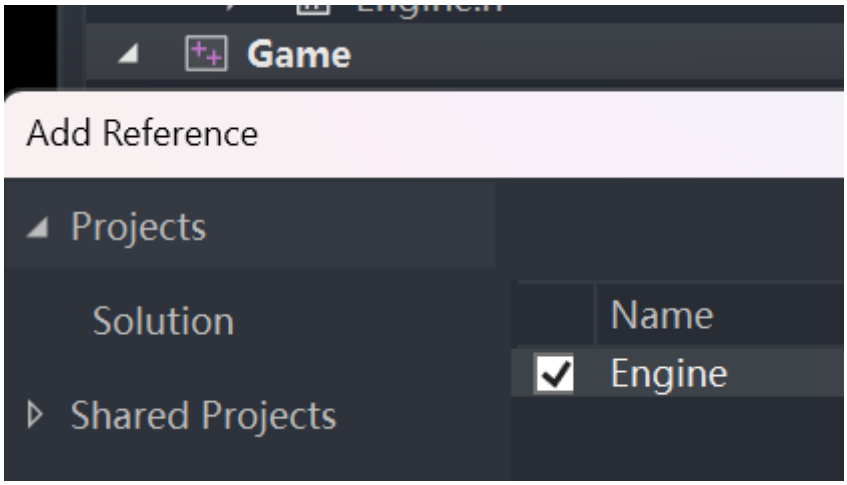
2. 链接

现在我们需要链接了。
点击Engine的build，可以看到输出：

```
1>Engine.vcxproj → C:\Dev\YouTube\C++\Game\x64\Debug\Engine.lib
```

这个.lib文件正是我们想要链接的，我们可以到链接器设置把它作为输入，但我们不需要这么做，VS可以自动化这个操作，因为这个项目是在实际的解决方案中：

Game→Add→Reference，然后点OK。



这会把那个.lib文件链接到我们的可执行文件中，就像我们已经把它添加到链接器中输入一样。
一个好处是Engine现在是Game的依赖，所以Game依赖于Engine，意味着如果Engine内部的某些东西发生了变化，然后我们去编译Game(Game实际要编译Engine和Game自己)。

证明：Clean Solution，然后build Game，可以看到：

```
1>------ Build started: Project: Engine, Configuration: Debug x64 ------
1>Engine.cpp
1>Engine.vcxproj → C:\Dev\YouTube\C++\Game\x64\Debug\Engine.lib
2>------ Build started: Project: Game, Configuration: Debug x64 ------
2>Application.cpp
2>Game.vcxproj → C:\Dev\YouTube\C++\Game\bin\x64Debug\Game.exe
```

它实际上做的是先构建Engine再构建Game，Game需要Engine才能工作，因为Game引用了Engine而需要链接它。