c 언어와 어셈블리 언어를 이용한 코드를 DLL과 lib로 저장, 실행하기
-chat gpt사용-

1. 코드 작성

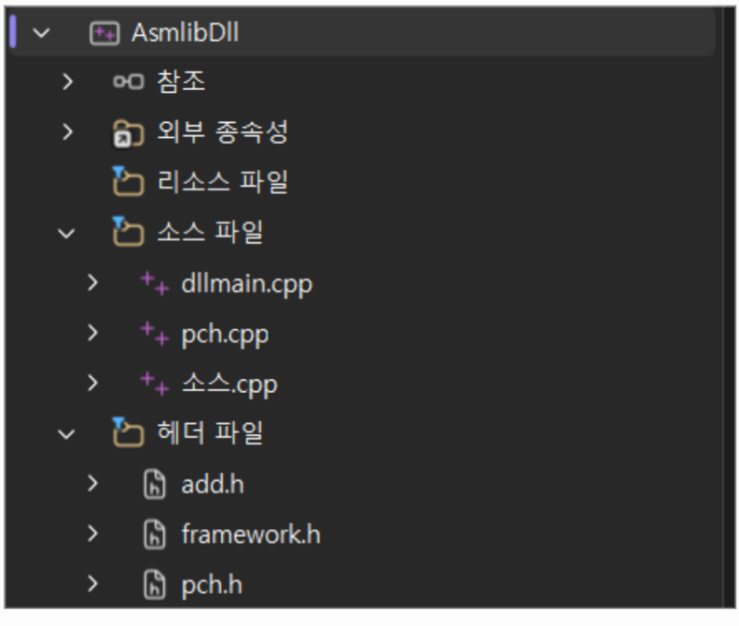
```cpp
#include <iostream>
#include "add.h"     // DLL 함수 선언 불러오기

#pragma comment(lib, "AsmlibDll.lib")   // DLL의 .lib 링크

int main()
{
    long long a = 10;
    long long b = 20;

    long long result = AddNumbers(a, b);

    std::cout << "DLL AddNumbers 테스트" << std::endl;
    std::cout << a << " + " << b << " = " << result << std::endl;

    return 0;
}
```

2. 코드 실행을 위한 환경 준비

— visual studio 2022에서 DLL동적 라이브러리를 검색하여 들어간뒤
프로젝트를 만들어 masm을 사용하도록 지정하여 DLL이 작동하도록 먼저 만든다.

Name	Path
☐ ImageContentTask(.targets, .props)	$(VCTargetsPath)₩BuildCustomizations₩ImageContentTask.targets
☐ lc(.targets, .props)	$(VCTargetsPath)₩BuildCustomizations₩lc.targets
☑ marmasm(.targets, .props)	$(VCTargetsPath)₩BuildCustomizations₩marmasm.targets
☑ masm(.targets, .props)	$(VCTargetsPath)₩BuildCustomizations₩masm.targets
☐ MeshContentTask(.targets, .props)	$(VCTargetsPath)₩BuildCustomizations₩MeshContentTask.targets
☐ ShaderGraphContentTask(.targets, .props)	$(VCTargetsPath)₩BuildCustomizations₩ShaderGraphContentTask.targets

3. 세부 파일 생성



이렇게 DLL을 빌드하기위하여 세부 파일들을 만들고(프로젝트형식으로)
그 파일들안에 각각

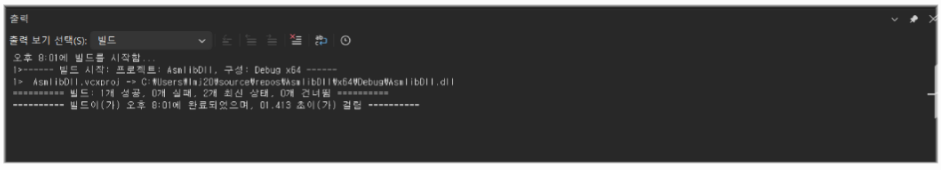
소스.cpp

```cpp
#include "pch.h"
#include "Add.h"

extern "C" __declspec(dllexport) long long AddNumbers(long long a, long long b)
{
    return a + b;
}

```

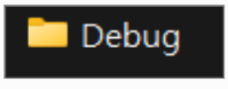
헤더.h

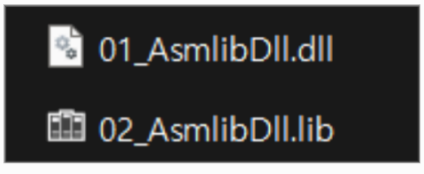
```cpp
#pragma once

extern "C" __declspec(dllimport) long long AddNumbers(long long a, long long b);

```

이런식으로 먼저 만들고 빌드하여



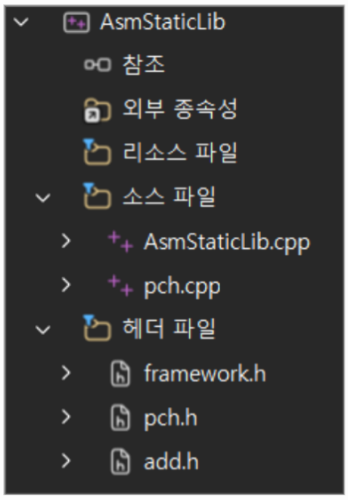
이렇게 빌드 성공이 뜨면





이런식으로 debug라는 파일안에 동적 라이브러리랑 DLL파일이 생성된다.
— 동적 라이브러리랑 정적 라이브러리의 차이점

구분 동적 라이브러리 / 정적 라이브러리
실행 시 필요 여부 - ✅ 실행할 때 필요 / ✅ 실행할 때 필요
파일- 형태 .dll / .lib
EXE -크기 작음 / 큼
프로그램 -간 공유 ✅ 가능 / ❌ 불가능
배포할 때- DLL 같이 보내야 함 / EXE 하나만 보내면 됨
2-1) 정적 라이브러리 만들기

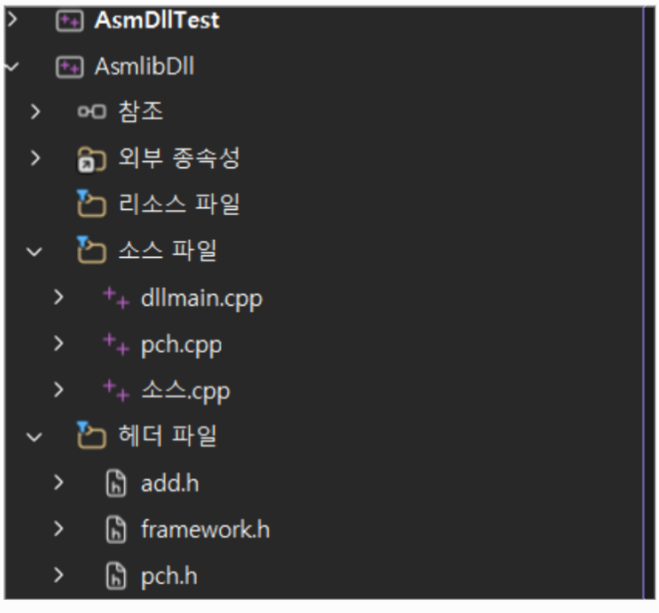


이렇게 파일목록들을 만들어주고
해더 파일에는 DLL과 연결하기 위해 같은 add.h를 만들고
AsmStaticLib.cpp에는

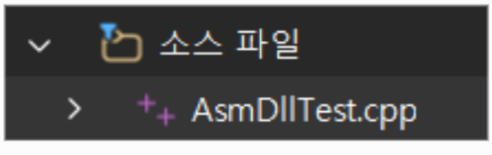
```cpp
#include "pch.h"
#include "add.h"

long long AddNumbers(long long a, long long b)
{
    return a + b;
}

```

를 넣고 빌드하면

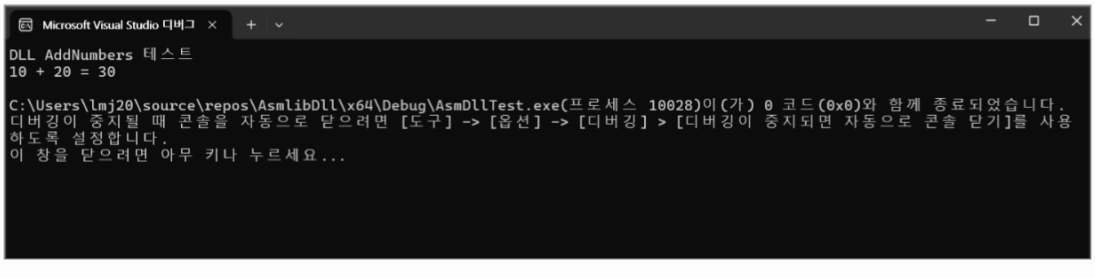
이런식으로 빌드성공 문구가 뜸
3. 실질적인 코드 실행



이렇게 실행할 코드를 적을 파일들을 만들고(프로젝트형식으로) 그리고



안에맨 처음에 만들었던 코드를 올려두고
위와 같은 똑같은 헤더파일 add.h를 올려주고 실행을 하게 되면



이렇게 뜬다.