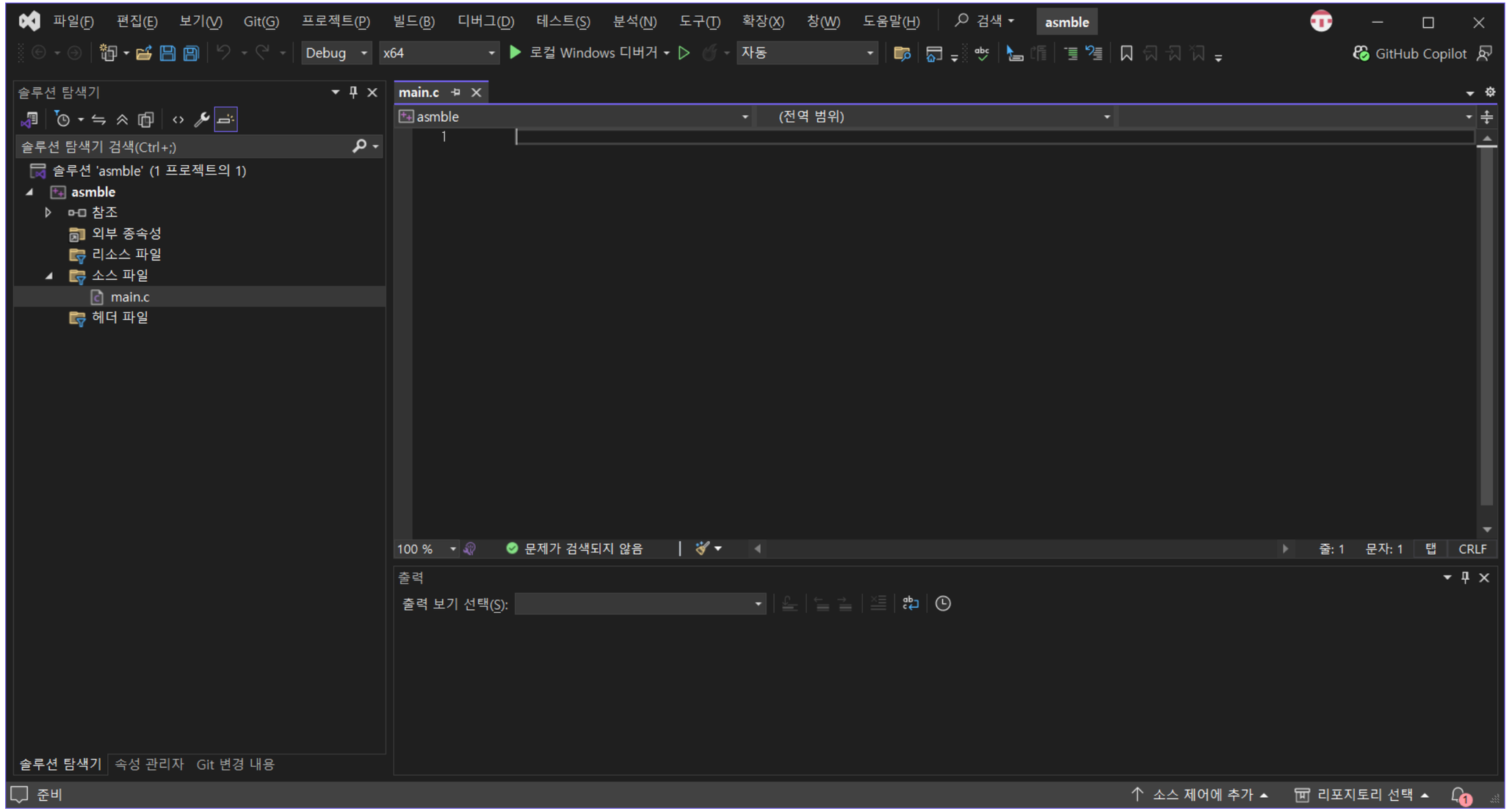
C+ASM

2022664008 김수민

1. C,ASM 결합



1.1 먼저 새 프로젝트를 만든 뒤 main.c를 생성합니다

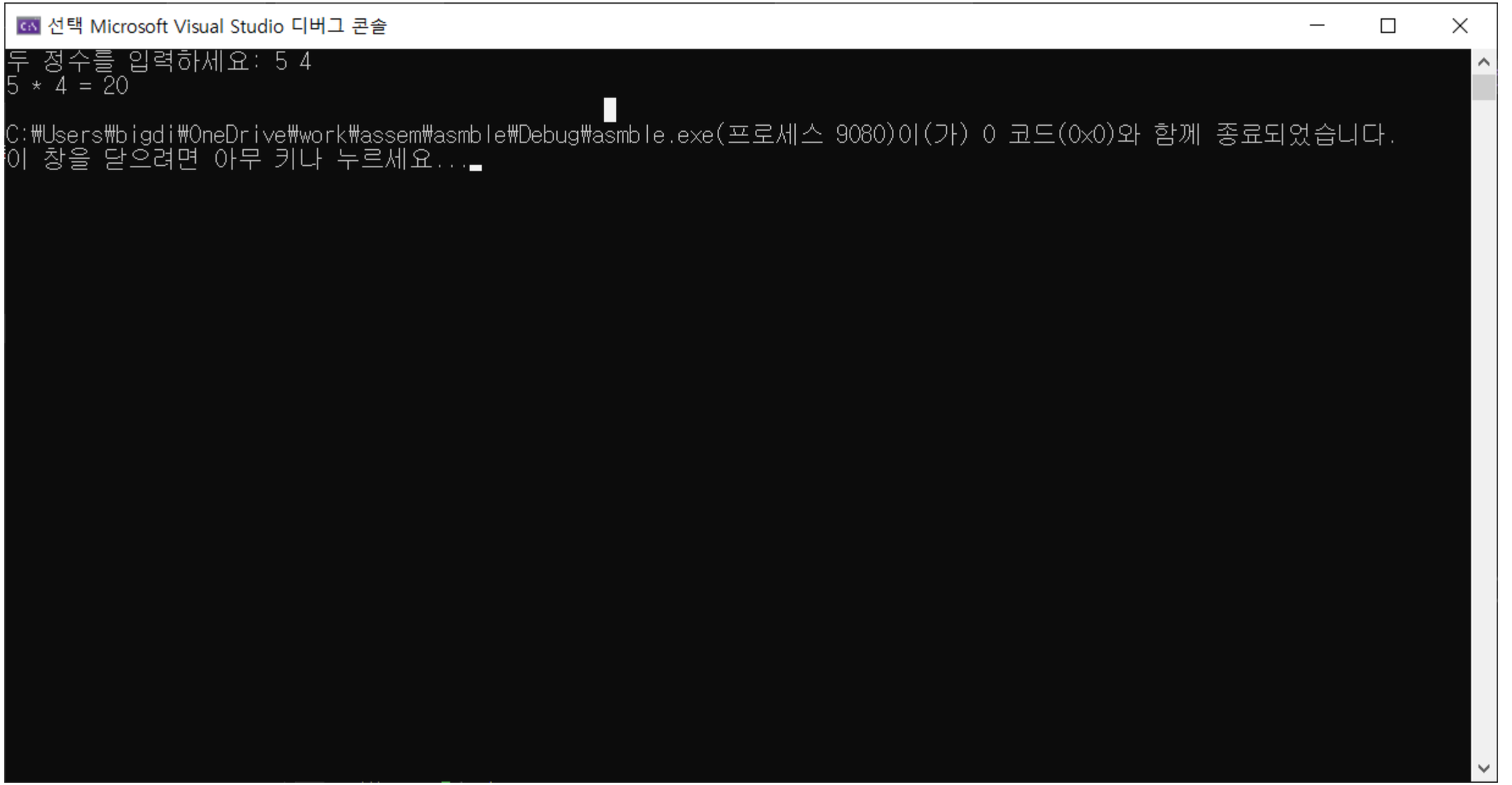
asmble (전역 범위) main()

```
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);
```

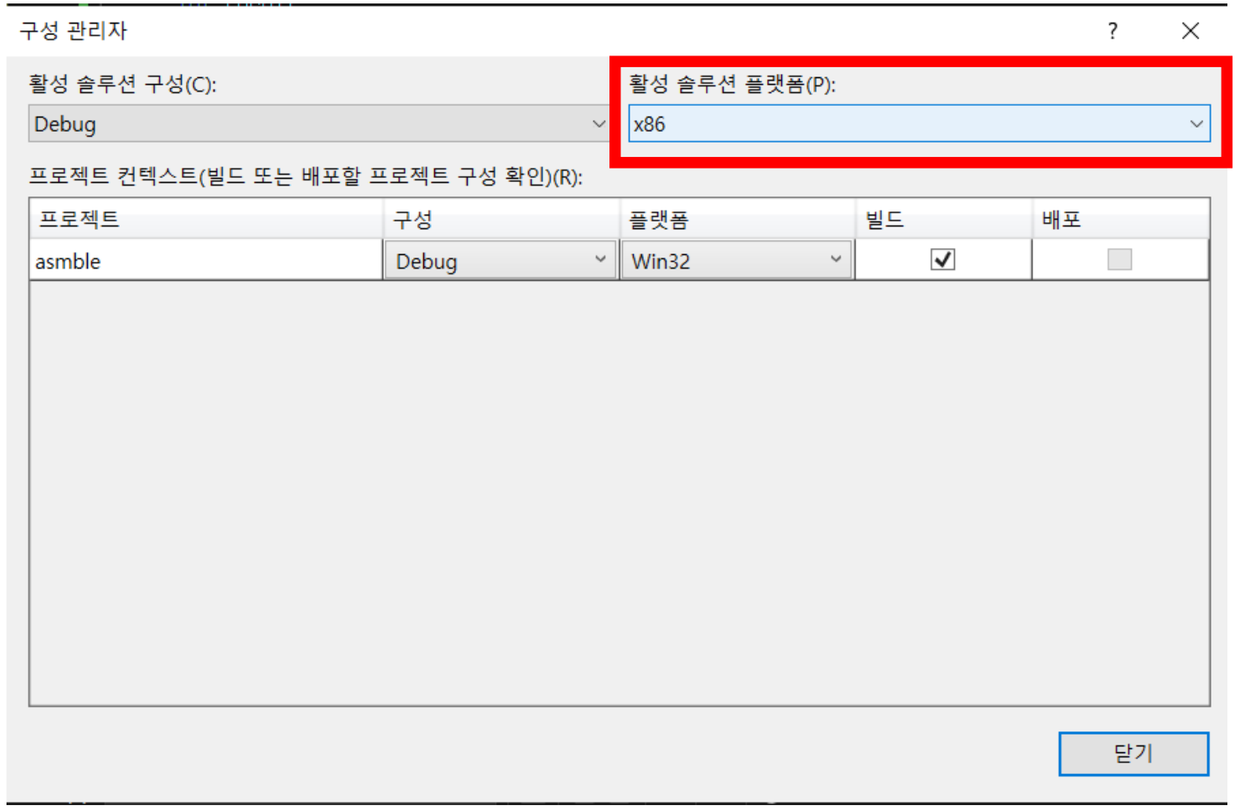
```
int r = multiply(x, y);

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

__asm을 활용하여 asm 코드 작성

1.2 두 수를 곱하는 프로그램을 작성



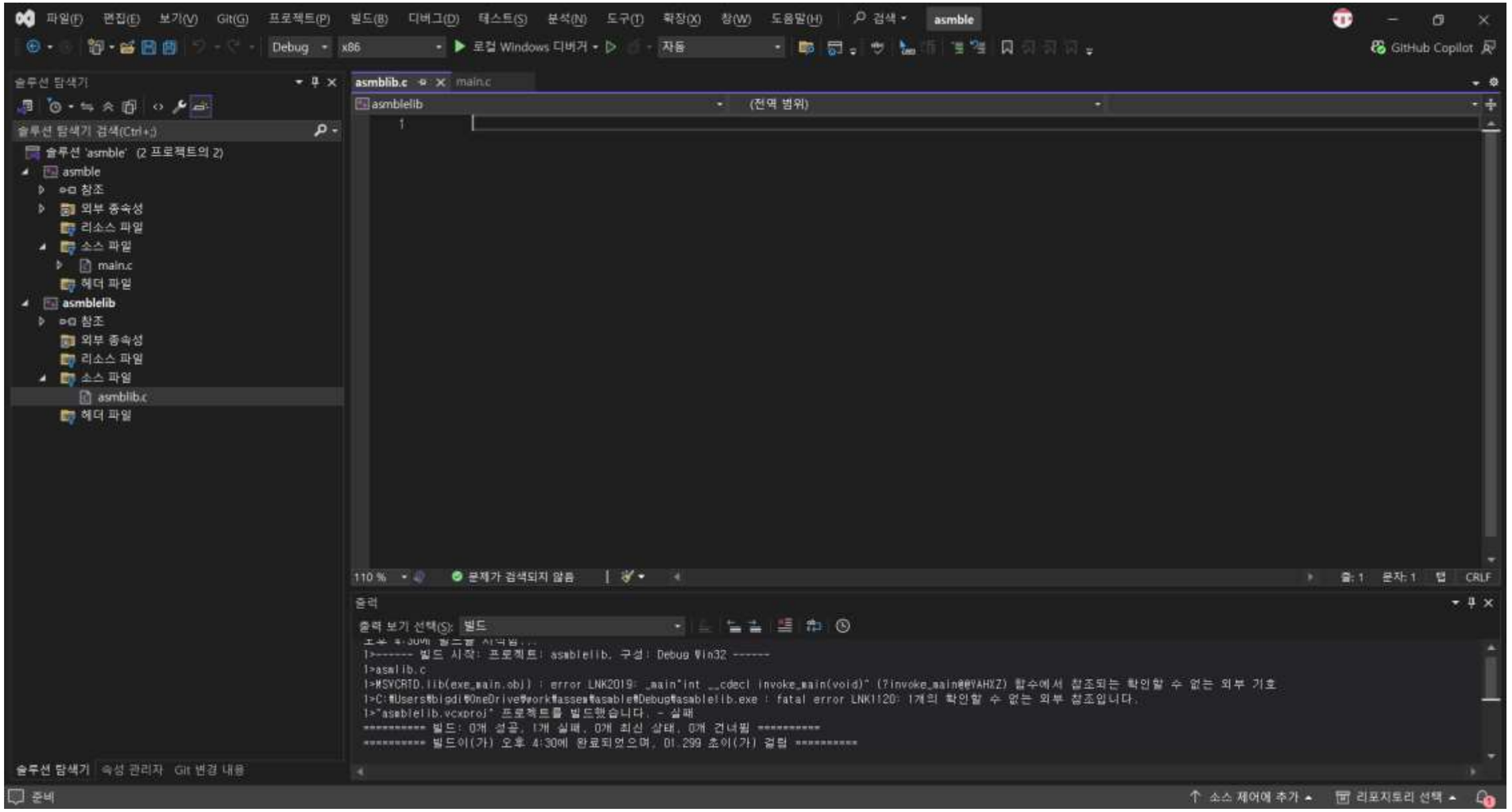
1.3 정상적으로 실행되는 모습



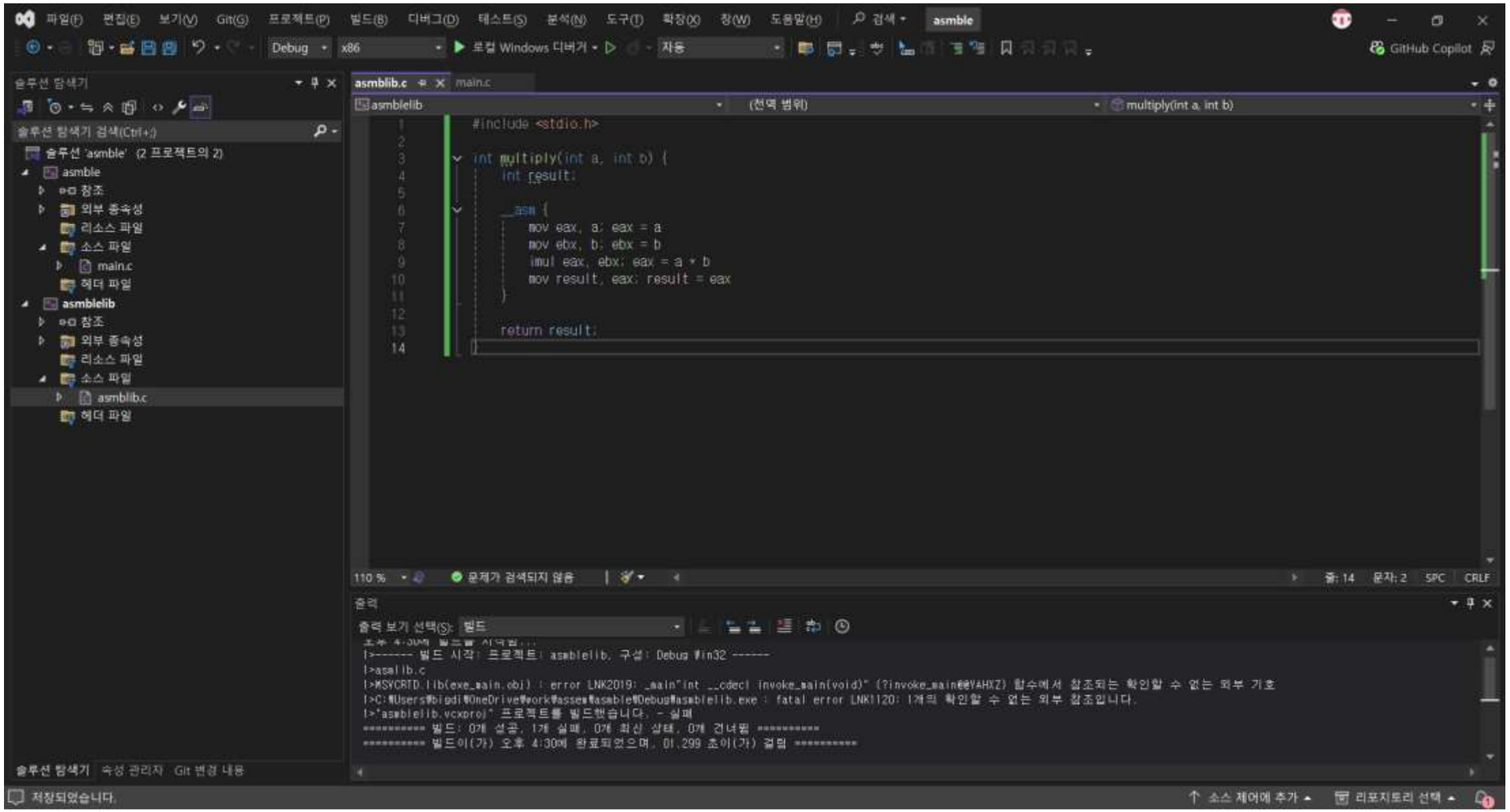
x86 또는 32로 변경

※ 혹시 실행이 안된다면 구성 관리자를 확인!!

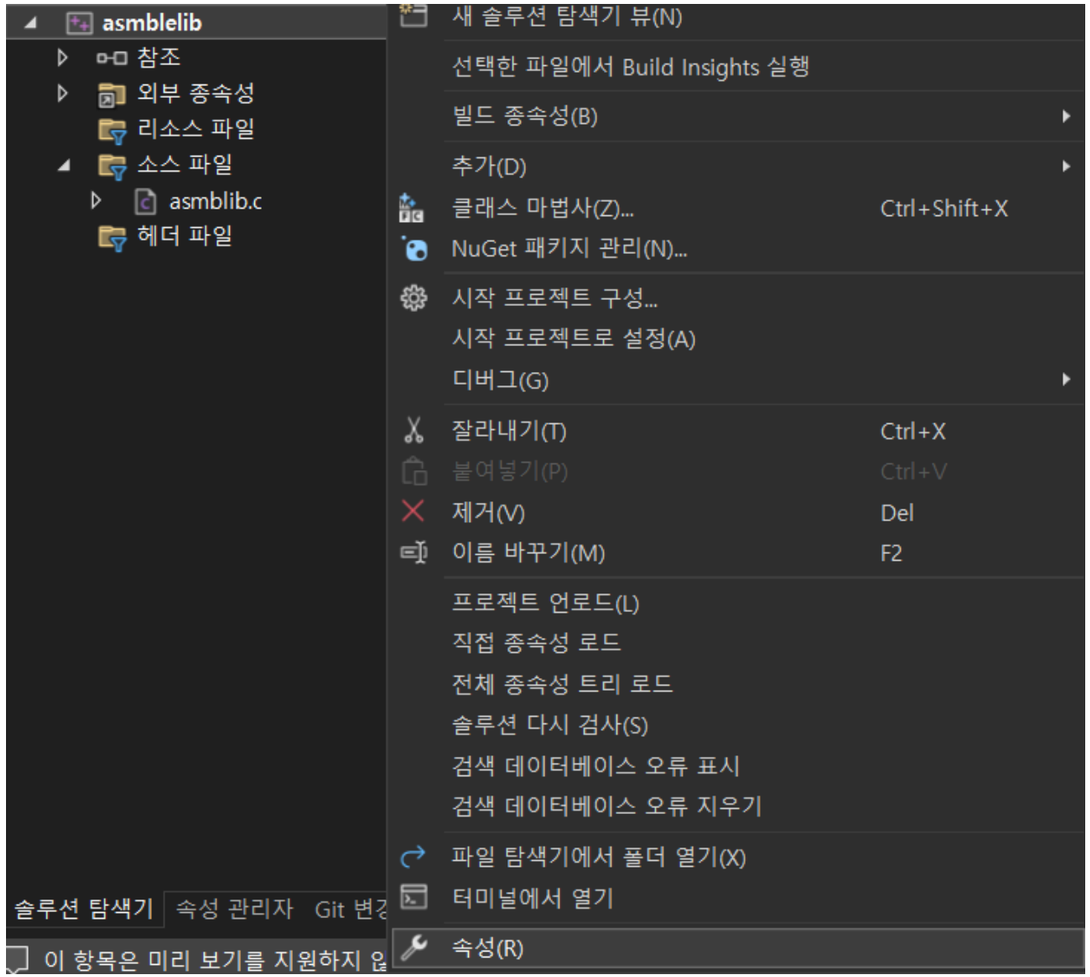
2. 라이브러리

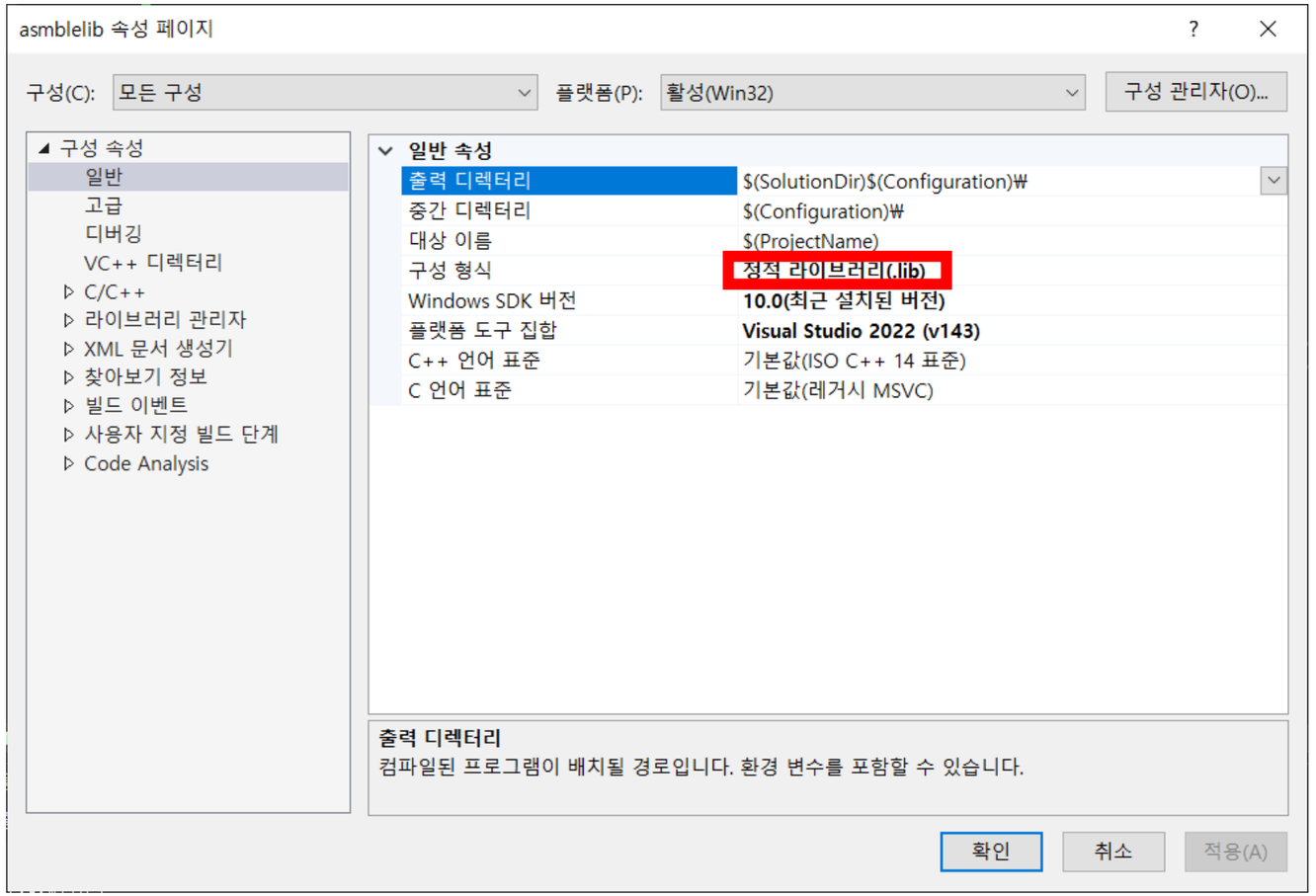


2.1 같은 솔루션에 새로운 프로젝트 생성 후 asmblib.c 생성

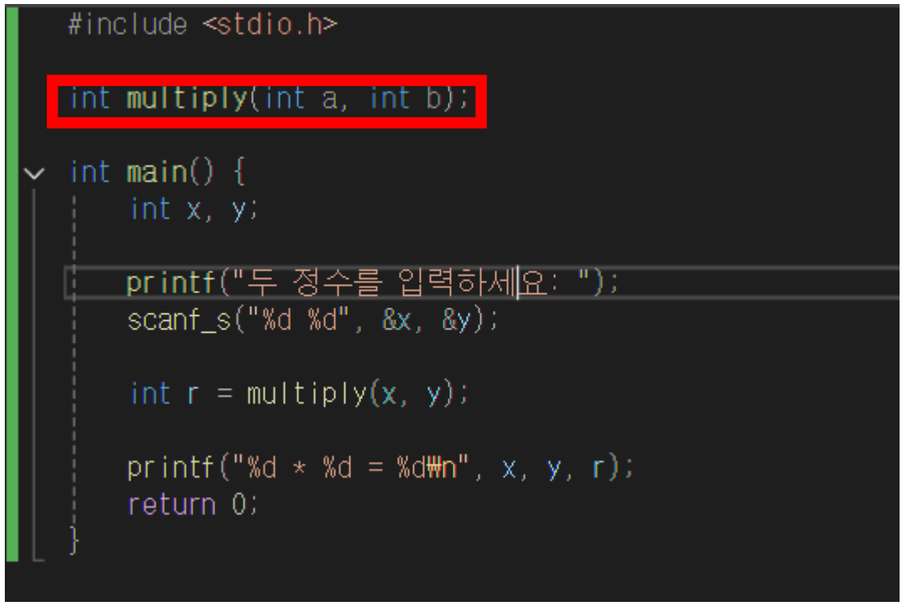


2.2 asmblib.c에 main.c의 main()을 제외 한 코드 작성



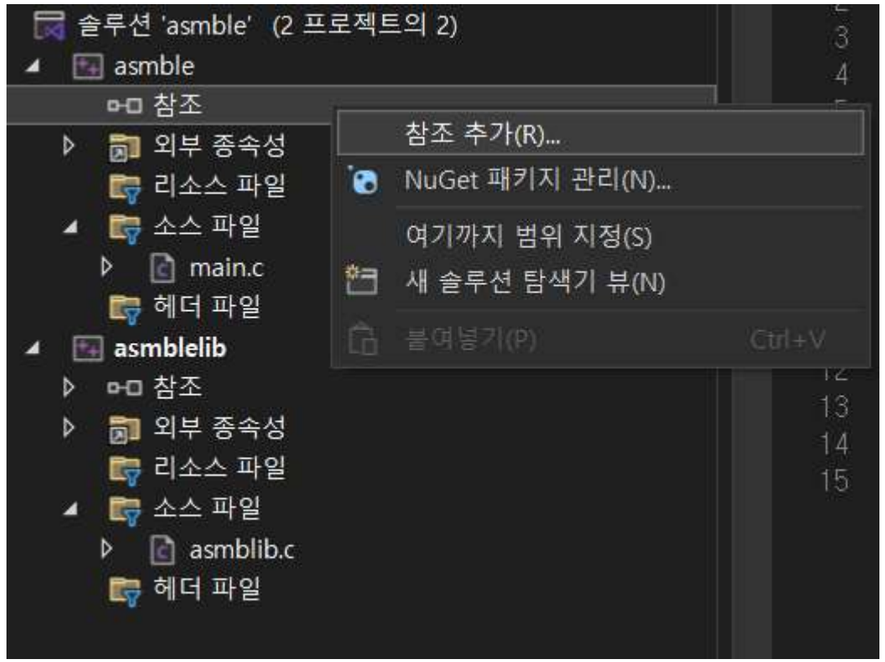


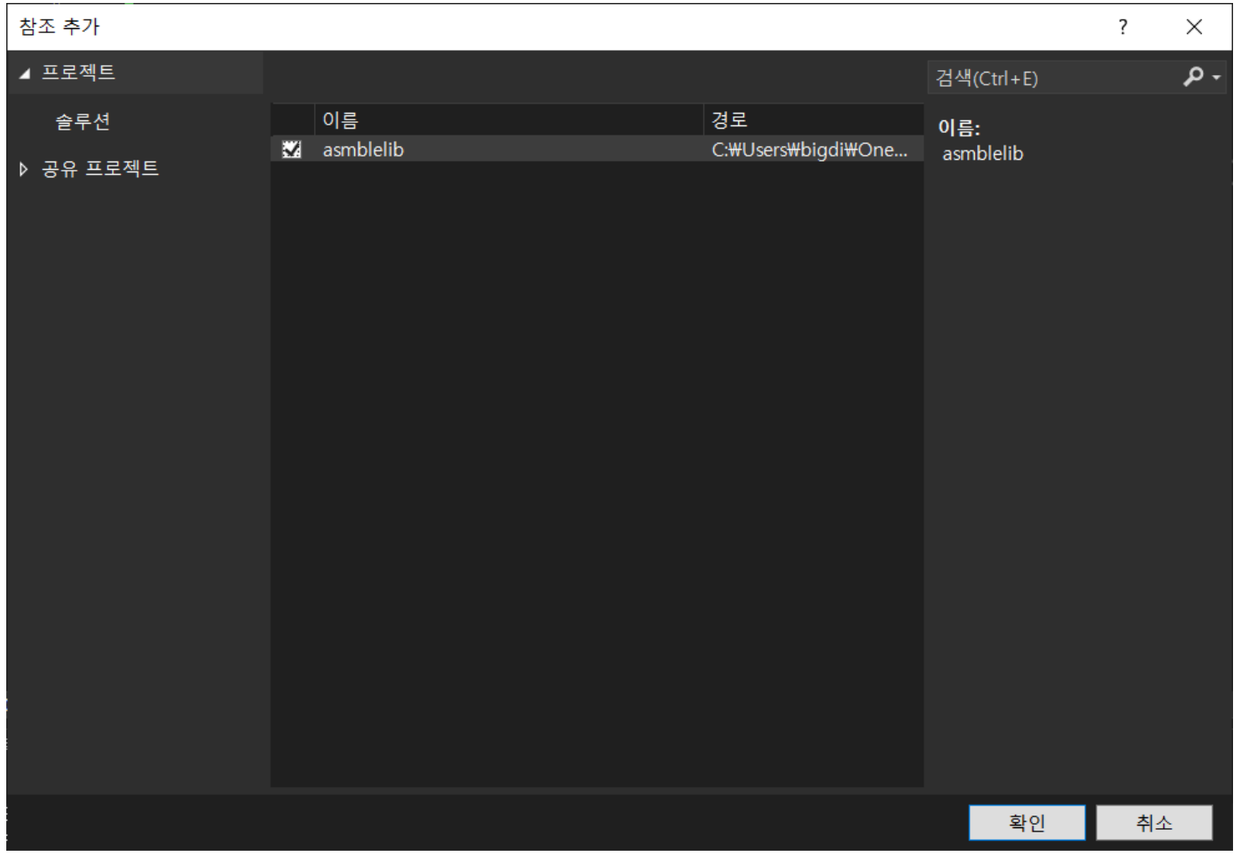
2.3 프로젝트에 우클릭 후 속성>구성형식을 정적 라이브러리로 변경



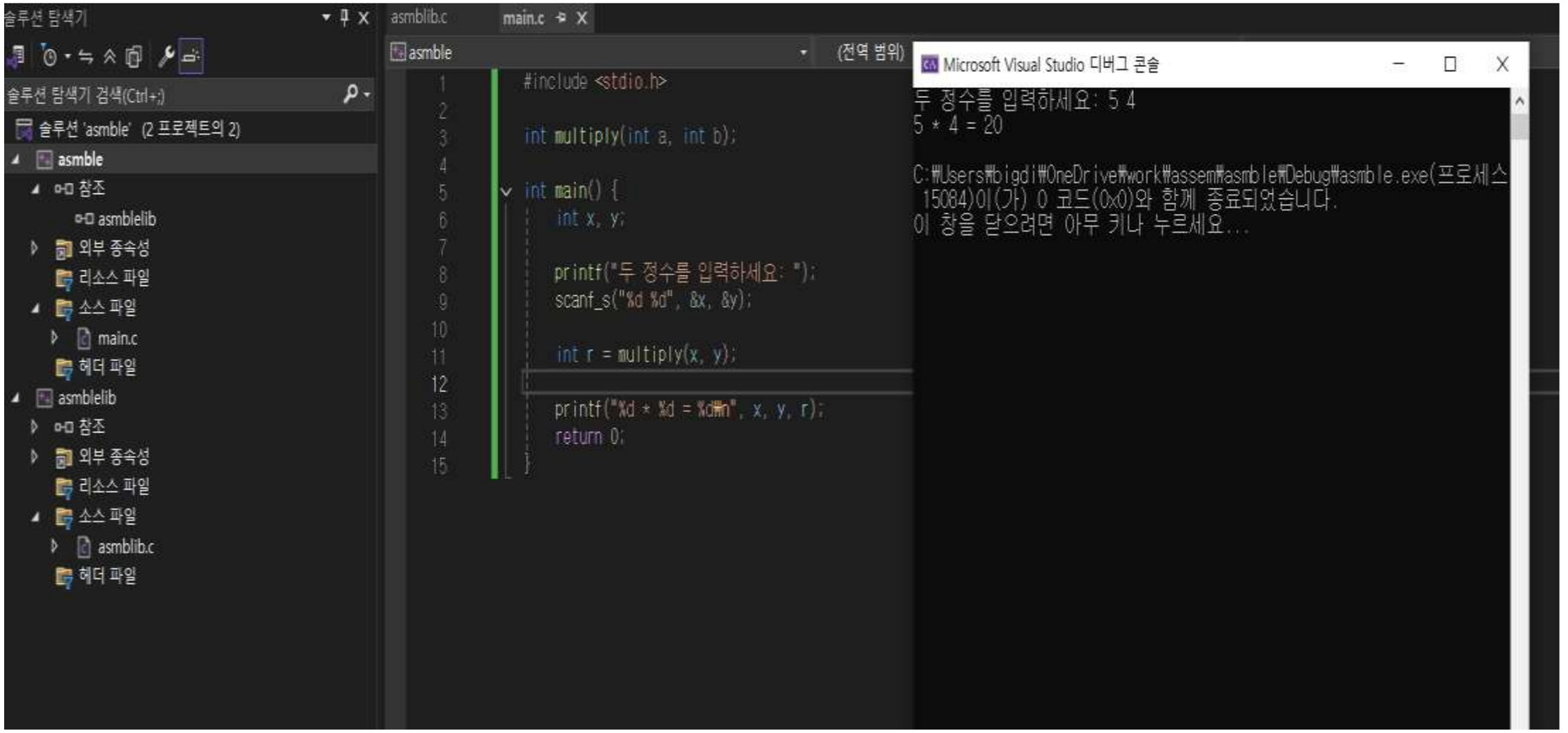
```c
#include <stdio.h>

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



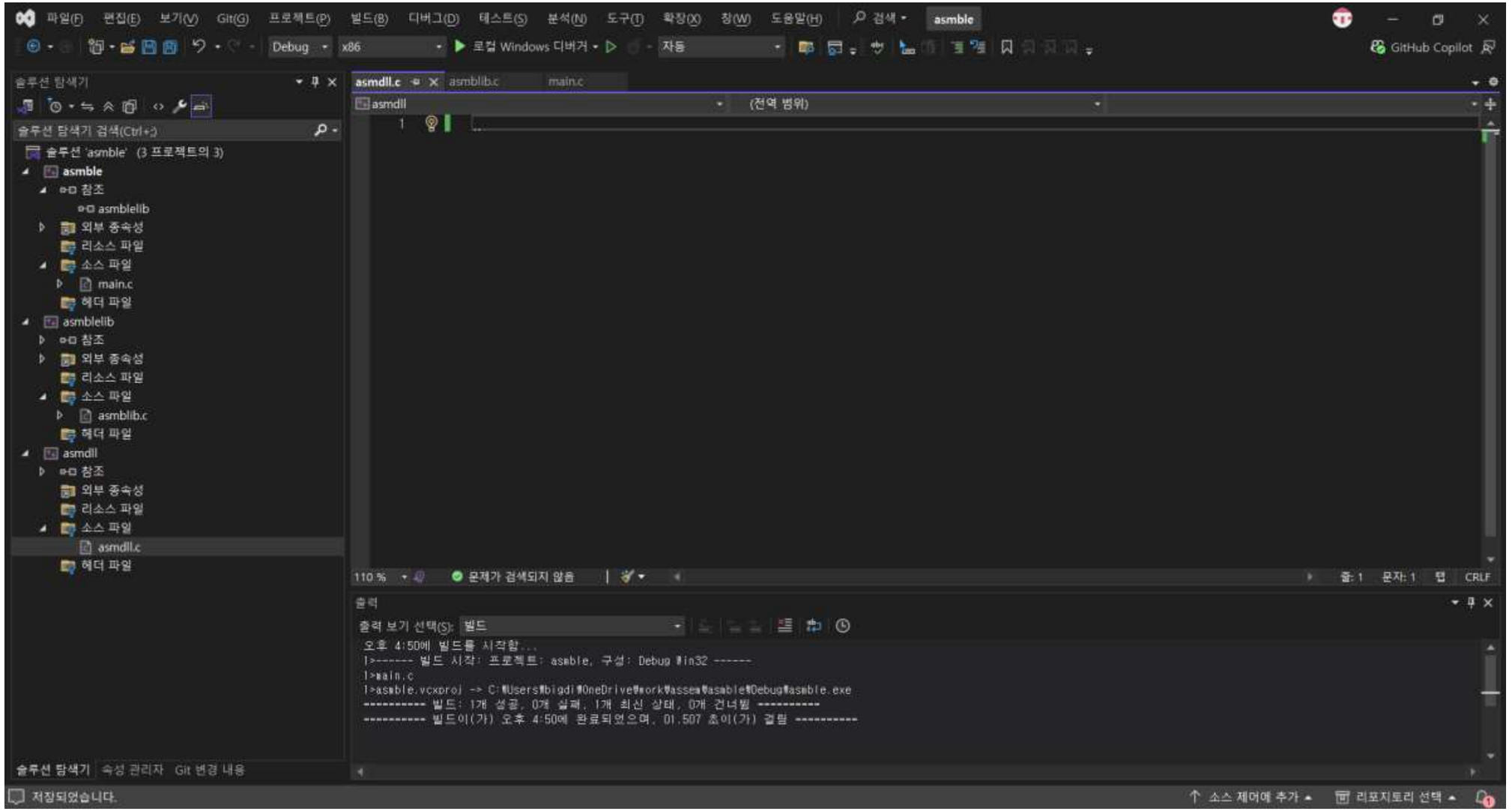


2.5 main.c가 있는 프로젝트에 전에 만들어 놓은 라이브러리를 참조하여 링킹

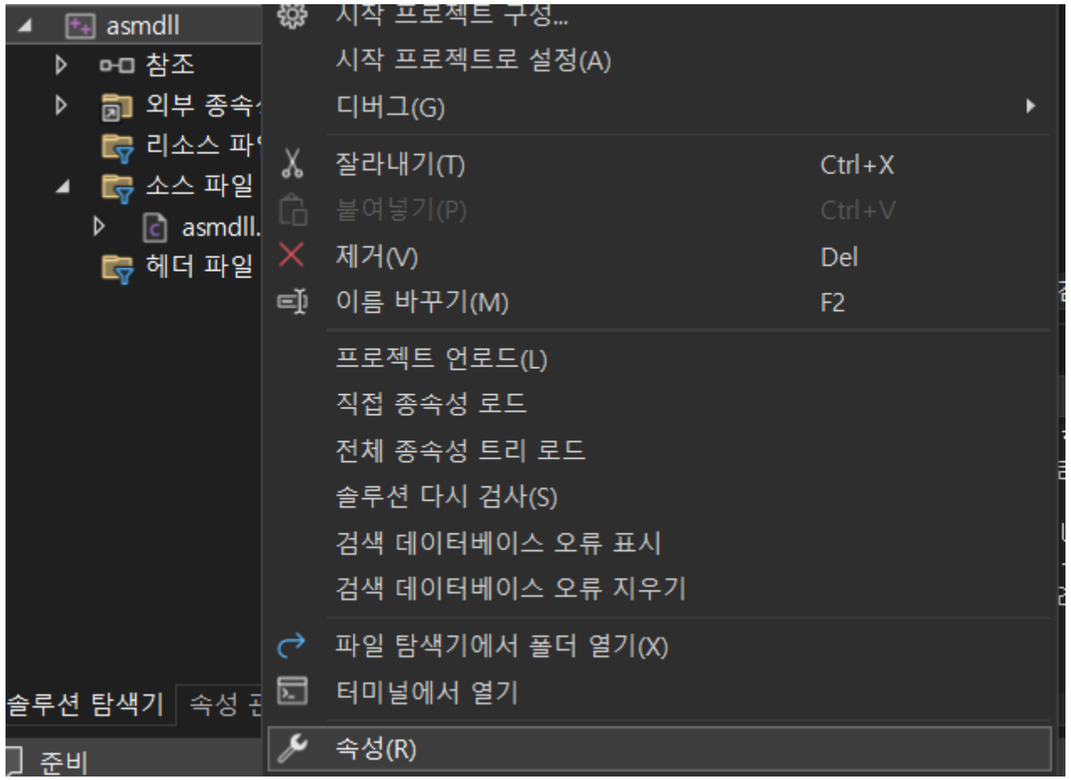


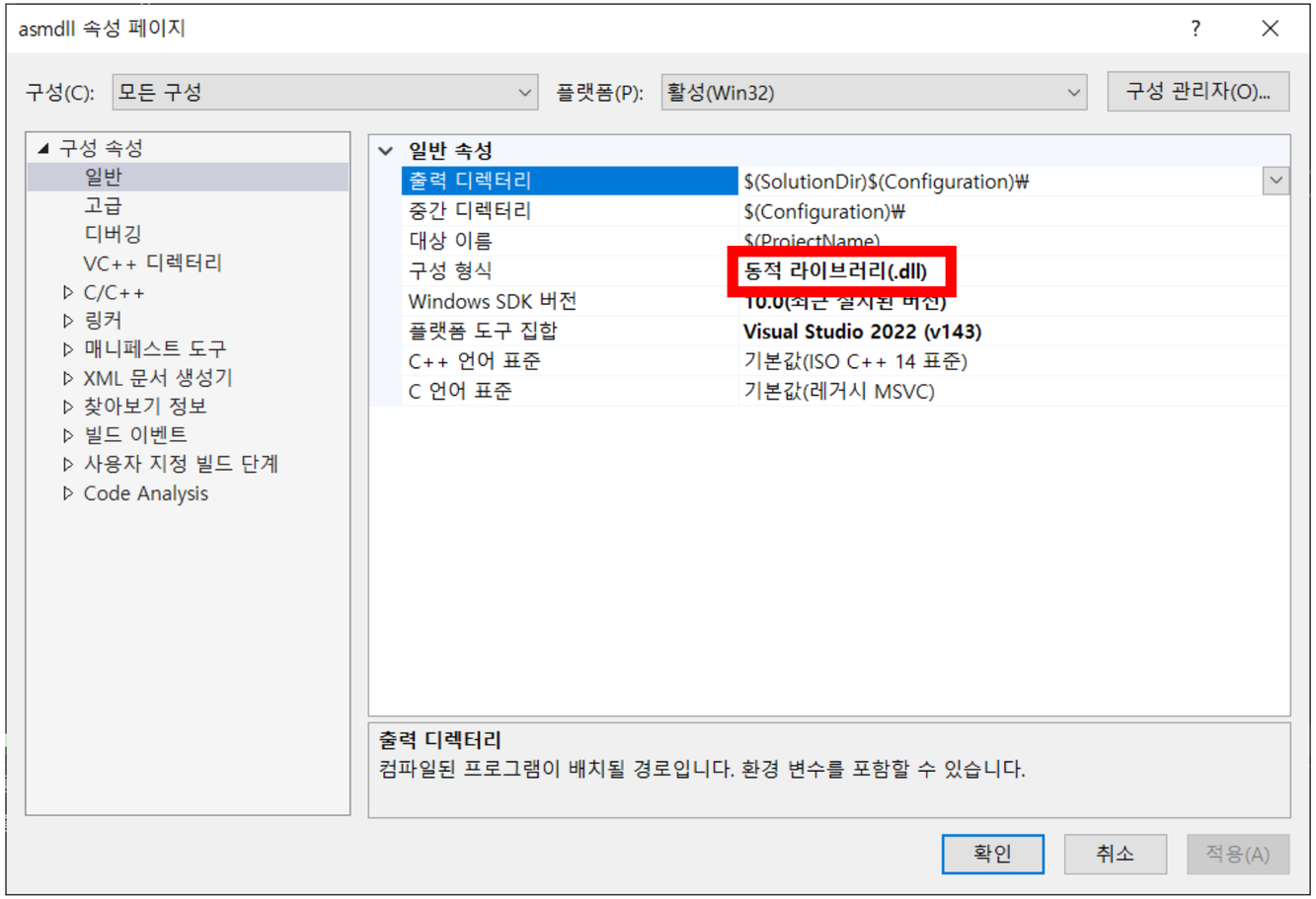
2.6 main.c 실행 시 정상작동 하는 모습

3. DDL



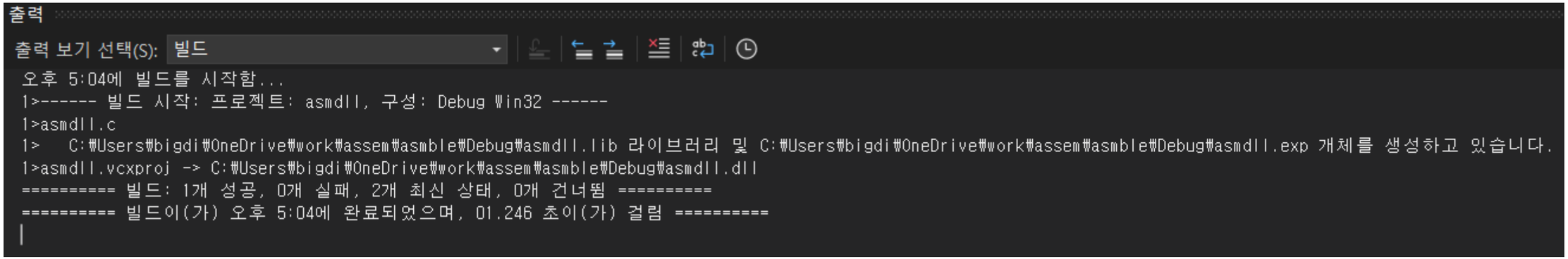
3.1 dll용 프로젝트 생성 후 asmdll.c 생성





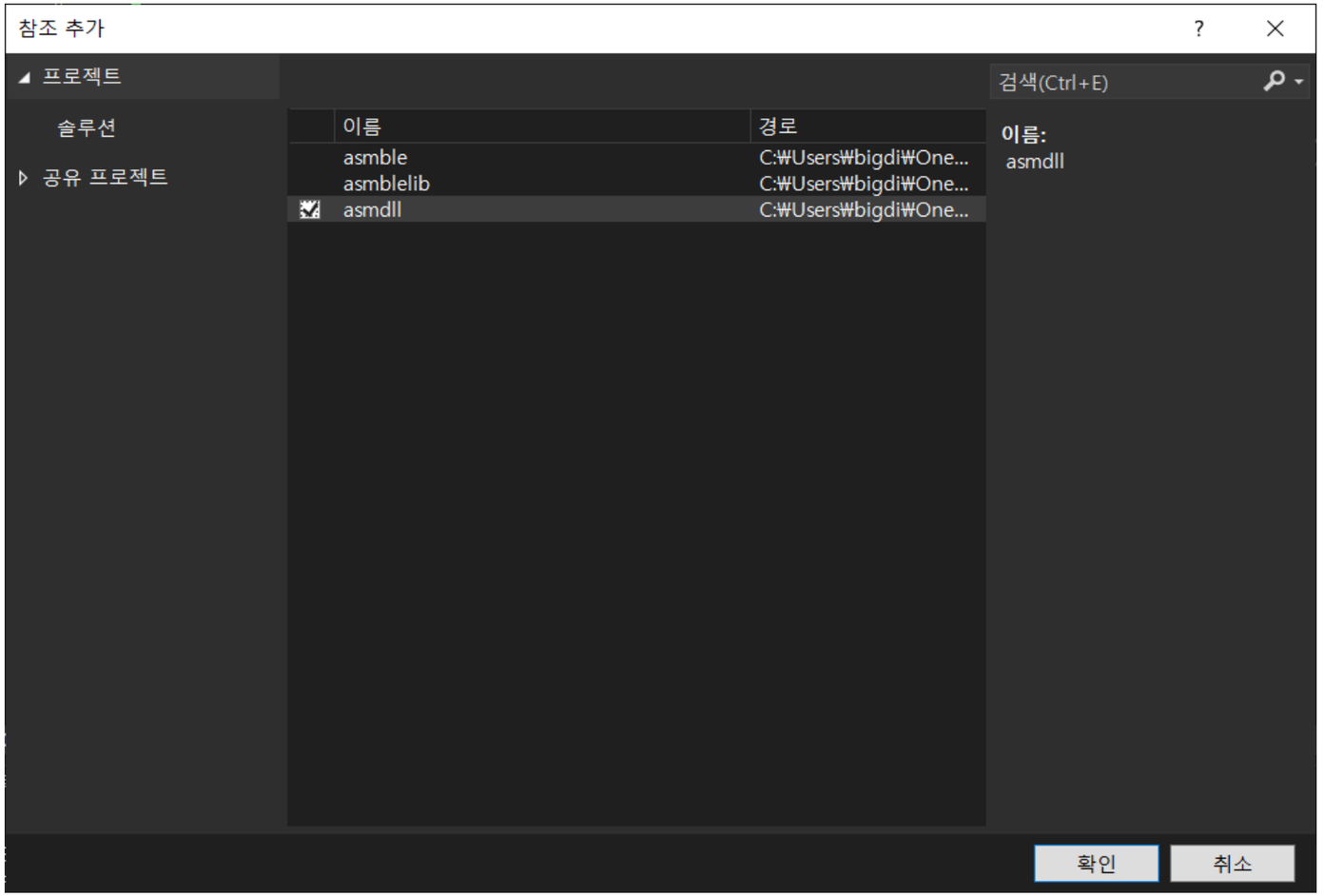
3.2 프로젝트에 우클릭 후 속성>구성형식을 동적 라이브러리로 변경

```
#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;
}
```

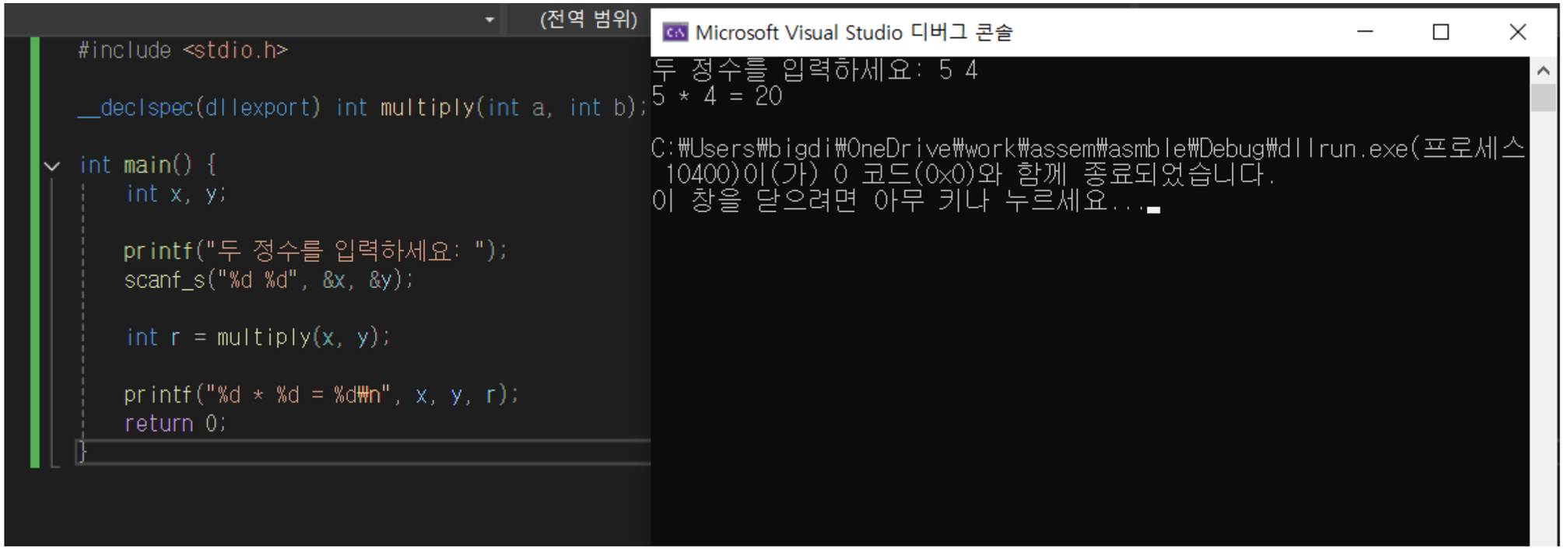


3.3 함수 앞에 __declspec(dllexport)를 추가하고 빌드하여 dll,lib 생성

```c
#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



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