

공통과제를 위한 Google Test 세팅 가이드

C++ 진영에서 가장 많이 사용되는 Unit Test Framework

- Google의 Testing Technology 팀에서 만든 C++ Test Framework
- Windows / Linux / Mac 등 다양한 플랫폼 사용 가능

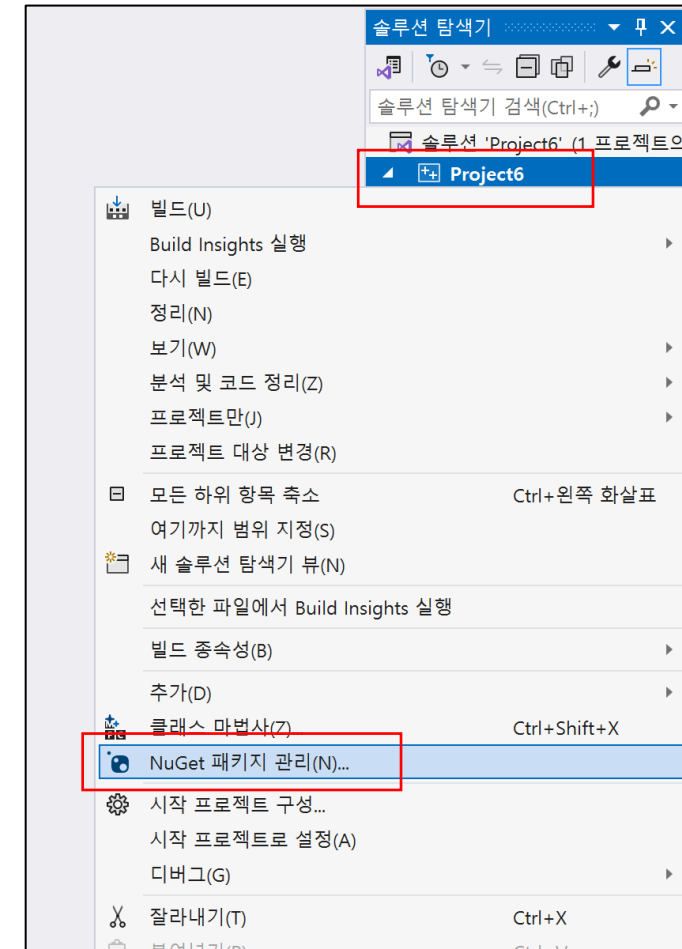
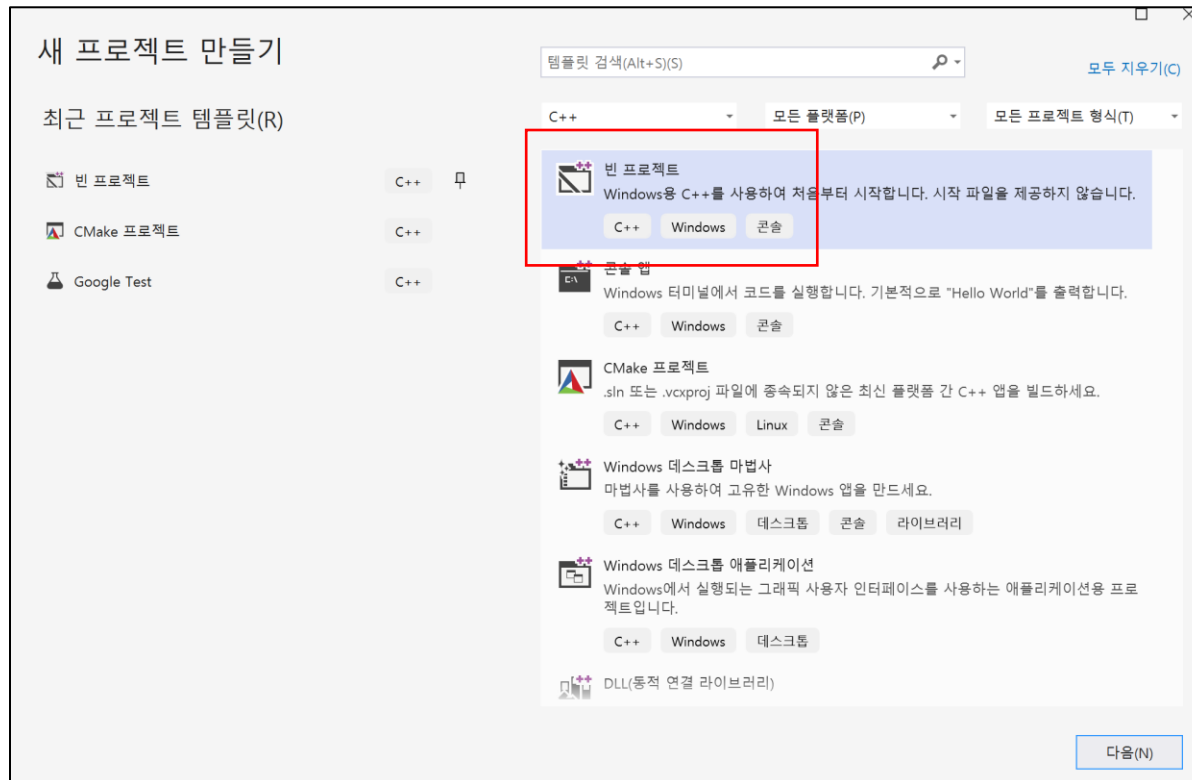
Google Test 사용 가이드 (Github & Document)

<https://github.com/google/googletest>

<https://google.github.io/googletest/>

Google Test 설치 방법 - 1

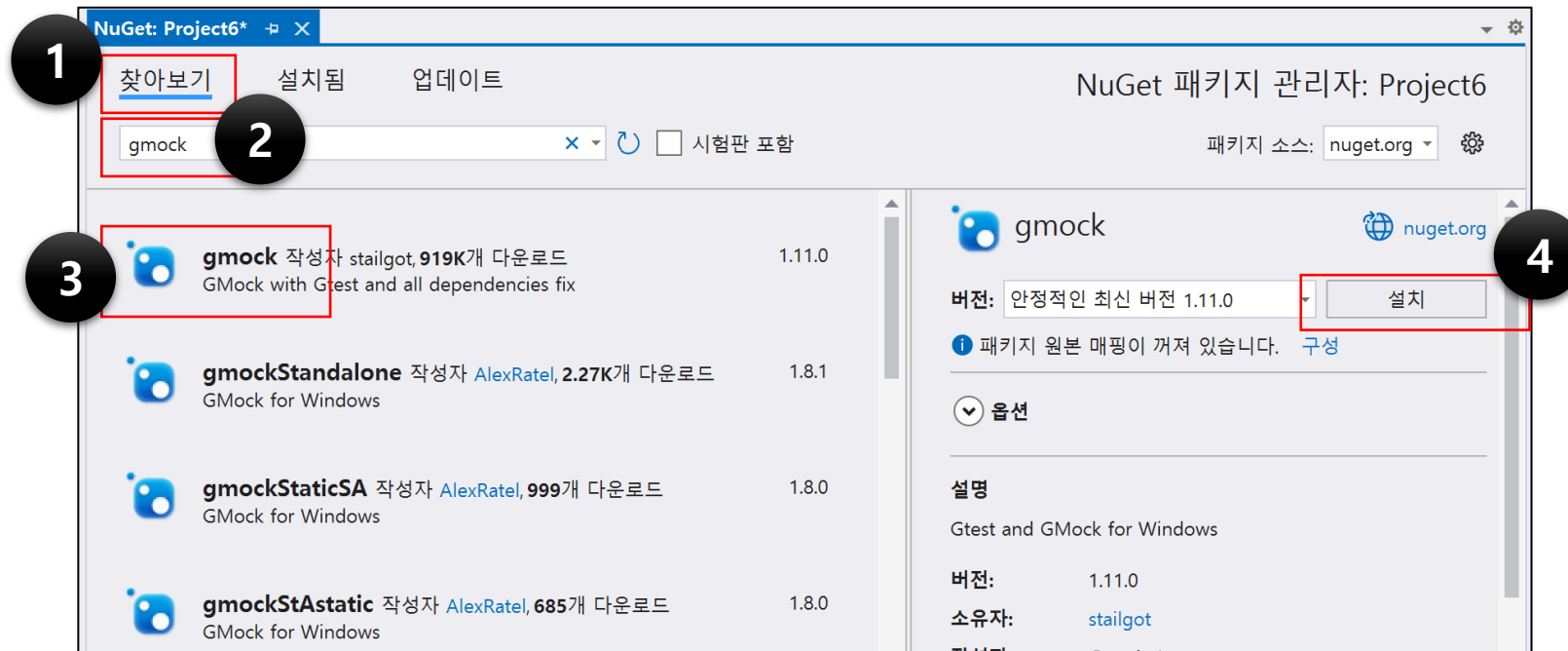
- 빈 프로젝트 생성
- 프로젝트 > NuGet 패키지관리



Google Test 설치 방법 - 2

NuGet 패키지 관리자

MS 개발환경에서 Library 를 쉽게 설치할 수 있도록 만들어진 도구
gmock 을 설치한다. (5초 소요)

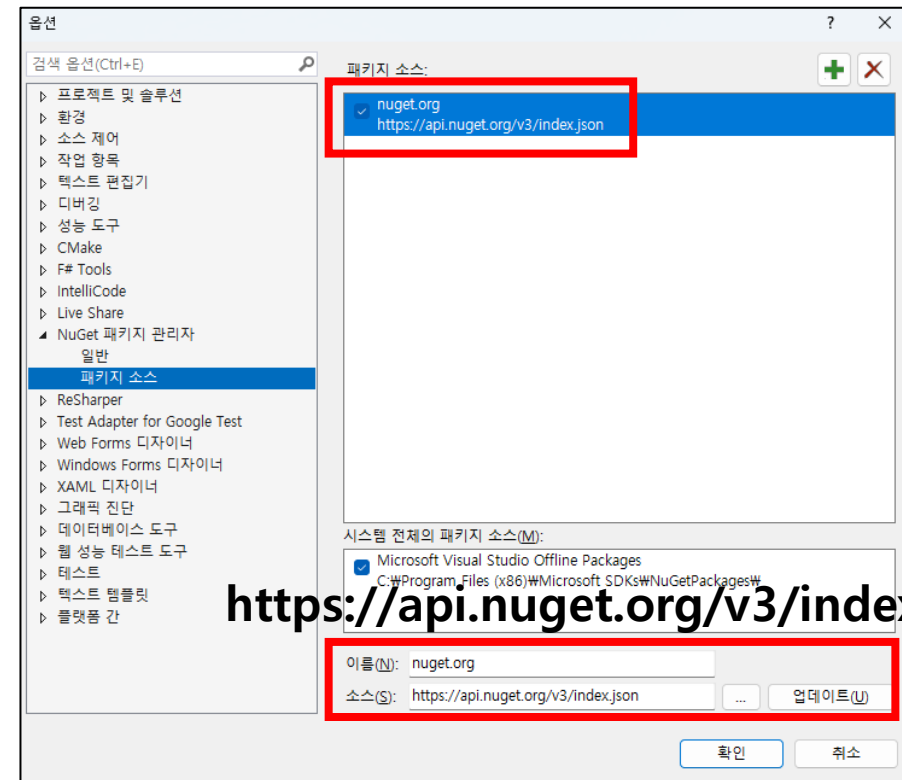
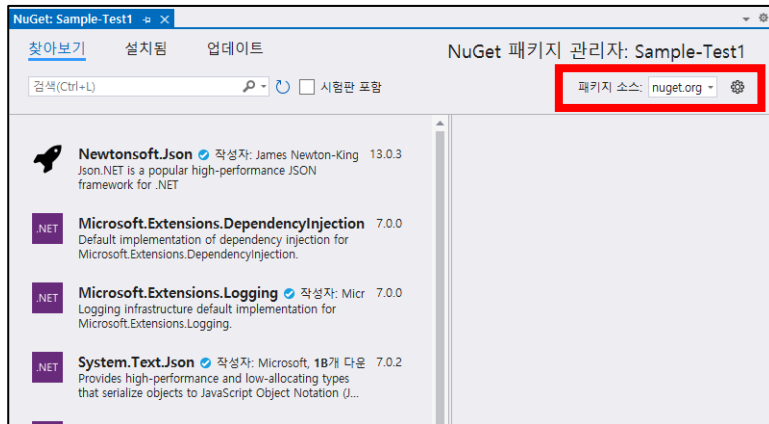


[Trouble Shooting] NuGet 저장소 확인

Nuget에 검색이 안되는 경우

현재 다음과 같이 패키지 소스 경로가 맞는지 확인한다.

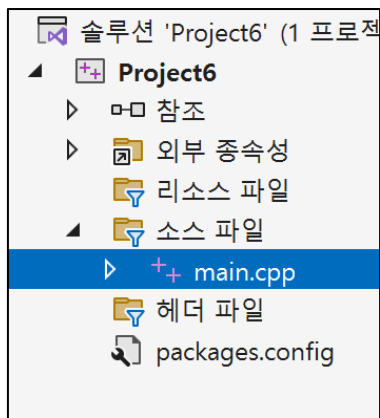
다르다면 아래와 같이 수정한다.



Google Test 설치 방법 - 3

기본 코드 작성

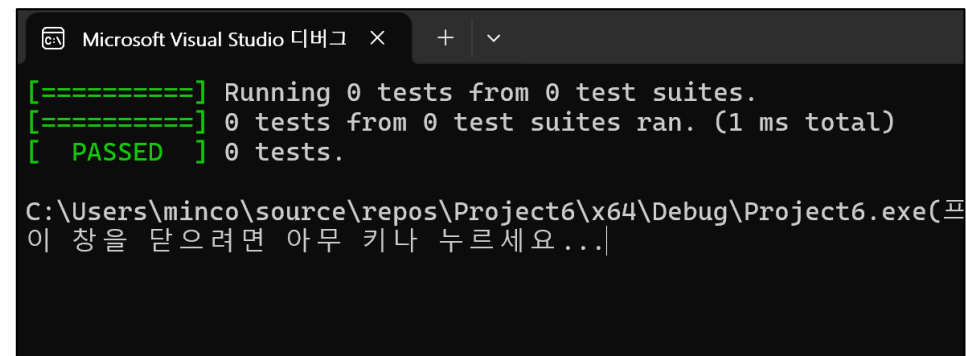
main 코드를 작성하여 Google Test가 잘 동작되는지 확인한다.



```
#include "gmock/gmock.h"

int main() {
    testing::InitGoogleMock();
    return RUN_ALL_TESTS();
}
```

main.cpp 작성

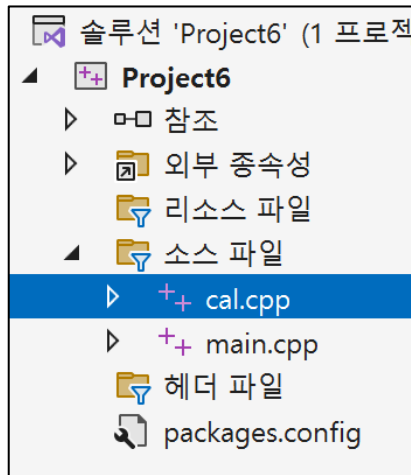


실행결과

테스트 대상이 되는 코드 추가

두 수의 합을 구하는 클래스를 하나 추가한다.

Header에 선언부, Source파일에 정의부로 구현해야하지만,
개발 편의상 cpp파일에 모두 구현한다.



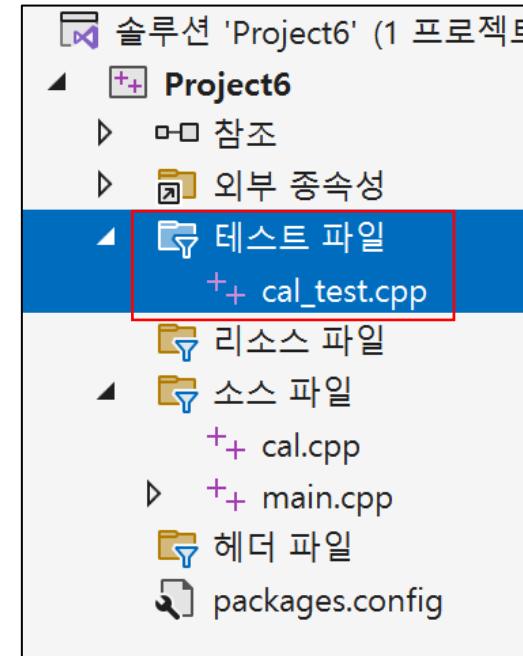
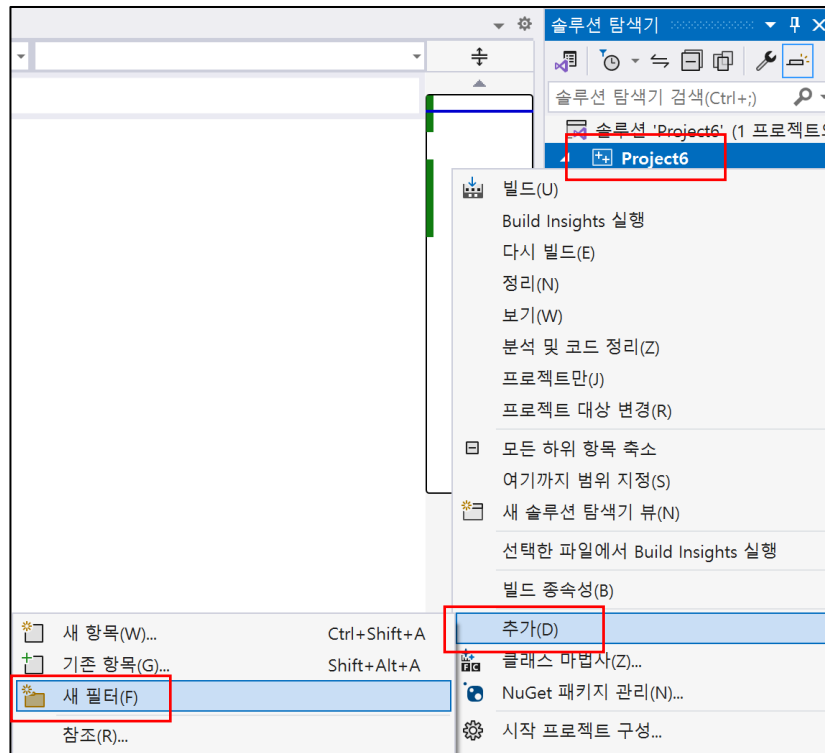
```
class Cal {  
public:  
    int getSum(int a, int b) {  
        return a + b;  
    }  
};
```

cal.cpp 추가하기

Google Test 설치 방법 - 5

필터와 Test 파일 추가하기

폴더 아이콘의 필터를 추가한다. (이름 : 테스트 파일)
필터 내부에 cal_test.cpp 파일을 추가한다.



Google Test 설치 방법 - 6

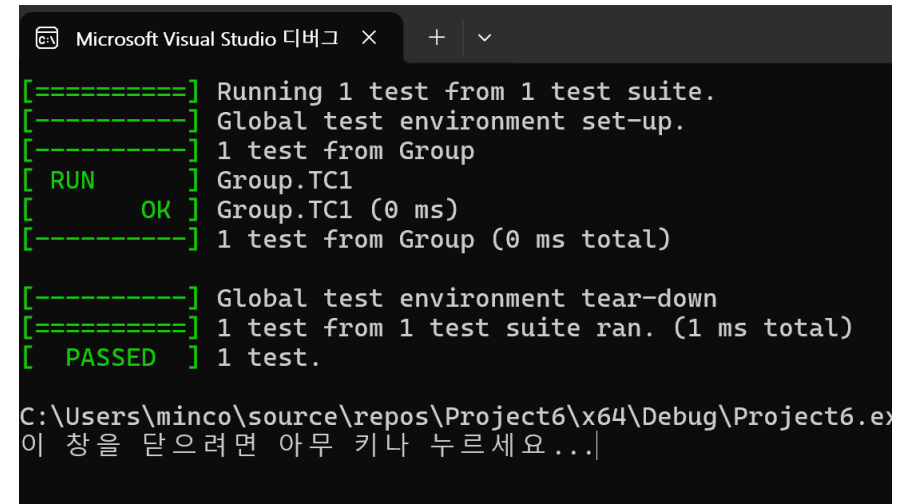
Test Case 추가 및 중간 점검

Test Case 하나를 추가하고 동작이 잘 되는지 중간 점검을 한다.

```
#include "gmock/gmock.h"

TEST(Group, TC1) {
    ...
    EXPECT_EQ(1, 1);
}
```

cal_test.cpp 파일에
기본 코드작성



```
Microsoft Visual Studio 디버그
[=====] Running 1 test from 1 test suite.
[-----] Global test environment set-up.
[-----] 1 test from Group
[ RUN     ] Group.TC1
[      OK ] Group.TC1 (0 ms)
[-----] 1 test from Group (0 ms total)

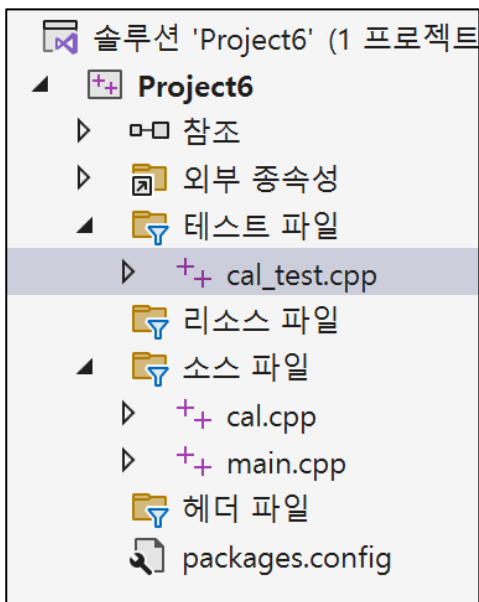
[-----] Global test environment tear-down
[=====] 1 test from 1 test suite ran. (1 ms total)
[ PASSED ] 1 test.

C:\Users\minco\source\repos\Project6\x64\Debug\Project6.exe
이 창을 닫으려면 아무 키나 누르세요...
```

Google Test 설치 방법 - 7

Cal 클래스 테스트하기

Unit Test 코드를 완성하여 최종 테스트를 해본다.



```
#include "gmock/gmock.h"
#include "cal.cpp"

TEST(Group, TC1) {
    Cal cal;

    int ret = cal.getSum(10, 20);

    EXPECT_EQ(30, ret);
}
```

cal_test.cpp 파일
코드 수정

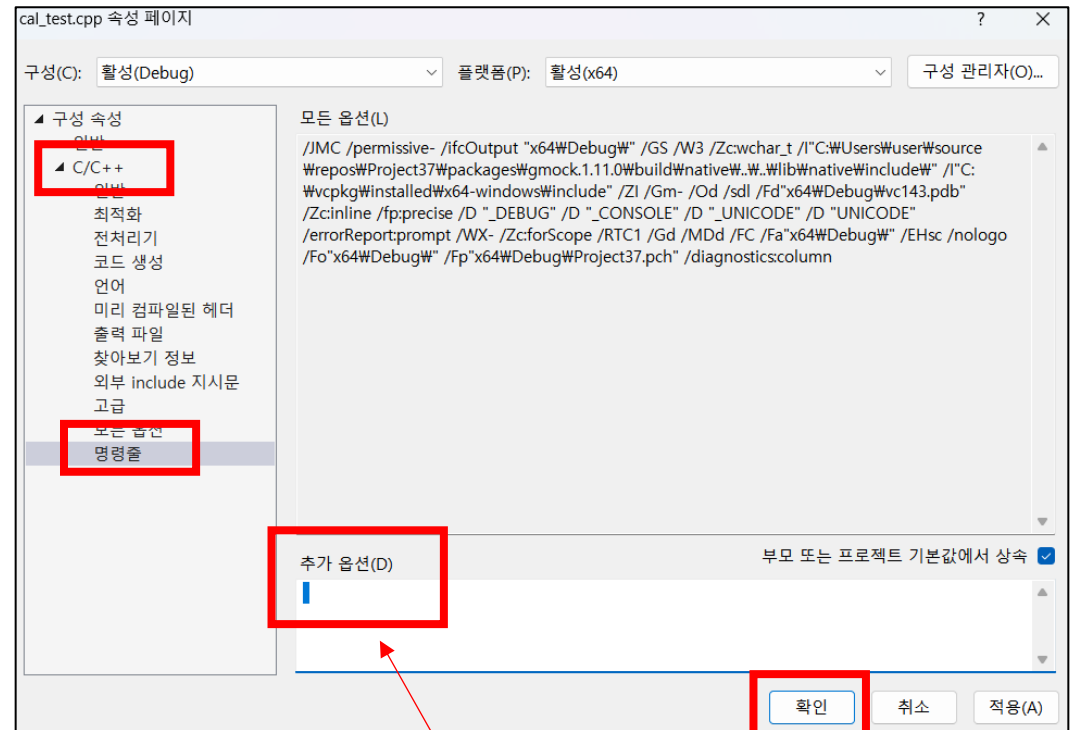
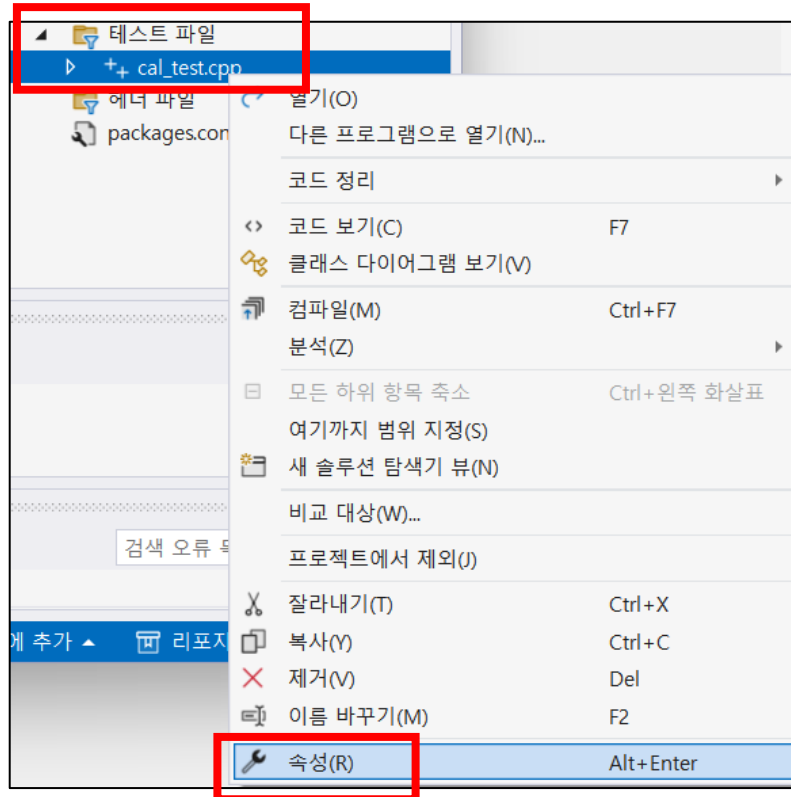
```
[=====] Running 1 test from 1 test suite.
[-----] Global test environment set-up.
[-----] 1 test from Group
[ RUN      ] Group.TC1
[ OK       ] Group.TC1 (0 ms)
[-----] 1 test from Group (0 ms total)

[-----] Global test environment tear-down
[=====] 1 test from 1 test suite ran. (2 ms total)
[ PASSED  ] 1 test.
```

[필수] Visual Studio에서 cpp 파일 include 이슈 해결방법

헤더파일이 아닌, cpp 파일을 include 하는 경우에는, 아래와 같이 세팅을 해야한다.

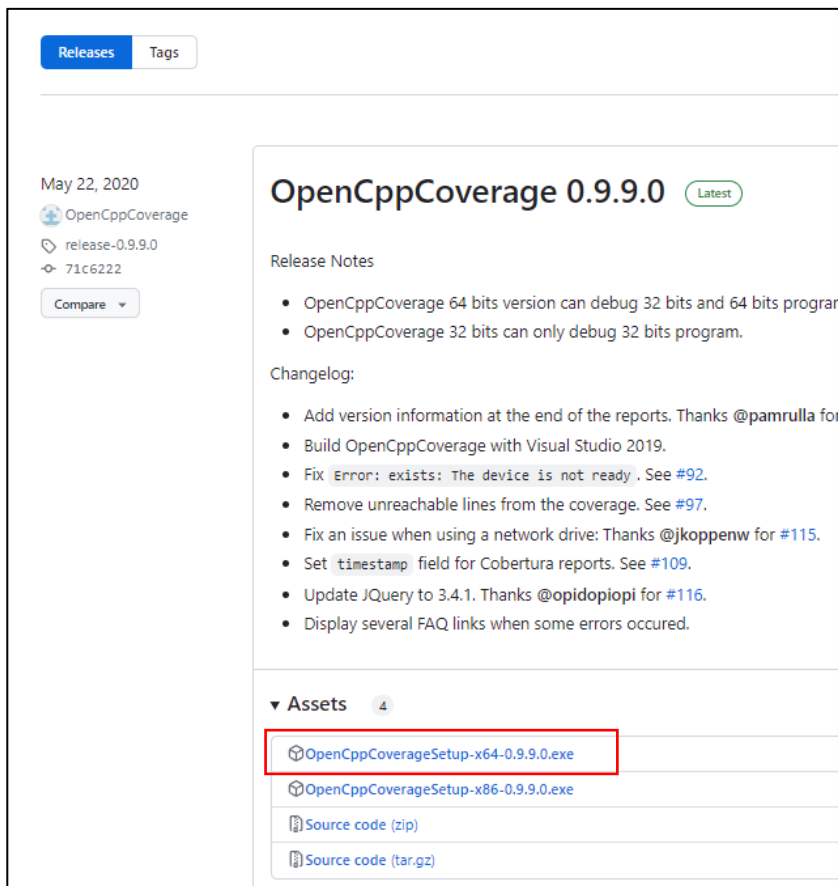
- 아래 세팅을 하지 않으면, UnitTest가 최신 컴파일된 소스코드를 인식하지 못하고, 과거의 UnitTest결과만 출력된다.
- 테스트 파일 (test.cpp) 파일 속성 > 명령줄 > 추가옵션에 아무 Text나 넣어준다.
 - 추가 옵션에 무언가 적혀있으면, 빌드 시마다 _test.cpp 파일 컴파일 재수행



띄어쓰기 한칸 입력후 "확인"

코드 커버리지 측정도구 설치하기

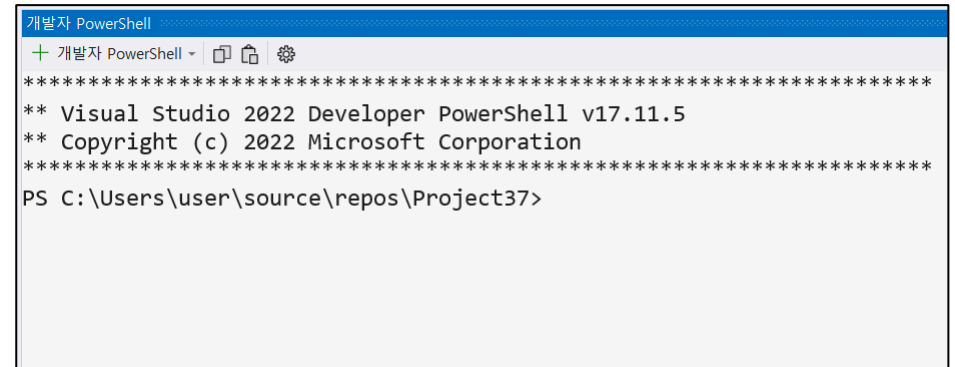
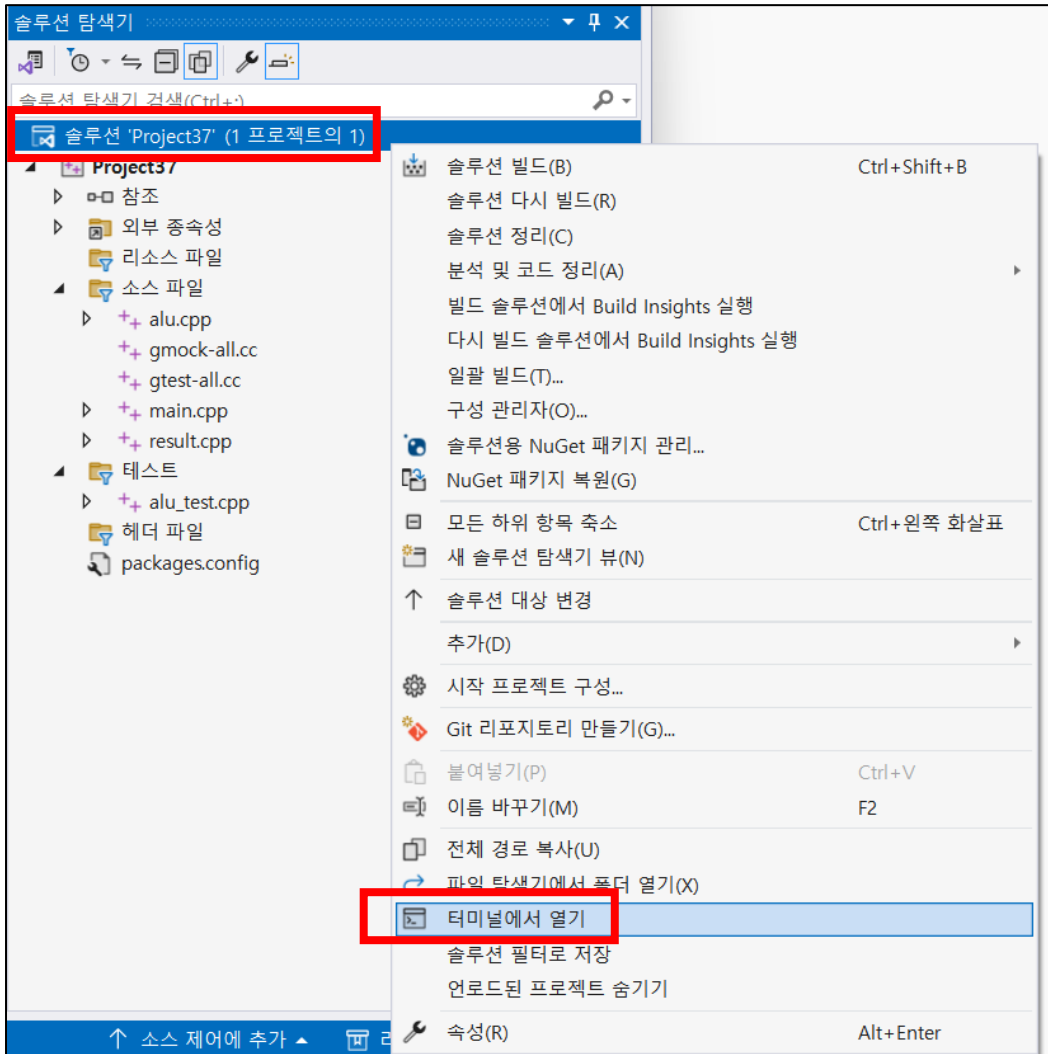
<https://github.com/OpenCppCoverage/OpenCppCoverage/releases>



64bit exe 파일로 다운로드를 받는다.
설치 후, **Visual Studio**를 꺾다가 다시 켜다.

코드 커버리지 실행 준비 – 터미널 열기

프로젝트가 아닌 “솔루션” 에서 터미널 열기



코드 커버리지 측정 명령어 입력

`OpenCppCoverage.exe --sources C:*.cpp --export_type=html:coverage -- .\x64\Debug\실행파일이름.exe`

복사 붙여넣기를 위한 명령어 링크

<https://gist.github.com/mincoding1/3d187a62a3ea90c5c21aa68016c408b6>

```
개발자 PowerShell
+ 개발자 PowerShell | |
*****
** Visual Studio 2022 Developer PowerShell v17.11.5
** Copyright (c) 2022 Microsoft Corporation
*****
PS C:\Users\user\source\repos\Project37> OpenCppCoverage.exe --sources C:*.cpp --export_type=html:coverage -- .\x64\Debug\
```

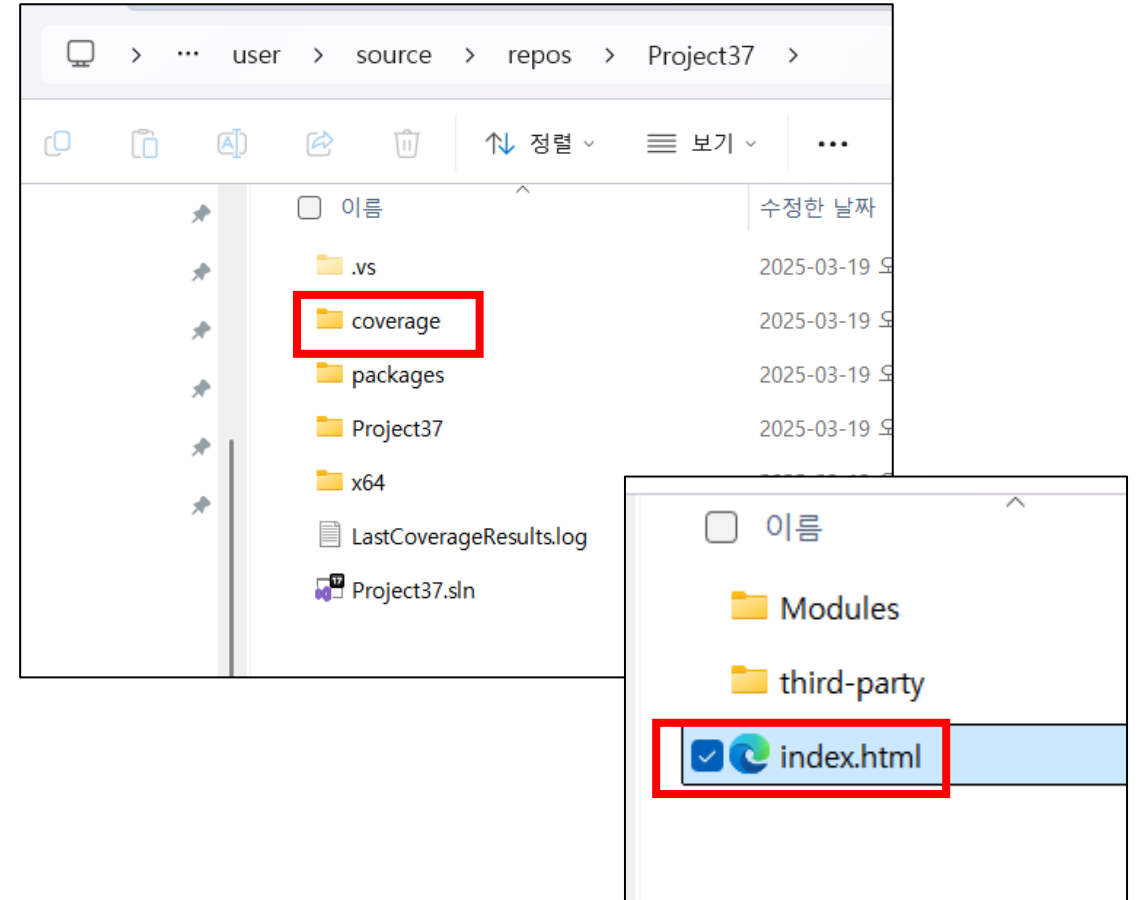
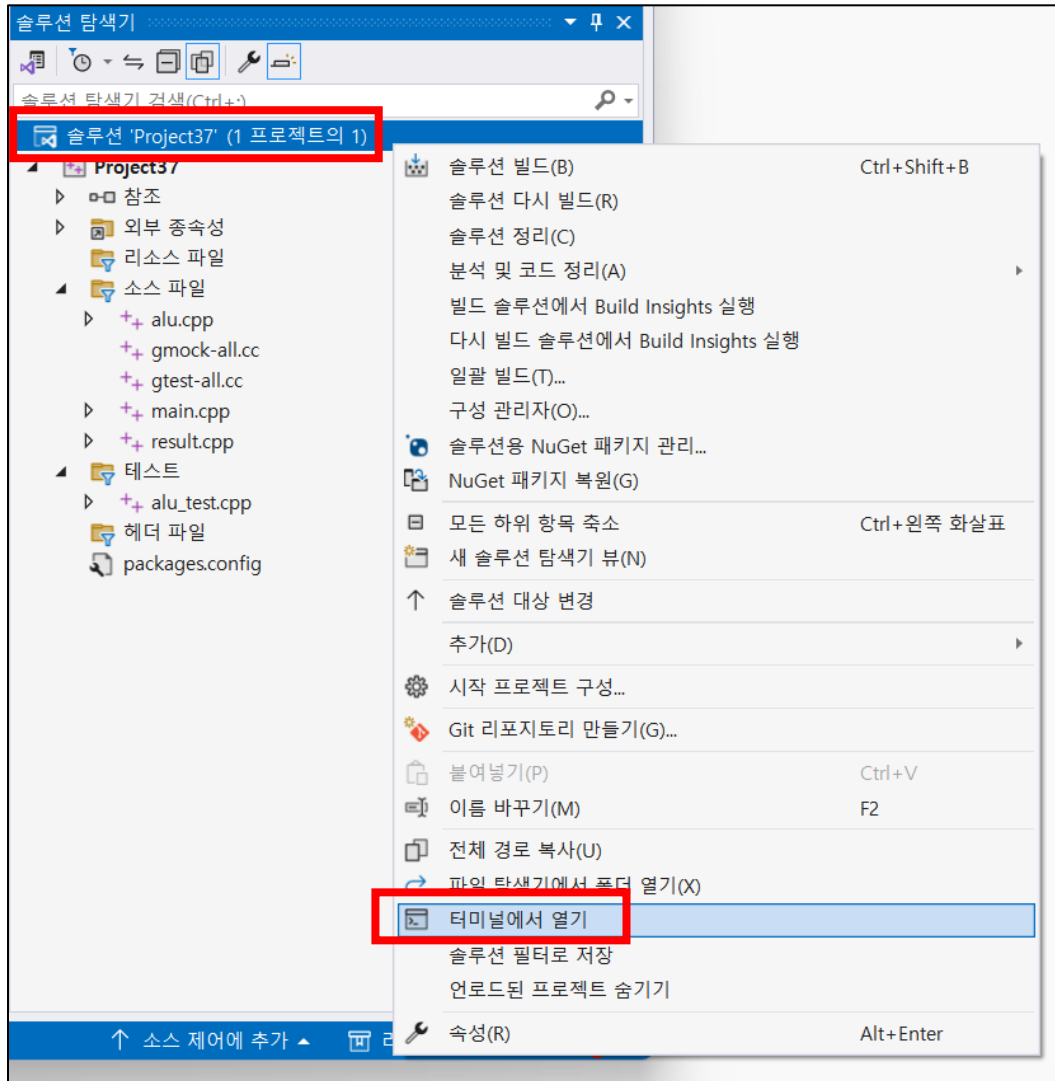


탭키 여러번 눌러
.exe파일이 나오도록하기

```
개발자 PowerShell
+ 개발자 PowerShell | |
*****
** Visual Studio 2022 Developer PowerShell v17.11.5
** Copyright (c) 2022 Microsoft Corporation
*****
PS C:\Users\user\source\repos\Project37> OpenCppCoverage.exe --sources C:*.cpp --export_type=html:coverage -- .\x64\Debug\Project37.exe
```

코드 커버리지 확인

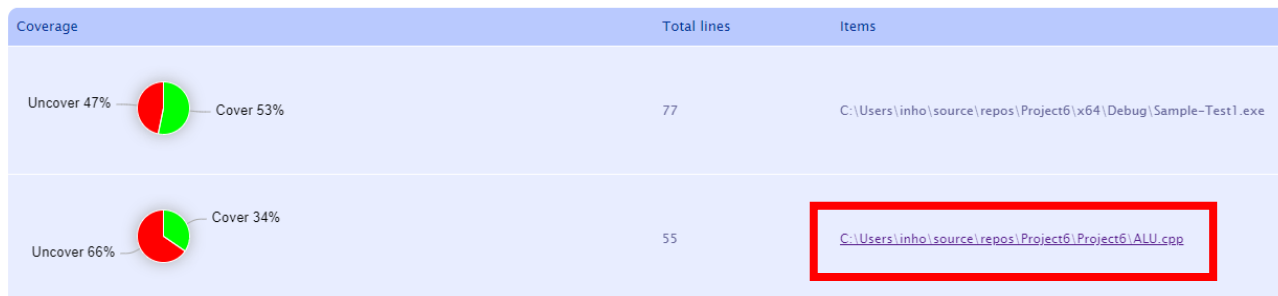
커버리지 결과 확인



Code Coverage 결과

- 초록색 : 테스트가 닿은 곳
- 빨간색 : 테스트가 닿지 못한 곳

Sample-Test1.exe



alu.cpp 파일을 찾아 클릭한다.

```

1. #pragma once
2.
3. #include <string>
4. #include "Result.cpp"
5.
6. class ALU
7. {
8.     int operand1 = -1;
9.     int operand2 = -1;
10.    std::string OPCODE = "";
11.
12. public:
13.    void setOperand1(int operand1) {
14.        this->operand1 = operand1;
15.    }
16.
17.    void setOperand2(int operand2) {
18.        this->operand2 = operand2;
19.    }
20.
21.    void setOPCODE(std::string OPCODE) {
22.        this->OPCODE = OPCODE;
23.    }
24.
25.    void enableSignal(Result *r) {
26.        if (OPCODE == "ADD" && OPCODE != "MUL" && OPCODE != "SUB") {
27.            if (operand1 != -1 && operand2 != -1) {
28.                int result = operand1 + operand2;
29.                r->setResult(result);
30.                r->setStatus(0);
31.            }
32.            else if (operand1 == -1) {
33.                r->setResult(65535);
34.                r->setStatus(1);
35.            }
36.            else if (operand2 == -1) {
37.                r->setResult(65535);
38.                r->setStatus(2);
39.            }
40.        }
41.        else if (OPCODE != "ADD" && OPCODE == "MUL" && OPCODE != "SUB") {
42.            if (operand1 != -1 && operand2 != -1) {
43.                int result = operand1 * operand2;
44.                r->setResult(result);
45.                r->setStatus(0);
46.            }
47.            else if (operand1 == -1) {
48.                r->setResult(65535);
49.                r->setStatus(1);
50.            }
51.            else if (operand2 == -1) {

```


감사합니다.