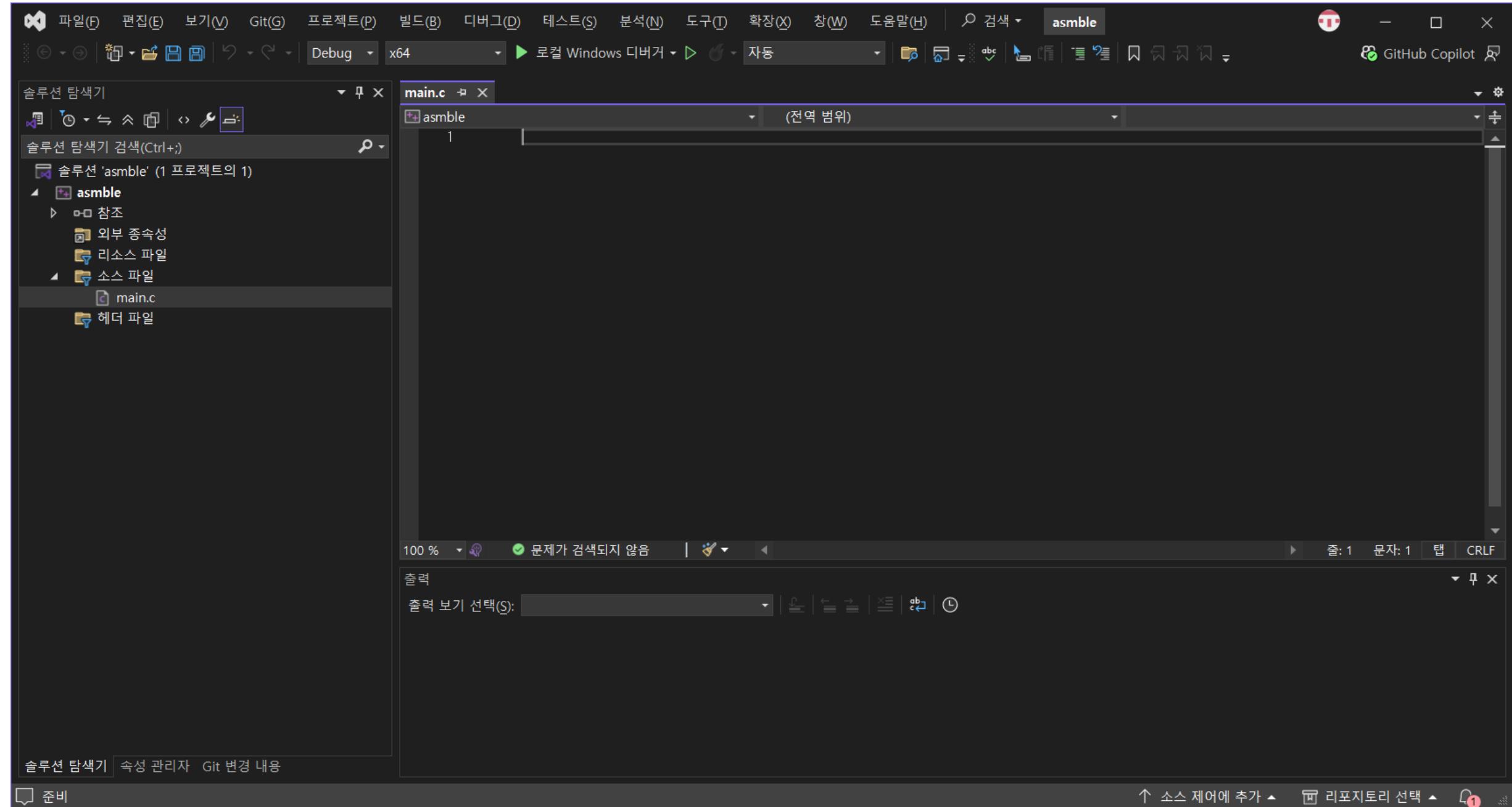


# C+ASM

2022664008 김수민

# 1. C,ASM 결합



1.1 먼저 새 프로젝트를 만든 뒤 main.c를 생성합니다

```
asmble (전역 범위) main0
1 #include <stdio.h>
2
3 int multiply(int a, int b) {
4     int result;
5
6     __asm {
7         mov eax, a; eax = a
8         mov ebx, b; ebx = b
9         imul eax, ebx; eax = a * b
10        mov result, eax; result = eax
11    }
12
13    return result;
14 }
15
16 int main() {
17     int x, y;
18
19     printf("두 정수를 입력하세요: ");
20     scanf("%d %d", &x, &y);
```

```
int r = multiply(x, y);
printf("%d * %d = %d\n", x, y, r);
return 0;
```

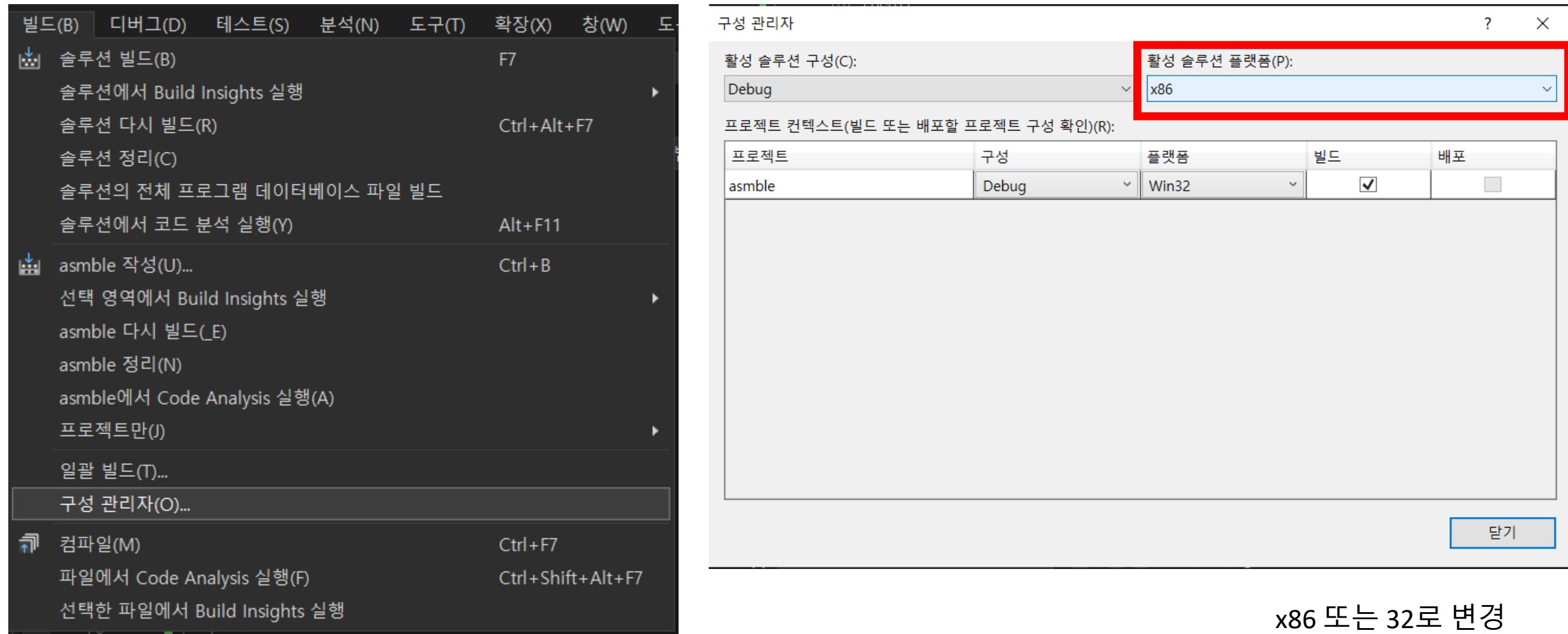
\_\_asm을 활용하여 asm 코드 작성

## 1.2 두 수를 곱하는 프로그램을 작성

선택 Microsoft Visual Studio 디버그 콘솔

```
두 정수를 입력하세요: 5 4
5 * 4 = 20
C:\Users\bigdi\OneDrive\work\assem\asmb1e\Debug\asmb1e.exe(프로세스 9080)이(가) 0 코드(0x0)와 함께 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요...
```

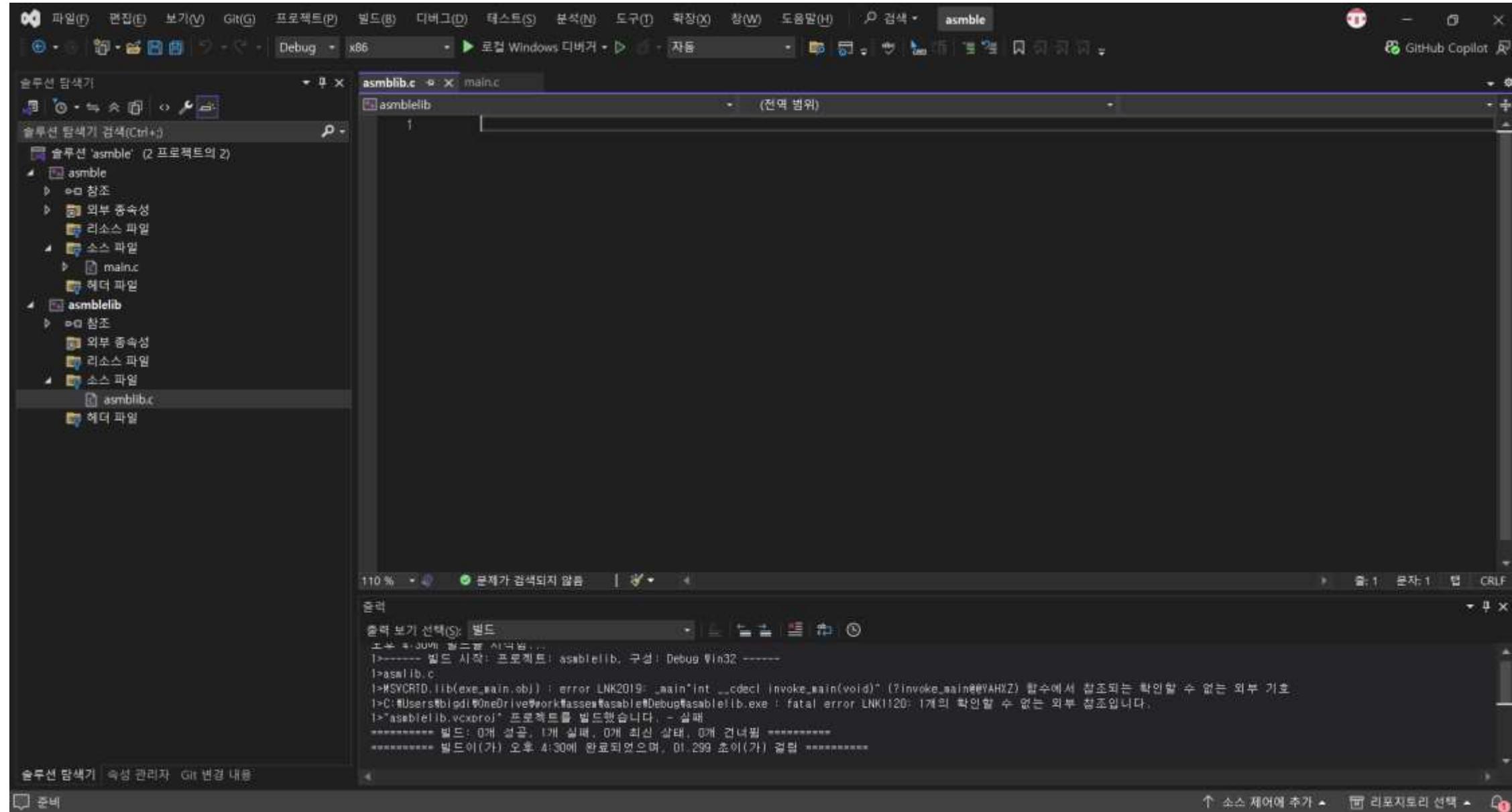
### 1.3 정상적으로 실행되는 모습



x86 또는 32로 변경

※ 혹시 실행이 안된다면 구성 관리자를 확인!!

## 2. 라이브러리



2.1 같은 솔루션에 새로운 프로젝트 생성 후 asmbllib.c 생성

The screenshot shows a Windows-based IDE interface. The top menu bar includes: 파일(F), 편집(E), 보기(V), Git(G), 프로젝트(P), 빌드(B), 디버그(D), 테스트(S), 분석(N), 도구(I), 확장(X), 도움말(H), 검색(S). The title bar says "asmble". The toolbar includes icons for file operations like Open, Save, and Build. The status bar at the bottom shows: 110%, 문제가 검색되지 않음, 줄: 14, 문자: 2, SPC, CRLF.

**Solution Explorer:** Shows the project structure with two projects: "asmble" and "asmlib".

- asmble:** Includes a "참조" folder, a "외부 종속성" folder, a "리소스 파일" folder, a "소스 파일" folder containing "main.c", and a "헤더 파일" folder.
- asmlib:** Includes a "참조" folder, a "외부 종속성" folder, a "리소스 파일" folder, a "소스 파일" folder containing "asmlib.c", and a "헤더 파일" folder.

**Editor:** The main editor window displays the "asmlib.c" file. It contains the following C code:

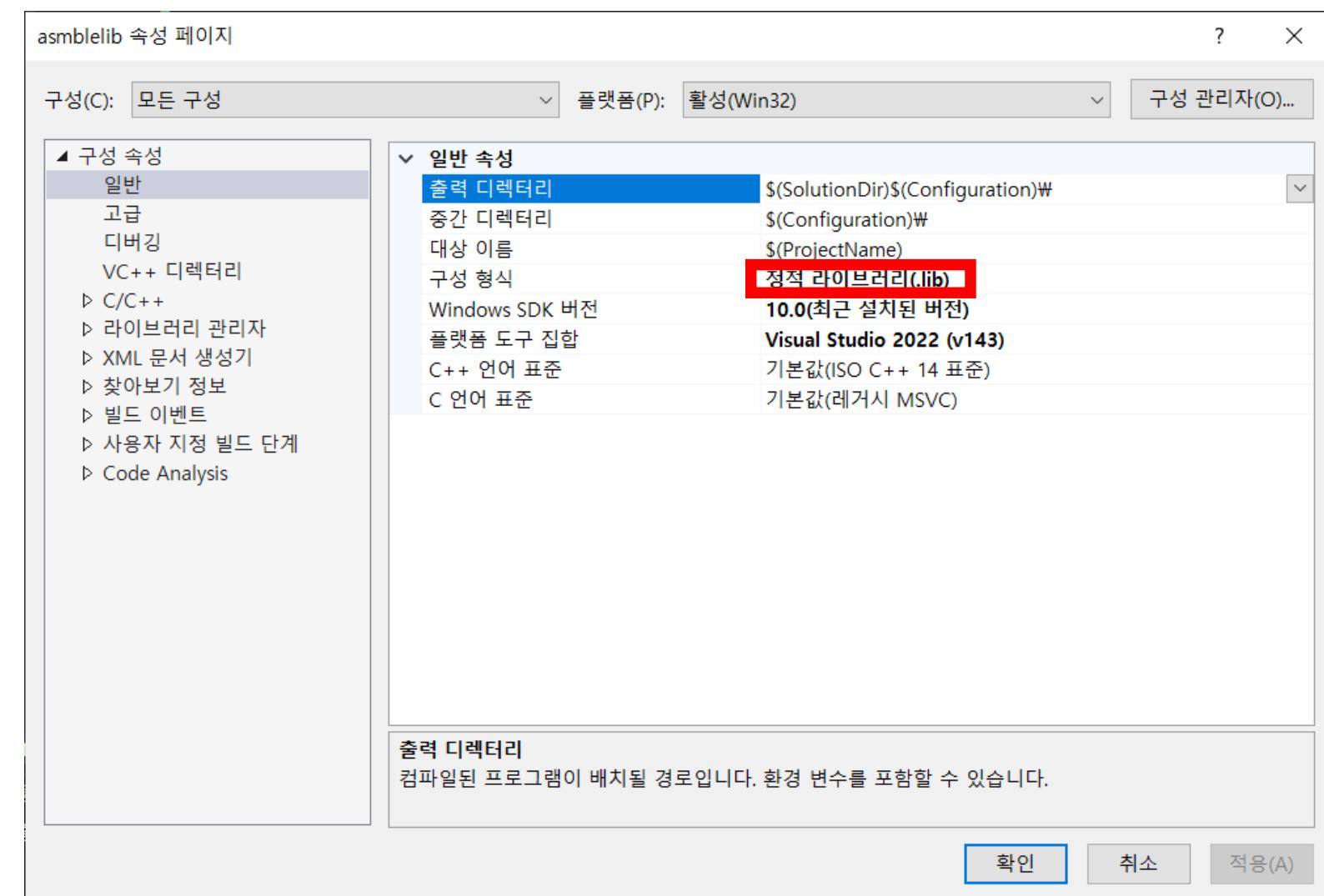
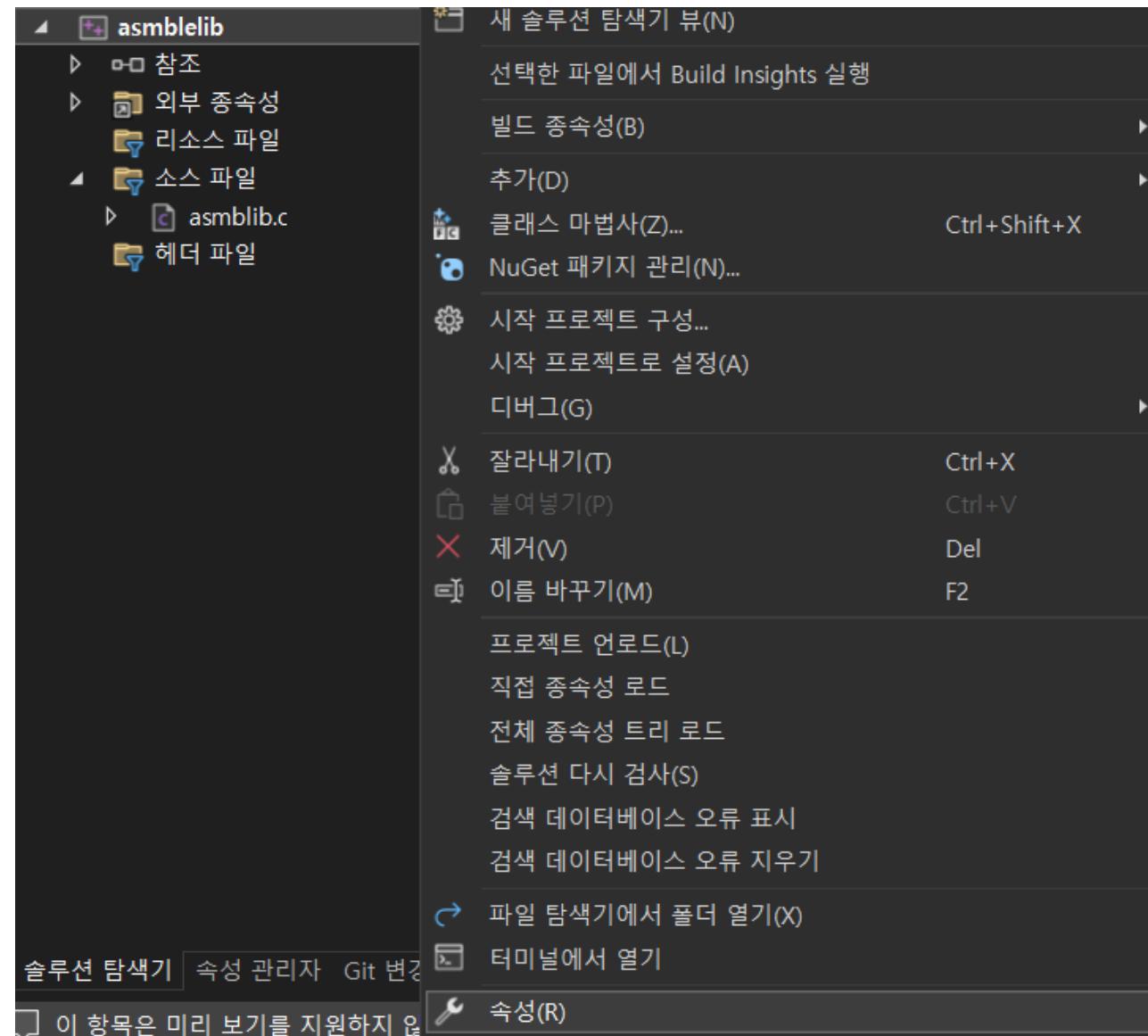
```
#include <stdio.h>
int multiply(int a, int b) {
    int result;
    asm {
        mov eax, a; eax = a;
        mov ebx, b; ebx = b;
        imul eax, ebx; eax = a * b
        mov result, eax; result = eax
    }
    return result;
}
```

**Output Window:** Shows the build log for the "asmlib" project.

```
소스 제어에 빌드 시작...
>----- 빌드 시작: 프로젝트: asmlib.lib, 구성을: Debug #fn32 -----
t>asmlib.c
>MSVCRTD.lib(exe_main.obj) : error LNK2019: _main"int __cdecl invoke_main(void)" (?invoke_main@@YAHXZ) 할수에서 참조되는 확인할 수 없는 외부 기호
!>C:\Users\biedi\OneDrive\work\asser\asmlib\Debug\asmlib.lib : fatal error LNK1120: 1개의 확인할 수 없는 외부 참조입니다.
!>"asmlib.lib.vcxproj" 프로젝트를 빌드했습니다. - 실패
***** 빌드: 0개 성공, 1개 실패, 0개 최신 상태, 0개 간 nợ됨 *****
***** 빌드(1) 오후 4:30에 완료되었습니다. 0.299 초(1) 걸림 *****
```

**Status Bar:** Shows: 저장되었습니다.

2.2 asmlib.c에 main.c의 main()을 제외 한 코드 작성



2.3 프로젝트에 우클릭 후 속성>구성형식을 정적 라이브러리로 변경

```
#include <stdio.h>

int multiply(int a, int b);

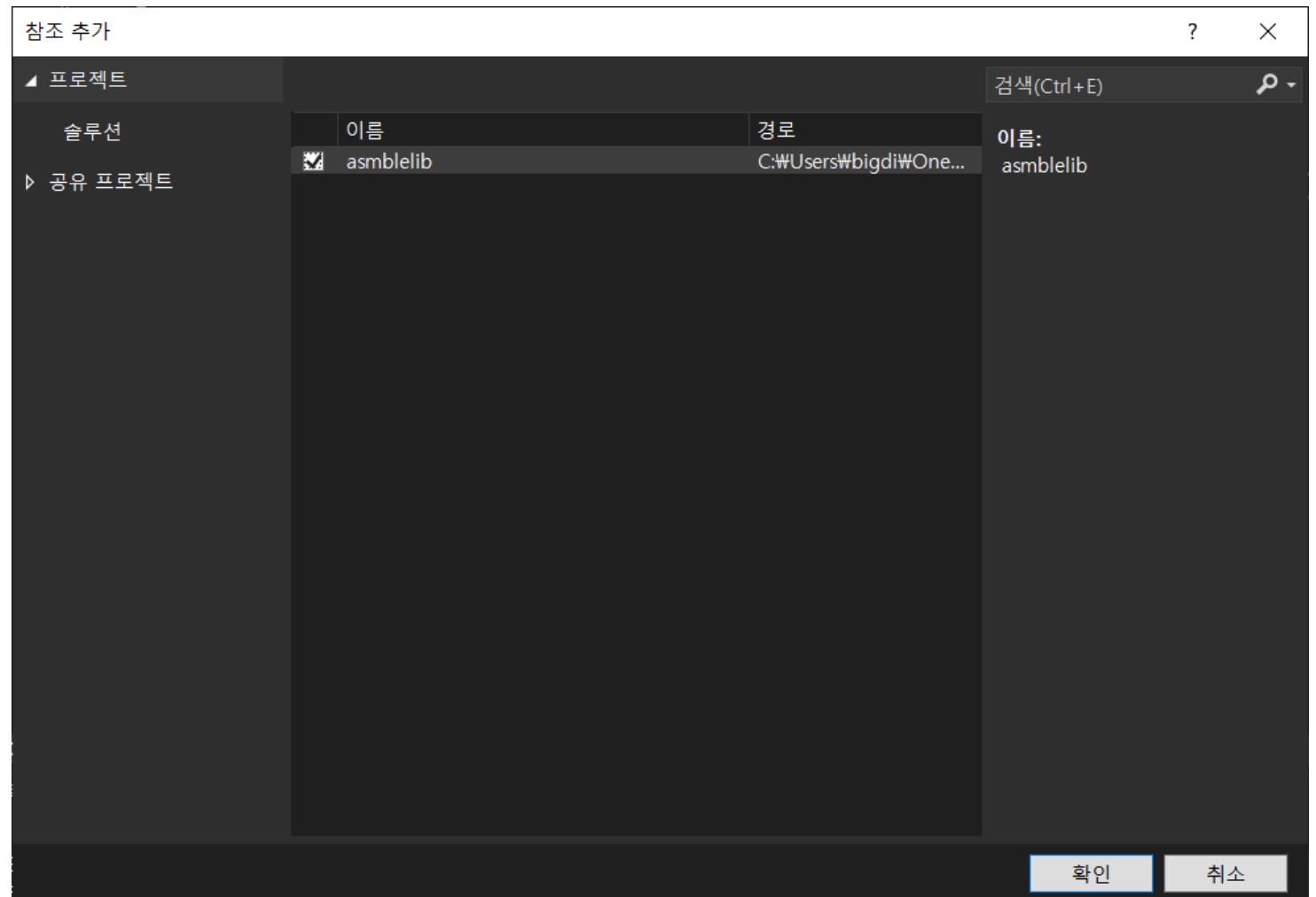
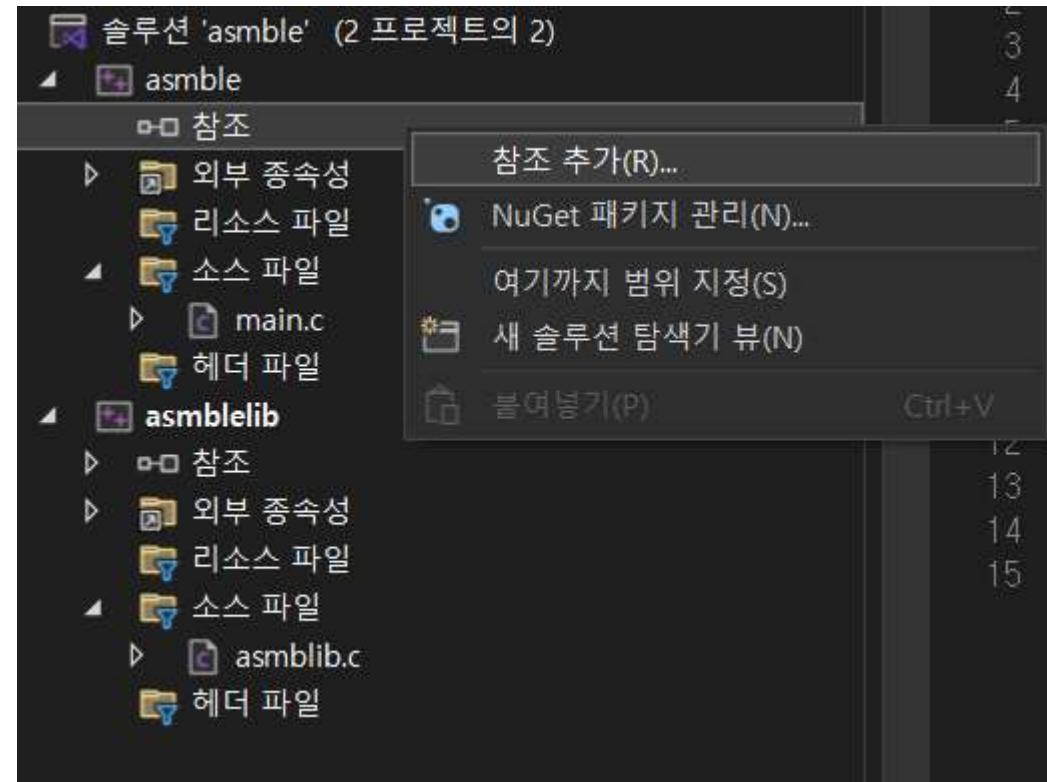
int main() {
    int x, y;

    printf("두 정수를 입력하세요: ");
    scanf_s("%d %d", &x, &y);

    int r = multiply(x, y);

    printf("%d * %d = %d\n", x, y, r);
    return 0;
}
```

## 2.4 main.c 코드를 위와같이 변경



2.5 main.c가 있는 프로젝트에 전에 만들어 놓은 라이브러리  
를 참조하여 링킹

The screenshot shows the Microsoft Visual Studio interface with the following details:

- Solution Explorer:** Shows the project structure for 'asmble'. It includes a solution named 'asmble' (2 projects), a main project 'asmble' containing 'asmblelib' as a reference, and files like 'main.c' and 'asmplib.c'. There are also external dependencies and resource files.
- Code Editor:** The 'main.c' file is open, displaying the following C code:

```
#include <stdio.h>

int multiply(int a, int b);

int main() {
    int x, y;

    printf("두 정수를 입력하세요: ");
    scanf_s("%d %d", &x, &y);

    int r = multiply(x, y);

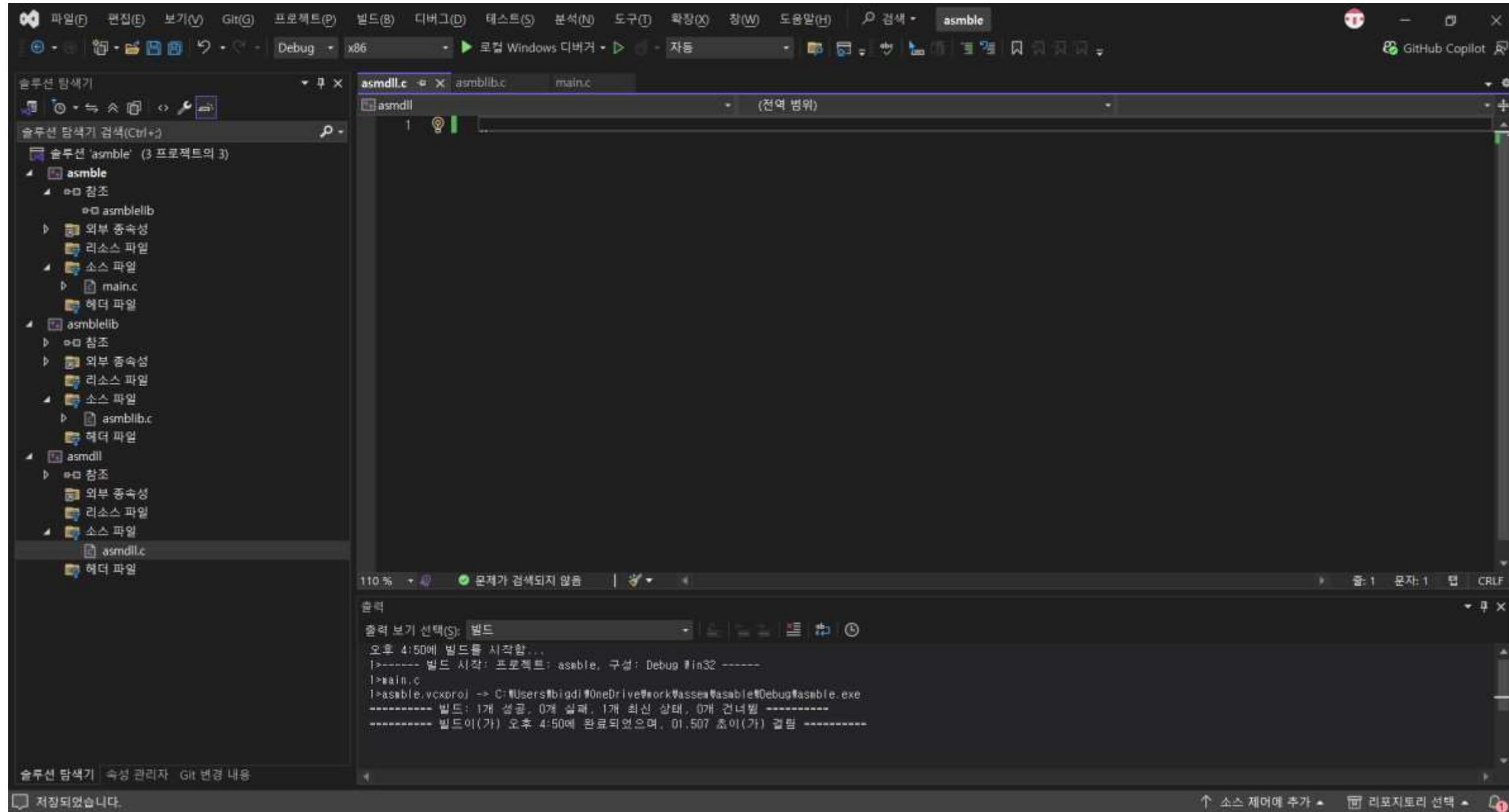
    printf("%d * %d = %d\n", x, y, r);
    return 0;
}
```
- Output Window:** The 'Microsoft Visual Studio 디버그 콘솔' window shows the application's output. It prompts the user for two integers, performs the multiplication, and prints the result. The output text is in Korean.

Output text from the console window:

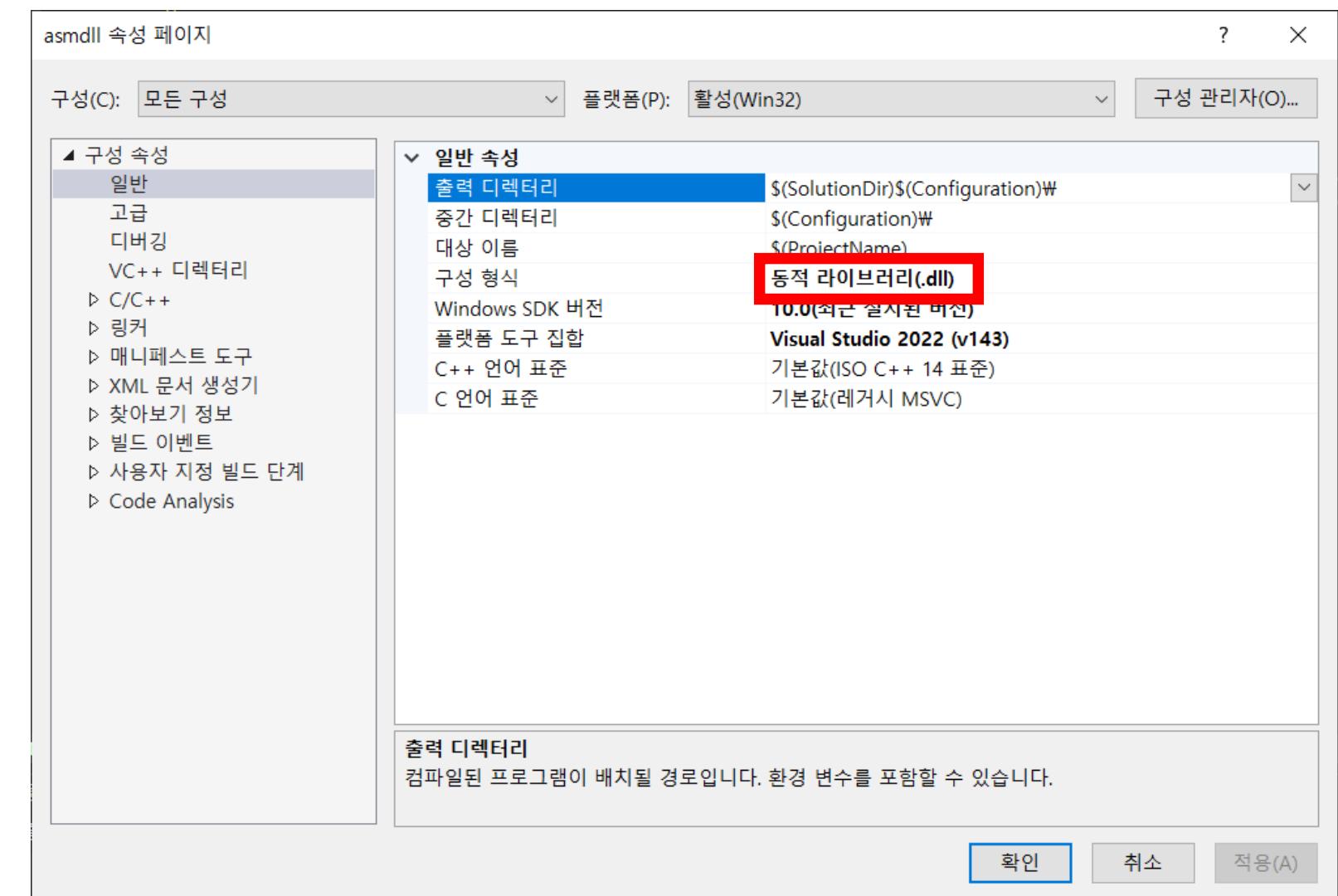
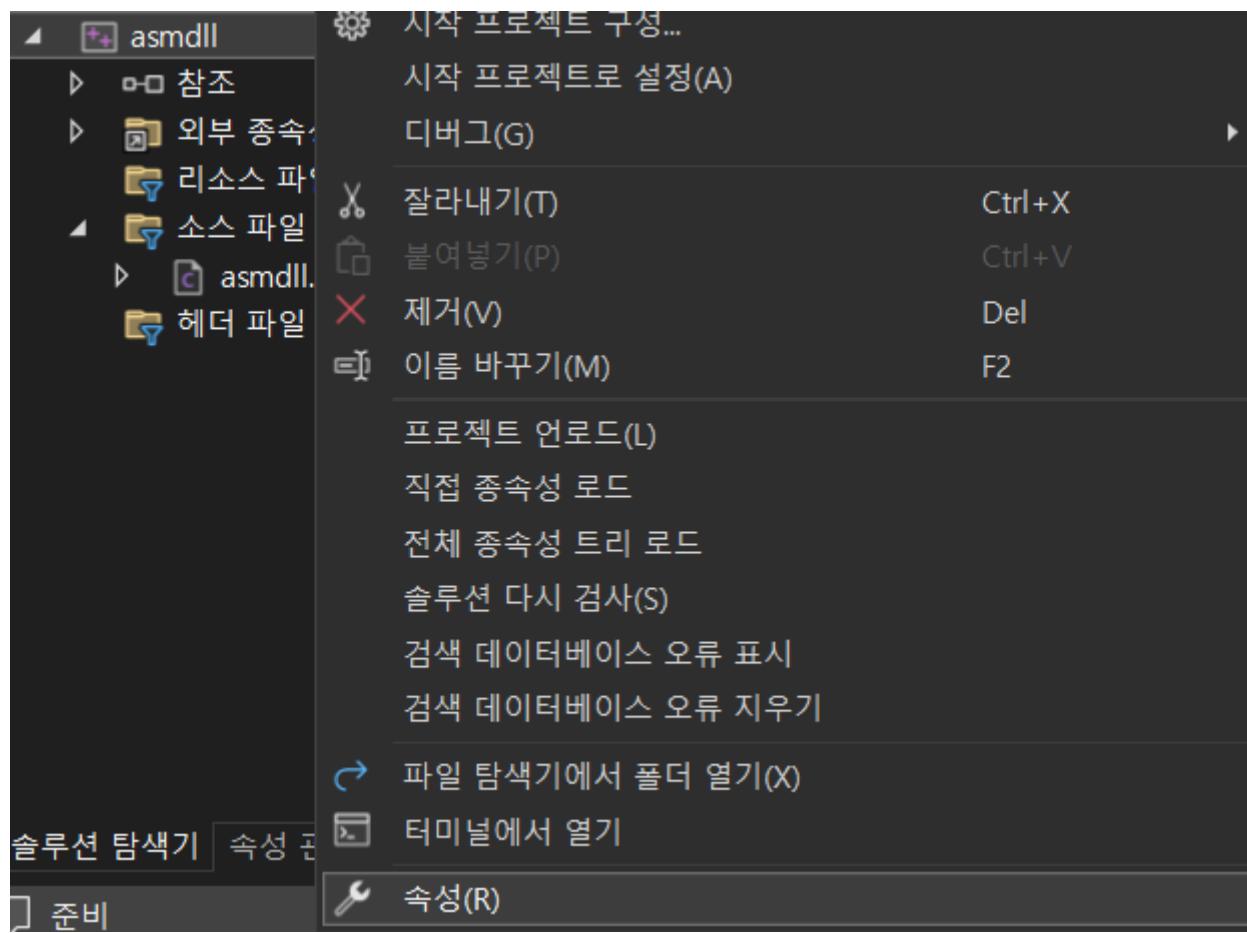
```
두 정수를 입력하세요: 5 4
5 * 4 = 20
C:\Users\bigdi\OneDrive\work\asmb\Debug\asmble.exe(프로세스
15084)이(가) 0 코드(0x0)와 함께 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요...
```

## 2.6 main.c 실행 시 정상작동 하는 모습

# 3. DDL



## 3.1 dll용 프로젝트 생성 후 asmdll.c 생성



3.2 프로젝트에 우클릭 후 속성>구성형식을 동적 라이브러리로 변경

```
#include <stdio.h>

__declspec(dllexport) int multiply(int a, int b) {
    int result;

    __asm {
        mov eax, a; eax = a
        mov ebx, b; ebx = b
        imul eax, ebx; eax = a * b
        mov result, eax; result = eax
    }

    return result;
}
```

출력

출력 보기 선택(S): 빌드

오후 5:04에 빌드를 시작함...

1>----- 빌드 시작: 프로젝트: asmdll, 구성: Debug Win32 -----  
1>asmdll.c  
1> C:\Users\bigdi\OneDrive\work\assem\asmble\Debug\asmdll.lib 라이브러리 및 C:\Users\bigdi\OneDrive\work\assem\asmble\Debug\asmdll.exp 개체를 생성하고 있습니다.  
1>asmdll.vcxproj -> C:\Users\bigdi\OneDrive\work\assem\asmble\Debug\asmdll.dll  
===== 빌드: 1개 성공, 0개 실패, 2개 최신 상태, 0개 건너뜀 =====  
===== 빌드이(가) 오후 5:04에 완료되었으며, 01.246 초이(가) 걸림 =====

3.3 함수 앞에 `__declspec(dllexport)`를 추가하고 빌드하여  
dll,lib 생성

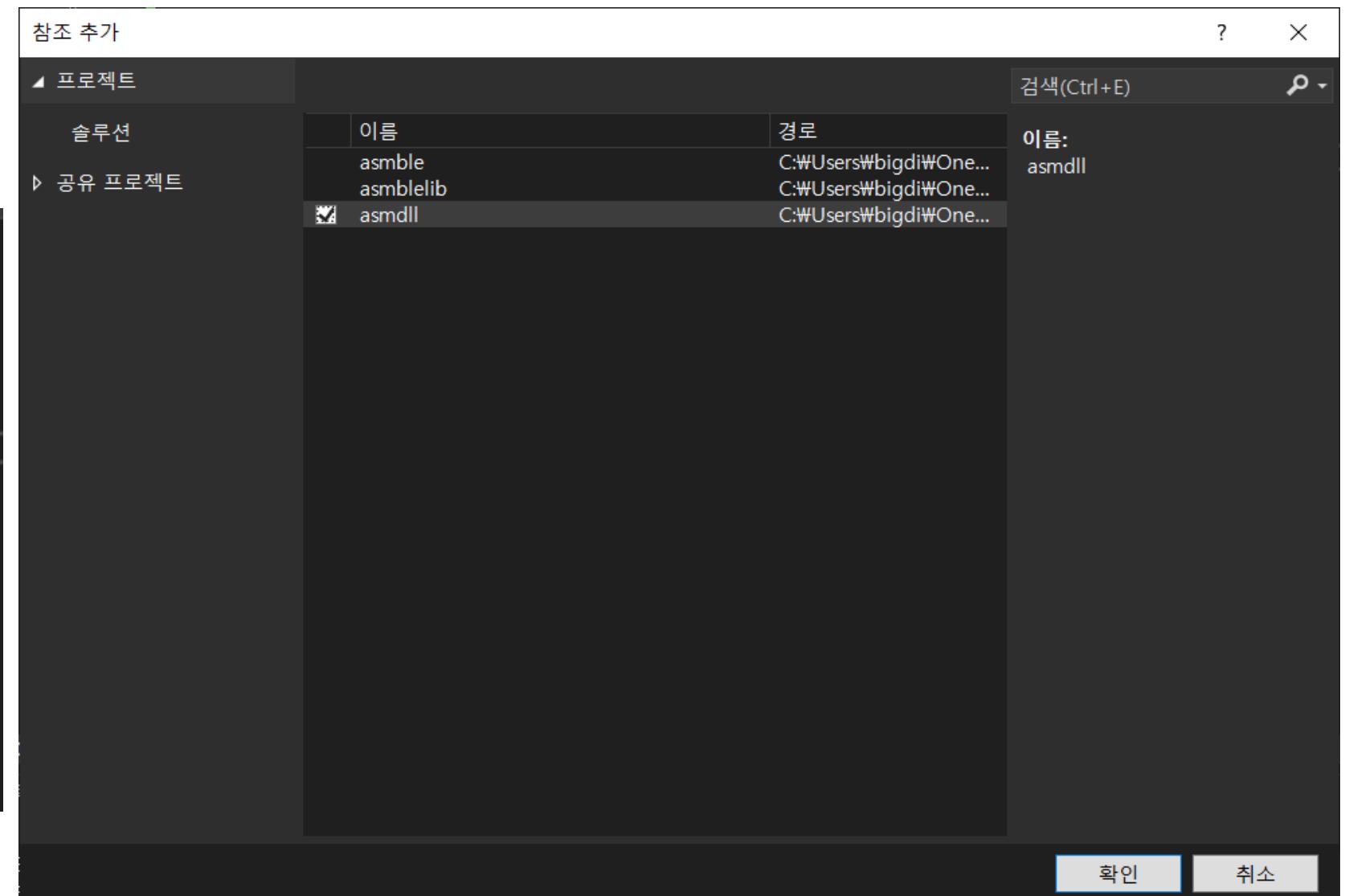
```
#include <stdio.h>

__declspec(dllexport) int multiply(int a, int b);

int main() {
    int x, y;
    printf("두 정수를 입력하세요: ");
    scanf_s("%d %d", &x, &y);

    int r = multiply(x, y);

    printf("%d * %d = %d\n", x, y, r);
    return 0;
}
```



### 3.4 main.c 코드 변경 후 참조 추가

The screenshot shows the Microsoft Visual Studio interface with two main windows. On the left is the '전역 범위' (Global) editor containing C code. On the right is the 'Microsoft Visual Studio 디버그 콘솔' (Debug Console) window.

**Code (전역 범위):**

```
#include <stdio.h>

__declspec(dllexport) int multiply(int a, int b);

int main() {
    int x, y;

    printf("두 정수를 입력하세요: ");
    scanf_s("%d %d", &x, &y);

    int r = multiply(x, y);

    printf("%d * %d = %d\n", x, y, r);
    return 0;
}
```

**Debug Console Output:**

```
Microsoft Visual Studio 디버그 콘솔
두 정수를 입력하세요: 5 4
5 * 4 = 20
C:\Users\bigdi\OneDrive\work\assembly\Debug\dllrun.exe(프로세스
10400)이(가) 0 코드(0x0)와 함께 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요...
```

### 3.5 정상적으로 출력되는 모습

# 정적 vs 동적

항목	정적 라이브러리 (.lib)	DLL (동적 라이브러리 .(dll))
링크 시점	컴파일/링크 시 EXE에 포함됨	실행 시(runtime)에 동적으로 로딩됨
코드 위치	EXE 내부에 직접 복사됨	DLL 파일 내부에 코드가 존재
실행 시 파일 필요 여부	EXE 하나만 있어도 실행 가능	EXE + DLL 파일이 모두 필요
업데이트 방식	라이브러리 변경 시 EXE 재빌드 필요	DLL만 교체하면 EXE는 수정 없이 사용 가능

# 점수

10/10

왜냐하면 교수님이 요구하신 조건  
을 모두 충족하였고 설명도 잘 됐다  
고 생각하기 때문입니다