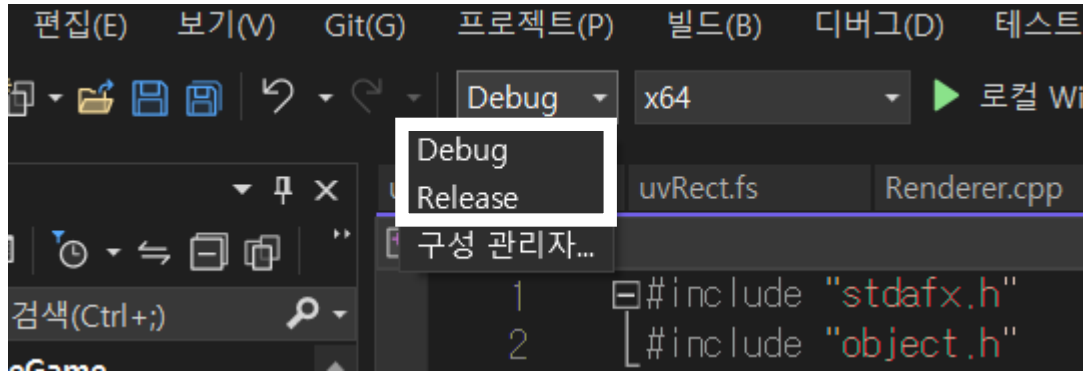


여러가지 Visual Studio 이용 팁

1. Debug 모드, Release 모드의 차이
2. 소스코드 배포 방법 (필요 없는 파일 지우기)
3. 실행 파일(.exe 파일) 배포 방법
4. ★★디버거 사용 방법★★

Debug 모드, Release 모드의 차이



- Debug 모드: 개발 도중 사용하기에 최적화 된 모드
- Release 모드: 개발이 끝나고 프로그램 배포에 최적화 된 모드

Debug 모드, Release 모드의 차이

Debug 모드

- 프로그램 실행 시 오류 체크를 위한 여러가지 과정 포함
- 코드에 최적화가 들어가지 않음
- 더 메모리를 많이 차지함
- 더 느림

Debug 모드, Release 모드의 차이

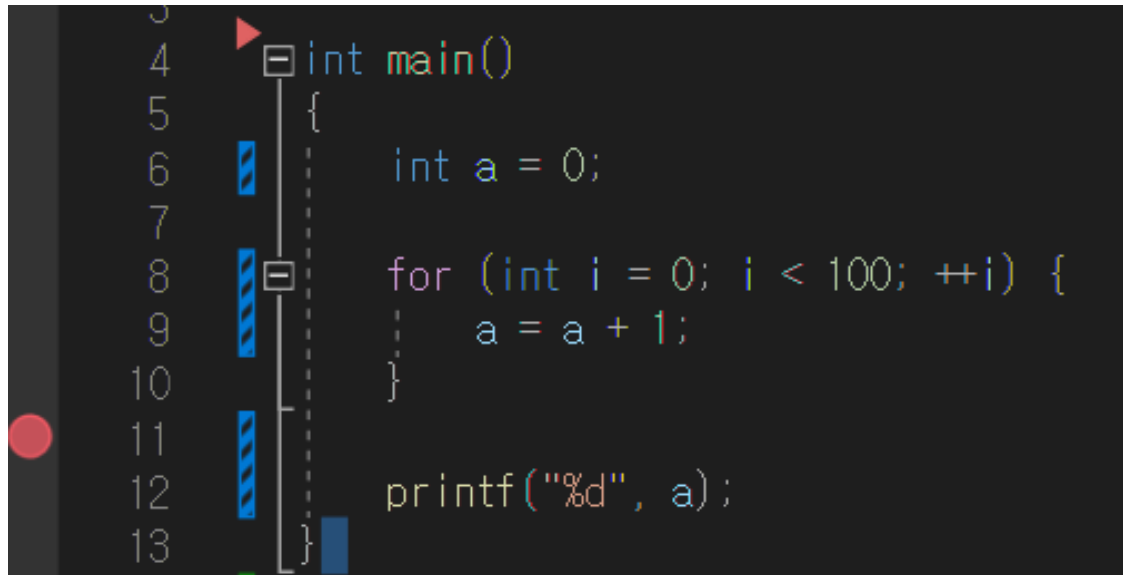
Release 모드

- 오류 체크를 위한 과정이 없음
- 불필요한 동작을 줄이는 최적화 기술
- 더 적은 메모리 사용
- 더 빠름

Debug 모드, Release 모드의 차이

개발 단계에서는 버그 체크가 필요하므로 Debug 모드를 사용하고,
버그가 없음이 확인되고 안정된 상태인 배포 단계에서는 Release
모드를 사용하자!

참고



```
3  
4 int main()  
5 {  
6     int a = 0;  
7  
8     for (int i = 0; i < 100; ++i) {  
9         a = a + 1;  
10    }  
11  
12    printf("%d", a);  
13 }
```

위 코드가 어셈블리 코드에서 실제로는 어떻게 동작할까?

참고

```
int a = 0;
00007FF6D587197B  mov     dword ptr [a],0

for (int i = 0; i < 100; ++i) {
00007FF6D5871982  mov     dword ptr [rbp+24h],0
00007FF6D5871989  jmp     main+33h (07FF6D5871993h)
00007FF6D587198B  mov     eax,dword ptr [rbp+24h]
00007FF6D587198E  inc     eax
00007FF6D5871990  mov     dword ptr [rbp+24h],eax
00007FF6D5871993  cmp     dword ptr [rbp+24h],64h ▶
00007FF6D5871997  jge     main+43h (07FF6D58719A3h)
    a = a + 1;
00007FF6D5871999  mov     eax,dword ptr [a]
00007FF6D587199C  inc     eax
00007FF6D587199E  mov     dword ptr [a],eax
}
00007FF6D58719A1  jmp     main+2Bh (07FF6D587198Bh)

printf("%d", a);
00007FF6D58719A3  mov     edx,dword ptr [a]
```

<- Debug 모드

프로그래머의 명령에 따라
a에 1을 더하는 작업을 100
회 실시해 준다.

참고

```
#include <iostream>
using namespace std;

int main()
{
00007FF7ED3B1070  sub     rsp,28h
    int a = 0;

    for (int i = 0; i < 100; ++i) {
        a = a + 1;
    }

    printf("%d", a);
00007FF7ED3B1074  mov     edx,64h
```

<- Release 모드

그냥 a에 100을 더하고 끝
낸다. (컴파일러의 최적화)

Release 모드가 빠를 수 밖에 없다.

소스코드 배포 방법 (필요 없는 파일 지우기)

 IOCP-GameServer

종류:	파일 폴더
위치:	C:\Users\이승준\Desktop\Coding
크기:	3.20GB (3,444,305,821 바이트)
디스크 할당 크기:	3.20GB (3,445,780,480 바이트)
내용:	파일 1,055, 폴더 361

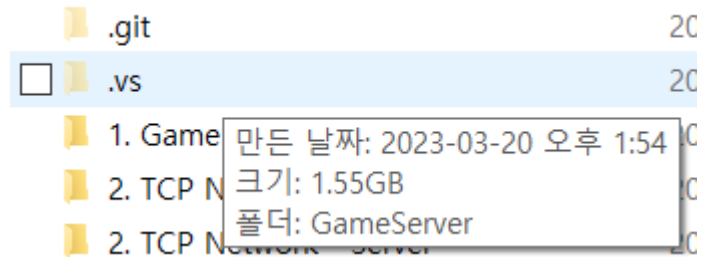
만든 날짜: 2023년 3월 6일 월요일, 오후 5:21:24

특성:	<input checked="" type="checkbox"/> 읽기 전용(폴더의 파일에만 적용)(R)
	<input type="checkbox"/> 숨김(H)

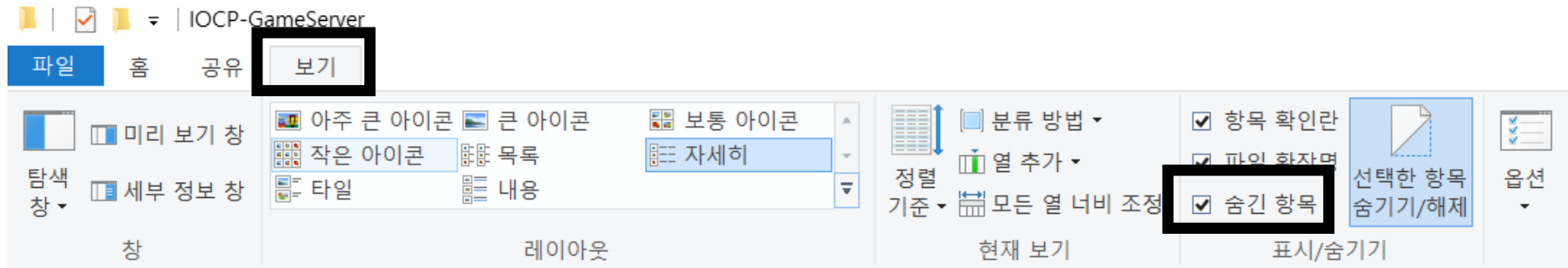
고급(D)...

<- 코드밖에 없는 파일이 3.2GB
를 차지한다.

소스코드 배포 방법 (필요 없는 파일 지우기)

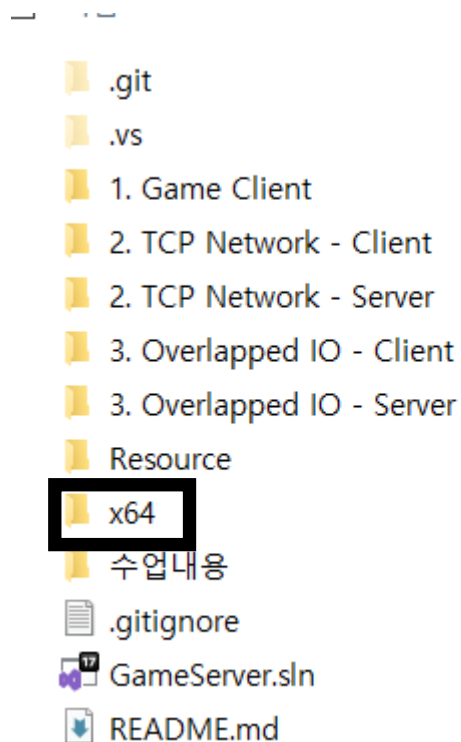


<- 폴더 내부에 들어가 보면 .vs 폴더가 1.55GB나 차지하고 있는 것을 볼 수 있다.
다 지워주자.



(.vs 폴더가 안 보인다면 보기->숨긴 항목을 체크해 주도록 하자)

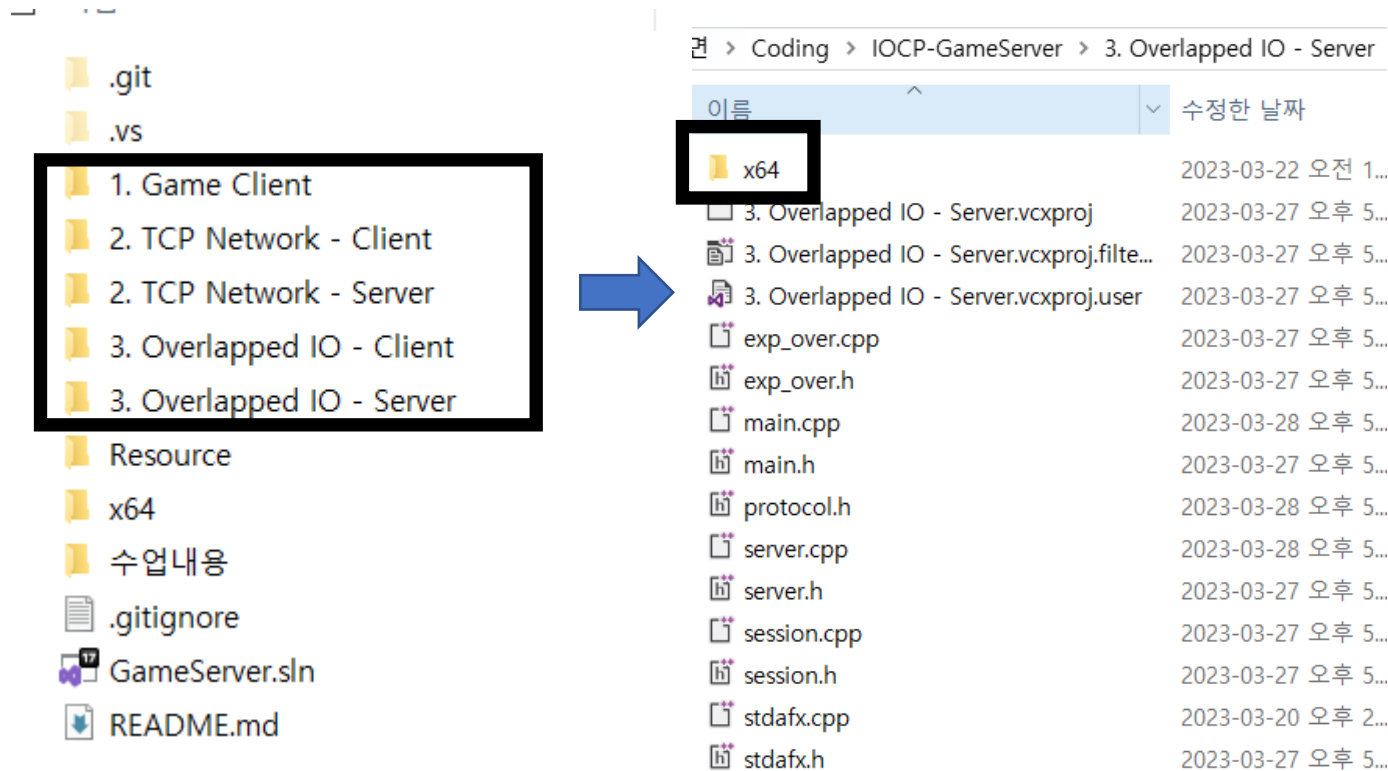
소스코드 배포 방법 (필요 없는 파일 지우기)



다시 솔루션 폴더가 있는 경로로 돌아와서, x64, Debug, Release 폴더를 싹 지워준다.

여기에는 컴파일 시그 프로그램의 실행 파일(.exe)이 저장되는데, 컴파일 시마다 새로 생성되므로 지워도 상관 없다.

소스코드 배포 방법 (필요 없는 파일 지우기)



마지막으로 프로젝트 폴더 안
으로 들어와서 x64, Debug,
Release 폴더를 지워준다.

소스코드 배포 방법 (필요 없는 파일 지우기)



<- 정리를 완료한 폴더의 크기이다.

(해당 폴더에는 이미지 등 리소스가 포함되어 있기 때문에 실제로는 KB 단위까지 줄어들 수도 있다.)

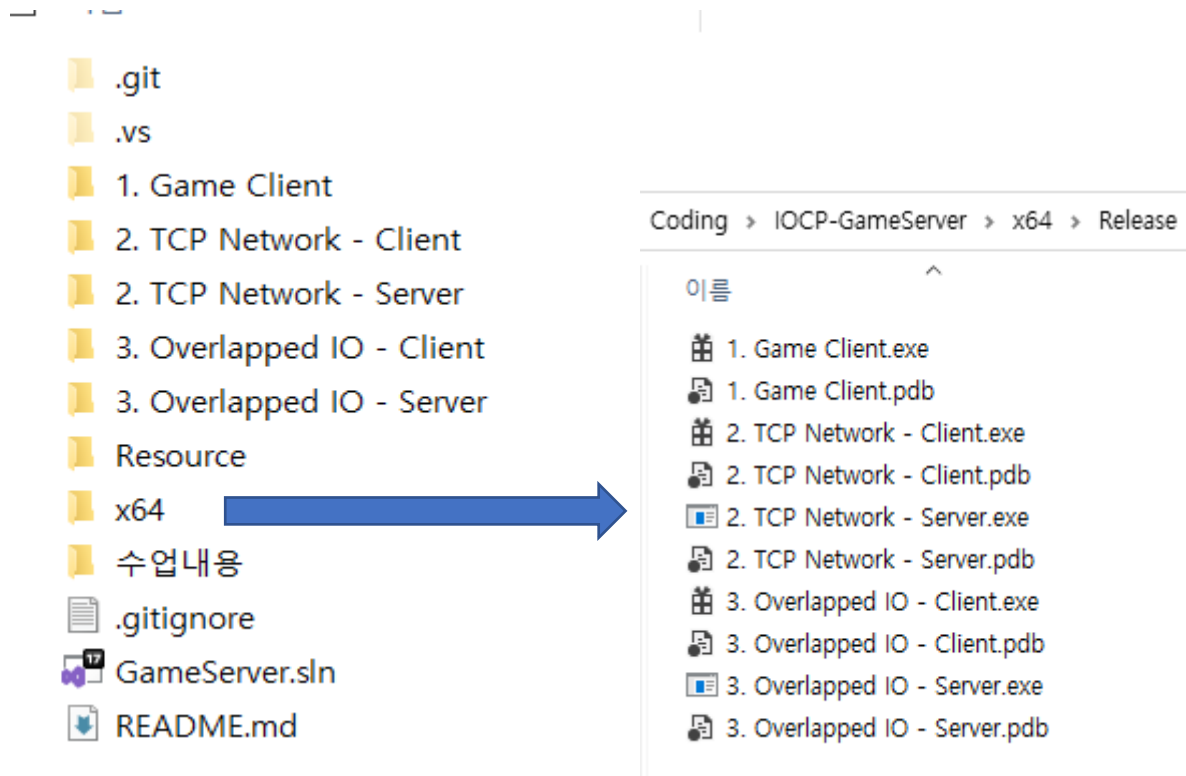
실행 파일(.exe 파일) 배포 방법

- *+ fadeFilter.cpp
- ▢ fadeFilter.h
- *+ framework.cpp
- ▢ framework.h
- *+ globalLoadingScene.cpp
- ▢ globalLoadingScene.h
- *+ light.cpp
- ▢ light.h
- *+ loginScene.cpp
- ▢ loginScene.h
- *+ material.cpp
- ▢ material.h
- *+ mesh.cpp
- ▢ mesh.h
- *+ monster.cpp
- ▢ monster.h
- *+ object.cpp
- ▢ object.h
- *+ particle.cpp
- ▢ particle.h
- *+ particleMesh.cpp
- ▢ particleMesh.h
- *+ particleSystem.cpp
- ▢ particleSystem.h
- *+ player.cpp

게임을 만들었는데 소스코드를 실행하기 위해 이용자가 비주얼 스튜디오를 설치하고 코드를 직접 실행해야 하는 일이 생기면 무척 번거로울 것이다.

다행히 컴파일 시에 Visual Studio가 실행 파일을 직접 생성해준다.

실행 파일(.exe 파일) 배포 방법



앞서 솔루션 폴더가 있는 경로에 있는 x64, Debug, Release 폴더에 실행 파일(.exe)이 저장된다고 서술했는데, 바로 이 실행 파일을 배포하면 된다.

같이 들어있는 .pdb 파일은 실행 파일의 디버그 정보가 포함된 파일로, 포함하지 않아도 상관 없다.

언급했듯이, 성능을 위해 가급적 Release 폴더에 있는 실행 파일을 배포하도록 하자.

Coding > IOCP-GameServer > 1. Game Client		
이름	수정한 날짜	
x64	2023-03-19 오전 12:00	
1. Game Client.aps	2023-03-13 오후 12:48	
1. Game Client.cpp	2023-03-14 오후 7:57	
1. Game Client.h	2023-03-14 오후 7:57	
1. Game Client.ico	2023-03-14 오후 7:57	
1. Game Client.rc	2023-03-14 오후 7:57	
1. Game Client.vcxproj	2023-04-09 오전 12:29	
1. Game Client.vcxproj.filters	2023-03-14 오후 7:57	
1. Game Client.vcxproj.user	2023-03-12 오후 1:17	
board.cpp	2023-03-14 오후 7:57	
board.h	2023-03-14 오후 7:57	
Chessboard.png	2023-03-14 오후 7:57	
framework.cpp	2023-04-09 오전 12:29	
framework.h	2023-03-14 오후 7:57	
mainScene.cpp	2023-03-19 오전 12:01	
mainScene.h	2023-03-14 오후 7:57	
object.cpp	2023-03-14 오후 7:57	
object.h	2023-03-14 오후 7:57	
piece.cpp	2023-03-14 오후 7:57	
piece.h	2023-03-14 오후 7:57	
Piece.png	2023-03-14 오후 7:57	
Resource.h	2023-03-14 오후 7:57	
scene.cpp	2023-03-14 오후 7:57	
scene.h	2023-03-14 오후 7:57	
small.ico	2023-03-14 오후 7:57	
stdafx.cpp	2023-03-14 오후 7:57	
stdafx.h	2023-03-14 오후 7:57	

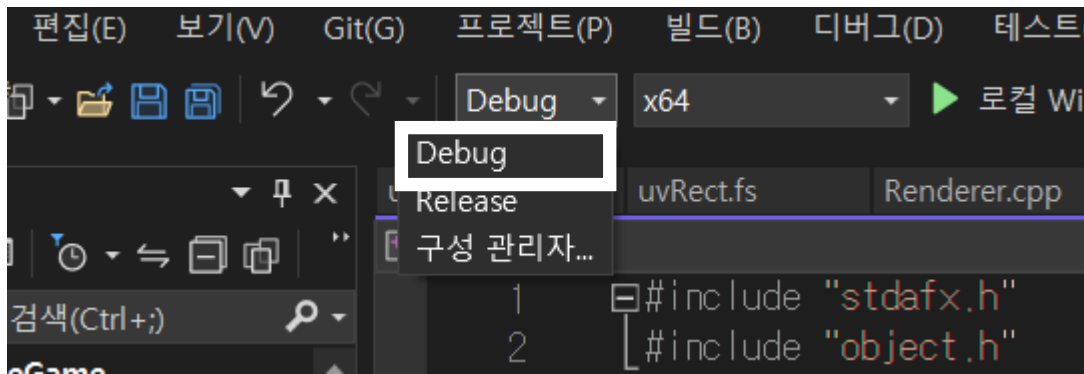


Coding > IOCP-GameServer > x64 > Release		
이름	수정한 날짜	유형
1. Game Client.exe	2023-03-19 오전 12:01	응용 프로그램
Chessboard.png	2023-03-14 오후 7:57	PNL
Piece.png	2023-03-14 오후 7:57	PNL

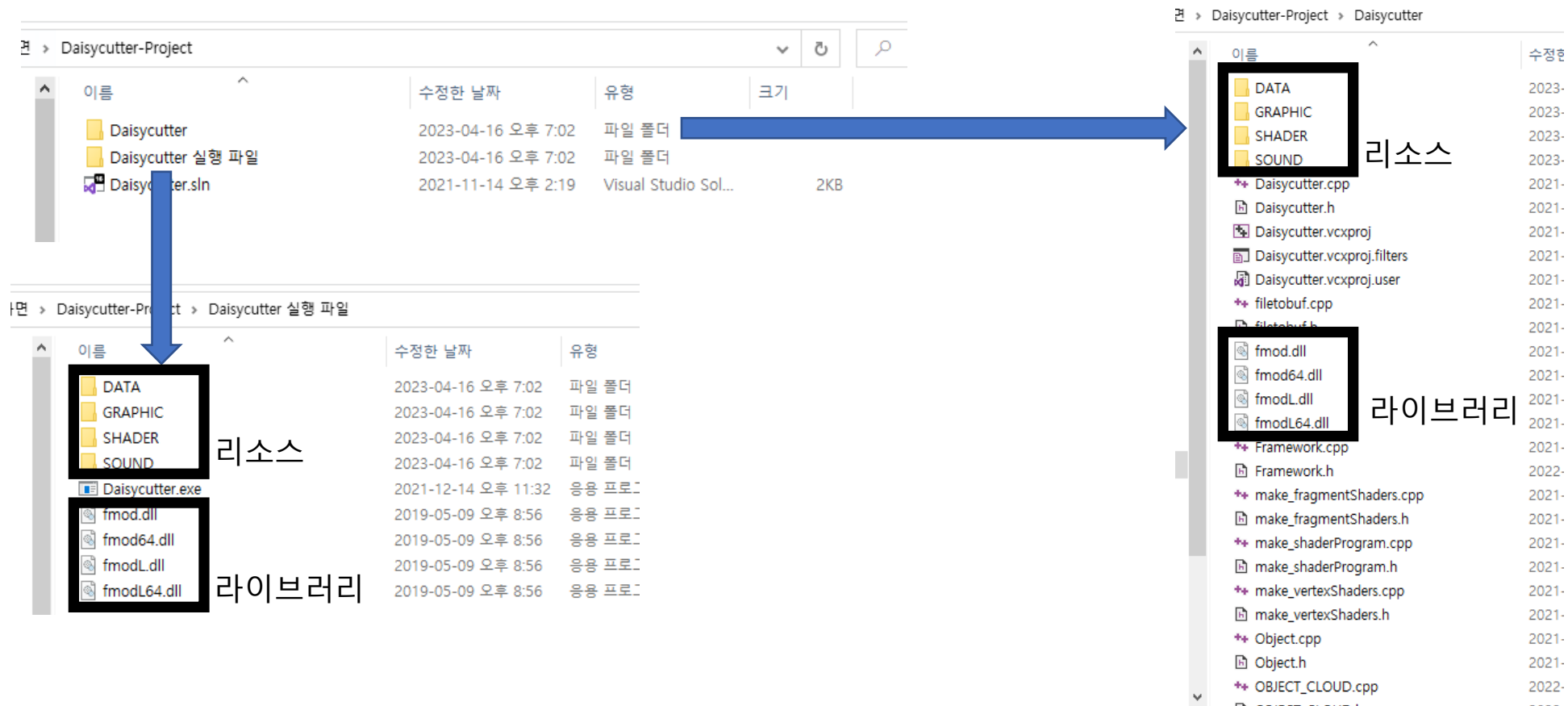
만약 소스 코드를 실행하는 데 이미지나 라이브러리 등의 리소스가 필요하다면, 배포할 때도 이 리소스를 포함해 줘야 한다.

이 때 소스 코드의 경로와 실행 파일(.exe)의 경로가 같다고 생각하고 리소스를 배치해 주면 된다.

실행 파일을 배포하기 전에 실행 파일을 다른 경로로 옮겨
제대로 실행 되는지 꼭 확인해 보자.
마지막으로 컴파일 된 실행 파일이 저장됨을 잊지 말자!
컴파일 하지 않고 저장만 한다면 새 실행 파일은 생성되지 않는다.
옛날 버전을 배포하는 실수를 하지 말자!

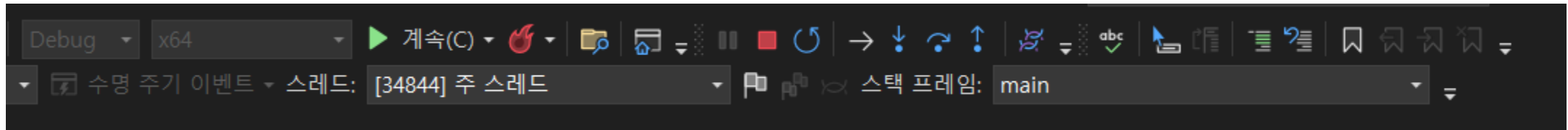


디버그 모드로 컴파일하면 릴리즈 폴더는 생성되지 않는다.
Debug / Release 어떤 상태로 컴파일 중인지 항상 체크하자!



온라인으로 과제 제출 시 이와 같이 소스코드를 정리하고, 실행 파일(.exe)을 포함한 후 압축해서 제출하면 된다.

디버거 사용 방법



비주얼 스튜디오는 원하는 시점에 프로그램을 멈추고, 현재 어떤 변수의 값이 어떤 가를 확인할 수 있는 디버거를 제공한다.

중단점

```
int main()
{
    int a = INT_MAX - 5;

    for (int i = 0; i < 10; ++i) {
        a += 1;
    }

    cout << a << endl;
}
```



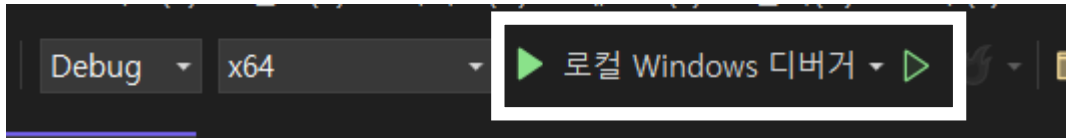
해당 코드를 실행해 보면 매우 큰 값이 출력 될 것이라고 생각했지만 예상과는 다르게 매우 작은 값이 출력된다.

디버깅 해 보자.

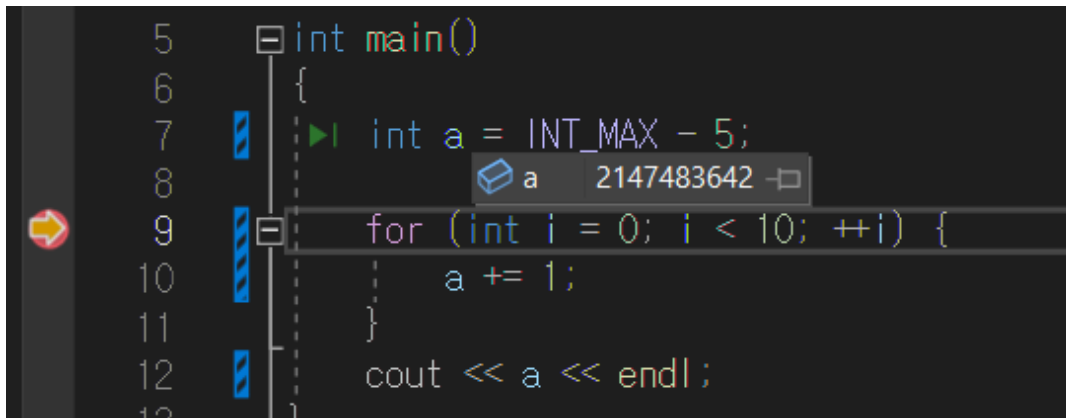
프로그램을 멈추고 싶은 코드의 좌측에 마우스 좌클릭을 한다.

빨간 원이 생기면 정상적으로 중단점이 설정 된 것이다.

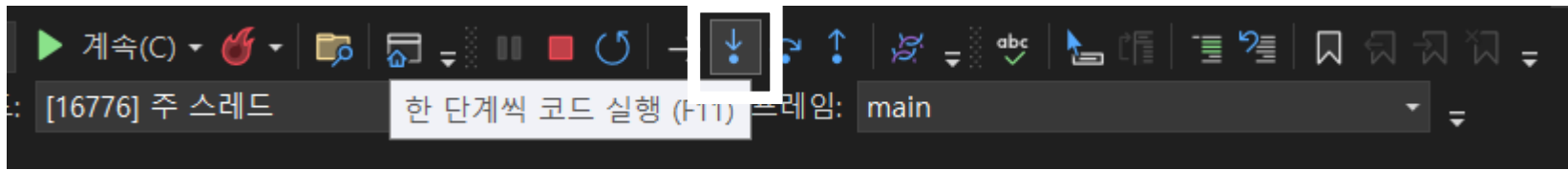
```
5 int main()
6 {
7     int a = INT_MAX - 5;
8
9     for (int i = 0; i < 10; ++i) {
10         a += 1;
11     }
12     cout << a << endl;
13 }
14
```



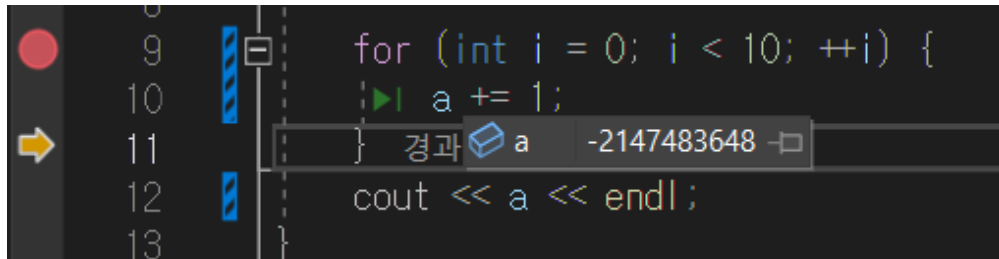
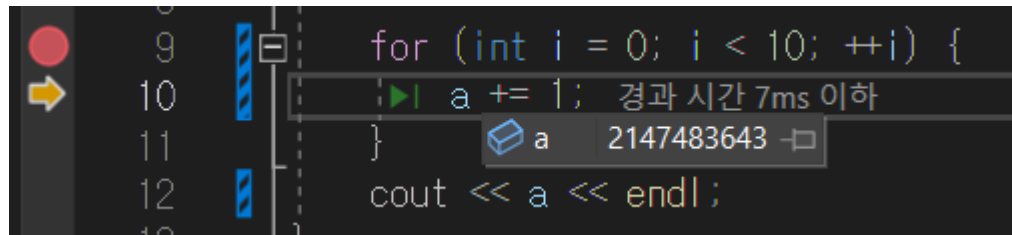
이 상태로 'F5' 또는 상단의 '로컬 Windows 디버거' 버튼을 클릭한다.



해당 코드에 첫 번째로 도착했을 때 프로그램이 멈추게 되며, 이 상태에서 변수 위에 마우스를 올리면 현 시점에서 변수의 값이 어떤 지 확인할 수 있다.

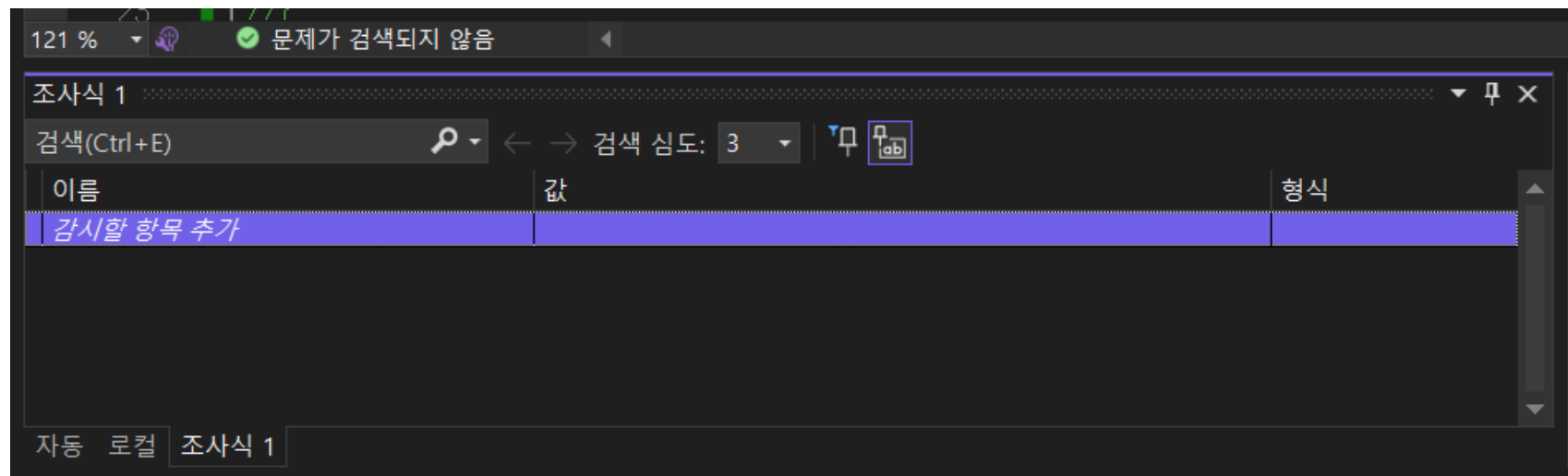


이 상태로 'F11' 또는 상단의 '한 단계씩 코드 실행' 버튼을 클릭하면 코드가 한 줄 씩 실행된다.

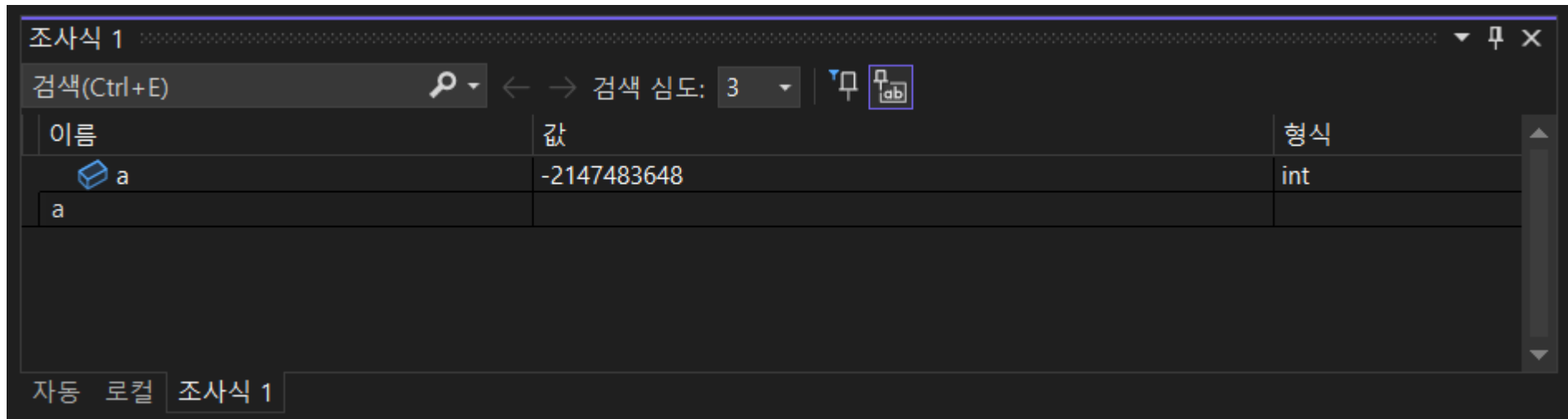


한 줄 씩 실행하며 for문에서 증가하는 a의 값을 확인해 보면, i가 5가 되는 시점에서 스택 오버플로우가 발생했음을 확인할 수 있다.

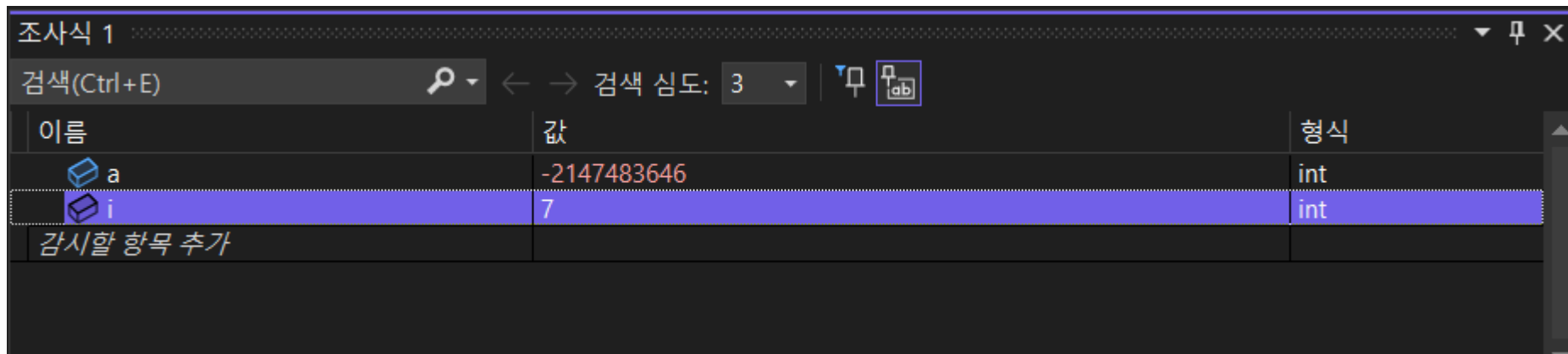
조사식



조사식을 통해 디버거가 실행되는 도중 특정 변수의 변화를 계속 관찰할 수 있다.



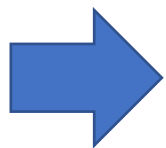
변수를 드래그&드롭 하거나 직접 입력하면 조사식에 변수가 등록된다.



조사식을 통해 코드의 진행에 따른 변수의 변화를 쉽게 관찰할 수 있다.

호출 스택

```
5 void func1()
6 {
7     int* i = nullptr;
8     *i = 3;
9 }
10 void func2()
11 {
12     func1();
13 }
14 int main()
15 {
16     func2();
17 }
18
```



```
void func1()
{
    int* i = nullptr;
    *i = 3;
}

void func2()
{
    func1();
}

int main()
{
    func2();
}
```

예외가 발생함

예외가 throw됨: 쓰기 액세스 위반입니다.
i이(가) nullptr였습니다.

호출 스택 표시 | 세부 정보 복사 | Live Share 세션을

▲ 예외 설정

- ☒ 이 예외 형식이 throw되면 중단

(주소가 없는 위치에 대입하려 하고 있음)

중단점이 설정되어 있지 않더라도, 디버거 실행 중 프로그램에 오류가 발생하면 자동으로 프로그램이 멈춘다.

```
5 void func1()
6 {
7     int* i = nullptr;
8     *i = 3;
9 }
10 void func2()
11 {
12     func1();
13 }
14 int main()
15 {
16     func2();
17 }
18
```



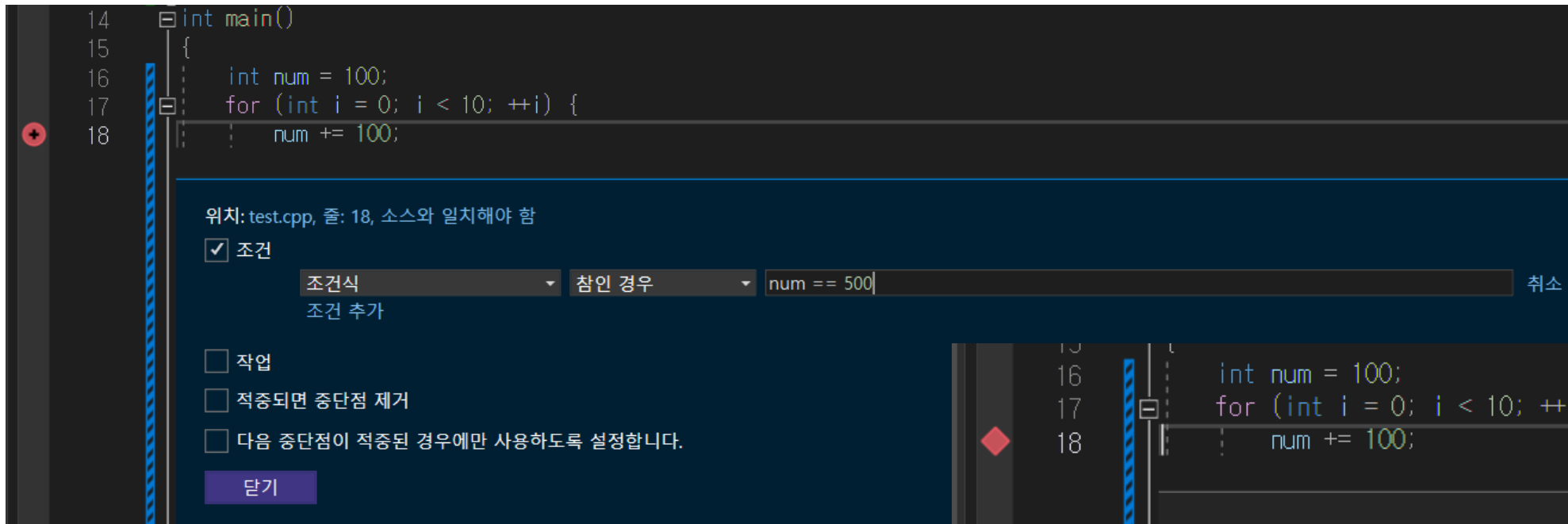
호출 스택

이름	언어
Algorithm.exe!func1() 줄 8	C++
Algorithm.exe!func2() 줄 13	C++
Algorithm.exe!main() 줄 17	C++
[외부 코드]	

호출 스택 중단점 예외 설정 명령 창 직접 실행 창 출력

이와 같은 이유나 중단점 적중 등의 이유로 프로그램이 정지했을 때, '호출 스택' 창에서 어떤 과정을 통해 현재 함수에 들어오게 되었는지 확인할 수 있다.

호출 스택을 확인해 보면 main() -> func2() -> func1() 과정을 통해 func1() 함수에 들어왔음을 확인할 수 있는데, 이는 복잡한 프로그램에서 오류의 발생 흐름을 추적하는데 큰 도움이 된다.



이 밖에도 Visual Studio는 여러 가지 편리한 디버깅 기능을 제공한다. 이를 통해 프로그램 작성 시 발생하는 오류를 보다 쉽게 추적할 수 있다.

