

1주차 : Visual Studio와 I/O

강사: 표지원

0. 시작하기에 앞서...

시작하기에 앞서...

알로하에 오신 모든 신입생, 재학생 여러분들
진심으로 축하하고 환영합니다~!~!~!

(와아아아)



(대충 환영한다는 짤)

저를 포함한 운영진 모두가 여러분을 학교에서 보고 싶어하지만.....!!

상황이 상황인지라 그러지 못해 너무 아쉽네요 $\pi\pi\pi\pi$... 어쩔 수 없죠.. (아니 코로나 뭐냐고;;;;)

시작하기에 앞서...

앞으로 예정되어있는 거의 모든 회합은,
스스로 자습이 가능할 정도의 **강의자료**를 **배부**하는 것으로 대체하고
철저한 멘토링 과정을 통해 멘티가 잘 따라 올 수 있도록 최선을 다 해 드릴 것입니다.

또한! 저희 모두가 모든 과정 속에서 **"질문"**이 특히 **매우 중요**하다고 생각하기 때문에,
멘토링 방 이외에 **전체 질문 방**을 두어, 활발한 의사소통이 일어 날 수 있게 할 것입니다.여러분의 피드백이 이런 상황에서 더욱 중요하니 자주 의견을 전달해 주시면 감사하겠습니다.

자 그래서...

원래 개강 예정일이었던 3월 2일을 지나, 드디어 첫 회합날이 다가왔습니다!

이번 시간에는...

1. Visual Studio 소개 및 설치

2. C문법을 통한 기초 I/O

이거 두개 할거예요 ㅎㅎㅎ 레츠기릿

1. Visual Studio 소개 및 설치

Visual Studio?

우리가 **앞으로 배우고 적용해 볼 일**들은 한 마디로 **"컴퓨터에 명령을 내리는 일"**입니다.
컴퓨터는 0과 1밖에 모르는데, 우리 중 어느 누구도 0과 1로 명령을 내리고 싶지 않을 거예요.
초기에는 0과 1로만 계산했는데, 시간이 흘러 과학자들이 좀 더 나은 방법을 만들어냈고
그게 바로, 프로그래밍 언어입니다! 프로그래밍 언어라는 말은 다들 들어 보셨을 거예요.

아마 컴퓨터소프트웨어학부 신입생이시라면, 지금 소입설 시간에 Python을 배우고 있을 것이고,
C언어나 Java와 같은 말도 들어 본 적이 있겠죠. (알고만 있다면 이미 당신은 1주차 끝)

"코딩"하면 생각나는 그 멋진 영어 문장들 그게 바로 **프로그래밍 언어**입니다!

Visual Studio!

알로하 에서는 프로그래밍 언어 중 C/C++을 주로 사용할 건데, Visual Studio라는 프로그램을 저희가 하고 싶은 명령을 언어로 적으면 이 언어를 아주 잘 번역해줘요.

물론 CodeBlocks나 Dev C++처럼 C++을 번역해주는 다른 통역가들이 많지만, VS가 제일 편하기 때문에, 애를 주로 사용할 거예요.

우리는 이것을 VS 홈페이지에서 최신버전으로 편하게 설치 할 수 있습니다 ㅎㅎㅎㅎ



우선 말씀드립니다!



C++만을 위한 환경을 만든다면, Visual Studio는 여유잡아 7GB정도의 공간을 요구합니다. (비워두세용)

또한 소요되는 시간이 매우 길어요. WIFI 환경을 이용합시당. (저는 30분 정도 걸렸습니다.)



우선 말씀드립니다!



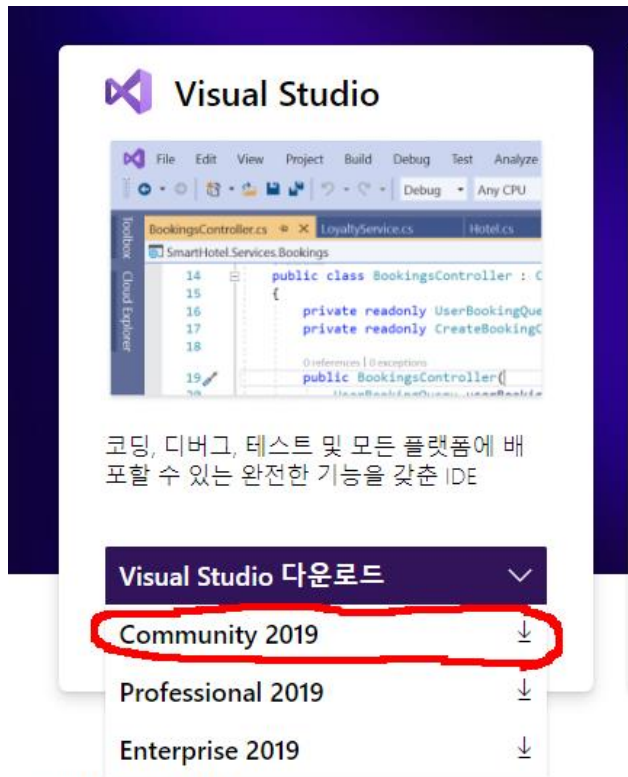
Visual Studio Community 2019(혹은 C++언어 기반 컴파일러)는
두 번째 회합 이전까지는 꼭!!!! 설치해 주셔야 합니다.

만약 설치 중에 화면이 넘어가지 않는다. 혹은 **에러가 뜬다면?**

→ **곧바로 질문방**에 알려주세요!!

아무 선배의 답이 없다면, 010 - 2221 - 7086으로 문자 혹은 전화주세요.

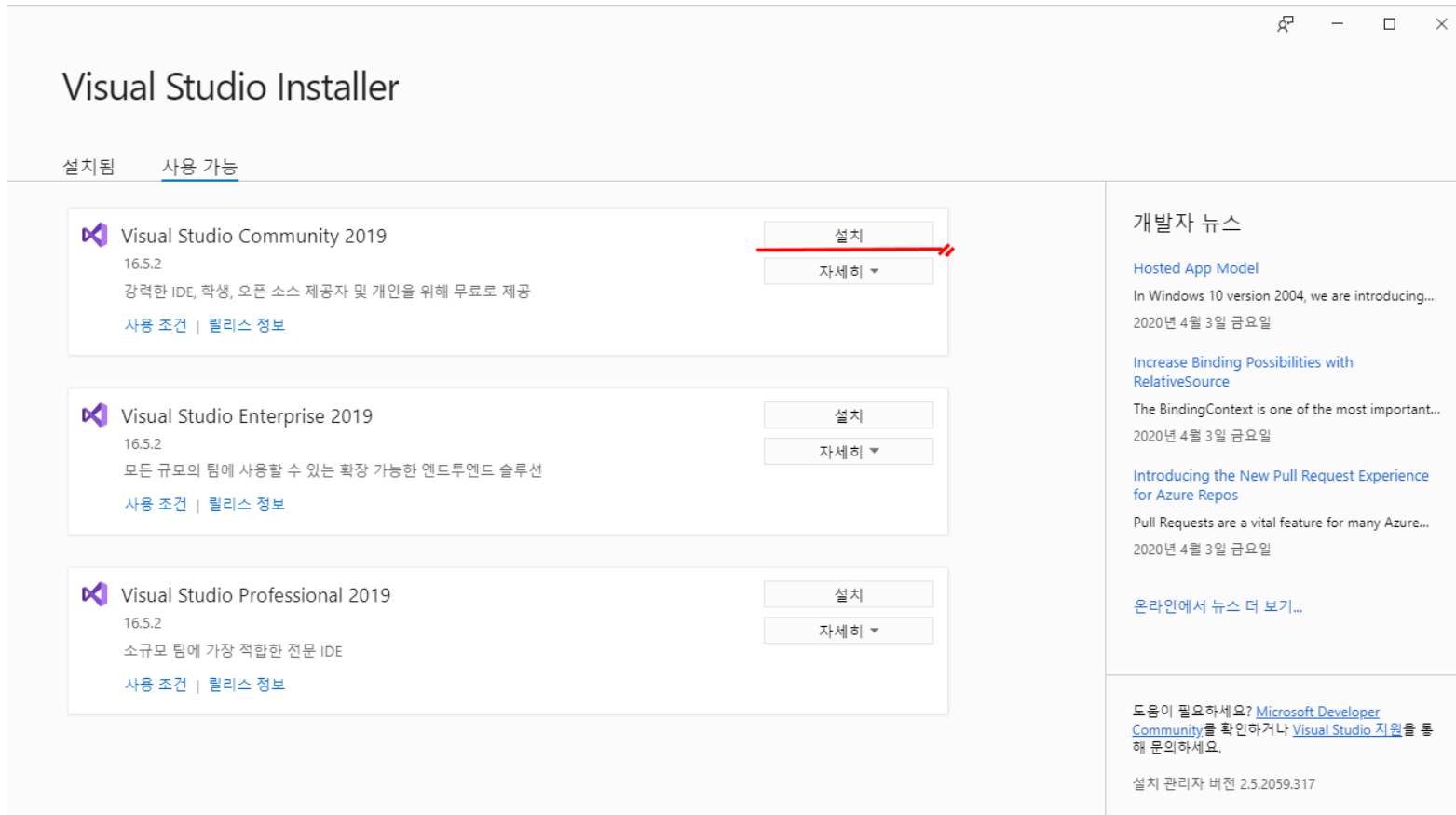
Visual Studio를 설치해 봅시다



개발자 서비스

먼저, <https://visualstudio.microsoft.com/ko/> 으로 들어가신 후 왼쪽 사진에 빨간 동그라미 처리 되어있는 부분을 클릭하여, VS 인스톨러를 받아주세요.

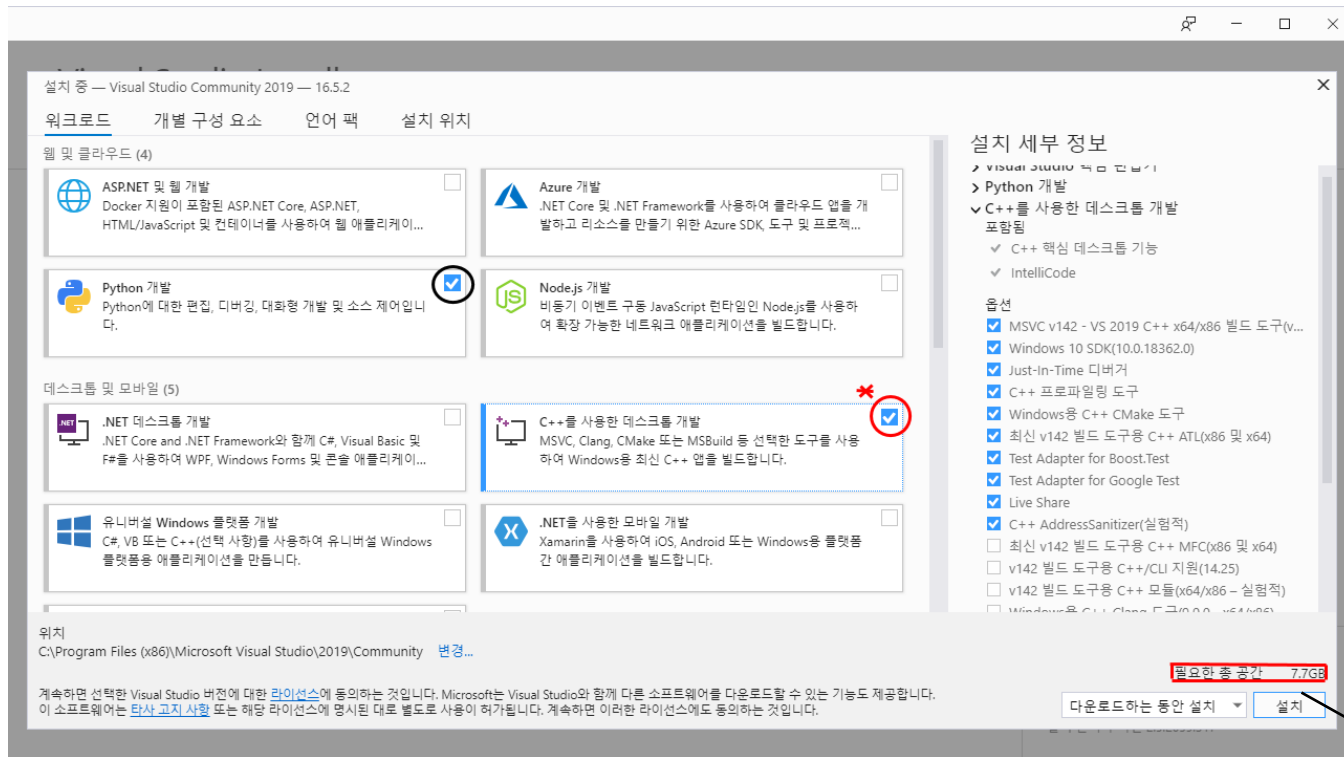
Visual Studio를 설치해 봅시다



그러면 다음과 같은 화면이 뜰 거예요.

여기서, 빨간 부분을 눌러주세요.

Visual Studio를 설치해 봅시다



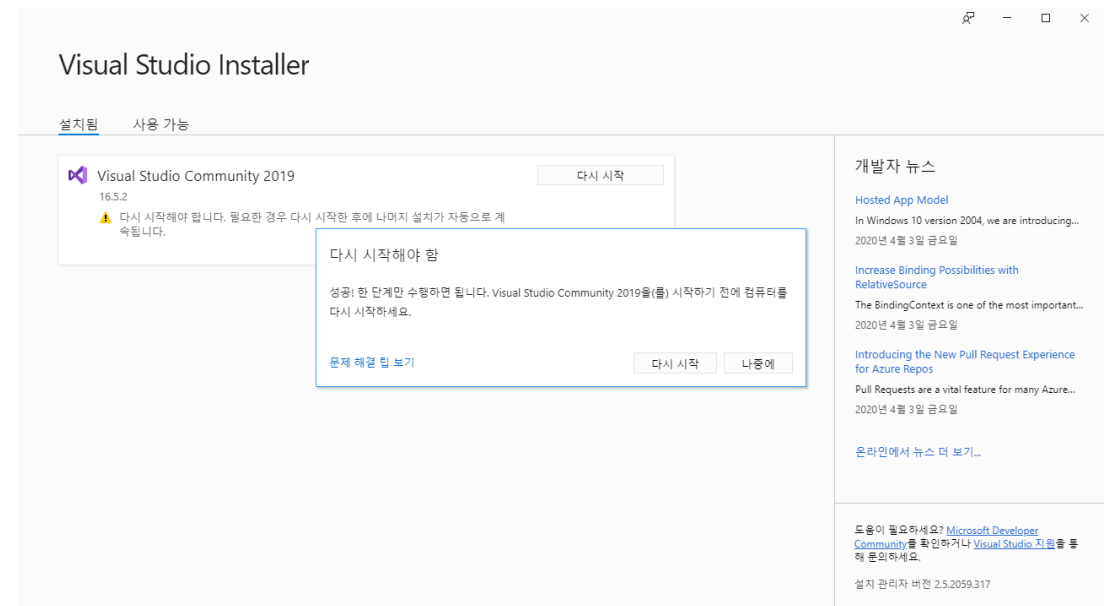
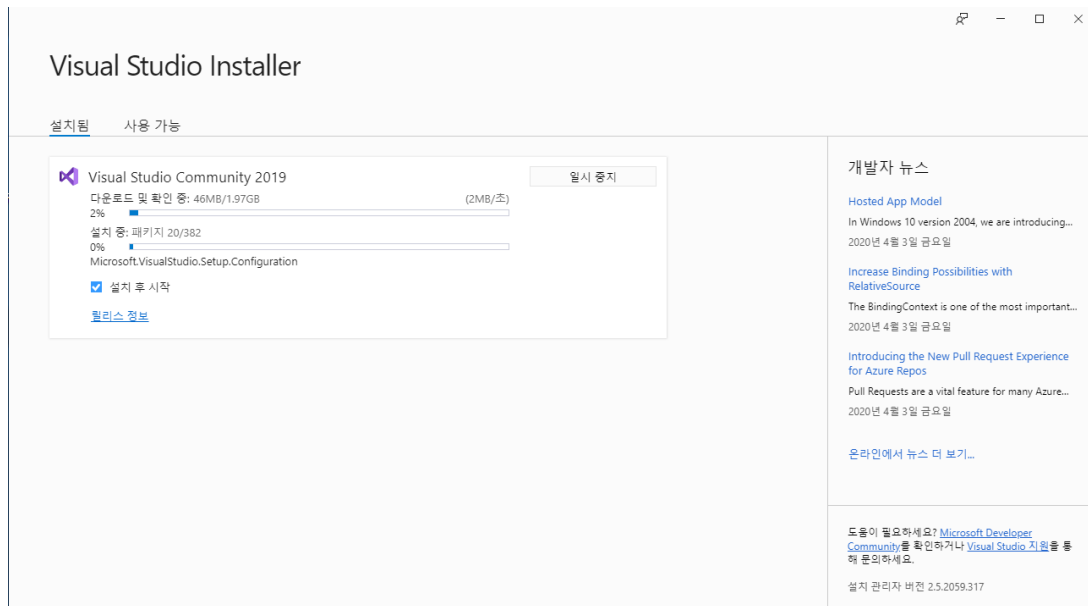
이 부분에서 우리가 사용할 기능을
고를 수가 있는데,

필수적으로 Visual C++은 하셔야 하고,
만약 Python도 Visual Studio에서 쓰고싶다면
체크하셔서 설치하시면 되겠습니다.

(대신, 시간과 용량이 더 들어가요.)

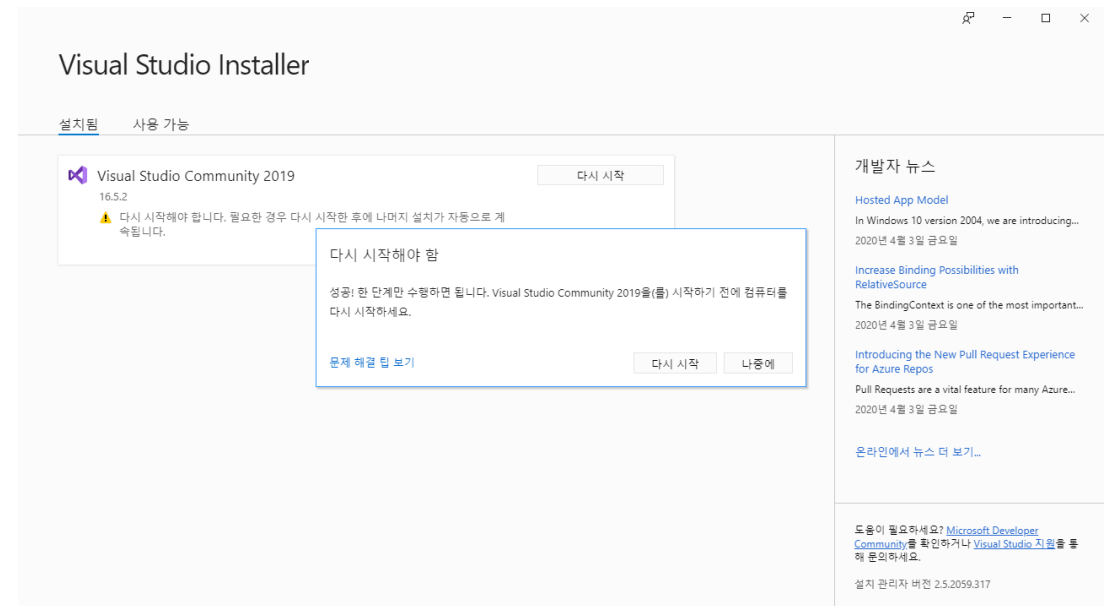
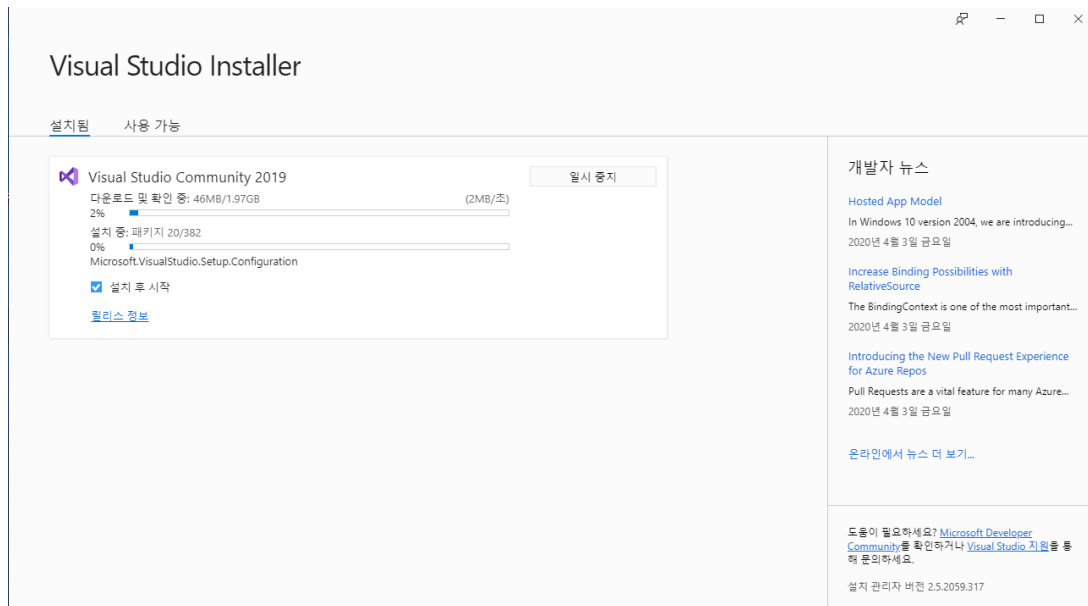
요거 7.7GB입니다!

Visual Studio를 설치해 봅시다



후루룩 넘기시고
다시시작 안하면 진행 안되므로 재부팅 해주세요

Visual Studio를 설치해 봅시다



후루룩 넘기시고
다시시작 안하면 진행 안되므로 재부팅 해주세요

Visual Studio를 설치해 봅시다

x

Visual Studio

시작합니다.

모든 개발자 서비스에 연결하세요.

Azure 크레딧 사용을 시작하고, 프라이빗 Git 리포지토리에 코드를
게시하고, 설정을 동기화하고, IDE 잠금을 해제하려면 로그인합니다.

[자세히](#)

설치가 완료되면 왼쪽과 같은 창이 나옵니다.

우선 지금은 나중에 로그인을 누르도록 해요.

로그인()

계정이 없는 경우 [새로 만드세요!](#)

[나중에 로그인](#)

Visual Studio를 설치해 봅시다

Visual Studio

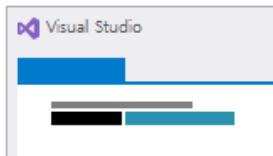
친숙한 환경에서 시작

개발 설정(V):

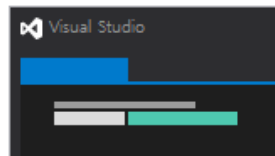
Visual C++

색 테마 선택

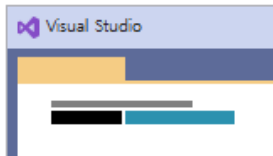
☐ 광원



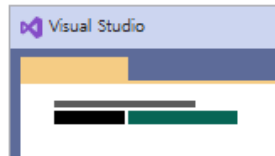
☒ 어둡게



☐ 파랑



☐ 파랑(추가 대비)



나중에 언제든지 이 설정을 변경할 수 있습니다.

Visual Studio 시작(S)

또한 우리는 C/C++ 환경을 원하기 때문에 빨간 네모부분을 반드시 사진과 같게 해주세요.

아래 배경 색상은 개인 취향대로 골라주고 넘어가 주시면 됩니다.
저는 코드가 눈에 더 잘 보이는 어두운 배경을 골랐습니다.

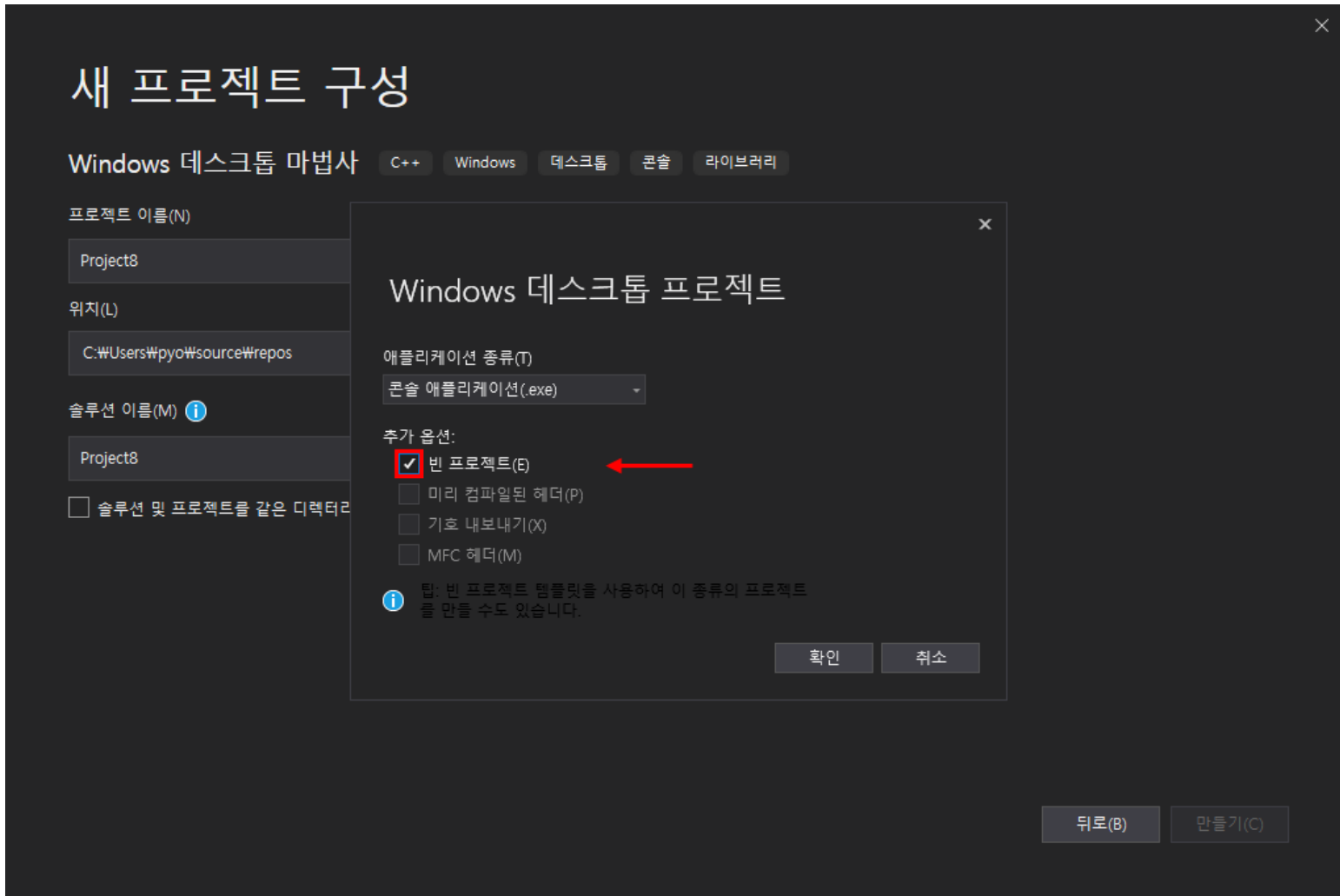
Visual Studio를 설치해 봅시다



자! 이제 프로젝트 생성입니다.

새 프로젝트 만들기를 누르고,
데스크톱 마법사로 열어주세요.

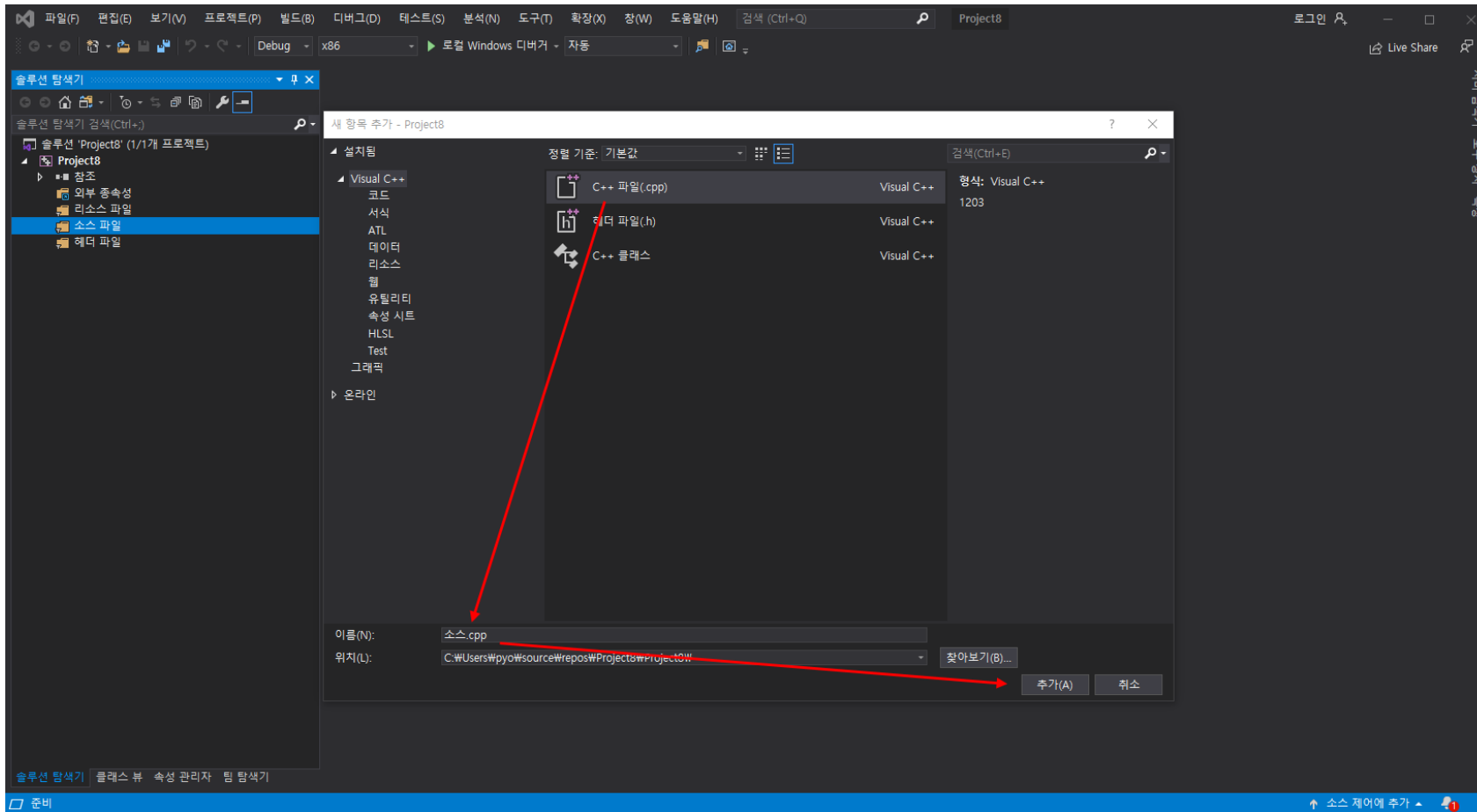
Visual Studio를 설치해 봅시다



원하는 프로젝트 이름을 정하시고,

꼭! 빈 프로젝트 부분에만
체크해서 생성해 주세요.

Visual Studio를 설치해 봅시다



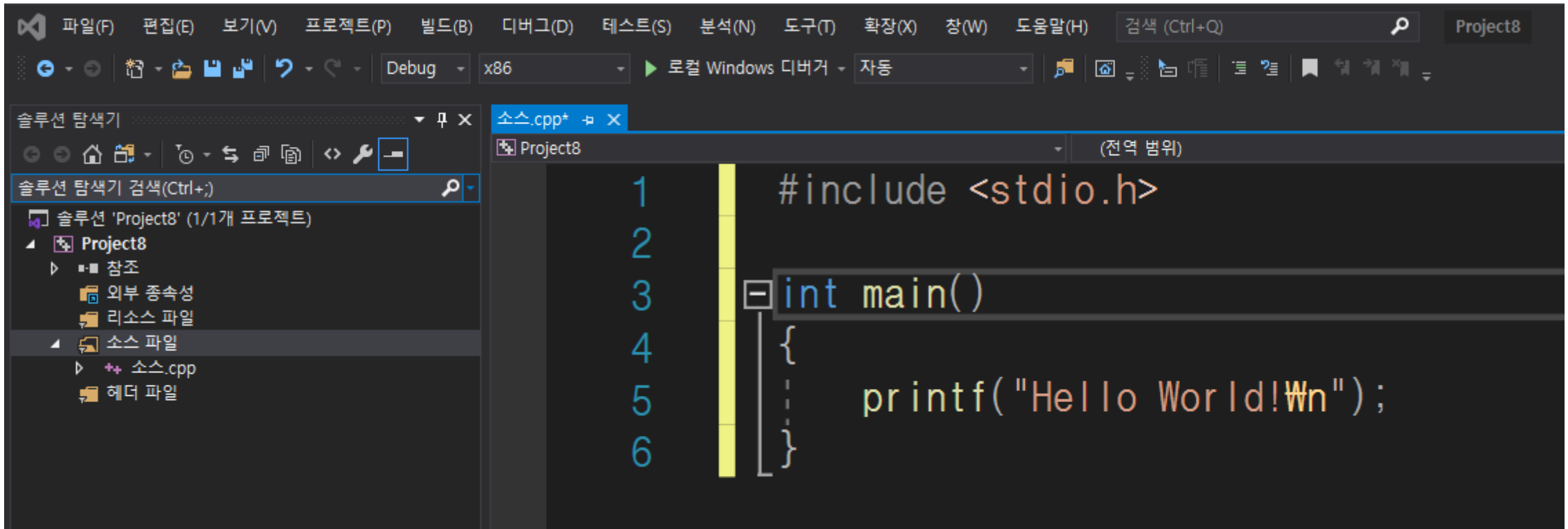
이제 소스파일을 생성해봅시다.

메인에서 Ctrl + Shift + A

혹은

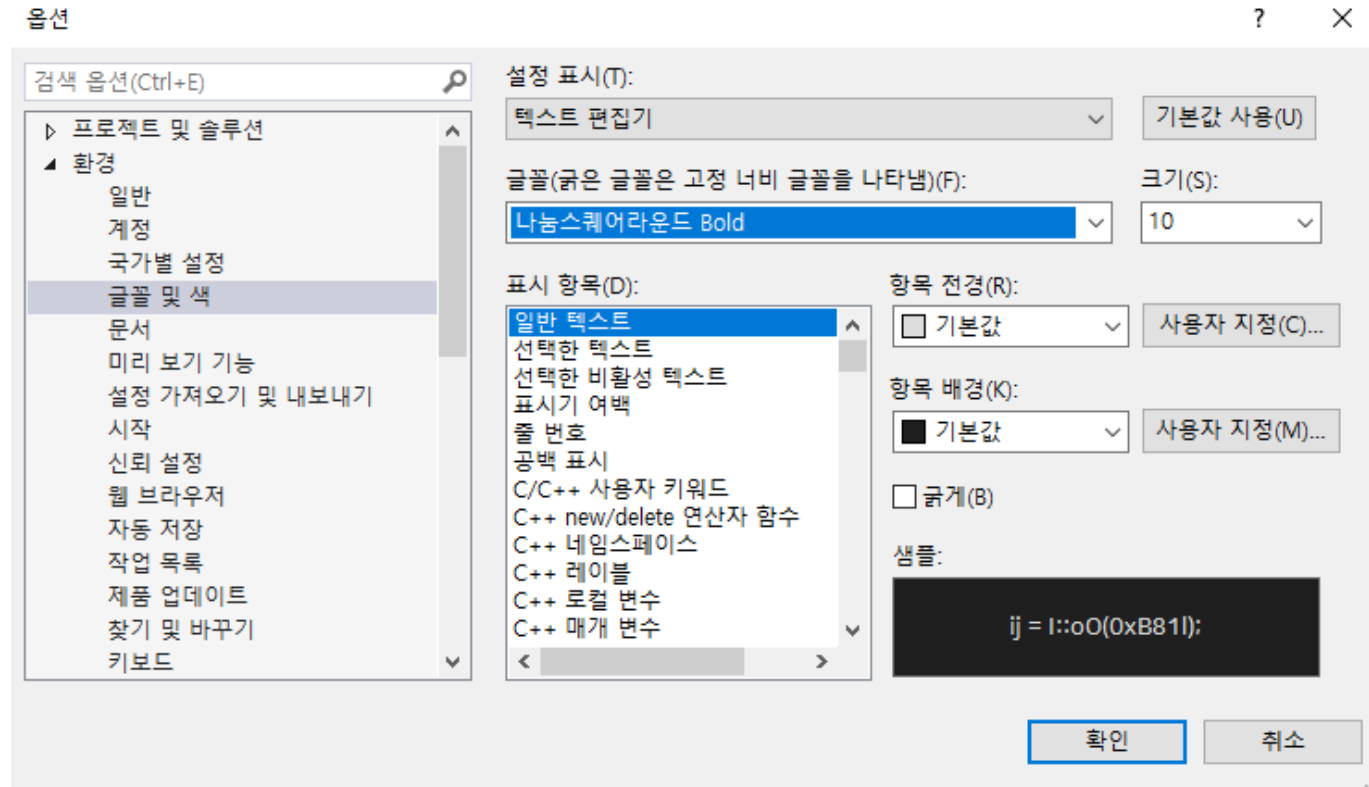
소스파일(우클릭) - 추가 - 새 항목
으로 생성할 수 있습니다.

Visual Studio를 설치해 봅시다



그럼 소스창이 열리고, 이제 배우기 위한 모든 준비는 끝 마쳤습니다.

Visual Studio를 설치해 봅시다



(전 이게 이쁘더라구요)

추가적으로, 폰트도 설정할 수도 있는데, 편의를 위해서
네이버 [D2Coding](#) 체를 다운받아 사용 할 수도 있습니다.

2. C문법을 통한 기초 I/O

C문법을 통한 기초 I/O

이번 시간이 첫 회합인 만큼, 오늘은 아주 가벼운 얘기를 해볼까 해요!

바로 **I/O**, **입력** 및 **출력**에 대한 내용이죠. (Input / Output)

모든 코드는, 자발적 참여를 위해서 코드를 캡처 해서 붙였습니다. (히히)

C문법을 통한 기초 I/O

```
1  #include <stdio.h>
2  // 헤더파일
3
4  int main()
5  {
6      int x;
7      // 입력을 받을 변수를, 정수형으로 정해서 선언했다.
8
9      scanf("%d", &x);
10     // 입력이다!
11     printf("%d", x);
12     // 출력이다!
13
14     return 0;
15 }
```

이 수업을 듣는 여러분 시점에서 말을 해볼게요.

여러분은 이제 막 소입설 시간에 Python을 배워서 입출력을 받고, 다양한 것들을 표현해 보았을거예요. 정수, 실수형에 대한 내용도 어느정도 알고는 있겠죠.

물론 C를 이전에 접해봤다면 더더더욱 쉽겠지만!
Python의 내용만 알고 있어도 괜찮습니다. (ez)

지금부터 x라는 정수형 변수에, 값을 받고, 출력해보고자 합니다.

C문법을 통한 기초 I/O

```
1  #include <stdio.h>
2  // 헤더파일
3
4  int main()
5  {
6      int x;
7      // 입력을 받을 변수를, 정수형으로 정해서 선언했다.
8
9      scanf("%d", &x);
10     // 입력이다!
11     printf("%d", x);
12     // 출력이다!
13
14     return 0;
15 }
```

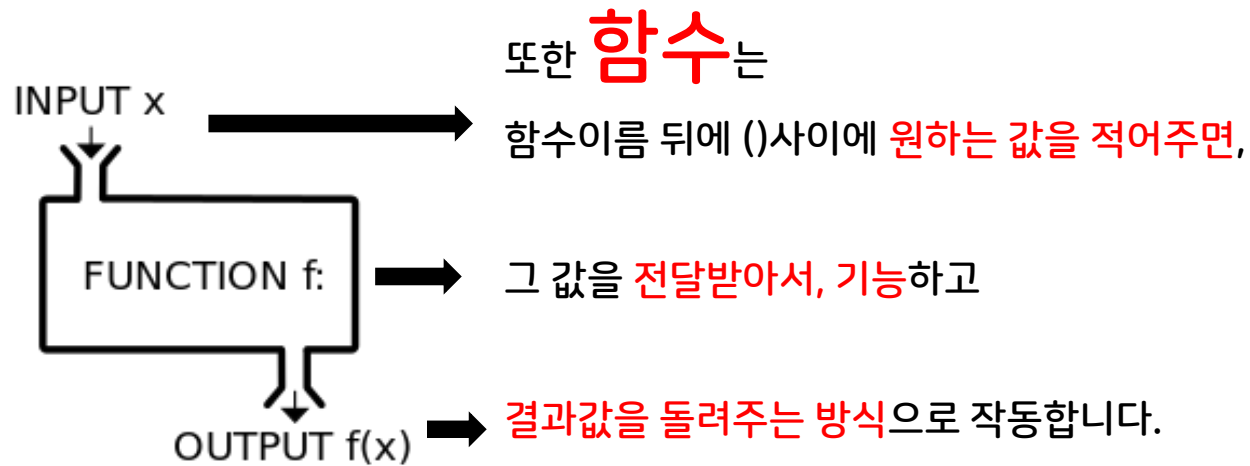
우선 첫 줄 #include <stdio.h>

우리는 이제 C언어를 쓴다고 하면,
대부분의 명령이 **어떠한 내용을 담는 함수**를 통해 표현 될 거예요.
그 함수는 **헤더**라는 곳에 이런저런 함수들이 모아져 있는데,
우리는 항상 시작할 때 저렇게 적어주어야 한답니다.

(헤더도 종류별로 있습니다.)

(*부록 참조)

C문법을 통한 기초 I/O



근데 대부분의 함수는 미리 정의가 되어있어요!

따라서 우리는, 그 함수가

1. 어떤 기능을 하는지 만 알고
2. 쓰는 법만 알면 되는 것이죠.

(우리가 함수를 직접 만들 수도 있는데, 나~중에 보도록 해요.)

C문법을 통한 기초 I/O

```
1  #include <stdio.h>
2  // 헤더파일
3
4  int main()
5  {
6      int x;
7      // 입력을 받을 변수를, 정수형으로 정해서 선언했다.
8
9      scanf("%d", &x);
10     // 입력이다!
11     printf("%d", x);
12     // 출력이다!
13
14     return 0;
15 }
```

int main()

프로그램의 모든 동작은 main 함수를 실행함으로써 이루어집니다.
따라서, main 함수 안에 원하는 명령을 넣으면 돼요.

그리고, Python과 조금 다르게

반드시 **명령 한 줄을 적으면 ;을 반드시 적어주어야 합니다.**

그게 "하나의 명령이 끝났다."라는 표시이거든요.

변수를 선언할 때에는,

`TYPENAME_ VAR_NAME_ = VAL;` 순으로 적어요.

여기서는, 정수형(int) 변수인 `x`를 초기화하지 않고 선언했네요.

(값을 처음에 넣지 않았다는 의미예요.)

C문법을 통한 기초 I/O

```
1  #include <stdio.h>
2  // 헤더파일
3
4  int main()
5  {
6      int x;
7      // 입력을 받을 변수를, 정수형으로 정해서 선언했다.
8
9      scanf("%d", &x);
10     // 입력이다!
11     printf("%d", x);
12     // 출력이다!
13
14     return 0;
15 }
```

scanf("%d", &x)부분 입니다.

이 함수는 우리가 STDIO헤더에서 불러온 **표준 입력 함수**예요!

어려운가요? 그냥 우리가 적으면 그 **입력을 받게 해주는** 함수예요.

"와 "사이에, %를 이용하여 값을 받는 형식을 정해주고,

뒤에 순서대로 값이 어디에 들어가야 하는지를 열거해주면 되는데,
숫자 대입과 느낌이 매우 비슷합니다.

대신, 뒤에 **열거하는 부분**에는 변수명이 아닌,

&를 변수명과 붙여서 적어주어야 합니다.

&연산자는, 자세한 내용은 나중에 포인터 시간에 배우도록 합시다!

(**부록 참조)

C문법을 통한 기초 I/O

```
1  #include <stdio.h>
2  // 헤더파일
3
4  int main()
5  {
6      int x;
7      // 입력을 받을 변수를, 정수형으로 정해서 선언했다.
8
9      scanf("%d", &x);
10     // 입력이다!
11     printf("%d", x);
12     // 출력이다!
13
14     return 0;
15 }
```

이때! 형식지정자는

%d → 정수형 (int)

%lld → 정수형 (long long int)

%lf → 실수형 (double)

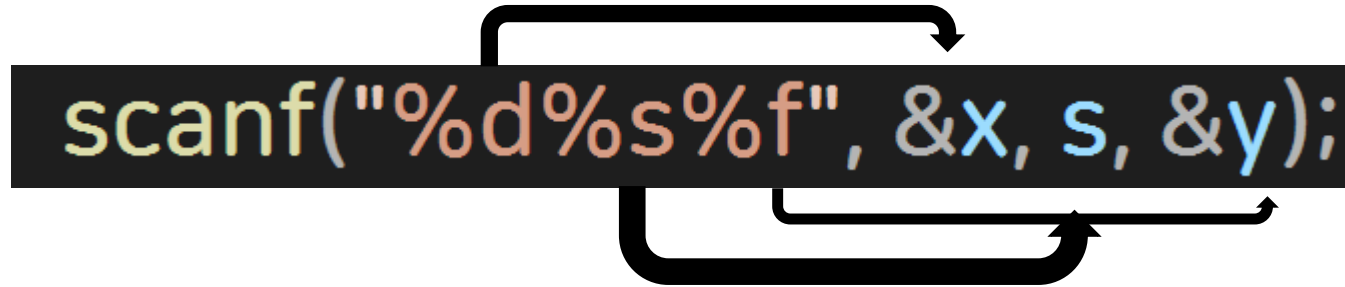
%s → 문자열 (문자열은 문자의 배열이에요. 나중에 배열 시간때 다뤄보도록 합시다!)

%c → 문자형 (char)

형식이든 입력이든 출력이든 항상 꼭 맞춰주어야 한답니다!

(***부록 참조)

C문법을 통한 기초 I/O



```
scanf("%d%s%f", &x, s, &y);
```

The diagram shows the `scanf` function call. A bracket above the format string `"%d%s%f"` points to the first argument `&x`. A bracket below the format string points to the second argument `s`. A bracket below the format string points to the third argument `&y`.

또한 여러 개를 입력 받을 수도 있습니다.

입력을 구분하는 기준은 **white space**(공백)입니다.

이때 **White Space**는, 스페이스바(' '), 엔터 키('\n'), 탭 키('\t')등을 모두 포함합니다.

나중을 위한 스포일러일 수도 있는데, 문자열은 & 안 붙일 수도 있습니다!!

(배열이거든요.)

C문법을 통한 기초 I/O



Visual Studio에서는 단순히 scanf로 적고 실행하면 오류가 뜰 거예요!
이유는 안전성 때문인데요, 우리는 이에 크게 신경 쓸 것 없이
설정 하나만 하고 넘어갑시다!

VS상단 메뉴부분에서, [도구] - [설정] 들어가고,
나오는 창에서 왼쪽 부분에, C/C++ - 전처리기 부분으로 들어가주세요,

Project8 속성 페이지

구성(C): 모든 구성 플랫폼(P): x86_64 (Win64) 구성 관리자(O)...

구성 속성

- 일반
- 고급
- 디버깅
- VC++ 디렉터리
- C/C++
 - 일반
 - 최적화
 - 전처리기
 - 코드 생성
 - 언어
 - 미리 컴파일된 헤더
 - 출력 파일
 - 찾아보기 정보
 - 고급
 - 모든 옵션
 - 명령줄
- 링커
- 매니페스트 도구
- XML 문서 생성기
- 찾아보기 정보
- 빌드 이벤트

전처리기 정의

전처리기 정의 해제

모든 전처리기 정의 해제

표준 포함 경로 무시

파일로 전처리

전처리 줄 번호 표시 안 함

주석 유지

_CRT_SECURE_NO_WARNINGS

전처리기 정의

소스 파일에 대한 전처리 기호를 정의합니다.

확인 취소 적용(A)

그리고, 왼쪽 사진처럼 모든 구성으로 한 뒤
전처리기 정의에

를 추가해 주시고 확인 눌러주세요!

그럼 다음으로 넘어가 보겠습니다.

C문법을 통한 기초 I/O

```
1  #include <stdio.h>
2  // 헤더파일
3
4  int main()
5  {
6      int x;
7      // 입력을 받을 변수를, 정수형으로 정해서 선언했다.
8
9      scanf("%d", &x);
10     // 입력이다!
11     printf("%d", x);
12     // 출력이다!
13
14     return 0;
15 }
```

`printf("%d", x)`

이 함수는 **표준 출력 함수**입니다.

"나 출력 할건데, 이 내용으로 해줘." 하고 전달만 해주면,
알아서 출력해 줍니다.

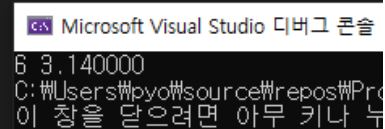
Python의 **print** 함수와 매우 유사해요. 대신 개행을 자동으로 하진 않습니다.
왼쪽을 보시면, 형식지정자를 통해 x라는 변수의 내용을 출력해주고 있습니다.

이때, scanf와는 다르게 **printf는 그냥 변수명으로 나열합니다.**

만약 형식을 다르게 한다면 오류가 난다거나 다른 값이 출력되겠죠?

C문법을 통한 기초 I/O

```
1  #include <stdio.h>
2  // 헤더파일
3
4  int main()
5  {
6      int x = 6;
7      double d = 3.14;
8
9      printf("%d %f", x, d);
10     // 출력이다!
11
12     return 0;
13 }
```



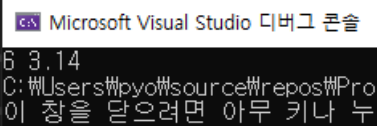
Microsoft Visual Studio 디버그 콘솔

6 3.140000
C:\Users\Wpyo\source\repos\Pro...
이 창을 닫으려면 아무 키나 누르세요

또한 printf는 **출력형식을 여러분이 정해줄 수도 있어요.**
기본적으로 실수형 변수는 %f를 통해서 출력하면,
소수점아래 7번째 자리에서 반올림합니다.
대신, 약간의 처리를 통해서 범위를 조정 할 수도 있죠.

C문법을 통한 기초 I/O

```
1  #include <stdio.h>
2  // 헤더파일
3
4  int main()
5  {
6      int x = 6;
7      double d = 3.14;
8
9      printf("%d %.2f", x, d);
10     // 출력이다!
11
12     return 0;
13 }
```



이렇게요.

%.2f의 .2는,

소수점 **2자리** **까지만 출력**하도록 지정 할 수 있게 해줍니다.

C문법을 통한 기초 I/O

```
1 #include <stdio.h>
2 // 헤더임동
3
4
5 int main()
6 {
7     int x;
8     // 변수라예
9
10    printf("Hi, everyone!\nMy name is PYO");
11    // 출력문이라능
12
13    return 0;
14 }
```

Microsoft Visual Studio

Hi, everyone!
My name is PYO
C:\Users\pyo\source\...
이 창을 닫으려면

한 가지 더!

우리는 다음줄로 넘어가는 내용을 적기 위해

개행('Wn')문자를 사용할 수 있습니다! (****부록참고)

다음줄로 넘겨야만 하는 부분이 있다면 꼭 써 줘야겠죠?

C문법을 통한 기초 I/O

```
1  #include <stdio.h>
2  // 헤더파일
3
4  int main()
5  {
6      int x;
7      // 입력을 받을 변수를, 정수형으로 정해서 선언했다.
8
9      scanf("%d", &x);
10     // 입력이다!
11     printf("%d", x);
12     // 출력이다!
13
14     return 0;
15 }
```

전체 코드를 한번 실행 해 볼까요?

Visual Studio에서,

저장 : Ctrl + S

되돌리기 : Ctrl + Z

디버깅 하지 않고 실행 : Ctrl + F5

한줄 한줄 디버깅 : F11

입니다!

먼저 저장을 해주고, Ctrl + F5로 실행결과를 보도록 해요.

(잘 되죠?)

C문법을 통한 기초 I/O

이렇게 **I/O**는 기본적인 것들만 다 다루어 보았어요.

물론 이게 전부는 아닙니다. 변수, 형식 및 함수와 연산자들을 더더욱 알아가다 보면,

다양한 **입출력방식**이 존재한다는 것을 알게 돼요!

더 재밌는 내용이 이제 **다음 주차**에 나올 테니, 기대해보도록 합시다.

끝

(수고했음)

뒤에 부록 있습니다!

C문법을 통한 기초 I/O (부록)

* 함수들을 사용하려면, 미리 정의 해 놓은 마치 백과사전과 같은 존재,
헤더를 불러와야 하는데 우리는 입력과 출력에 관한 함수가 필요하므로
STanDard Input & Output을 담은 stdio헤더를 불러옵니다.
#include <헤더명.h>라고 적으면,
“시작하기전에, 특정 헤더에 있는 함수들을 가져와줘” 라고 해주는 것입니다.

C문법을 통한 기초 I/O (부록)

```
scanf("%d", &x);
```

****잠깐! &를 붙이면 변수의 주소가 된다구요? 변수의 주소요? 무슨 말인지 잘 모르겠어요...**

사실, 모든 변수는 생성될 때에, 값을 담을 메모리를 부여 받는답니다.

그리고 그 메모리에 값을 저장하는 것이죠. 가령, 0101과같은 이진수 형태로 말이죠.

그리고 각 변수마다 “내가 어디 부분에 메모리를 갖고있어!”라는 정보를 가져요. 주소이죠.

우리 **int형식의 변수**는 4-byte의 크기를 갖는다고 알고있죠!

그럼 int x를 선언함과 동시에, x라는 변수는 **4-byte의 크기의 메모리**를 갖게 되는 것입니다.

입력을 해주는 함수에서는, **입력 받은 값을 바로 해당 주소에 써 주기 때문에**,

뒤에 변수명을 제공하지 않고, 그 **변수의 주소 값을** 제공하는 것 이랍니다.

C문법을 통한 기초 I/O (부록)

*** 형식에 따라, 메모리 저장 방식이 다르기 때문입니다.

**** 실제로, 'wn' 이외에도 다양한 **w가 붙은 문자**가 많습니다.

이를 **이스케이프 문자**라고 하는데요, 종류는 아래와 같습니다

wn	→	New line. 새로운 줄로 이동.
wb	→	back space. 커서 한 칸 왼쪽 이동하고 그 위치 문자 삭제.
wt	→	tap. 탭의 간격만큼 이동.
w"	→	큰 따옴표를 출력해줌. (ex. "w"문자w")
w'	→	작은 따옴표를 출력해줌. (ex. 'w'문자w')

실제로 여러분들은 wn이외에는 거의 사용하지할 일이 없을 거예요. 알아 두기만 합시다.