



# 알고리즘의 설계와 분석 (Design & Analysis of Algorithms)

Visual Studio 사용법 소개

Sogang University



# Visual Studio 설치 및 C 프로그래밍

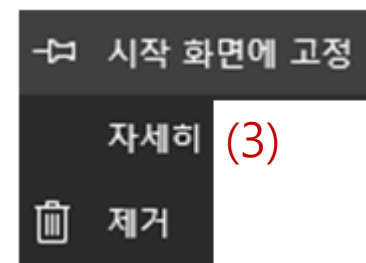
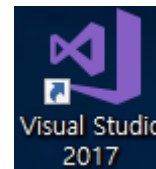
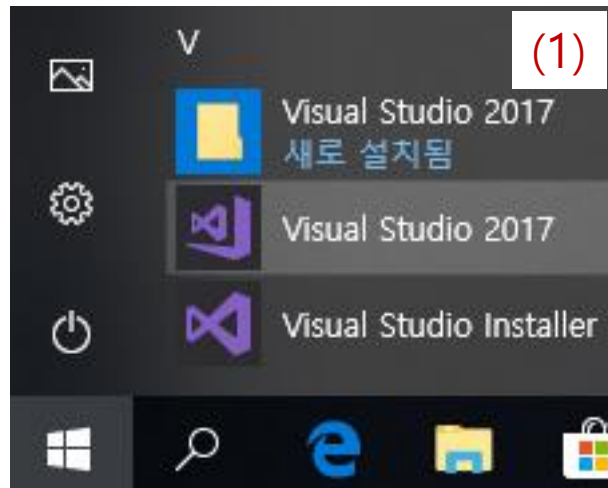
## ◆ 개요

- ◆ Visual Studio 2017(**VS2017**) Community Edition 설치를 먼저 확인한다.
- ◆ VS2017 Community Edition이 아닐 수도 있지만 큰 영향은 없다.

# Windows 10 바탕 화면 설정

## ◆ Visual Studio Icon 추가

- ◆ 바탕 화면이나 작업 표시줄에 Visual Studio icon을 추가하면 편리하다.
- ◆ 바탕 화면: 시작 메뉴 → V 항목에서 Visual Studio 2017을 찾아<sup>(1)</sup> → 좌 클릭한 채로 마우스 커서를 바탕화면으로 이동 → 클릭 해제 → 바탕화면에 단축 icon이 생긴다<sup>(2)</sup> → 단축 icon을 더블 클릭하여 VS2017 실행.
- ◆ 작업 표시줄: 위와 같은 방법으로 클릭하여 작업 표시줄로 커서 이동 → 클릭 해제(작업 표시줄 icon은 한번만 클릭하면 VS2017이 실행된다).
- ◆ 시작 화면: Visual Studio 2017을 찾아<sup>(1)</sup> → 우 클릭 → 클릭 시작 화면에 고정<sup>(3)</sup> → 시작 화면에 icon이 등록된다.

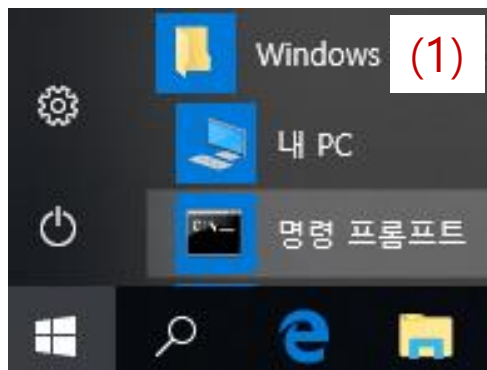


## ◆ 명령 프롬프트

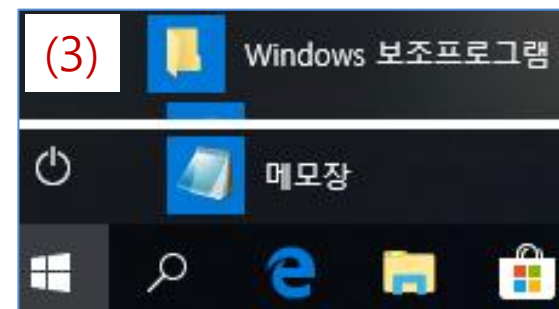
- ◆ 본 강의에서 작성한 프로그램은 **명령 프롬프트**에서 실행하여야 한다.
- ◆ 따라서, 이 도구도 바탕화면이나 작업 표시줄에 단축 icon을 만들어 주면 편리하다.
- ◆ **바탕 화면**: 시작 메뉴 → W 항목에서 **Windows** 시스템내 **명령프롬프트**를 찾아<sup>(1)</sup> → 좌 클릭한 채로 마우스 커서를 **바탕화면으로 이동** → 클릭 해제 → 바탕화면에 단축 icon이 생긴다<sup>(2)</sup> → 단축 icon을 더블 클릭.
- ◆ **작업 표시줄**: 위와 동일한 방법으로 icon을 **작업 표시줄로 이동**.
- ◆ **시작 화면**: **Visual Studio 2017** 등록 방법과 동일.

## ◆ 메모장

- ◆ 프로그램 작성에 메모장도 가끔 사용한다(**Windows** 보조프로그램<sup>(3)</sup>).
- ◆ 따라서, 이 도구 역시 바탕 화면(작업 표시줄)에 icon을 만들면 편리하다.

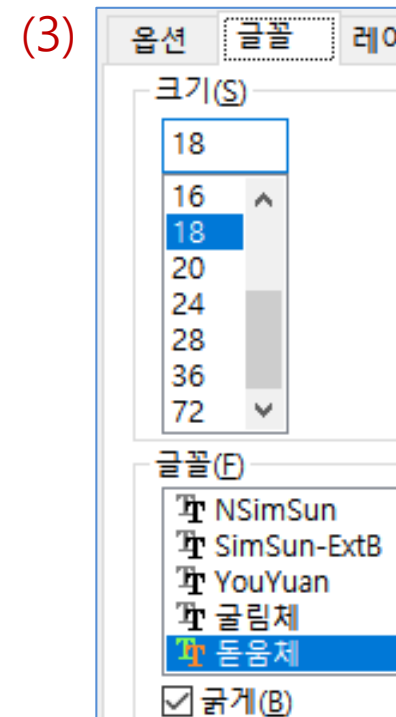
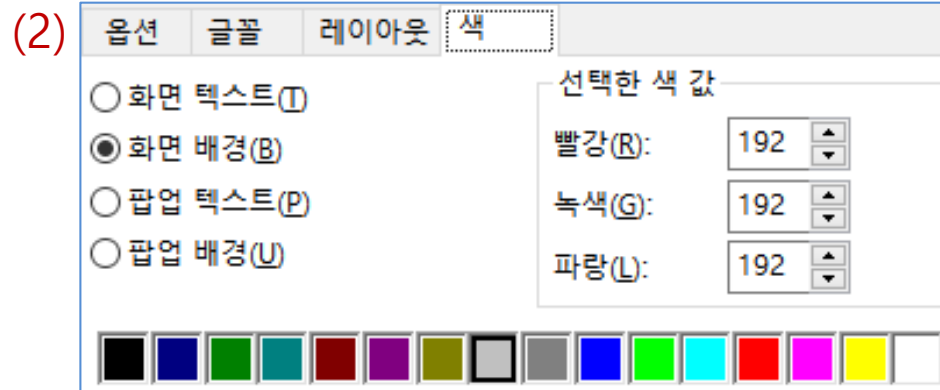
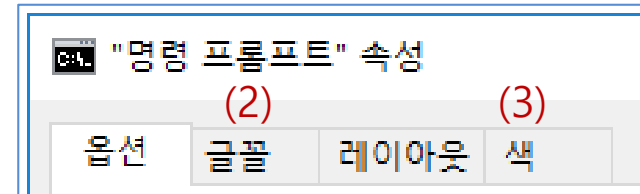
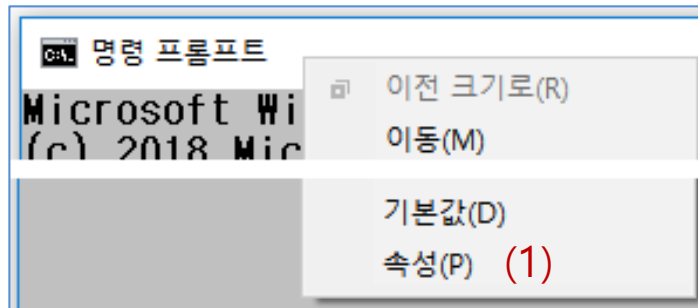


(2)



## ◆ 명령 프롬프트 설정 바꾸기

- ◆ 명령 프롬프트를 실행하면 바탕 색이 검정색인 창이 생기는데, 이 역시 글자 모양과 함께 변경할 수 있다.
- ◆ 명령 프롬프트 실행 → 마우스 우 클릭 → 속성(1) → 속성 창에서 글꼴(2) 또는 색(3)을 선택하여 글자 모양/크기, 색 등을 조정할 수 있다.





# C 프로그래밍을 위한 준비

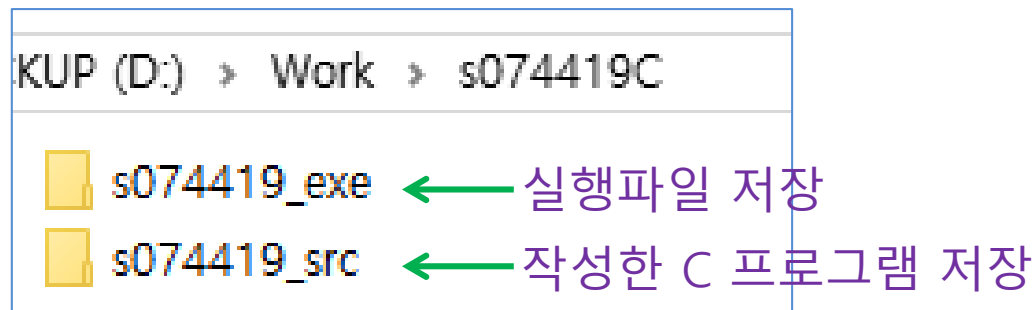
## ◆ 파일 관리

- ◆ 본 강의에서는 실습을 많이 수행하고, 실습을 통하여 프로그래밍을 익힐 것이다.
- ◆ 실습 시 작성한 프로그램은 지우지 말고 보관하여 두면 차후 다른 프로그램 작성에 도움이 될 수 있다.
- ◆ 컴퓨터는 고장이 생길 가능성이 늘 존재한다. 특히, 디스크가 고장이면 자신이 작성한 자료를 모두 잃게 된다.
- ◆ 특히, 학교의 실습용 컴퓨터는 효과적인 관리를 위하여, 학생들이 작성한 자료를 임의로 삭제할 수 있다.
- ◆ 따라서, 학생들은 자신이 작성한 모든 자료를 보관하여야 한다(back up).
- ◆ 이를 위하여, 수업을 마친 후 자신이 작성한 파일 또는 자료를 **usb 디스크**, **cloud**, 또는 **자신의 email** 등을 통하여 **항상 보관**하도록 한다.



## ◆ 폴더 생성 및 관리

- ◆ 본 강의를 위한 폴더를 다음과 같이 생성하여 관리하도록 한다<sup>(1)</sup>.
- ◆ 적당한 드라이브/폴더를 선택하여 **snnnnnnC**라는 폴더를 만든다(여기서, **nnnnnn**은 자신의 학번 뒤 6자리 숫자로 한다).
- ◆ 그리고, **snnnnnnC** 폴더에 **snnnnnn\_src**라는 폴더와 **snnnnnn\_exe**라는 폴더를 만든다(아래 예 참조).



- ◆ **snnnnnn\_src** : 이 폴더에 **작성한 C 프로그램을 저장**한다. 폴더 snnnnnnC 전체를 백업해도 되지만, 최소 이 폴더만 백업하면 충분하다.
- ◆ **snnnnnn\_exe** : 완성한 프로그램의 **실행 파일을 이 폴더에 저장**한다. 프로그래밍을 할 때 명령 프롬프트를 이 폴더에서 사용토록 하면 편리하다.
- ◆ 위 폴더는 자신의 **usb 메모리에 만들어도 무방하다**(단, **백업은 철저히**).
- ◆ 추가로 **snnnnnn\_prj** 라는 폴더를 만들텐데, 이는 다음에 설명한다.

(1) 물론 개인적으로 다르게 설정하여도 되지만, 여기서 안내하는 방법이 가장 효율적일 것이다.



# Visual Studio 프로젝트 만들기

## ◆ 프로젝트와 솔루션(1)

### ◆ 프로젝트

- ◆ Visual Studio에서 소스 코드, 비트맵, 아이콘, 구성 요소 등, 프로그램 (앱)을 빌드하는 데 필요한 항목을 저장한 논리적 컨테이너이다.
- ◆ 프로젝트 하나 당 하나의 실행 파일을 생성한다.
- ◆ Visual Studio에서 프로젝트 파일의 확장자는 **.vcxproj**이다.

### ◆ 솔루션

- ◆ 하나 이상의 프로젝트의 모임을 의미한다.
- ◆ 즉, 어떤 목적에 **K** 개의 프로그램이 필요하다면, 이를 위하여 **K** 개의 프로젝트를 생성하고 이들이 모여 하나의 솔루션을 구성하게 된다.
- ◆ Visual Studio에서 솔루션 파일의 확장자는 **.sln**이다.
- ◆ Visual Studio에서는 새 프로젝트를 만들면 이를 포함한 솔루션을 만드는데, 필요한 경우 다른 프로젝트를 추가로 솔루션에 포함 시킬 수 있다.
- ◆ 그러나, 본 강의에서는 하나의 프로젝트만을 생성하여 실습을 수행할 것이다.

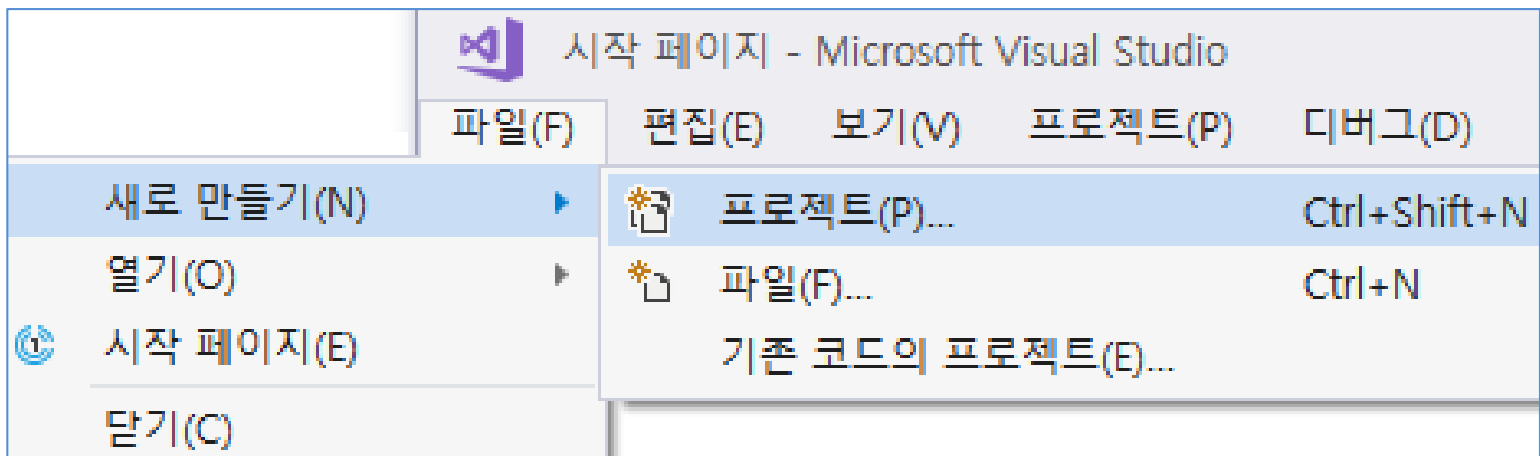
(1) <https://docs.microsoft.com/ko-kr/visualstudio/ide/creating-solutions-and-projects?view=vs-2017>





## ◆ 프로젝트 만들기

- ◆ 앞에 기술한 것처럼, 적절한 드라이브에 Work 폴더, 그리고 snnnnnnnC라는 폴더를 만들었다고 가정한다<sup>(1)</sup>.
- ◆ 앞으로 본 강의에서는 항상 이 폴더에서만 실습을 수행한다.
- ◆ 만일 자신이 사용하는 실습실 컴퓨터에 이 폴더가 삭제되어 있다면, 백업 받은 자료로 다시 복원하도록 하자<sup>(2)</sup>.
- ◆ 이제 VS2017을 실행한다.
- ◆ 열린 창에서 메뉴 바 → 파일 → 새로 만들기 → 프로젝트를 클릭한다.

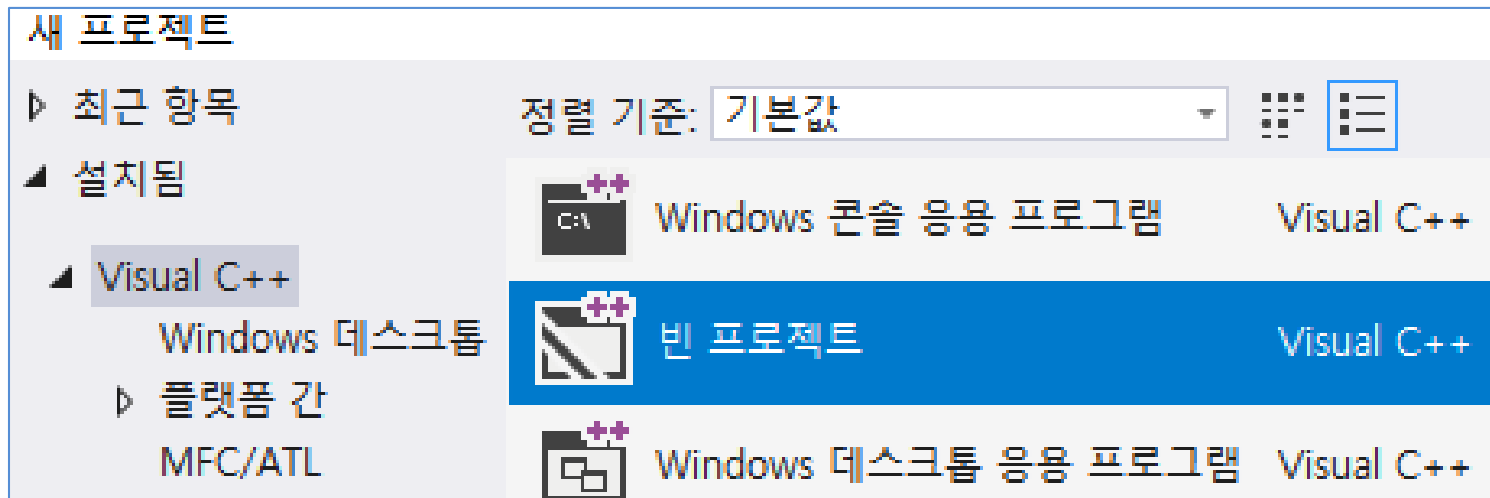


(1) nnnnnnn은 자신의 학번 뒤 6자리.

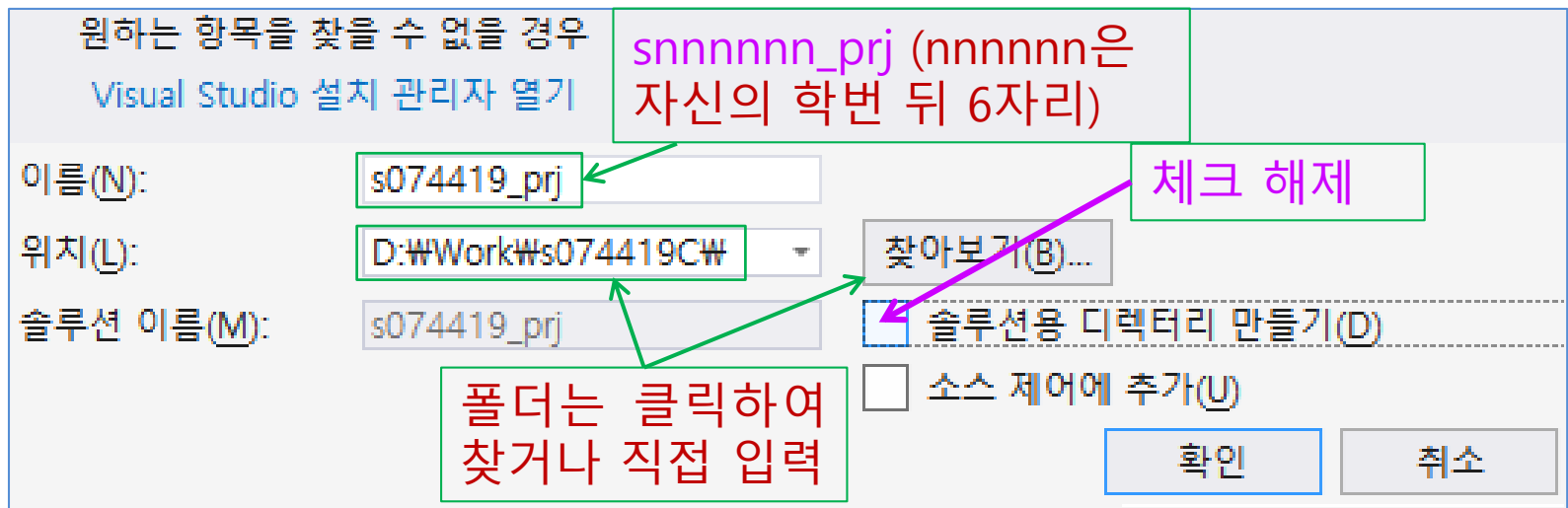
(2) 폴더가 지워질 가능성이 있기 때문에 매 수업을 마친 후 반드시 이 폴더를 자신의 usb, cloud, 또는 email에 백업 하도록 하자.



✦ Visial C++ → 빈 프로젝트 클릭



◆ 위와 같은 창의 아래 부분에 다음과 같이 입력한다<sup>(1)</sup>. 입력 후 클릭 **확인**.



(1) 실제로 이 두 창을 동일한 창이다. 글자를 크게 보이기 위하여 둘로 나눈 것이다.

- ◆ 확인: 아래와 같이 세팅하였는지 다시 한번 확인하자.

새 프로젝트

정렬 기준: 기본값

최근 항목

설치됨

Visual C++

Windows 데스크톱

플랫폼 간

MFC/ATL

테스트

Windows 콘솔 응용 프로그램 Visual C++

빈 프로젝트 Visual C++

Windows 데스크톱 응용 프로그램 Visual C++

원하는 항목을 찾을 수 없을 경우  
Visual Studio 설치 관리자 열기

이름(N): s074419\_prj

위치(L): D:\Work\s074419C\

찾아보기(B)...

솔루션 이름(M): s074419\_prj

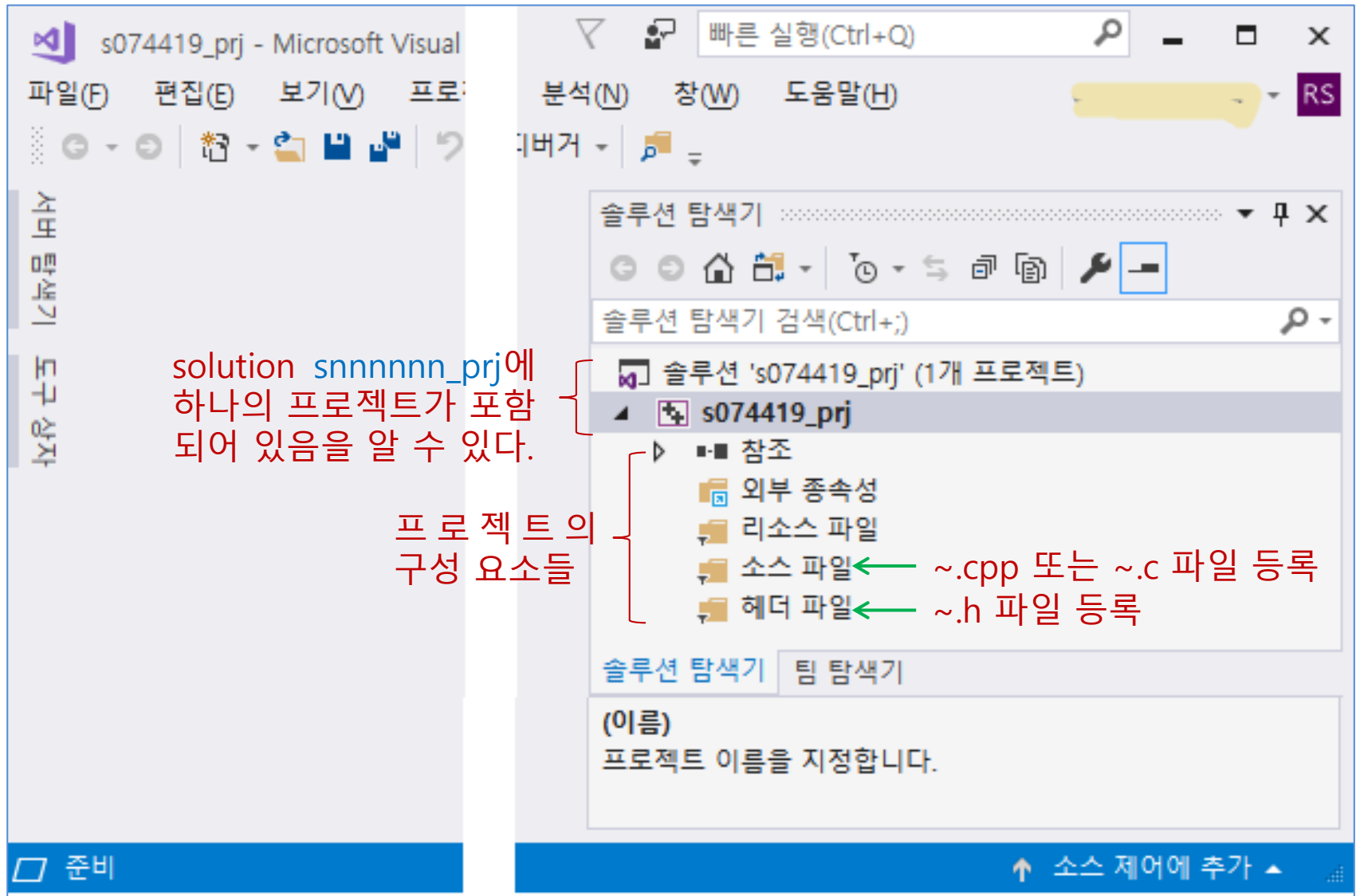
☒ 솔루션용 디렉터리 만들기(D)

☐ 소스 제어에 추가(U)

확인후 클릭 → 확인 취소

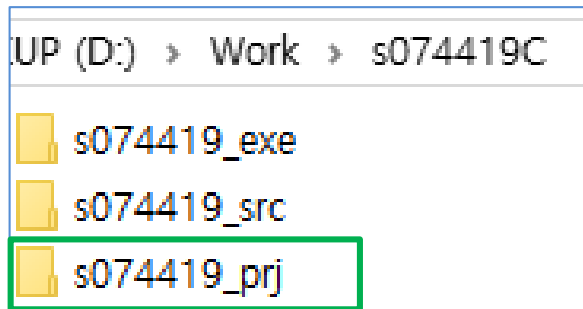
우리는 단 하나의 프로젝트만을 사용할 것이므로 이를 체크 하는 것이 의미가 없다(오히려 불편하다)

◆ 다음과 같은 창이 생긴다.

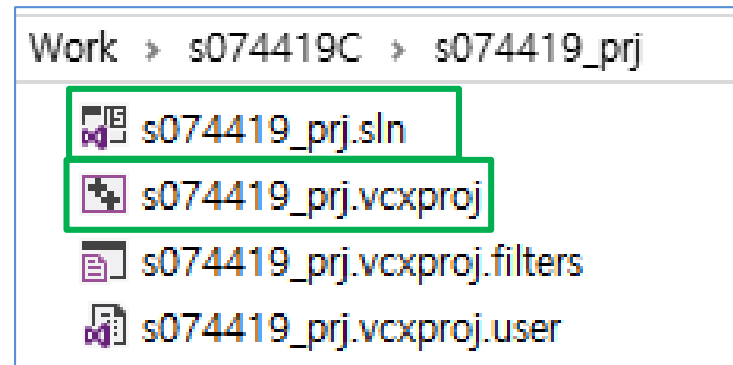


## ◆ 프로젝트 폴더 둘러보기(1)

- ◆ 폴더 `snnnnnnnC`에서 생성한 프로젝트 폴더 `snnnnnnn_prj`를 볼 수 있다(A).
- ◆ 폴더 `snnnnnnn_prj`에는 프로젝트에 관련된 파일이 생성되어 있다(B).



(A) 폴더 `snnnnnnnC`의 내용



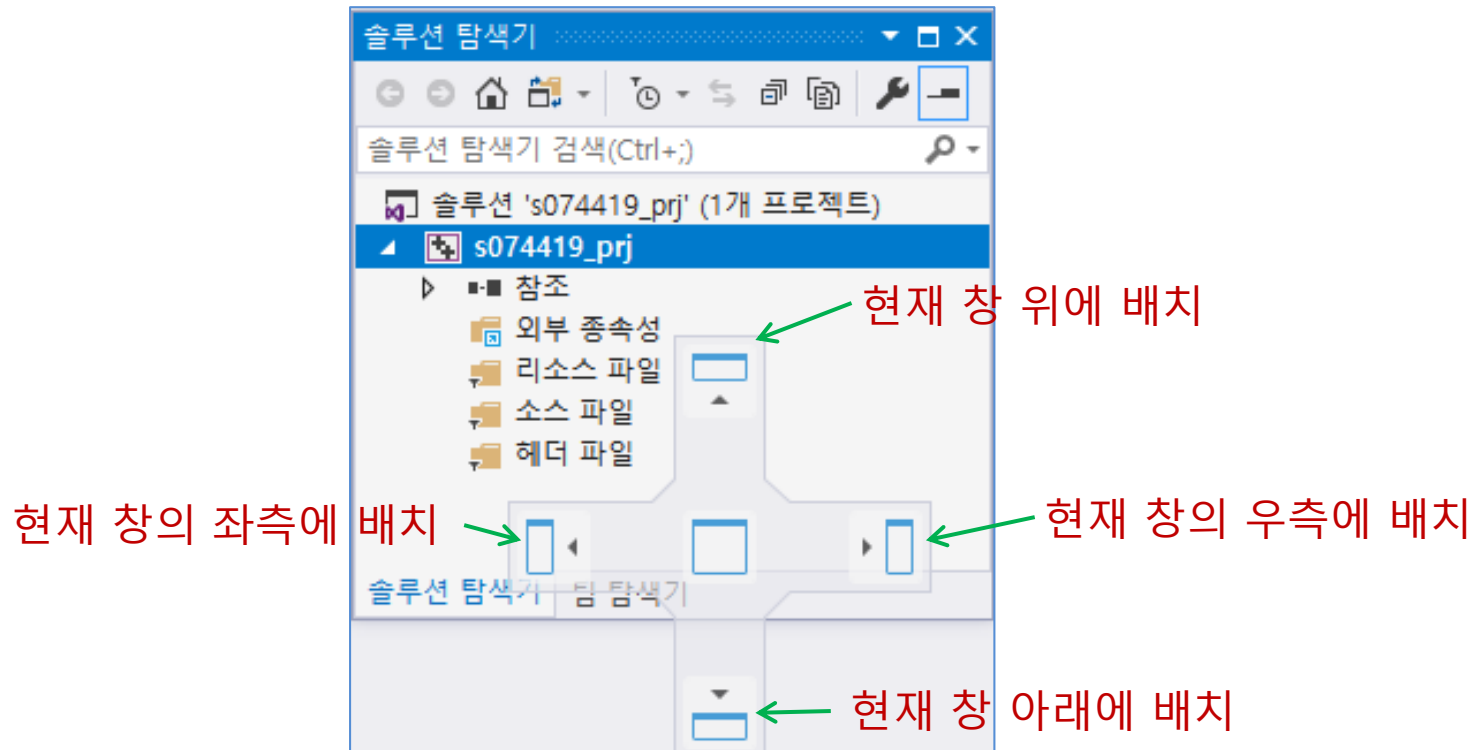
(B) 폴더 `snnnnnnn_prj`의 내용

- ◆ 폴더 `snnnnnnn_prj`에서 파일 `snnnnnnn_prj.vcxproj` 파일은 프로젝트 정보를 포함하고 있고, `snnnnnnn_prj.sln`은 솔루션 정보를 가지고 있다.
- ◆ 차후, `snnnnnnn_prj.sln`를 더블 클릭하여 Visual Studio를 열어 작업을 계속 수행할 수 있다.
- ◆ 해당 프로젝트에서 작업을 진행하면 추가로 파일이 생성될 수 있다. 특히, 빌드를 실행하면 **Debug**라는 폴더가 새로 생성되는데, 이 폴더에 실행 파일 `snnnnnnn_prj.exe`가 생성된다.

(1) nnnnnnn은 자신의 학번 뒤 6자리.

## ◆ Visual Studio 창 레이아웃 조정(1)

- ◆ Visual Studio에는 다수의 창들이 존재하는데, 필요한 창들을 적절히 배치하는 것이 효율적인 프로그래밍에 도움이 된다.
- ◆ 아래 그림은 솔루션 탐색기를 새 위치로 옮기는 과정을 보인 것이다.

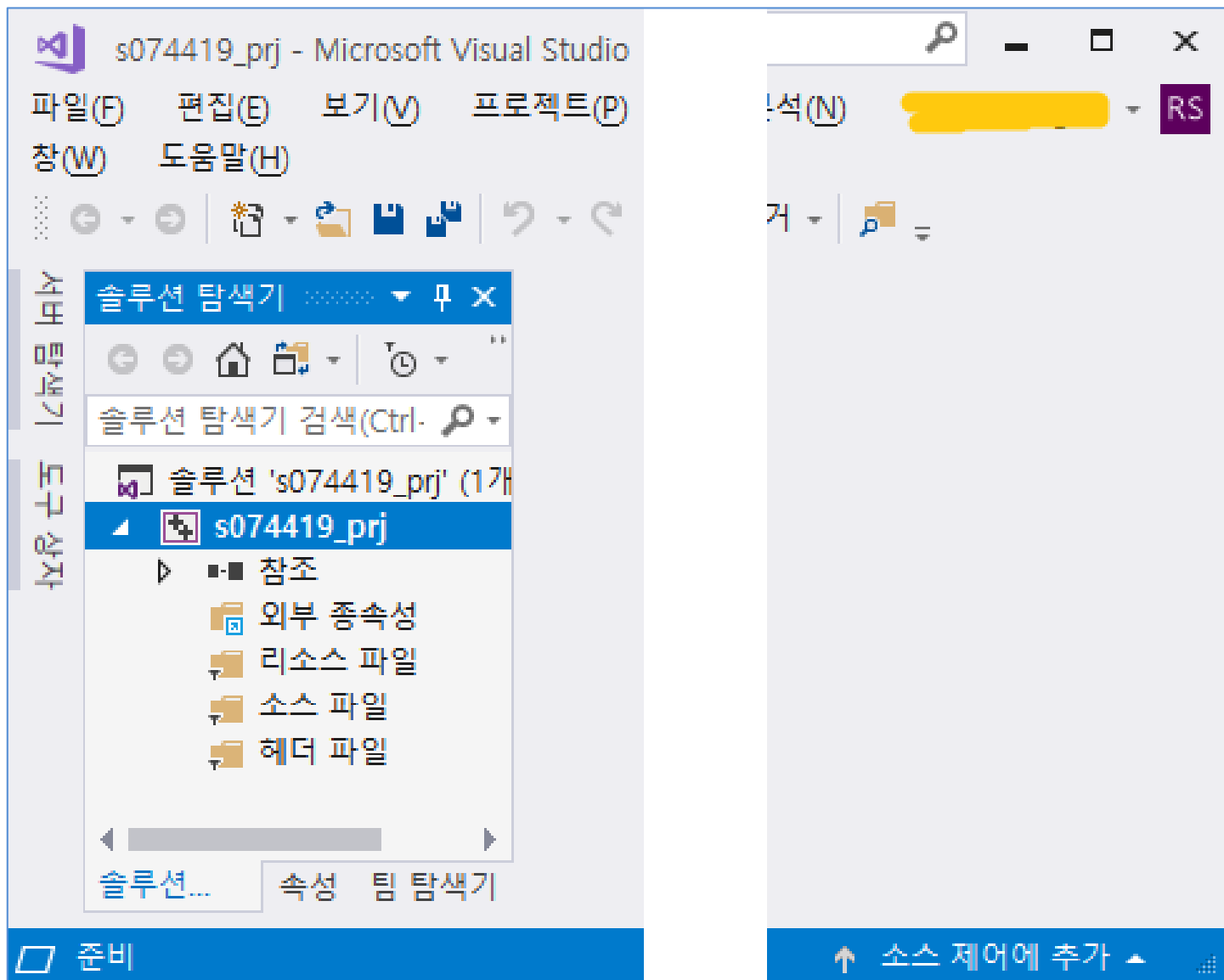


- ◆ 이 조정은 위 그림만으로는 이해하기 어려울 수 있는데, 실습 시간에 설명을 듣고 조정 방법을 익히도록한다.

(1) <https://docs.microsoft.com/ko-kr/visualstudio/ide/customizing-window-layouts-in-visual-studio?view=vs-2017>



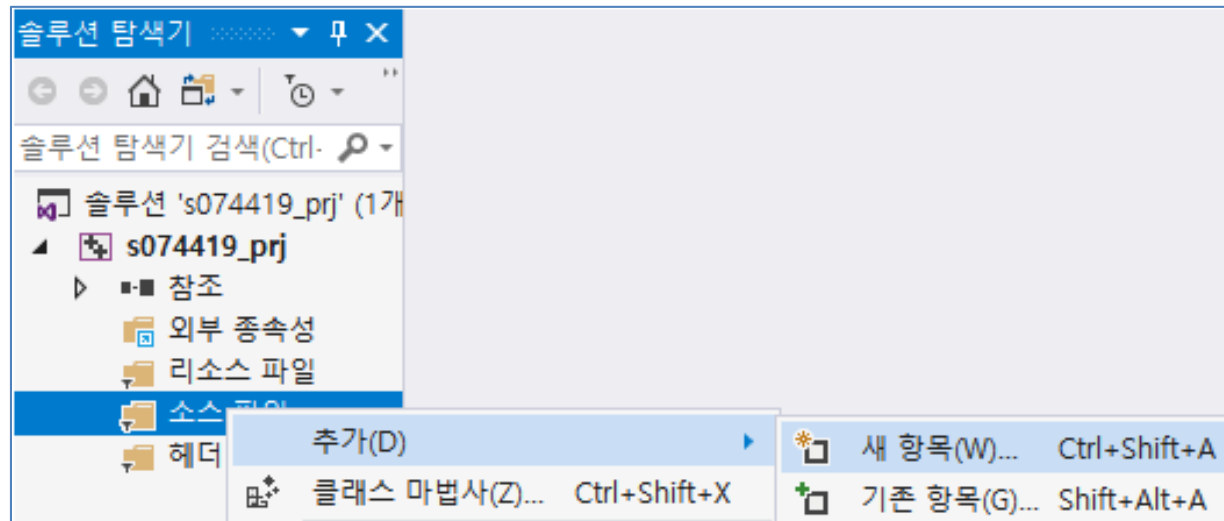
- ◆ 조정하는 방법을 익힌 후 솔루션 탐색기 등의 창들을 아래 그림과 같이 좌측에 배치하여 보자.



# 파일 생성 및 실행

## ◆ 파일 생성

- ◆ Visual Studio에서 프로그램 작성, 빌드, 실행시키는 방법을 기술한다.
- ◆ C 프로그램 파일의 확장자는 본래 **.c**이지만, 본 강의에서는 C++ 프로그램 파일의 확장자인 **.cpp**를 사용한다(1,2).
- ◆ 프로그램을 처음 만들려며 아래 그림처럼 **솔루션 탐색기** → **소스파일** → **추가** → **새 항목**을 클릭한다.



- (1) C++ 언어는 C 언어 구문을 거의 대부분 포함하기 때문에 확장자로 **.cpp**를 사용하여도 C 언어를 익히는데 문제가 없다.
- (2) 다만, ANSI C, C99, C11 등의 표준과 약간 다를 수 있어, 다른 컴파일러에서는 일부 오류가 발생할 가능성은 있다(이 경우 코드를 표준에 맞게 약간 수정해야 함).



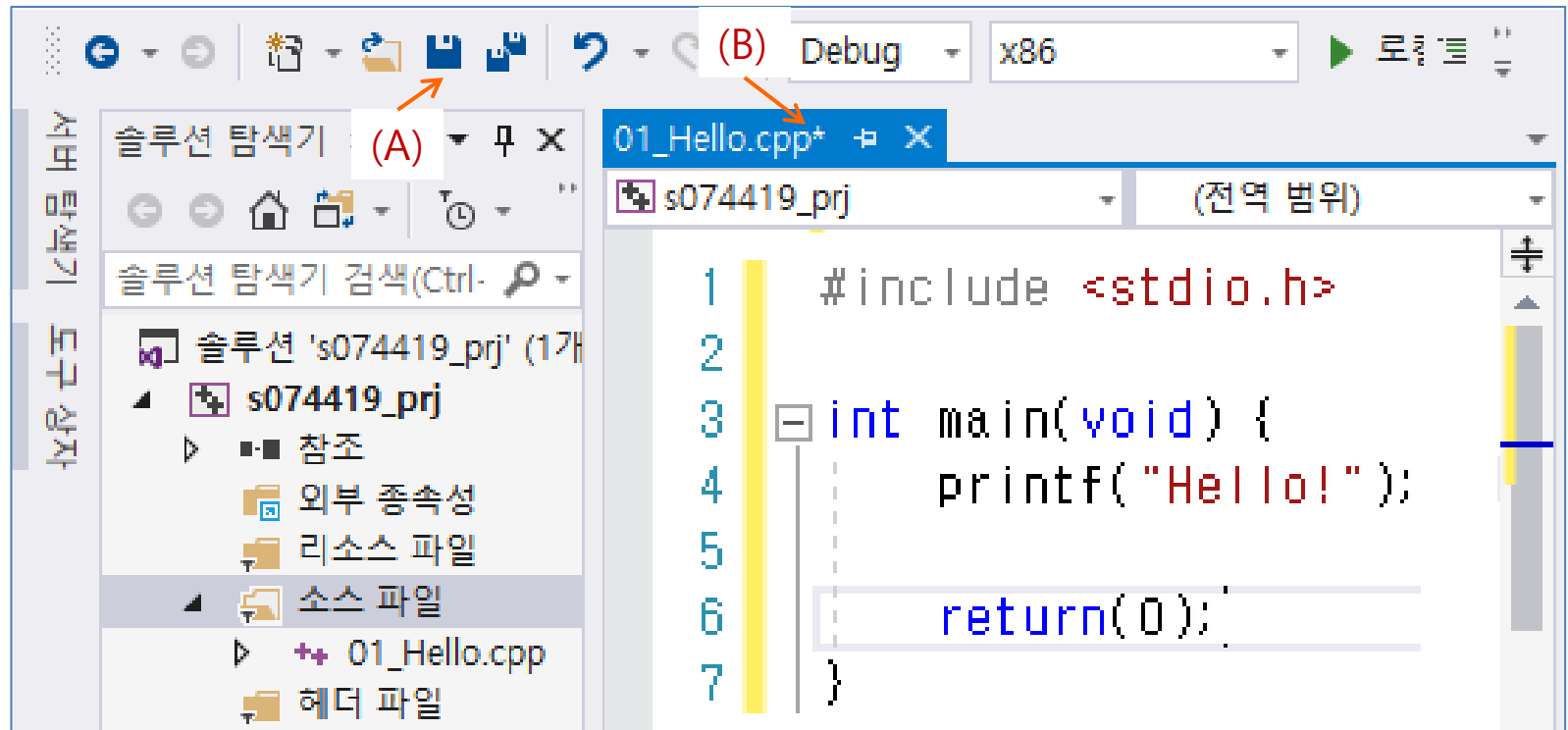


- 

- CSE3081 알고리즘의 설계와 분석

## ◆ 프로그램 입력/편집

- ◆ 파일 01\_Hello.cpp의 편집 창이 열리는데, 아래 그림의 내용을 입력하자.
- ◆ 파일 입력을 마치면 디스크 모양의 툴팁(A)을 클릭하여 저장한다.
  - ◆ 만일, 편집창이 없을 경우, 좌측 솔루션 탐색기에서 해당 파일을 더블 클릭하면 편집창이 열린다.
  - ◆ 파일 이름에 \*가 있으면 파일 내용이 바뀐 것을 의미한다(B).
  - ◆ 편집 기능은 매우 강력하며, 단축키를 익히면 더욱 편리하게 작성할 수 있다(1).



(1) <http://luyin.tistory.com/294>



## ◆ 소스 코드에 오류가 있는 경우

- ◆ 편집 창에 물결 모양의 빨강 밑줄이 생기며, 커서를 그곳으로 이동하면 오류에 대한 설명이 보인다.
- ◆ 따라서, 프로그램 작성 중에 문법적으로 잘못된 부분을 즉시 수정할 수 있어 대단히 편리하다.

### ◆ Examples

```
3  int main(void) {  
4      print ("Hello!");  
5  
6      retur  
7  }
```

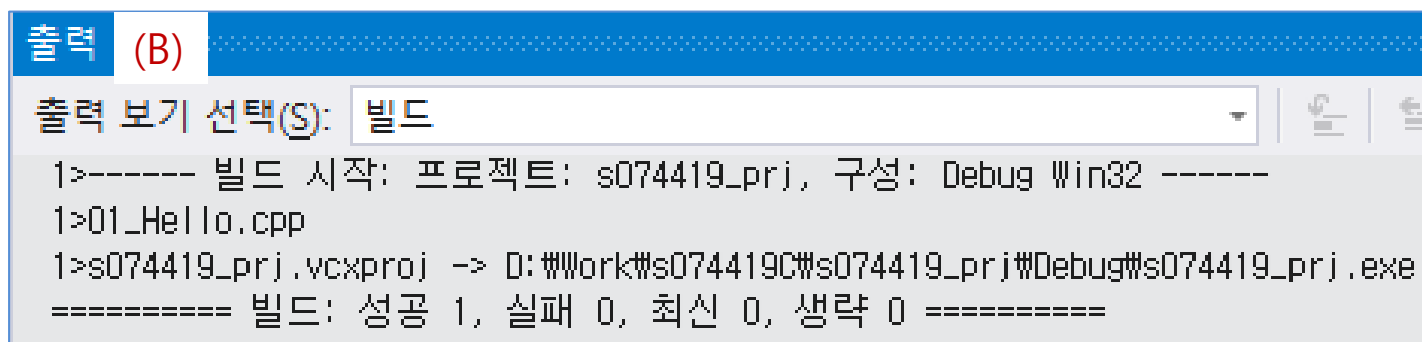
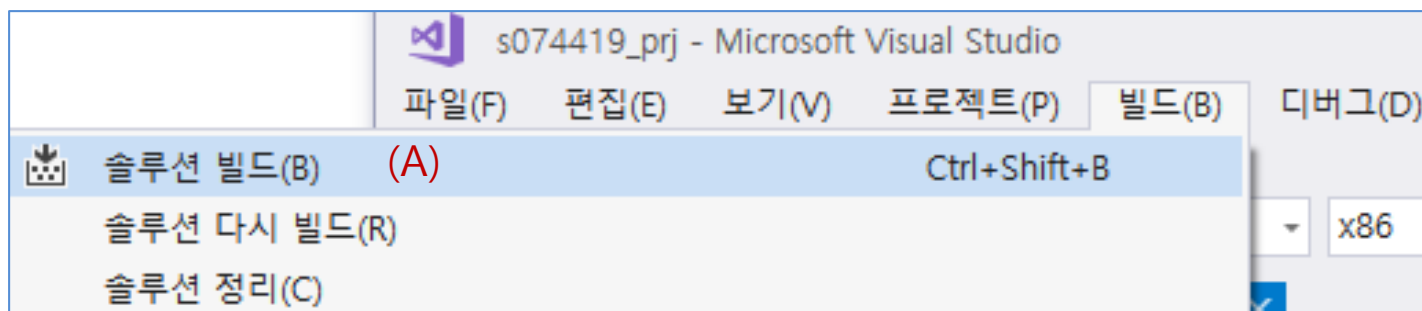
(const char [7])"Hello!"  
식별자 "print"이(가) 정의되어 있지 않습니다.

```
3  int main(void) {  
4      printf("Hello!")  
5  
6      return(0);  
7  }
```

';가 필요합니다.

## ◆ 프로그램 빌드

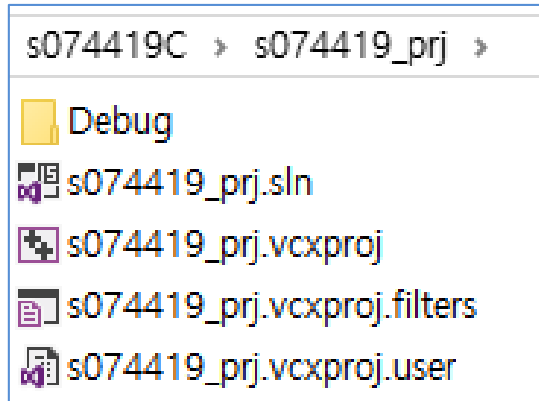
- ◆ Visual Studio에서 빌드는 컴파일, 링크 기능을 수행하는 기능이다.
- ◆ 메뉴 바 → 빌드 → 솔루션 빌드 클릭(단축키는 Ctrl+Shift+B)<sup>(A)</sup>
- ◆ 출력 창에 빌드 진행 상황이 보여진다<sup>(B, 1)</sup>. 오류가 있을 경우 어떤 오류인지 출력될 것이다.



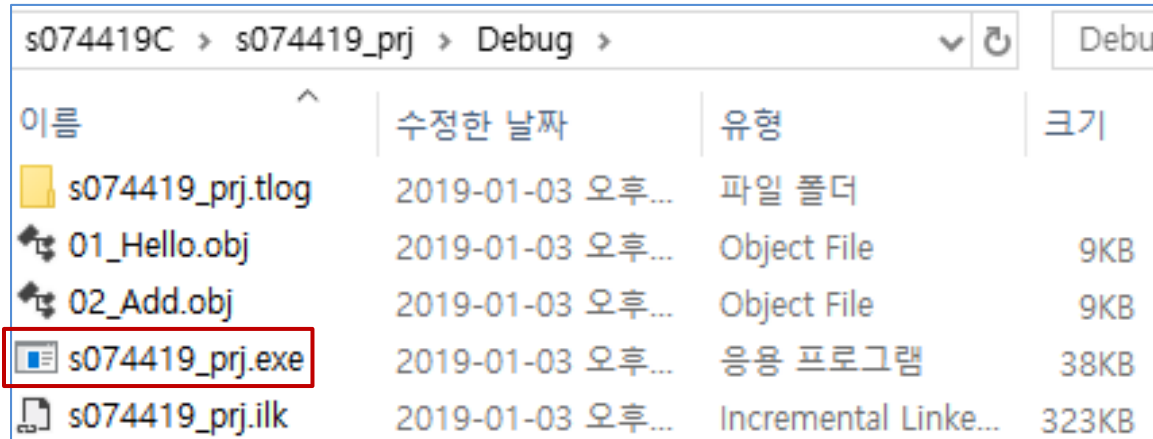
(1) 출력창의 위치는 가능한 소스 코드 편집창 밑에 위치하도록 한다.

## ◆ 빌드 후 프로젝트 폴더

- ◆ 빌드를 수행하면 프로젝트 폴더 내에 Debug라는 폴더가 생긴다(A).
- ◆ Debug 폴더에는 빌드 중 생성된 파일들과 최종적으로 만들어진 실행 파일 **snnnnnnn\_prj.exe**가 생긴다(B).

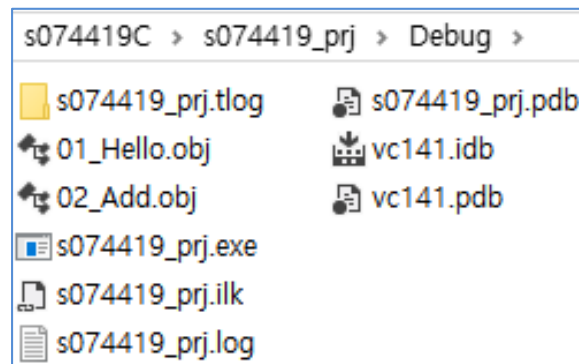


(A) 새로 생긴 Debug 폴더



(B) Debug 폴더의 내용

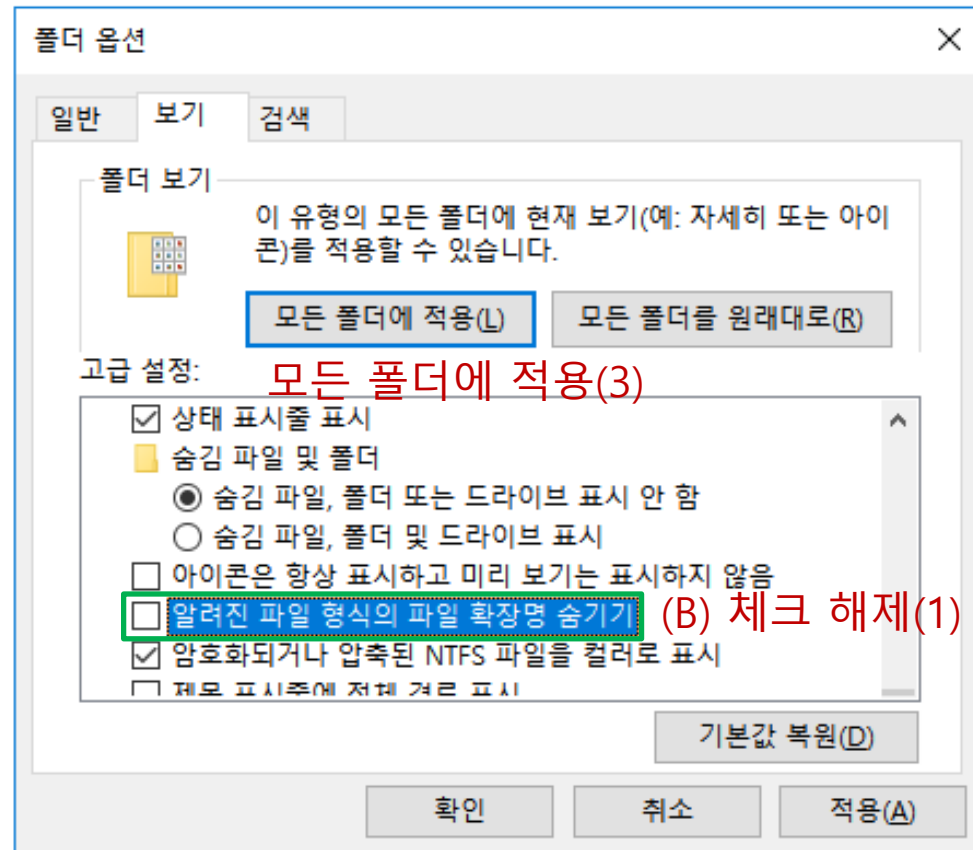
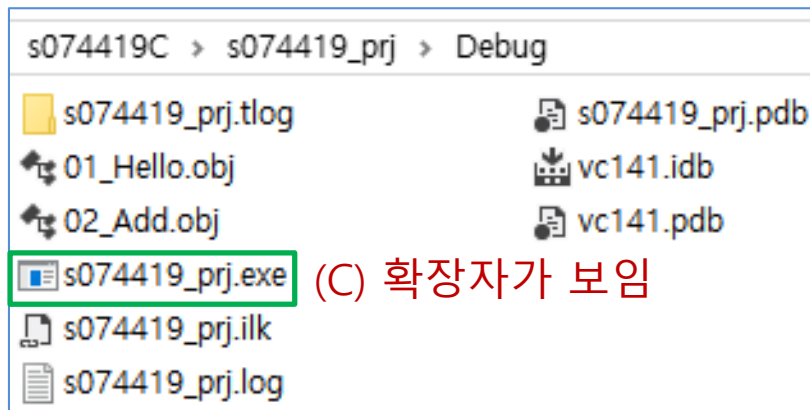
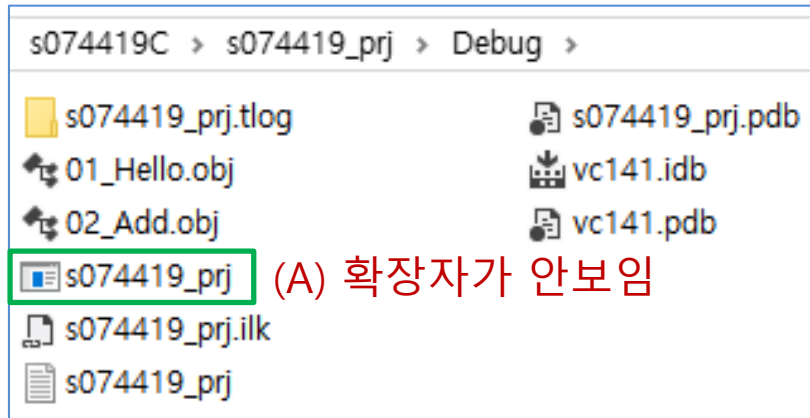
- ◆ 위 Debug 폴더처럼 파일들이 자세히 모드로 보이면, 폴더 창의 보기(1) → 목록을 클릭하여 아래 그림처럼 파일들만 보이게 하면 보기가 좋다.



(1) Win 7인 경우 폴더 창에서  
마우스 우 클릭 후 보기

## ◆ 파일의 확장자 보이게 하기

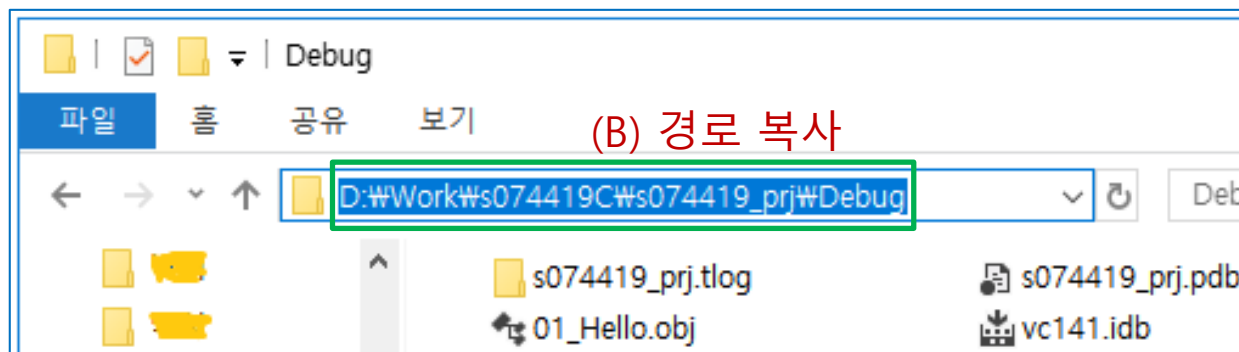
- ◆ 폴더의 파일을 볼 때, 폴더의 확장자가 보이지 않아 불편할 경우가 있다(A).
- ◆ 이 경우, **폴더의 보기** → **옵션(1)** → "**알려진 ... 숨기기**" 체크 해제(B) → **적용** → **모든 폴더에 적용** → **확인**을 클릭하면, 폴더의 파일에 **확장자가 표시된다(C)**.



(1) Win 7에서는 (도구가 안보일 경우: Alt) 도구 → 폴더 옵션

## ◆ 프로그램 실행

- ◆ 명령 프롬프트를 실행한다.
- ◆ 프로젝트 폴더가 C 드라이브가 아닌 D 드라이브에 있다면, D: 입력(A).
- ◆ Debug 폴더에서 폴더의 경로를 마우스로 클릭하고 Ctrl\_C(B).
- ◆ 명령 프롬프트에서 cd Ctrl\_V(경로 복사, 폴더를 바꾼다) (1, C).
- ◆ 실행 파일 이름 s074419\_prj 입력하면(D) 프로그램이 실행된다(E).



(1) Win 7에서는 명령 프롬프트 창에서 마우스 우 클릭 → 붙여넣기

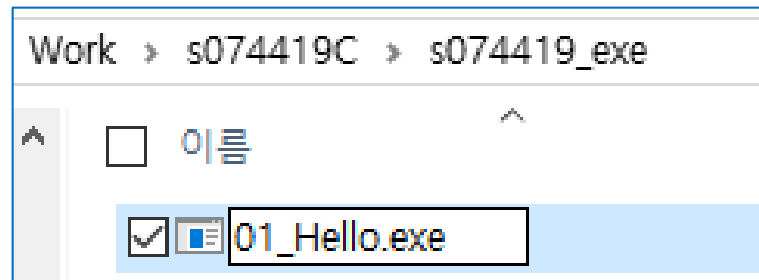
명령 프롬프트에서 폴더를 바꾸는 과정

```
CA. 명령 프롬프트
Microsoft Windows [Version 10.0.17134.472]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\csrim>D: (A) 드라이브 선택
D:\>cd D:\Work\074419C\074419_prj\Debug <Enter> (C) 경로 복사
D:\Work\074419C\074419_prj\Debug>s074419_prj <Enter> (D) 실행
Hello!
D:\Work\074419C\074419_prj\Debug>
```

(E) 실행 결과

## ◆ 실행 프로그램 보관

- ◆ snnnnnn\_prj에서 빌드한 실행 파일의 이름은 항상 snnnnnn\_prj.exe이다.
- ◆ 이 실행 파일을 snnnnnn\_exe에 복사해두자.
- ◆ 즉, Debug 폴더에 있는 snnnnnn\_prj.exe를 snnnnnn\_exe로 복사한다.
- ◆ 복사한 파일 snnnnnn\_prj.exe를 아래와 같이 01\_Hello.exe로 이름을 바꾼다(또는, 02\_01\_Hello.exe).
- ◆ 01\_Hello.exe 파일 역시 명령 프롬프트에서 실행할 수 있다.
- ◆ 이를 위해서, 명령 프롬프트에서 "cd 폴더 경로" 를 입력하여 대상 폴더를 01\_Hello.exe가 있는 폴더로 바꾸고 실행하여야 한다<sup>(1,2)</sup>.



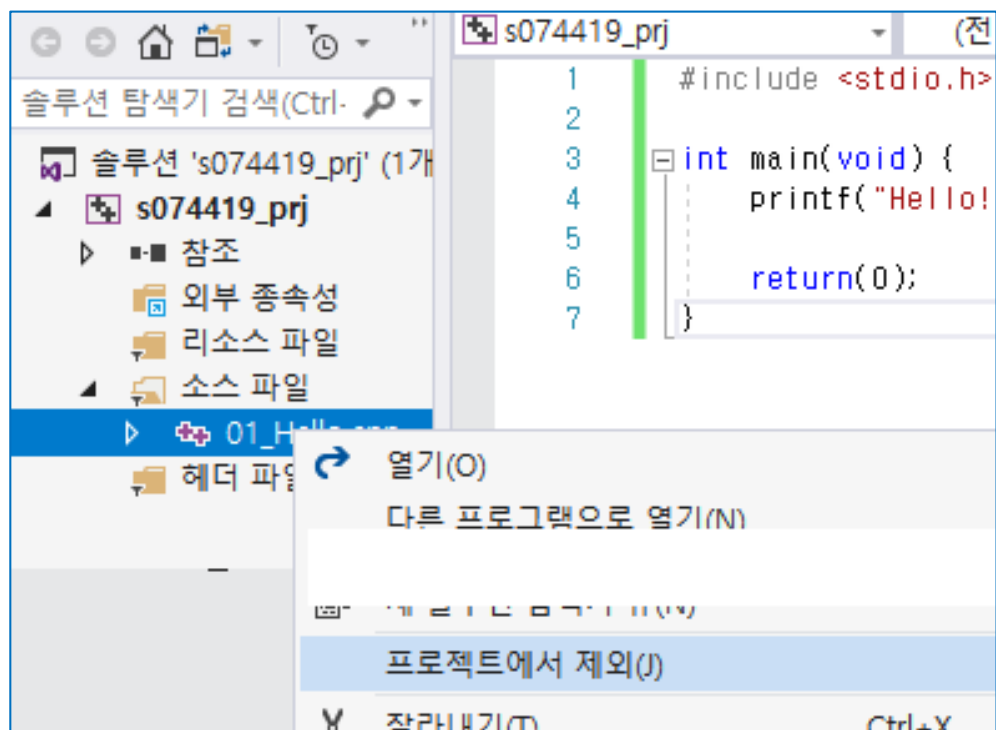
- (1) 주의. 여기서 빌드한 실행 파일은 이를 더블 클릭해서는 실행되는 것을 볼 수 없다 (우리가 보통 사용하는 프로그램 처럼 만들기 위해서는 초기에 프로젝트를 생성할 때, 빈 프로젝트가 아닌 다른 것을 선택해야 한다(10쪽 참조))
- (2) 우리가 작성한 프로그램은 콘솔 응용 프로그램으로 명령 프롬프트에서만 실행된다 (이에 반하여 데스크톱 응용 프로그램 등을 선택해야 더블 클릭으로 프로그램을 실행할 수 있는데, 이는 VS에 대해 상당한 지식이 필요하다).



# 새 프로그램 작성 및 실행

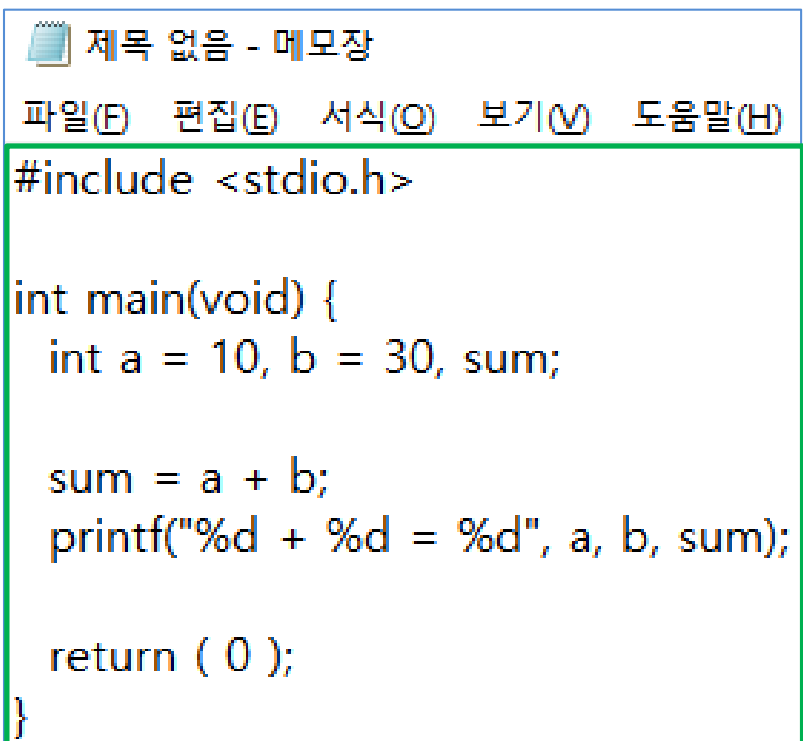
## ◆ 기존 소스 프로그램 제거

- ◆ 프로젝트 s074419\_prj에서 또 다른 프로그램을 작성하고 빌드할 수 있다.
- ◆ 이를 위해서는 이전에 작업한 소스 프로그램 01\_Hello.cpp를 먼저 프로젝트에서 제거하여야 한다.
- ◆ Visual Studio 솔루션 탐색기 → 소스 파일 → 01\_Hello.cpp 우 클릭 → 클릭 프로젝트에서 제외 → 파일이 프로젝트에서 제거된다.



## ◆ 새 프로그램 작성

- ◆ 01\_Hello.cpp 파일을 작성하는 방법과 동일하게 새로운 프로그램을 생성, 작성할 수 있다.
- ◆ 그러나, 이번에는 메모장으로 프로그램을 작성한 후 이를 Visual Studio에 등록하여 보자.
- ◆ 메모장을 열어 먼저 다음과 같이 입력한다<sup>(1)</sup>.



```
#include <stdio.h>

int main(void) {
    int a = 10, b = 30, sum;

    sum = a + b;
    printf("%d + %d = %d", a, b, sum);

    return ( 0 );
}
```

```
#include <stdio.h>

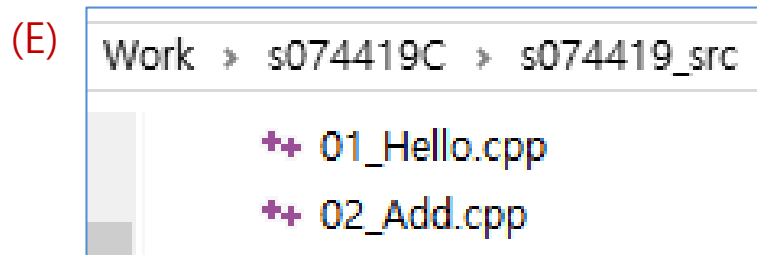
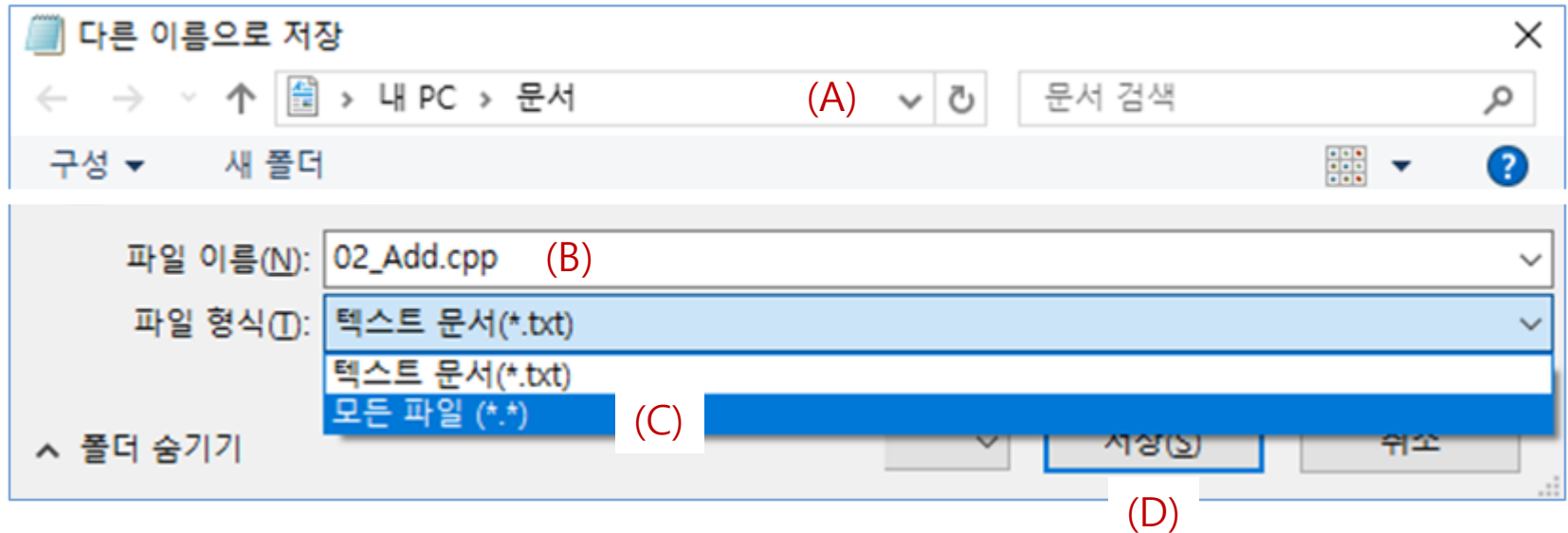
int main (void) {
    int a = 10, b = 30, sum;

    sum = a + b;
    printf("%d + %d = %d", a, b, sum);

    return(0);
}
```

(1) 메모장과 같은 형식의 파일을 만드는 편집기는 뭐든 사용할 수 있다. 그러나, 한글, Word 등과 같은 문서 편집기는 사용할 수 없다.

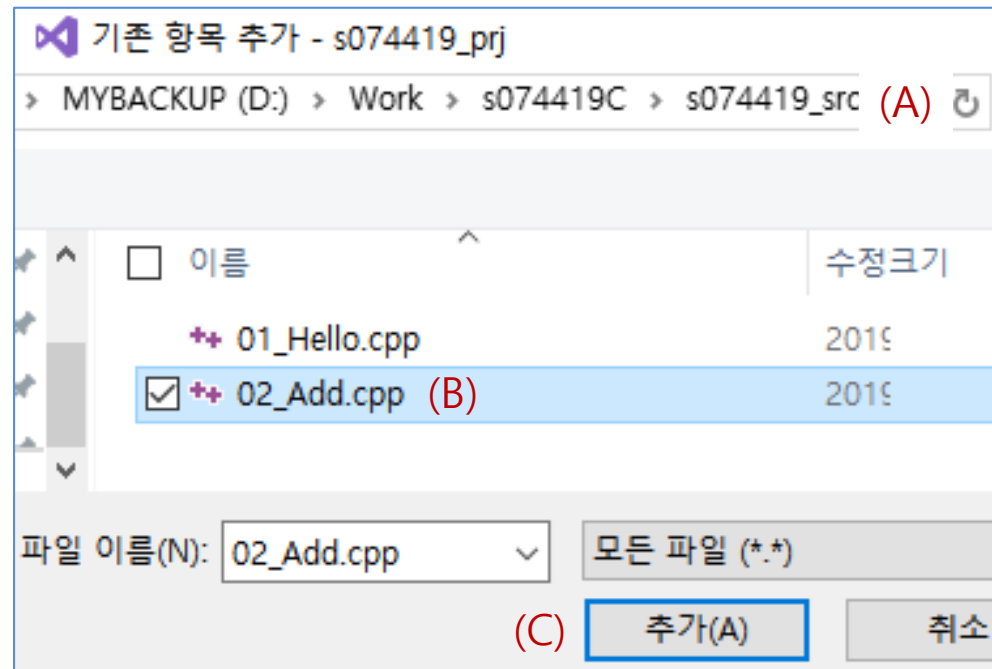
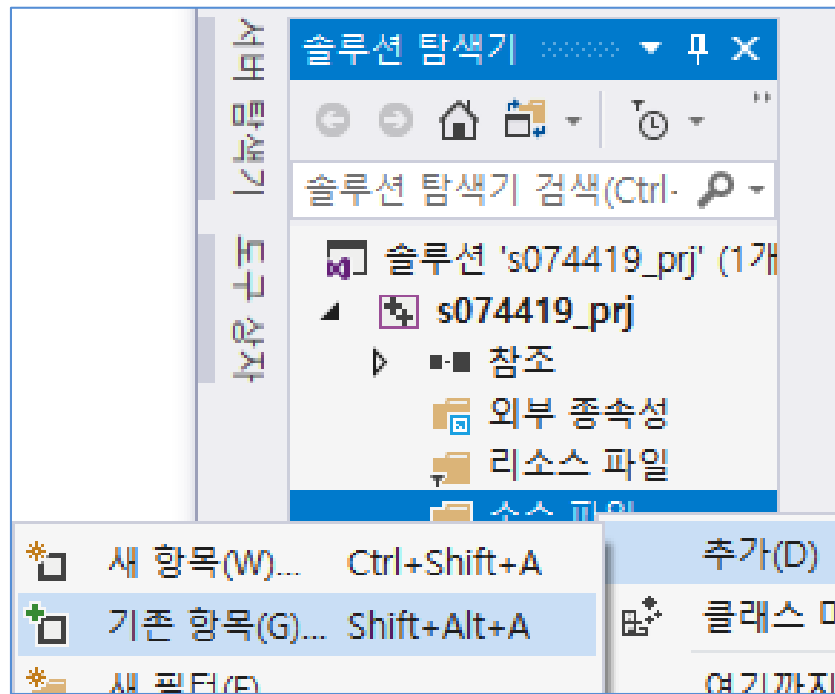
- ◆ 작성한 프로그램을 기존 만들어둔 폴더 `snnnnnnn_src`에 저장하자<sup>(1)</sup>.
- ◆ 그런데 메모장에서 파일을 저장하면 파일 이름 뒤에 확장자 `.txt`가 자동으로 붙기 때문에 주의가 필요하다.
- ◆ 메모장에서 **파일** → **클릭 다른 이름으로 저장** → 저장 폴더 `snnnnnnn_src`를 찾아 선택<sup>(A)</sup> → 파일이름 입력(`02_Add.cpp`)<sup>(B)</sup> → 파일 형식을 모든 파일`(*.*)`로 변경<sup>(C)</sup> → 저장<sup>(D)</sup>을 클릭하여 파일을 저장한다<sup>(E)</sup>.



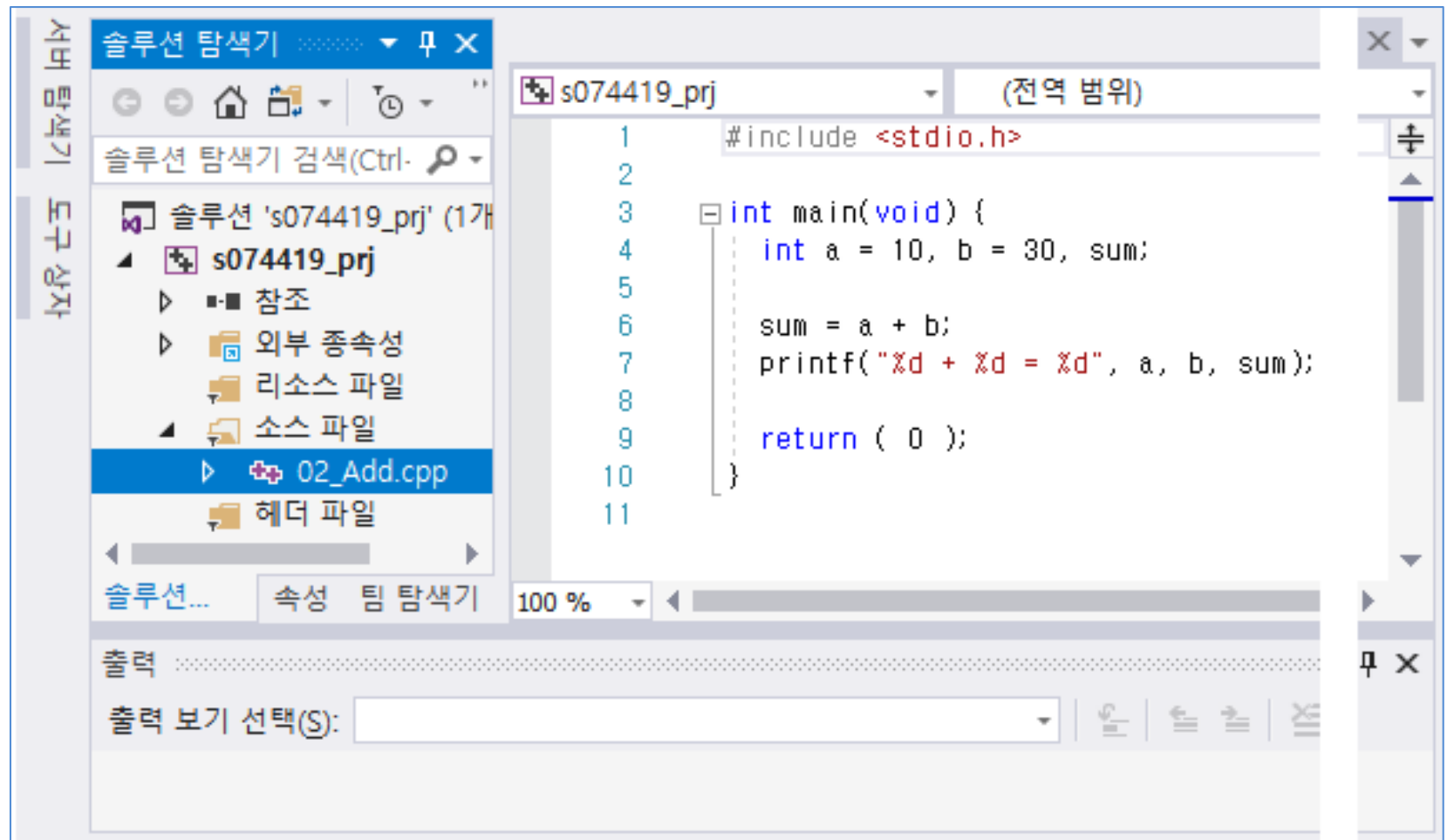
(1) 파일이름: `02_Add.cpp`(또는, `02_02_Add.cpp`)

## ◆ 프로그램 추가

- ◆ 새로 작성한 02\_Add.cpp를 프로젝트에 추가하자.
- ◆ 솔루션 탐색기 → 소스 파일 → 기존 항목 → 추가 → 기존 항목 추가 창에서 → 폴더 선택(A) → 파일 선택(B) → 추가(C)를 클릭한다.



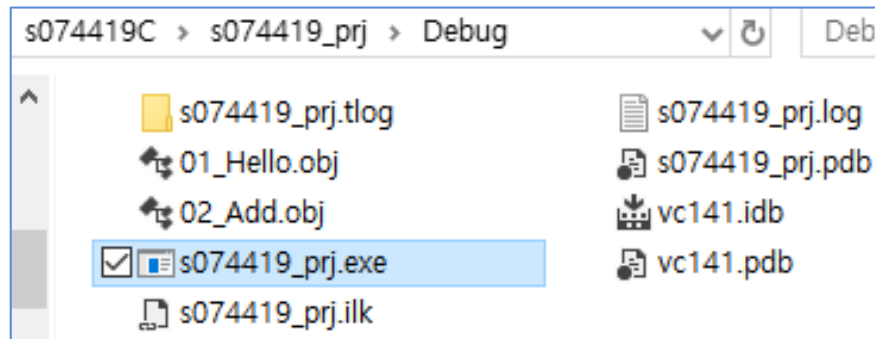
- ◆ 02\_Add.cpp를 추가한 후 Visual Studio 창을 아래에 보인다.
- ◆ 메뉴 바 → 빌드 → 솔루션 빌드 클릭하여 실행 파일을 만든다.



## ◆ 빌드 후 프로젝트 폴더 내용

### ◆ Debug 폴더

- ◆ 실행 파일은 그대로 s074419\_prj.exe이나 02\_Add.cpp에 대한 실행 파일이다.
- ◆ 02\_Add.obj 파일이 추가 되었다.
- ◆ .obj 파일은 실행 파일을 만들기 전 중단 단계 파일로 .cpp 파일 하나당 하나씩 생성되는데, 프로그램 완성 후 삭제해도 무방하다<sup>(1)</sup>.



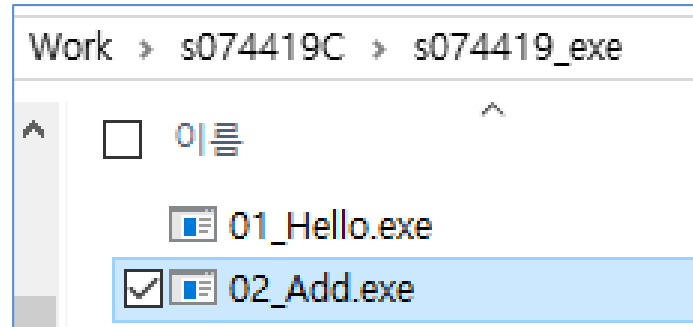
- ◆ 아래에 실행 파일을 명령 프롬프트에서 수행한 과정과 결과를 보인다.

```
C:\Users\Wcsr\im>d: (disk를 D로 바꾸기)
D:\>cd D:\Work\W\074419C\W\074419_prj\WDebug (대상 폴더 바꾸기)
D:\Work\W\074419C\W\074419_prj\WDebug>s074419_prj (실행 파일 수행)
10 + 30 = 40 (수행 결과 출력)
D:\Work\W\074419C\W\074419_prj\WDebug>
```

(1) 사실 Debug 폴더 전체를 삭제해도 무방하다(빌드를 수행하면 다시 생김).

## ◆ 실행 파일 보관

- ◆ 처음 작성한 프로그램과 마찬가지로, snnnnnnn\_exe 폴더에 실행 파일을 저장한다(파일 이름 변경 필요).



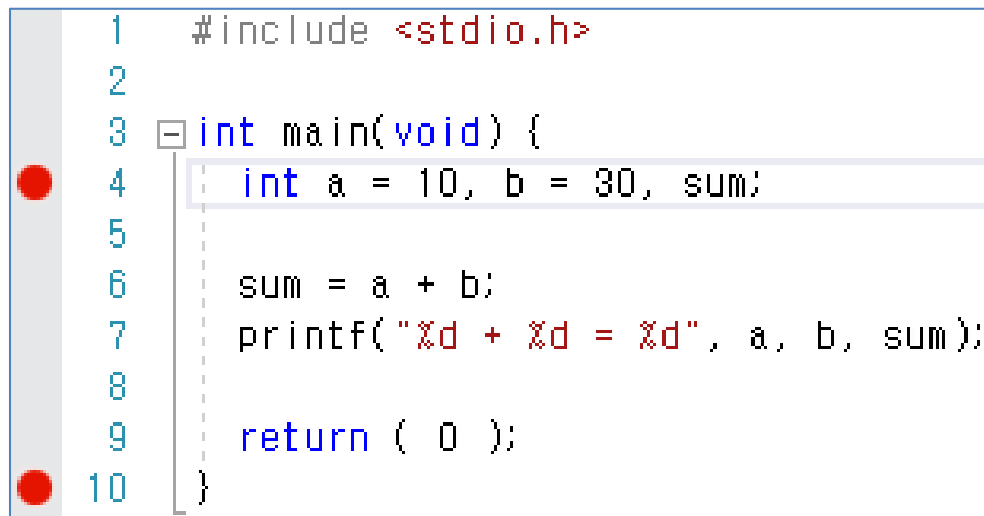
- ◆ 아래에 이 파일을 실행한 과정과 결과를 보인다.

```
C:\Users\Wcsr\im>d: ↵
D:\>cd D:\Work\Wcsr\im>s074419C\Wcsr\im>s074419_exe ↵
D:\Work\Wcsr\im>s074419C\Wcsr\im>s074419_exe>02_Add ↵
10 + 30 = 40
D:\Work\Wcsr\im>s074419C\Wcsr\im>s074419_exe>
```

# 편집 모드/디버깅 모드

## ◆ 편집 모드와 디버깅 모드

- ◆ Visual Studio에서는 프로그램의 statement를 하나 씩 수행하면서 오류 여부를 체크하는 모드가 있는데 이를 **디버깅 모드**라고 한다.
- ◆ 지금까지 사용하던 Visual Studio의 모드는 **편집 모드**라고 할 수 있다.
- ◆ 디버깅 모드는 프로그램의 실행 오류를 찾기 위하여 사용한다.
- ◆ 디버깅 모드에서 실행하려면 먼저 프로그램의 적당한 위치에 중단점 (break point)을 설정하여야 한다(단축키 **F9**으로 설정/해제)<sup>(1)</sup>.
- ◆ 아래 VS의 편집 모드에서 02\_Add.cpp에 중단 점을 설정한 예를 보인다.



```
1  #include <stdio.h>
2
3  int main(void) {
4      int a = 10, b = 30, sum;
5
6      sum = a + b;
7      printf("%d + %d = %d", a, b, sum);
8
9      return ( 0 );
10 }
```

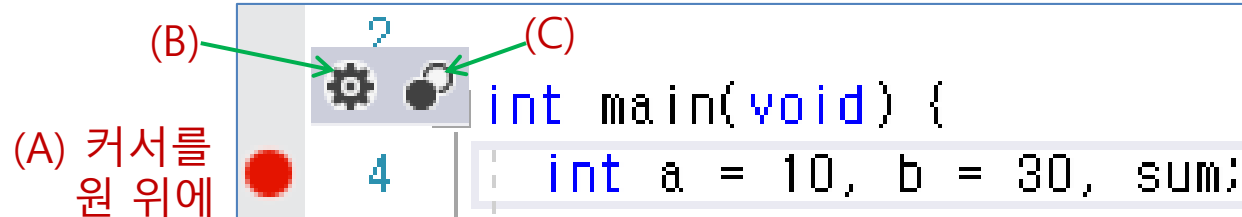
(1) 중단점은 프로그램의 해당 지점에서 실행을 잠시 멈추게 한다.





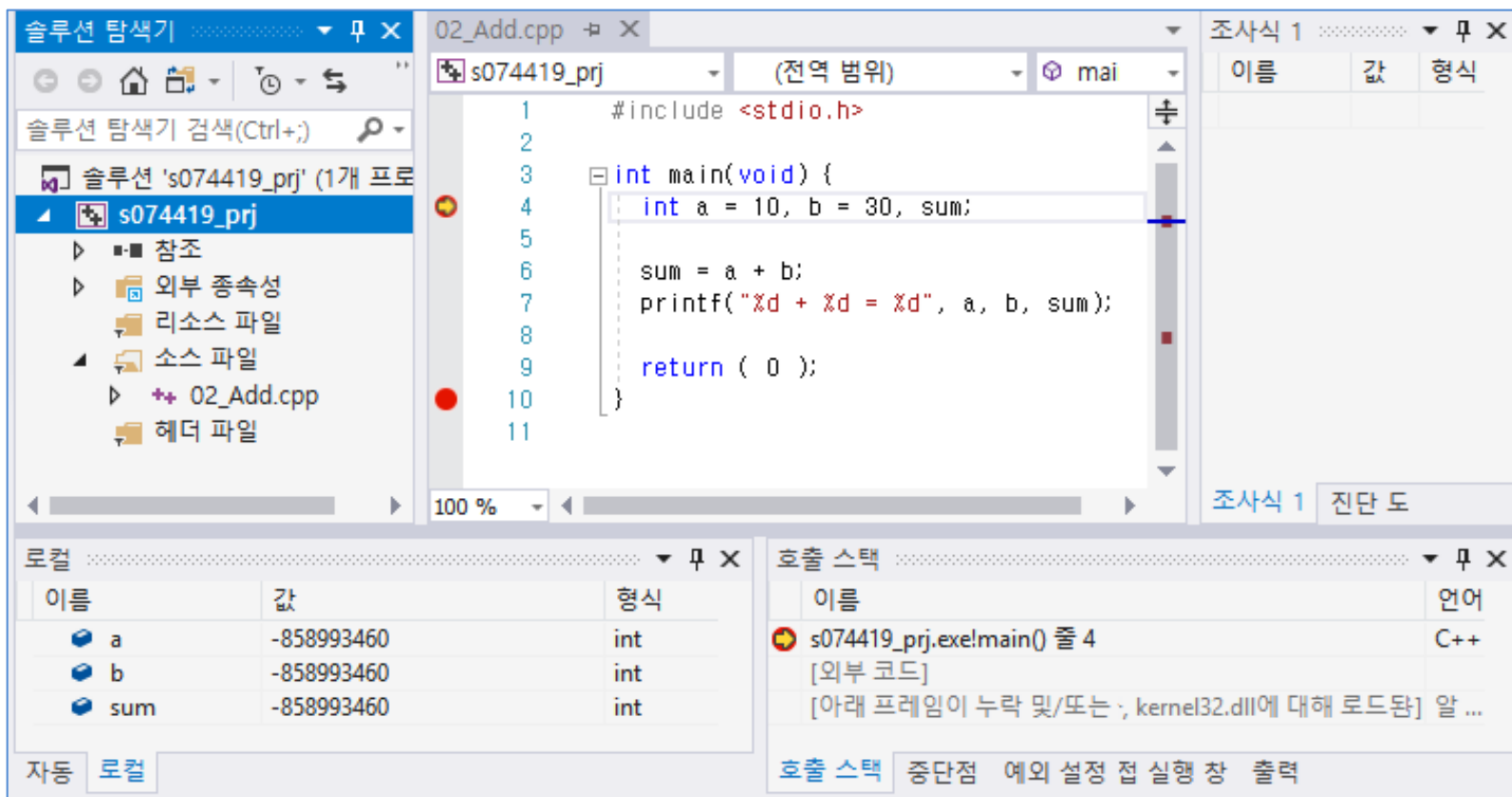
## ◆ 조건이 있는 중단점(Conditional Breakpoint)

- ◆ 중단점의 빨강 원에 커서를 가져가면(A) 아래와 같은 툴팁이 보인다. 이를 통하여 중단 점의 사용여부를 설정하거나(B), 어떤 조건을 주고 그 조건에 맞으면 중단 기능이 동작하게(C) 할 수 있다.



## ◆ 디버깅 모드 실행

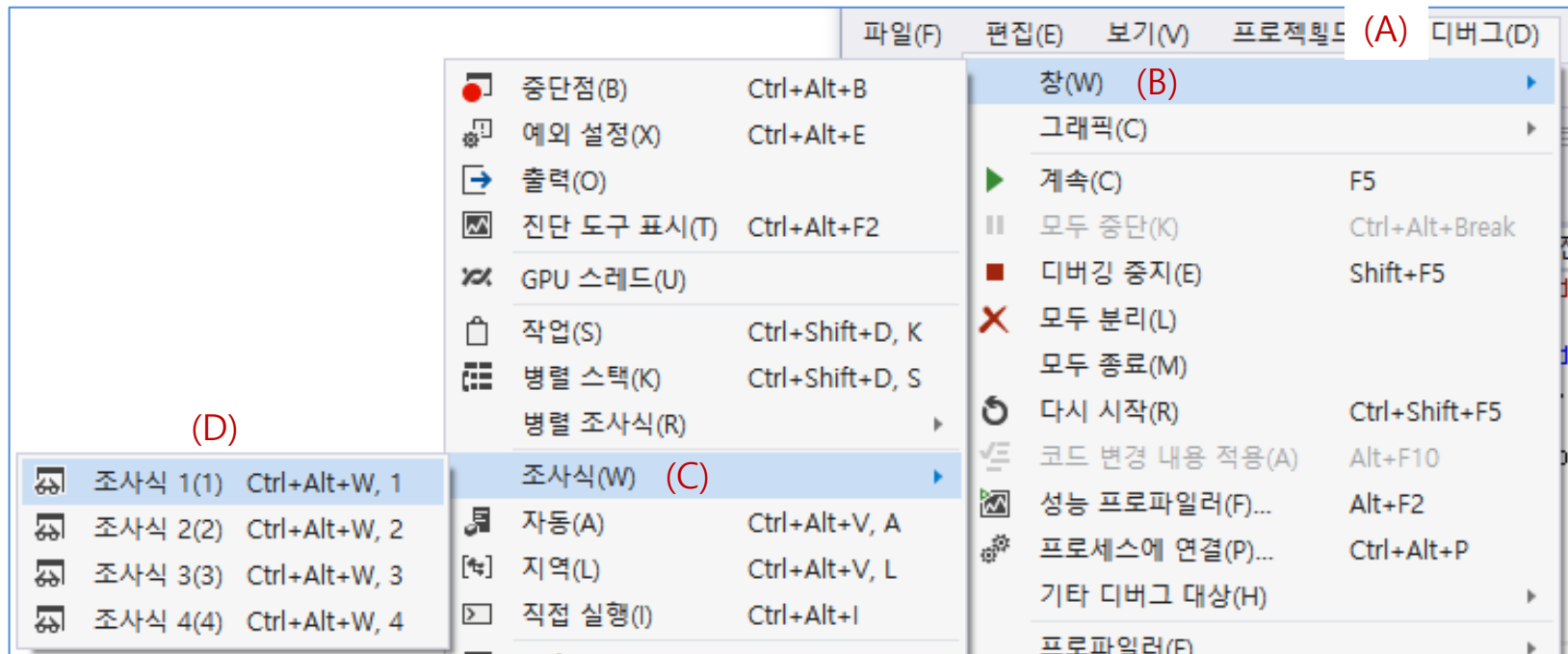
- ◆ 단축키 **F5**를 누르면 **디버깅 모드** (**Shift+F5**는 편집 모드로 되돌림).
- ◆ 디버깅 모드에서는 첫번째 중단점을 만날 때까지 프로그램을 실행하고 멈춘다(주의. 중단점이 있는 line의 명령은 실행하지 않는다).
- ◆ 디버깅 모드로 진입하면 VS 창이 바뀌게 되는데, 아래 그림처럼 레이아웃을 조정해보자(1).



(1) 우측의 조사식이 보이도록 배치한다(잘 안되면 이것 저것 시도해 보자).

## ◆ 조사식 창(Watch Window)

- ◆ 조사식 창은 프로그램 실행 중 변수들의 값의 변화를 알 수 있게 하여, 프로그램 오류를 신속히 찾을 수 있게 해주는 유용한 창이다.
- ◆ 조사식 창은 아래 그림과 같이 메뉴 바 → 디버그(A) → 창(W)(B) → 조사식(W)(C) → 조사식 1~4 중 선택(D) 하여 활성화 시킬 수 있다.
- ◆ 오직 디버깅 모드에서만 조사식(W)이 보이며, 총 4 개의 조사식 창을 열 수 있다.



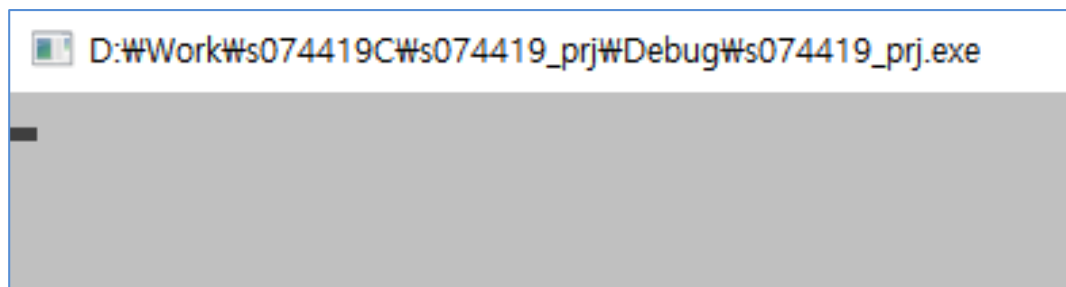
## ◆ 디버깅

- ◆ 디버깅 모드에 진입하면, **line 4**에서 실행을 멈춘다(A).
- ◆ 추가로, 명령 프롬프트와 유사한 **출력 창**이 VS 창과 별도로 **나타난다(B)**.
- ◆ 조사식 창에 프로그램에서 사용하는 변수를 그림과 같이 입력하면(C).
- ◆ 현재 변수 a, b, sum 값이 모두 초기화가 안되어 있는 상태로 보인다(D).

(A) 현재 멈춘 지점이 원 안 노랑 화살표로 표시된다.

이름	값	형식
a	-858993460	int
b	-858993460	int
sum	-858993460	int

(B) 디버그 모드에서는 프로그램의 입출력을 이 창을 통해서 행하여진다.



- ◆ 단축키 **F10**은 프로그램을 한 스텝(line) 진행하도록 한다.
- ◆ **F10**을 누르면 아래 그림과 같이 한 스텝 진행하여 **line 6에서 멈추고(A)**,  
line 4의 수행으로 인하여 **변수 a와 b 값이 초기화 된다(B)**.

(A)

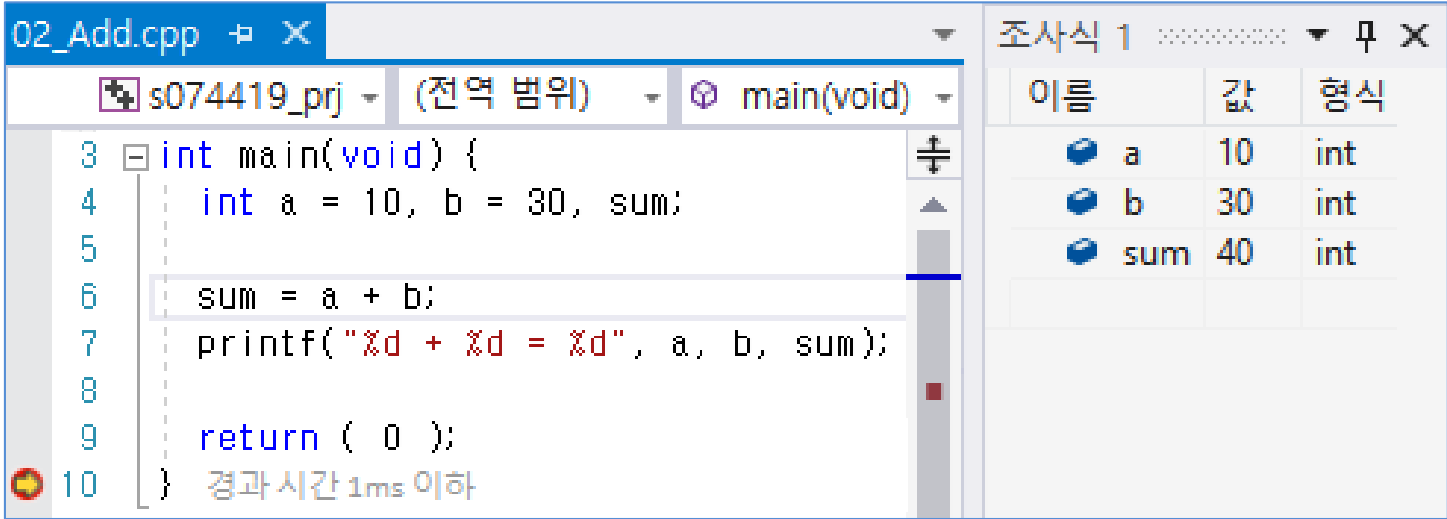
이름	값	형식
a	10	int
b	30	int
sum	-858993460	int

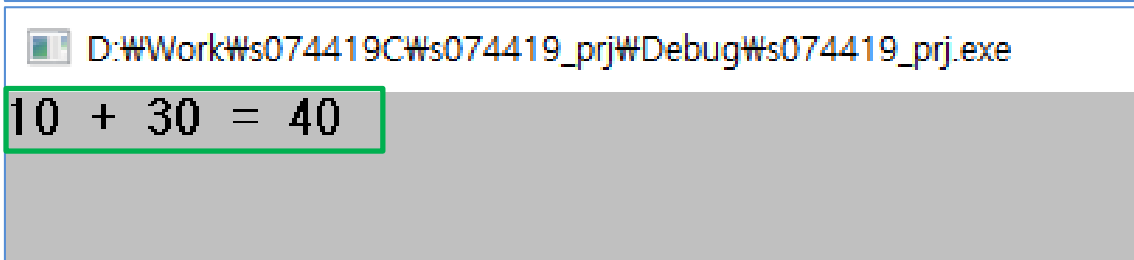
- ◆ 다시 **F10**을 누르면 **line 7에서 멈추며(A)**, **sum 값은 40으로 바뀐다(B)**.

(A)

이름	값	형식
a	10	int
b	30	int
sum	40	int

- ◆ 단축키 **F5**는 항상 현재 위치에서 다음 중단점까지 연속으로 실행한다.
- ◆ 이제, F5를 누르면 line 10에서 멈추고(A), line 7에서의 출력 결과가 출력창에 보이게 된다(B).

(A) 

(B) 

- ◆ 참고. 디버깅 모드에서도 소스 프로그램의 편집이 가능하다.
- ◆ 이제, 편집 모드로 돌아가려면 **Shift+F5**, 디버깅 모드에서 프로그램을 다시 실행하려면 **Ctrl+Shift+F5**를 누르면 된다.



## ◆ 디버깅 모드에서의 주요 단축키 모음

- ◆ 신속한 프로그램 개발을 위하여 디버깅 모드 사용에 익숙해져야 한다.
- ◆ 아래 디버깅 모드에서 유용한 주요 단축키 들을 보이니 가능한 암기 하도록 하자.

단축키	기능	비고
F7	빌드	수정한 파일이 있으면 다시 빌드
F5	빌드 + 실행	수정한 부분이 없으면 실행만
F9	중단점 설정	중단점(break point)
F10	줄 단위 실행	함수인 경우 함수 실행(step over)
F11	명령 단위 실행	함수이면 함수로 진입(step in)
F12	정의로 이동	변수, 함수 등의 선언으로 이동
Ctrl + '-' 키	이전 커서 위치로	F12 후 다시 돌아오는데 유용
Ctrl + F10	커서 위치까지 실행	현재 커서 위치까지 실행 후 멈춘다



# 실행시간 측정

## ◆ 실행 시간 측정

- ◆ 작성한 프로그램을 동일한 기능의 다른 프로그램과 그 성능을 비교할 때, 같은 입력에 대해 계산 시간, 사용한 메모리의 양을 측정한다.
- ◆ 프로그램의 실행 시간 측정 방법을 아래 예를 들어 보인다.

```
#include <time.h>                // time.h 필요

int main(void) {
    clock_t start, finish;
    double timeA = 0;

    start = clock(); // 코드 A 실행 바로 전 시각

체크할 코드 A



    finish = clock(); // 코드 A 실행 후 시각
    timeA += ((double)(finish - start)) / CLK_TCK;
    // timeA is a measure of execution time

    return(0);
}
```





## ◆ 두 군데 이상 체크할 경우

```
#include <time.h>                // time.h 필요

int main(void) {
    clock_t start, finish;
    double timeA = 0;

    start = clock(); // 코드 A1 실행 바로 전 시각
    

체크할 코드 A1


    finish = clock(); // 코드 A1 실행 후 시각
    timeA += ((double) (finish - start)) / CLK_TCK;

    . . .

    start = clock(); // 코드 A2 실행 바로 전 시각
    

체크할 코드 A2


    finish = clock(); // 코드 A2 실행 후 시각
    timeA += ((double) (finish - start)) / CLK_TCK;

    printf("Execution time is %.2f.", timeA);
    return(0);
}
```

# Multi-File Programming

## ◆ Multi-File Programming

- ◆ 대규모 프로그램을 작성하기 위하여 사용된다.
- ◆ 프로그램을 함수 또는 기능에 따라 다수의 파일로 프로젝트를 구성한다.
- ◆ 파일 간에는 헤더 파일(~.h 파일)을 사용하여 필요한 사항을 공유한다.
- ◆ 다음과 같은 헤더 파일을 사용할 수 있다(고정된 사항은 아님)

### global.h

```
// global 변수를 정의
// main 함수에 한번만 포함시킨다
int gx, gy;    // 0으로 초기화
double f = 10.0;
```

### extern.h

```
/* global 변수를 다른 파일에 알려줄
때 사용한다. global.h를 포함한 파일
을 제외한 모든 ~.cpp 파일에 포함시킨
다. */
extern int gx, gy; // extern은
extern double f;   // 초기화 불가
```

### include.h

```
// 필요한 헤더 파일을 포함
#include <stdio.h>
#include <string.h> // etc
```

### function.h

```
// 작성한 함수 정의 포함
void func1(...);
Int func2(...); // etc
```

### define.h

```
// 각종 정의 포함
#define PI 3.1315
#define ... // etc
```



## ◆ 프로그램 구성

- ◆ `main()`, `func1()`, `func2()` 등 세 개의 함수로 구성되어 있다고 하자.
- ◆ 함수들을 각각 다른 파일에 저장한다면, 다음과 같은 형태로 파일을 편집한다.

### main.cpp

```
#include "include.h"
#include "define.h"
#include "global.h"
#include "function.h"

int main(void) {
    ...
}
```

### func1.cpp

```
#include "include.h"
#include "define.h"
#include "extern.h"
#include "function.h"

void func1(...) {
    ...
}
```

### Visual Studio에 등록

- ~.cpp 파일들은 소스 파일에 등록하고, ~.h 파일은 헤더 파일에 등록한다.
- 소스 파일과 헤더 파일 폴더 안에 폴더를 추가하여, 파일들을 분류하여 등록할 수 있다.
- 이하 빌드 방법은 단일 파일 빌드와 동일하다.

### func2.cpp

```
#include "include.h"
#include "define.h"
#include "extern.h"
#include "function.h"

int func2(...) {
    ...
}
```