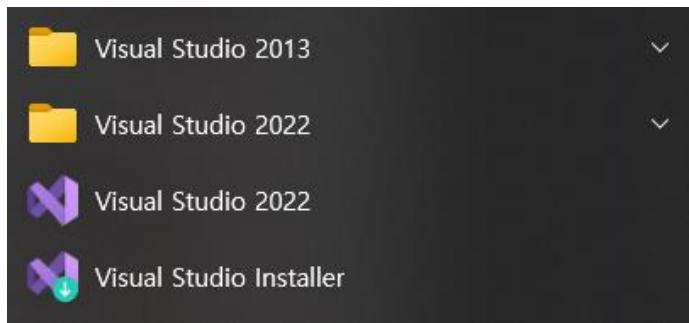


C Programming

#1/4 ver 0.1

Yongseok Chi



<https://visualstudio.microsoft.com/ko/vs/older-downloads/>

2013

en_visual_studio_ultimate_2013_x86_dvd_3175319.iso

Visual Studio 2013
및 기타 제품

다음 목록의 제품을 다운로드하려면 다운로드 단추를 클릭하고 메시지가 표시되면 Visual Studio 구독 계정으로 로그인하세요. Visual Studio 구독이 없는 경우 로그인 페이지에서 "새 Microsoft 계정 만들기"를 클릭하여 무료로 계정을 하나 만들 수 있습니다.

Visual Studio Professional 2013; Visual Studio Premium 2013; Visual Studio Ultimate 2013

Visual Studio Test Professional 2013

Visual Studio 2013 언어 팩

Visual Studio Test Professional 2013 언어 팩

Visual Studio Team Foundation Server 2013

다운로드

강의 자료 visual studio version 2013

2013 vs. 2022

: class 추가의 dialog 생성 방법 차이 => Dialog , Dialog id

: 120page 참고

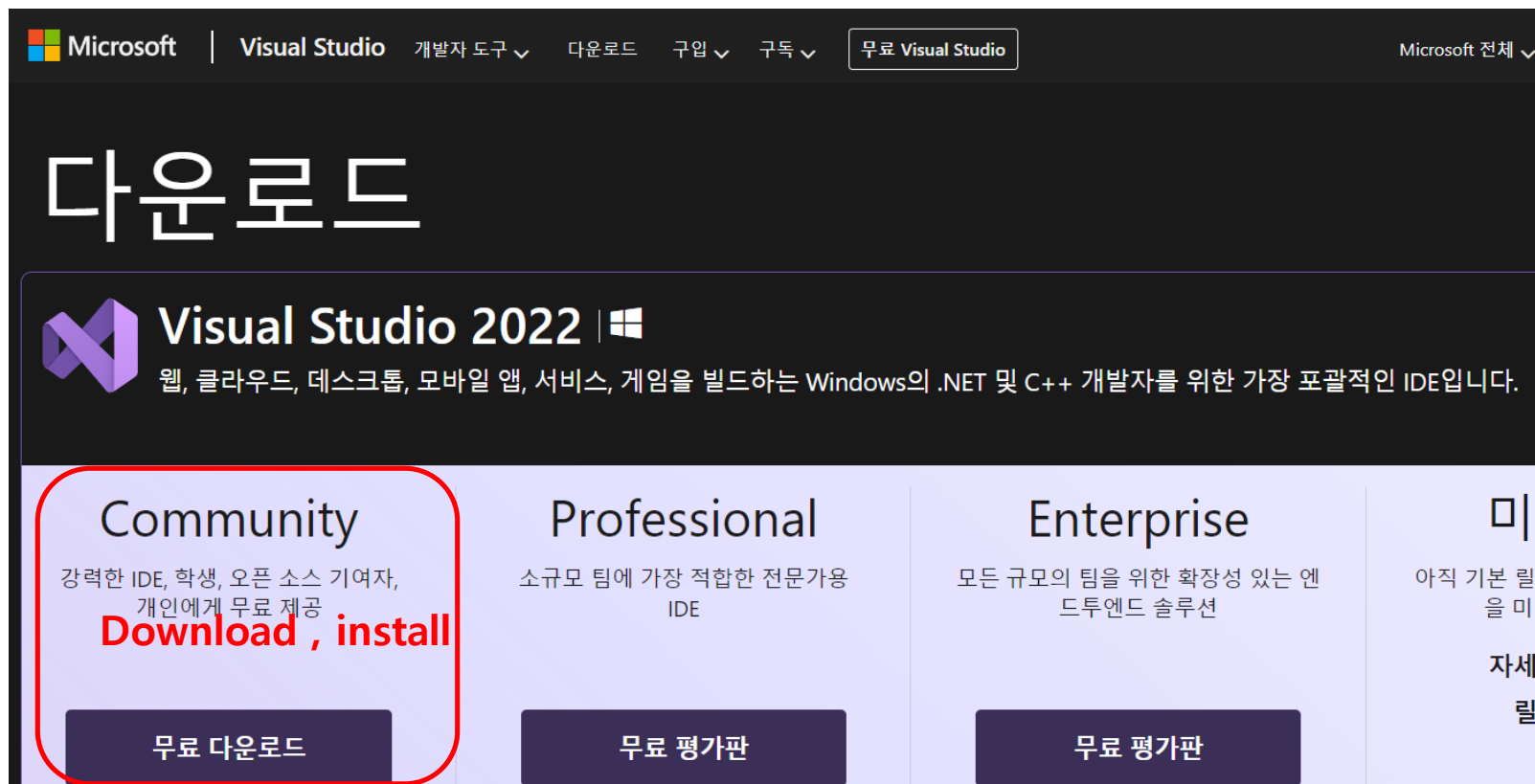
수업에서 2013 version 사용

Visual Studio Express 2013 for Windows Desktop

Reference

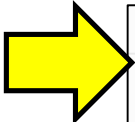
1. Reference

- (1) Preprocessor, msdn.microsoft.com/en-us/library/d9x1s805.aspx
- (2) 비주얼 스튜디오 <https://visualstudio.microsoft.com/ko/downloads/>
- (3) 강의자료 **Visual Studio 2013** version
- (4) Editor : Notepad++ 설치 <https://notepad-plus-plus.org/downloads/>
- (5) code 비교 : beyond compare <https://www.scootersoftware.com/>
- (6) project 분석 : source insight <https://www.sourceinsight.com/>





Reference

: 설치시 C++선택





워크로드 개별 구성 요소 언어 팩 설치 위치


 **Python 개발** ☐
Python에 대한 편집, 디버깅, 대화형 개발 및 소스 제어입니다.


 **Node.js 개발** ☐
비동기 이벤트 구동 JavaScript 런타임인 Node.js를 사용하여 확장 가능한 네트워크 애플리케이션을 빌드합니다.


데스크톱 및 모바일 (5)

 **.NET Multi-Platform App UI 개발** ☐
.NET MAUI와 함께 C#을 사용하여 단일 코드베이스에서 Android, iOS, Windows 및 Mac용 앱을 빌드합니다.

 **C++를 사용한 데스크톱 개발** ☒
MSVC, Clang, CMake 또는 MSBuild 등 선택한 도구를 사용하여 Windows용 최신 C++ 앱을 빌드합니다. **CHECK**

 **C++를 사용한 모바일 개발** ☐
C++를 사용하여 iOS, Android 또는 Windows용 플랫폼 간

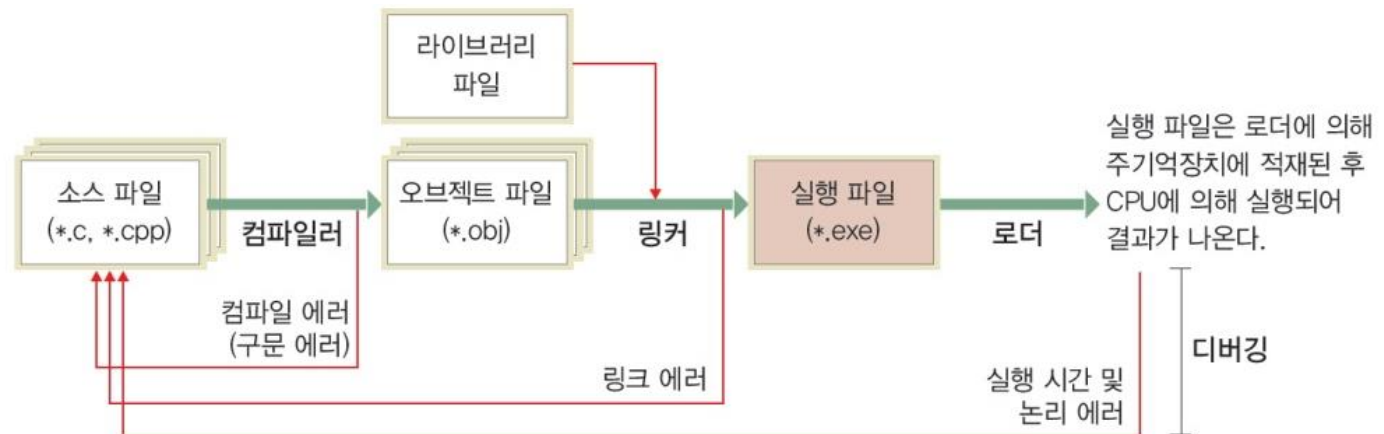
 **.NET 데스크톱 개발** ☒
.NET 및 .NET Framework와 함께 C#, Visual Basic 및 F#을 사용하여 WPF, Windows Forms 및 콘솔 애플리케이션을...

 **유니버설 Windows 플랫폼 개발** ☐
C#, VB 또는 C++(선택 사항)를 사용하여 유니버설 Windows 플랫폼용 애플리케이션을 만듭니다.

위치

1. 프로그램 개발이란?

- **프로그래밍 언어** : 컴퓨터(PC, Phone, CPU) 프로그램을 작성하기 위해 개발된 언어
- **기계어** : 0과 1의 2진 체계를 사용하는, 컴퓨터가 직접 이해할 수 있는 언어
- **Assembler language** : CPU 개발때 명령어로 구성된 언어(CPU Architecture-instruction)
- **고급(high-level) 언어** : 사람에게 가깝게 명령을 내릴 수 있어서 편리한 언어, C-language
 '나이가 20살 미만이라면' → 'if (age < 20)'로 표현
- **algorithm(알고리즘)** : 주어진 문제를 풀기 위한 방법. 논리 & 창의
- **flow chart(순서도)** : 미리 약속된 도형과 화살표를 이용하여 일의 흐름을 한눈에 볼 수 있도록 한 것
- **coding(코딩)** : 알고리즘을 특정 프로그래밍 언어로 옮기는 작업
 코딩 과정을 마친 프로그램 - source code(소스 코드)
- **compiling(컴파일)** : 고급 언어로 작성한 프로그램을 기계어로 번역하는 작업
- **linking(링킹)** : 여러 프로그램 파일을 하나로 합치고, 라이브러리를 프로그램 안에 포함시키는 과정

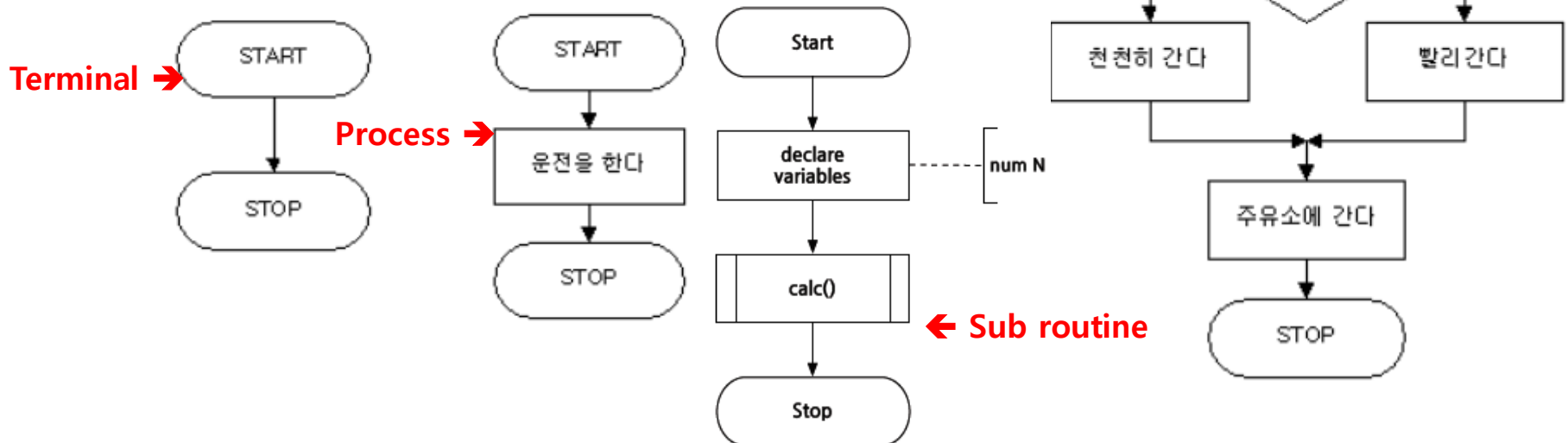


1. 프로그램 개발이란?

- **executing(실행)** : 링킹 과정을 마친 *.exe 파일을 CPU에 저장하고, CPU에 의해 실행됨
- **debugging(디버깅)** : 프로그램이 제대로 실행되는 것을 방해하는 버그(bug)를 찾아 제거하는 작업

• Flowchart

- ㉠ 단말기호(Terminal) : 프로세스의 시작과 끝
- ㉡ 처리기호(Process) : 모든 처리를 표시
- ㉢ 판단기호(Decision) : 논리를 판단하여 참(True 또는 Yes)과 거짓(False 또는 No)을 분기하여 표시
- ㉤ 서브루틴 : 함수, 정의된 보조 프로그램



1. 프로그램 개발이란?

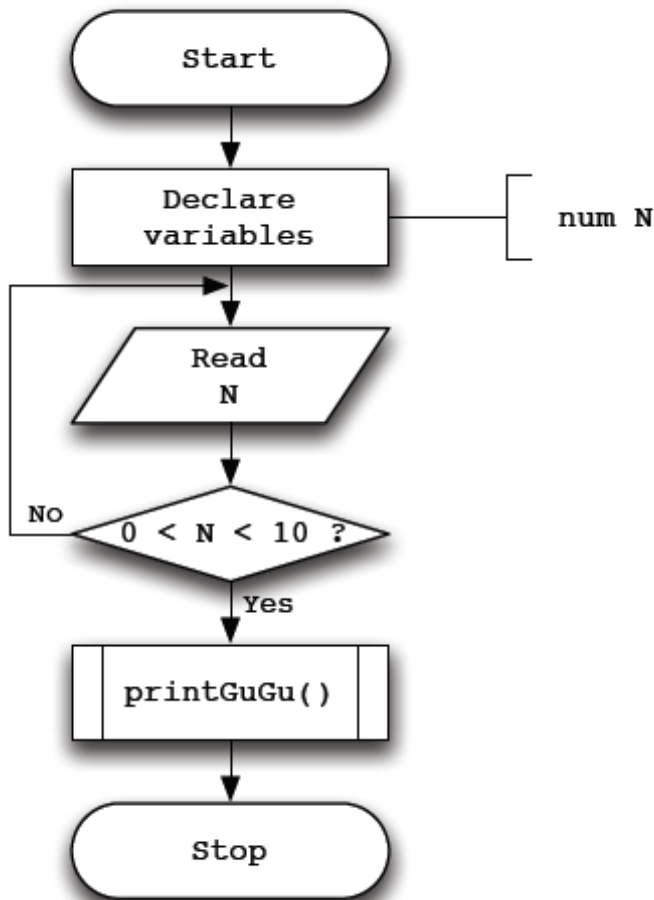
- Flowchart 실습

1-9까지의 정수 n 을 입력 받아,

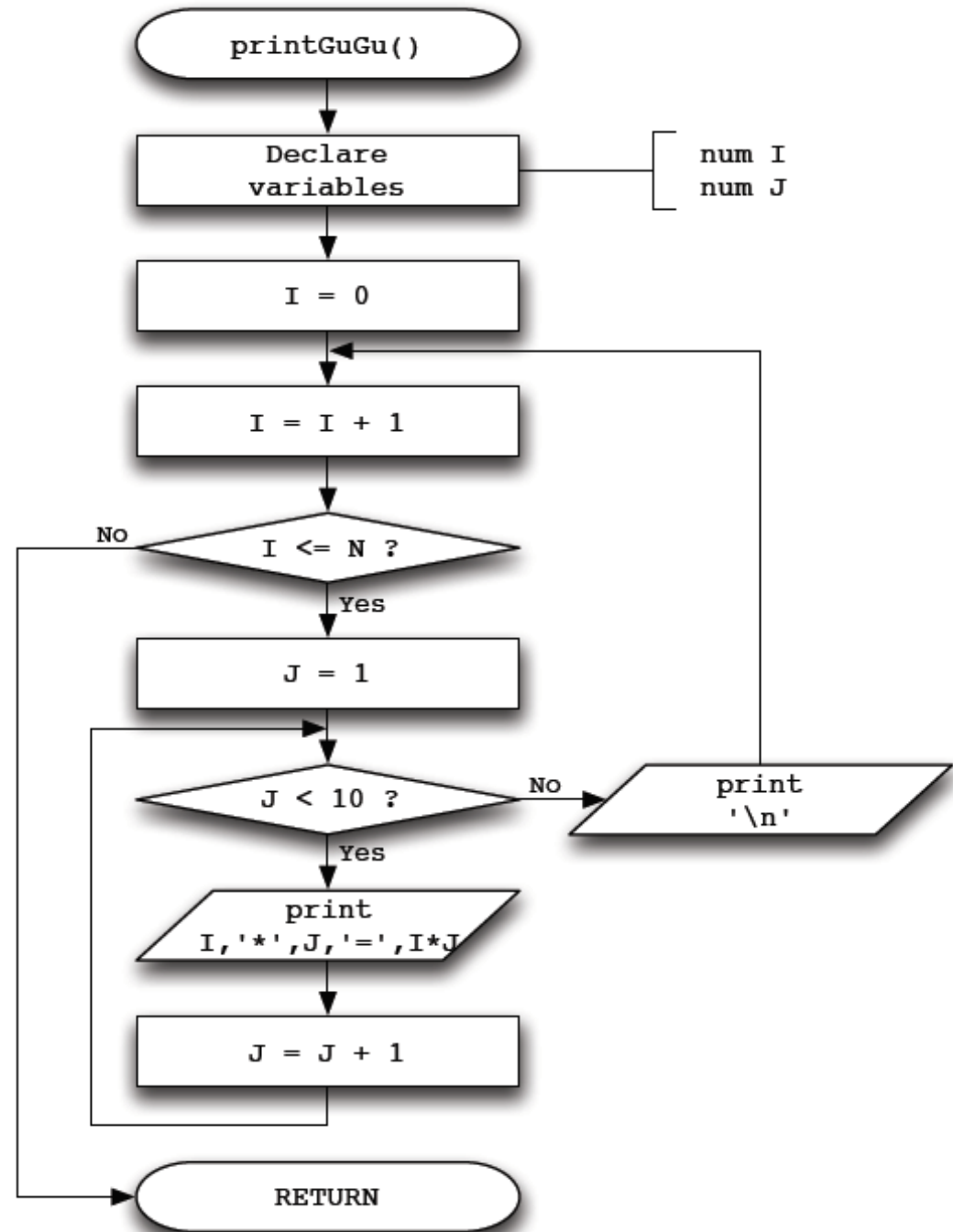
1단부터 n 단까지 구구단을 출력하는 프로그램의 순서도를 만들기

1-9까지의 정수 n을 입력 받아,

1단부터 n단까지 구구단을 출력

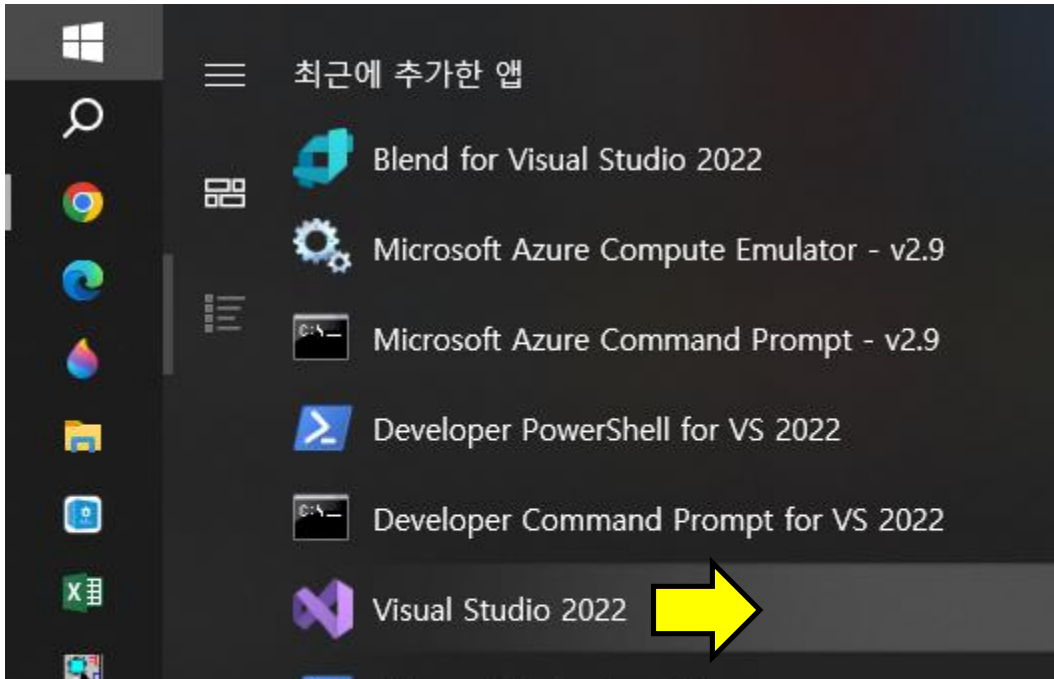


1단부터 n단까지 구구단을 출력



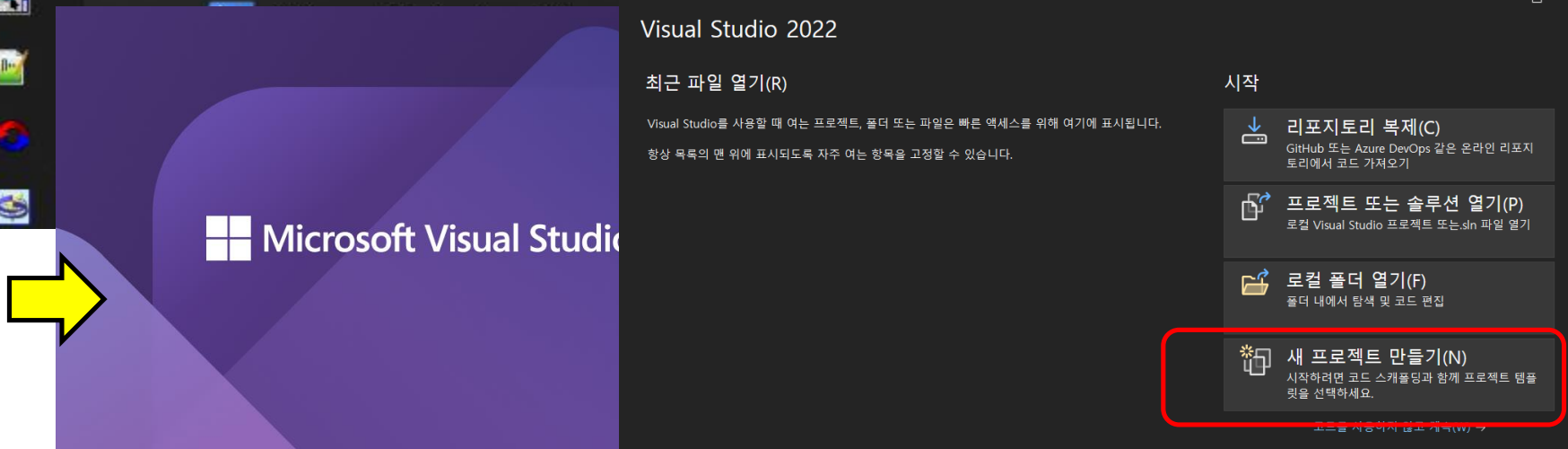
2. Visual studio 실행하기 Visual Studio Community 2022

2.1 Visual Studio 실행화면



Microsoft ID 로그인
: 없다면 추후에 (1달 정도 사용)

Microsoft Visual 실행 화면



2. Visual studio 실행하기 Visual Studio Community 2022

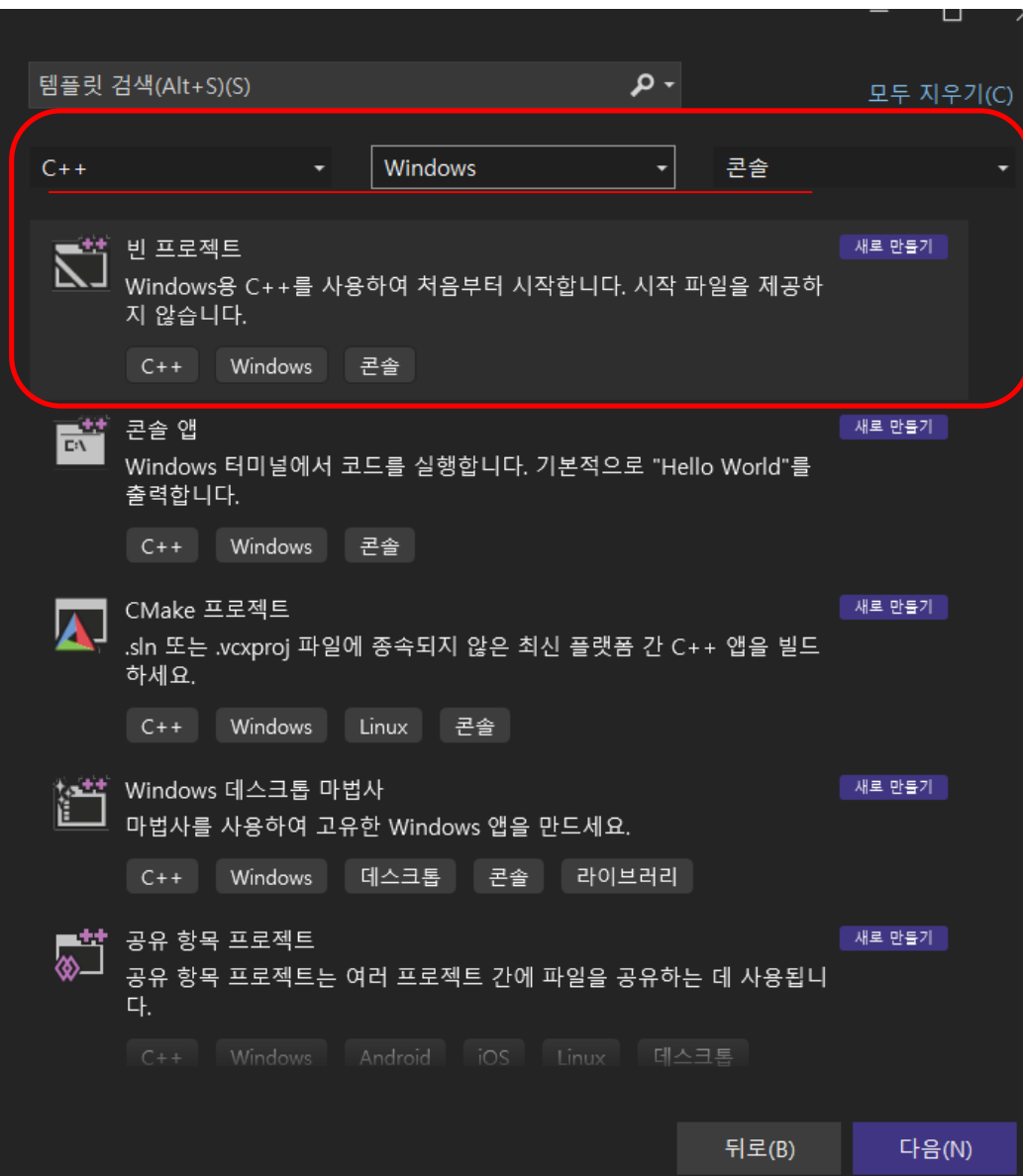
2.1 Visual Studio 실행화면

새 프로젝트 만들기

최근 프로젝트 템플릿(R)

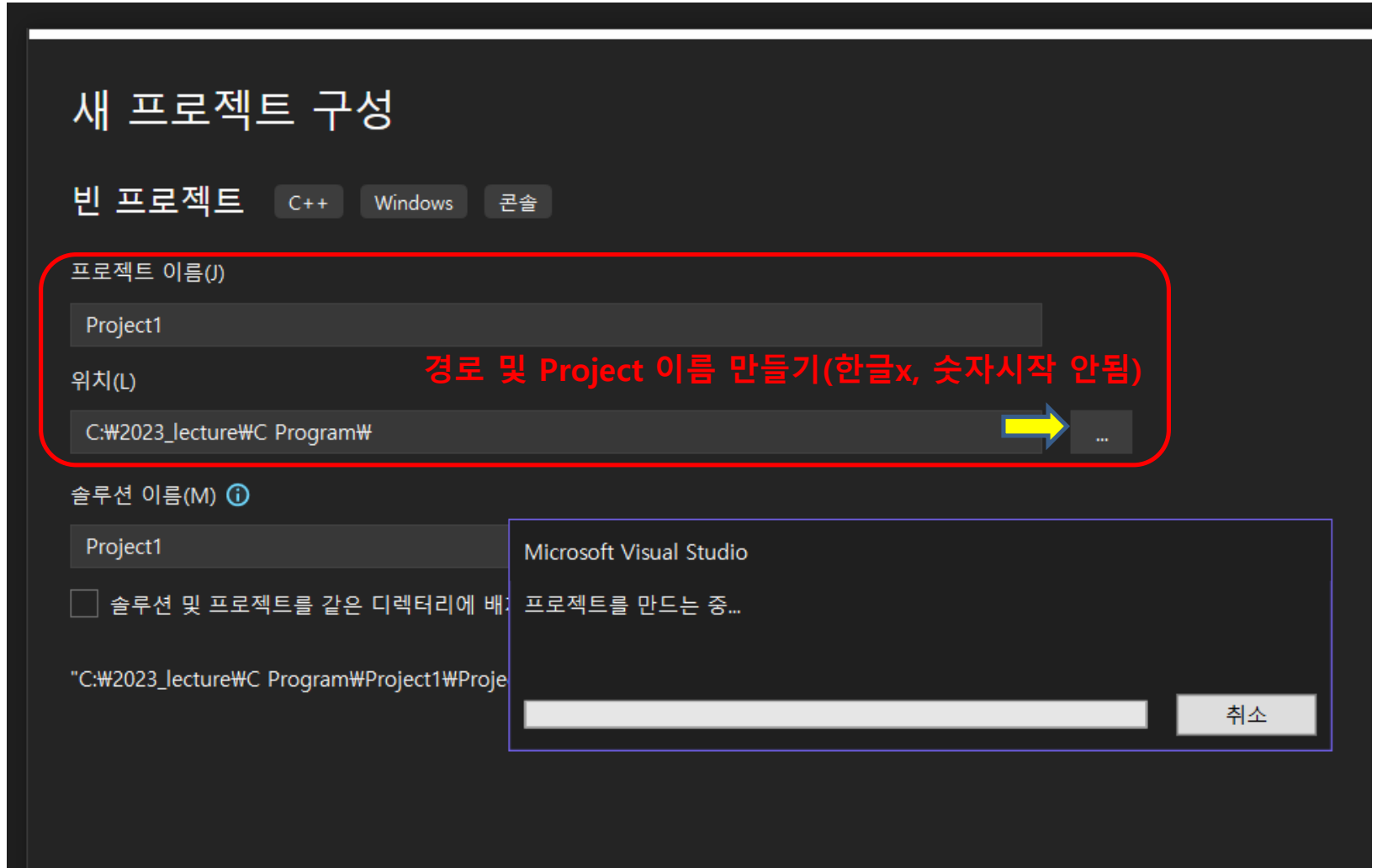
최근에 액세스한 템플릿 목록이 여기에 표시됩니다.

선택



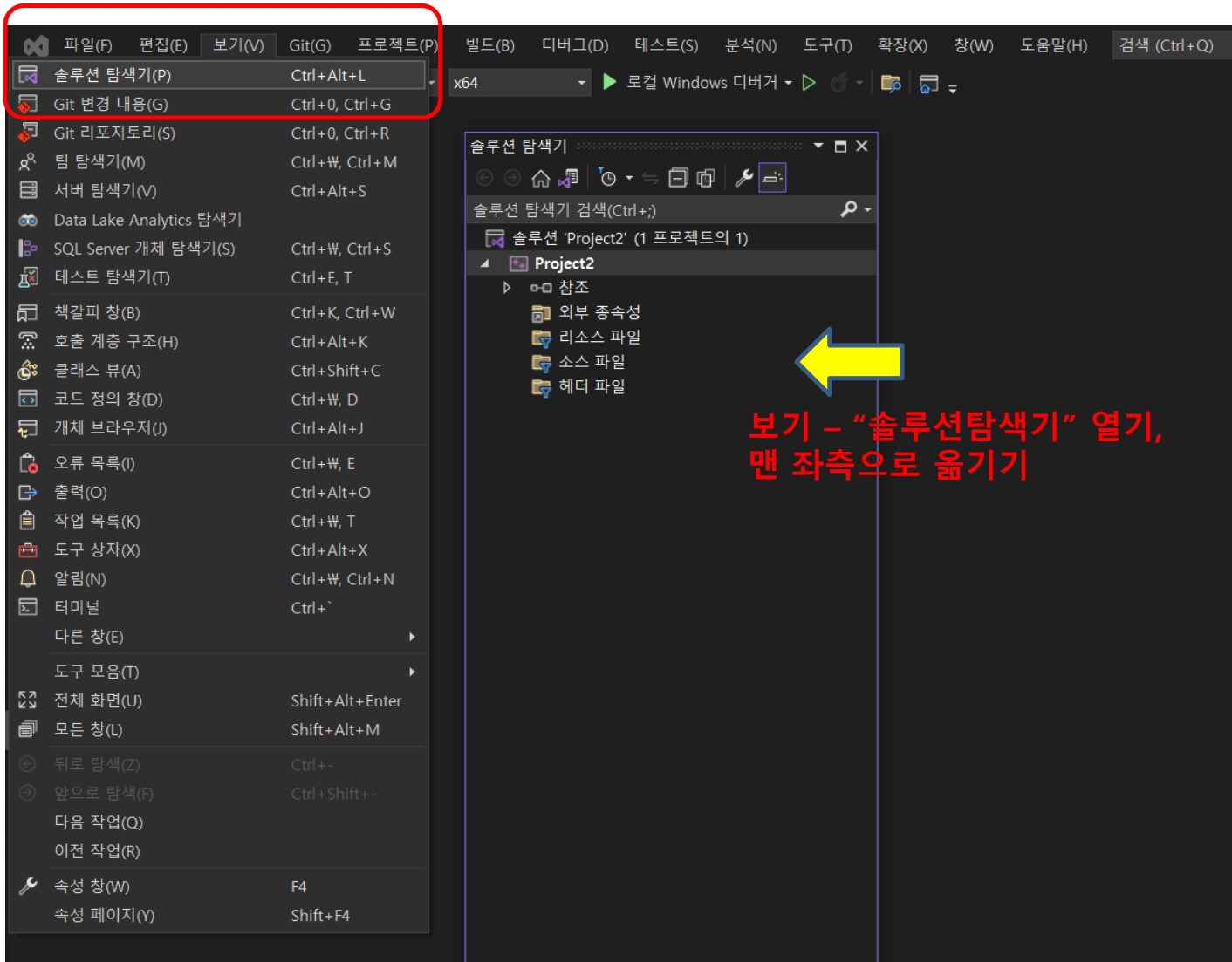
2. Visual studio 실행하기 Visual Studio Community 2022

2.1 Visual Studio 실행화면



2. Visual studio 실행하기 Visual Studio Community 2022

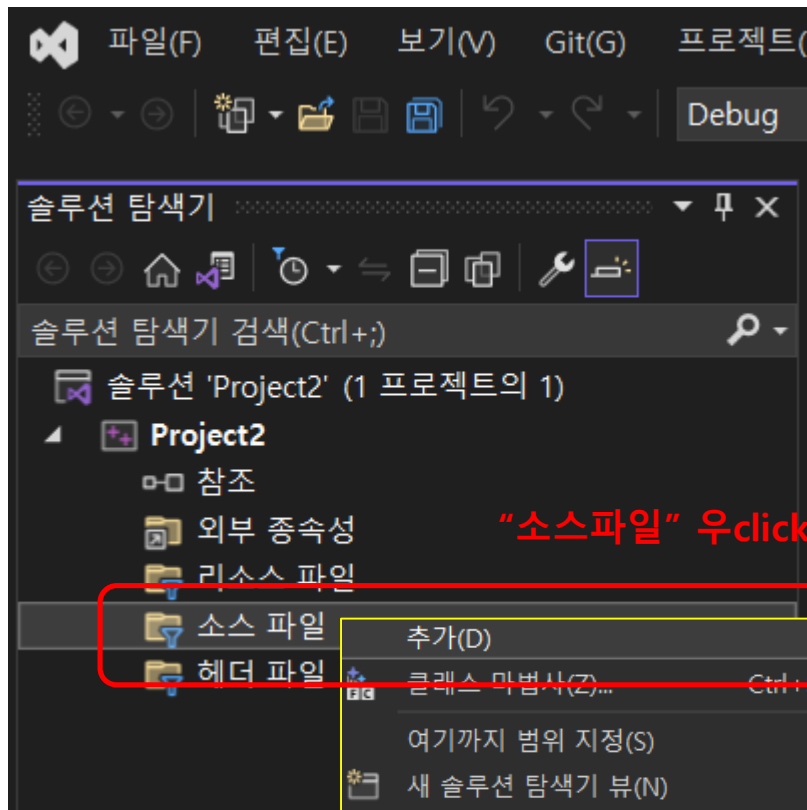
2.1 Visual Studio 실행화면



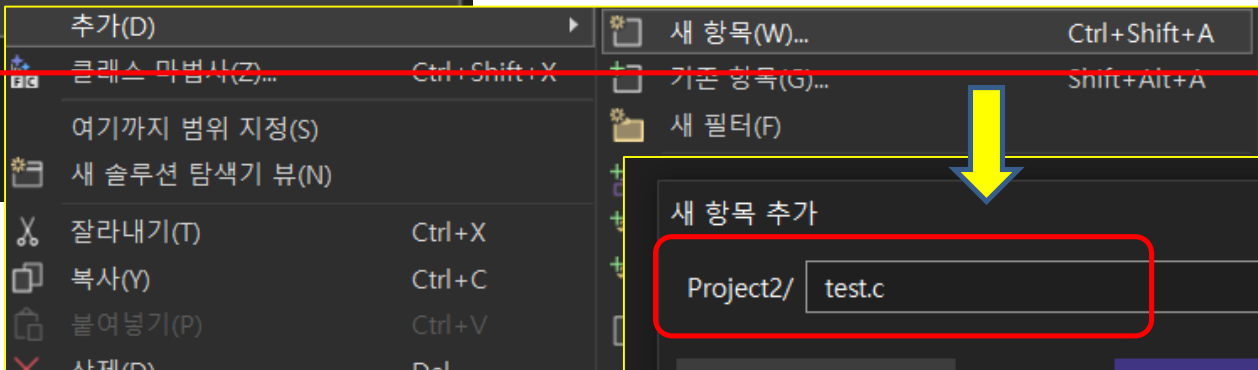
보기 - "솔루션탐색기" 열기,
맨 좌측으로 옮기기

2. Visual studio 실행하기 Visual Studio Community 2022

2.1 Visual Studio 실행화면



“소스파일” 우click - 추가 - 새항목



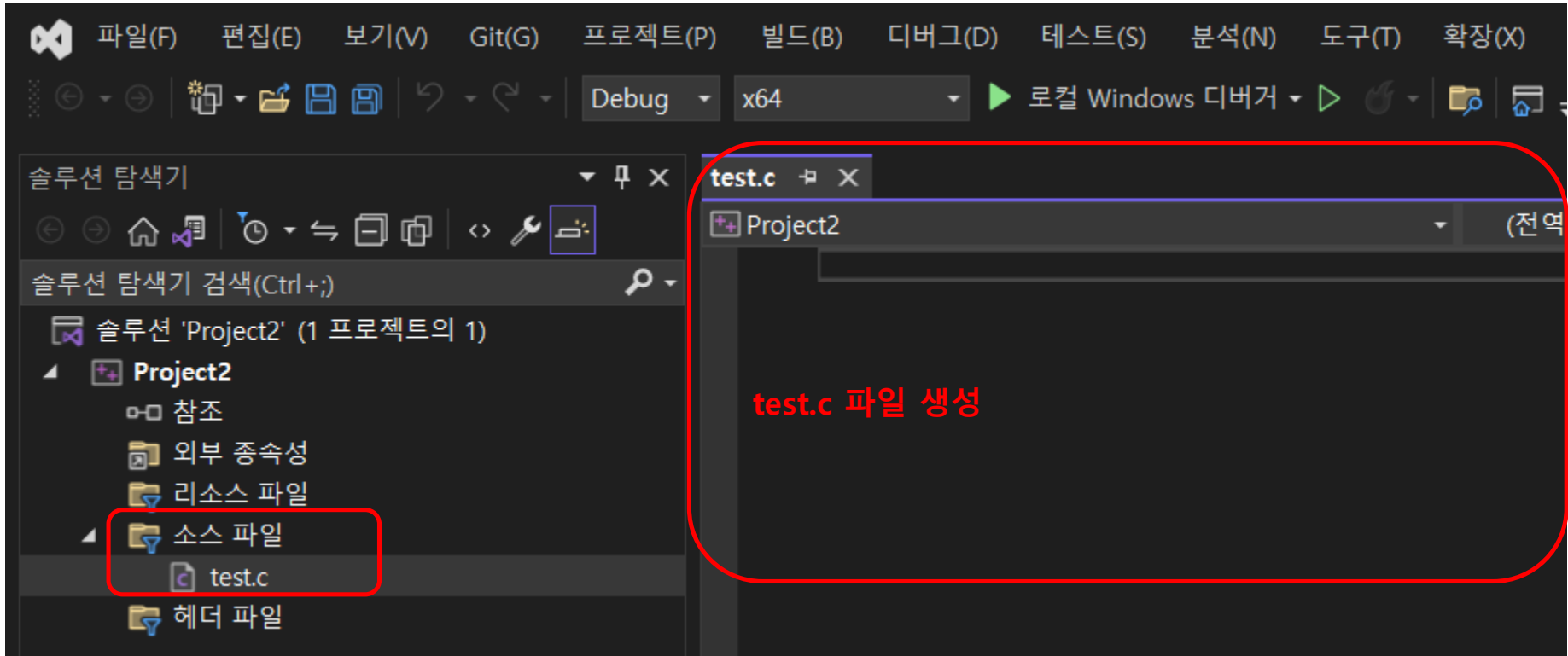
확장자가 **“.cpp”**면, C++ 언어에 맞게 컴파일 됨

C언어에서만 제공하는 기능을 사용하려면 반드시 확장자를 **.c**로 직접 입력

C를 사용하려면, 파일명을 test.c로 해야 C언어에 맞게 컴파일 됨.

2. Visual studio 실행하기 Visual Studio Community 2022

2.1 Visual Studio 실행화면



2. Visual studio 실행하기 Visual Studio Community 2022

2.1 Visual Studio 실행화면 (test.c 파일의 위치 확인하는 방법)

우 click

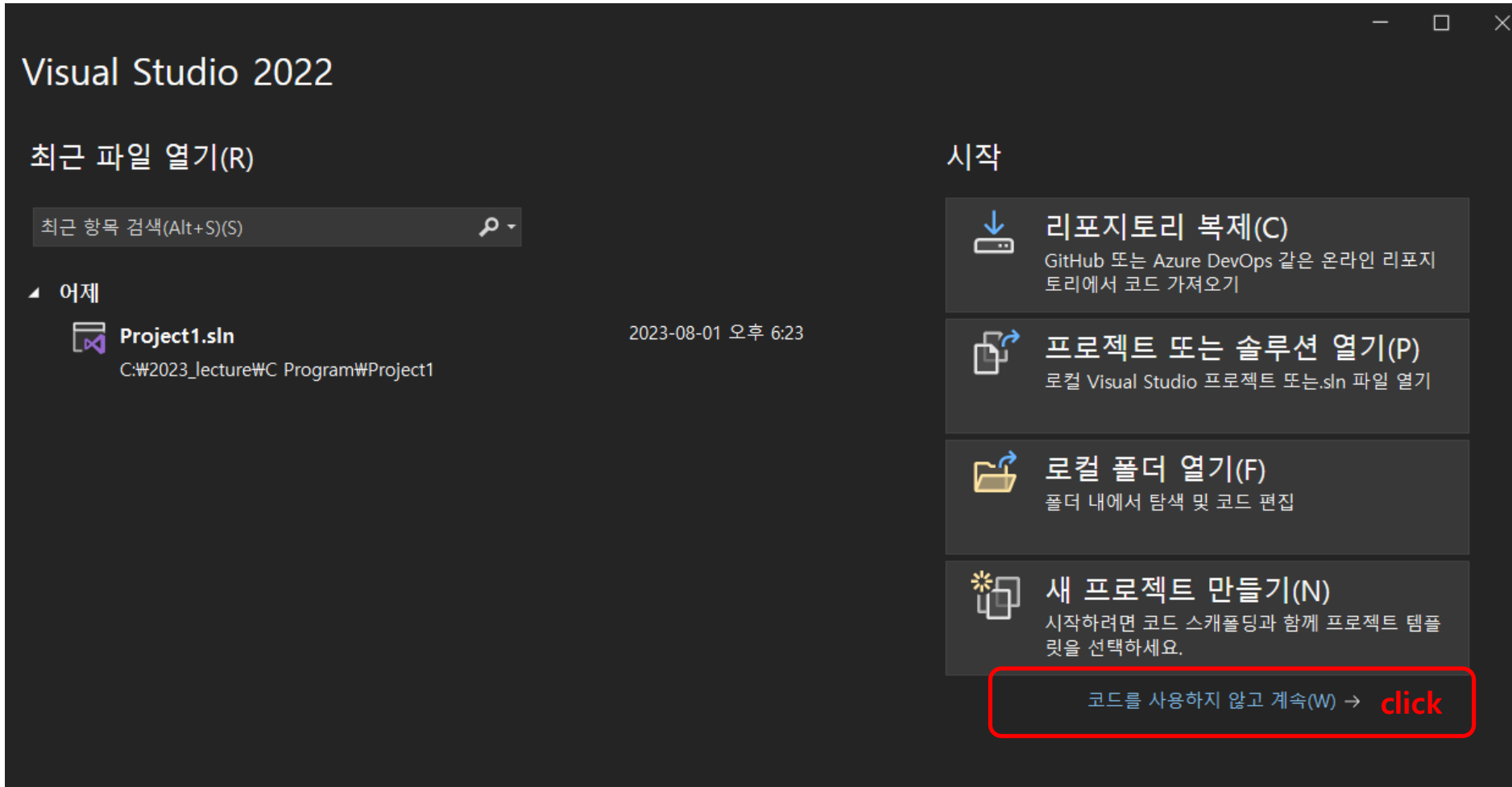
click

이름	수정한 날짜	유형
Project2.vcxproj	2023-08-02 오전 9:49	VC++ Project
Project2.vcxproj.filters	2023-08-02 오전 9:49	VC++ Project Fil...
Project2.vcxproj.user	2023-08-02 오전 9:49	Per-User Project ...
test.c	2023-08-02 오전 9:56	C Source

생성된 파일이 windows 탐색기에서 확인

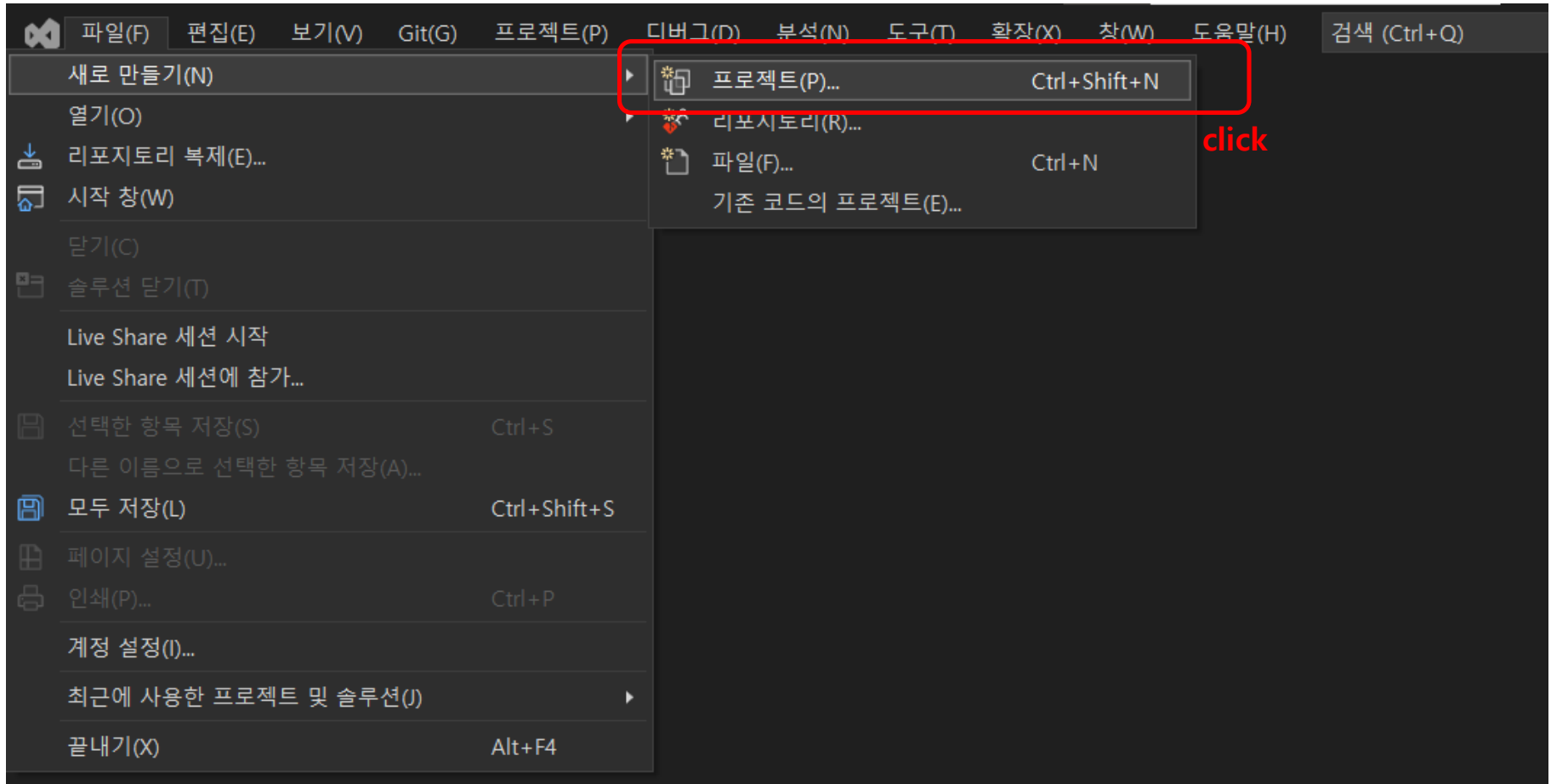
2. Visual studio 실행하기 Visual Studio Community 2022

2.1 프로젝트 만들기(방법2)



2. Visual studio 실행하기 Visual Studio Community 2022

2.1 프로젝트 만들기(방법2)



2. Visual studio 실행하기 Visual Studio 2013

2.2 Visual Studio 2013 실행

시작 페이지 - Microsoft Visual Studio (관리자)

파일(F) 편집(E) 보기(V) 디버그(D) 팀(M) 도구(T) 테스트(S) 분석(N) 창(W) 도움말(H)

새로 만들기(N) ▶ 프로젝트(P)... Ctrl+Shift+N

열기(O) **파일 → 새로 만들기 → 프로젝트** 웹 사이트(W)... Shift+Alt+N

닫기(C)

솔루션 닫기(T)

선택한 항목 저장(S)

새 프로젝트

최근 항목

설치됨

템플릿

- Visual Basic
- Visual C#
- Visual C++**
 - ATL
 - CLR
 - 일반
 - MFC
 - 테스트
 - Win32
- Visual F#
- SQL Server
- TypeScript
- Python
- 기타 프로젝트 형식
- 샘플

온라인

이름(N): ConsoleApplication3

위치(L): C:\Users\bradley\Documents\Visual Studio 2013\Projects

솔루션 이름(M): ConsoleApplication3

형식: Visual C++

Win32 콘솔 응용 프로그램을 만드는 프로젝트입니다.

여기서 Win32 Console Application 선택

온라인으로 전환하거나 템플릿을 찾으려면 여기를 클릭하십시오.

찾아보기(B)...

☒ 솔루션 디렉터리 만들기(D)

☐ 소스 제어에 추가(U)

확인 취소

2. Visual studio 실행하기 Visual Studio 2013

2.2 Visual Studio 2013 실행

시작 페이지 - Microsoft Visual Studio (관리자)

파일(F) 편집(E) 보기(V) 디버그(D) 팀(M) 도구(T) 테스트(S) 분석(N) 창(W) 도움말(H)

빠른 실행(Ctrl+Q)

새 프로젝트

최근 항목

설치됨

템플릿

Visual Basic

Visual C#

Visual C++

ATL

CLR

일반

MFC

테스트

Win32

Visual F#

SQL Server

TypeScript

Python

기타 프로젝트 형식

샘플

온라인

.NET Framework 4.5

정렬 기준: 기본값

설치된 템플릿 검색(Ctrl+E)

Win32 콘솔 응용 프로그램

MFC 응용 프로그램

Win32 프로젝트

빈 프로젝트

메이크파일 프로젝트

Visual C++

Visual C++

Visual C++

Visual C++

Visual C++

온라인으로 전환하거나 템플릿을 찾으려면 여기를 클릭하십시오.

이름(N): ConsoleApplication3

위치(L): C:\Users\Wbradley\Documents\Visual Studio 2013\Projects

솔루션 이름(M): ConsoleApplication3

File명 만들기 (한글 X, 숫자 시작 안됨)

개인 위치(한글 x, 숫자시작안됨)

☒ 솔루션 디렉토리 만들기(D)

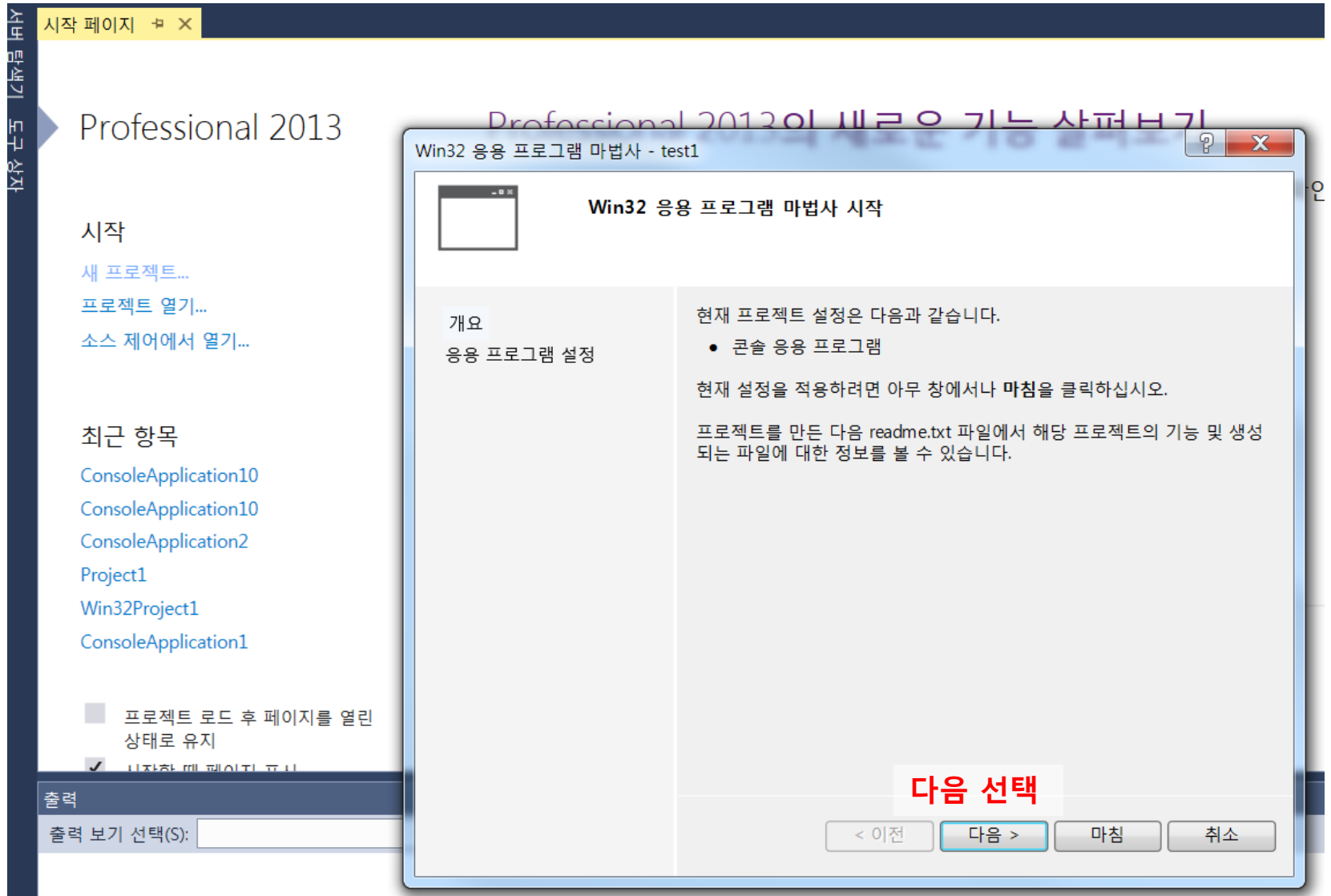
☐ 소스 제어에 추가(U)

확인

취소

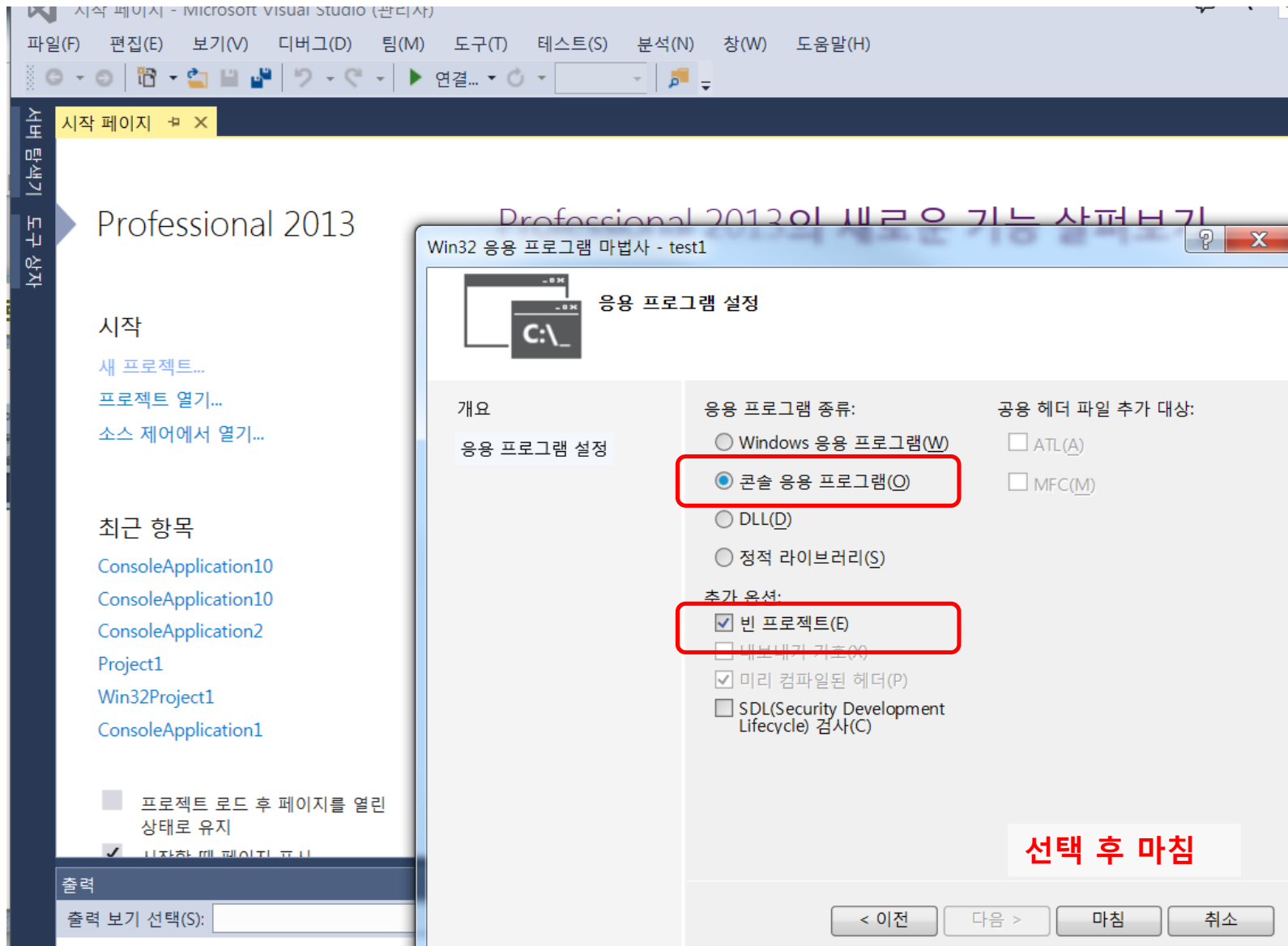
2. Visual studio 실행하기 Visual Studio 2013

2.2 Visual Studio 2013 실행



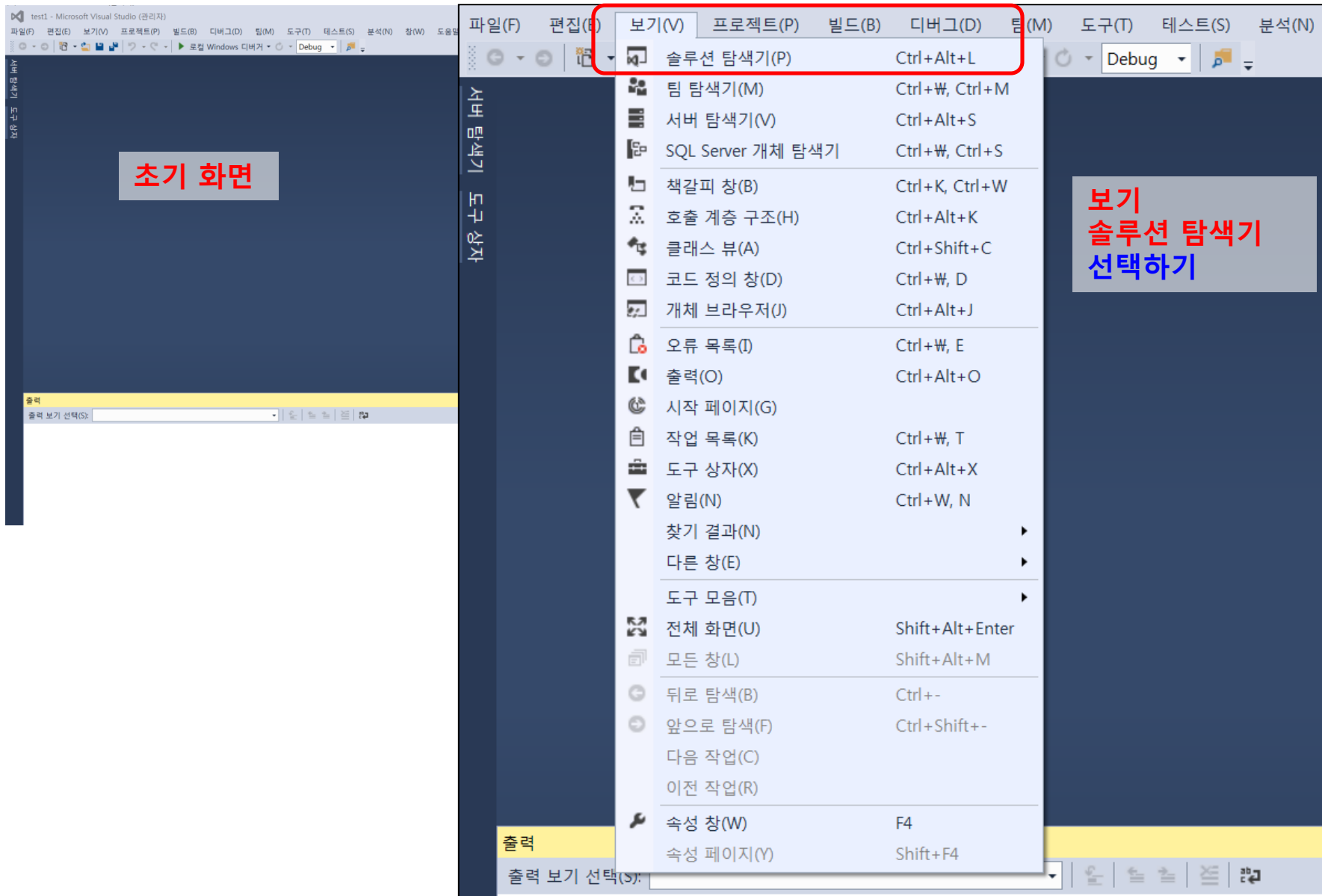
2. Visual studio 실행하기 Visual Studio 2013

2.2 Visual Studio 2013 실행



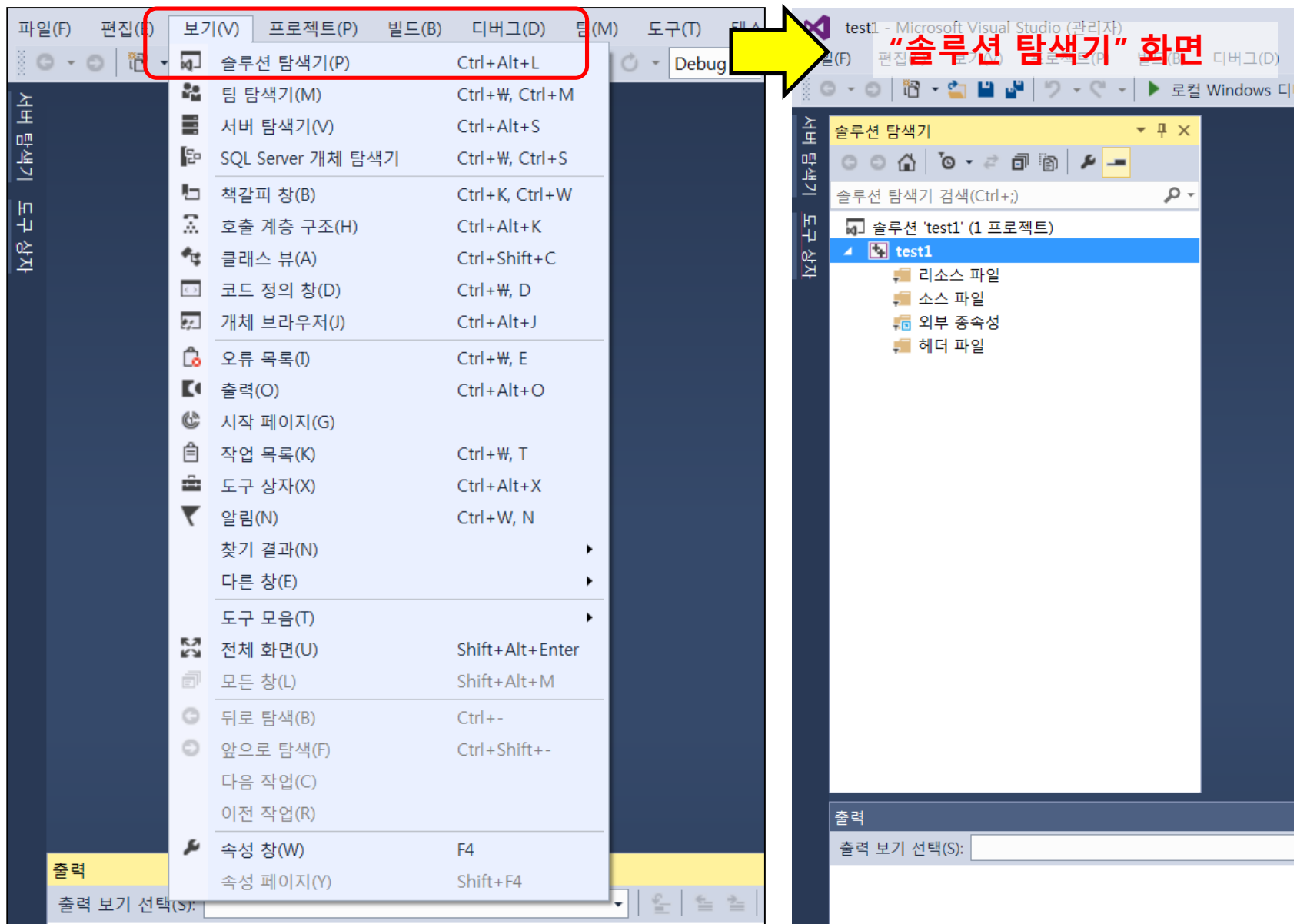
2. Visual studio 실행하기 Visual Studio 2013

2.2 Visual Studio 2013 실행



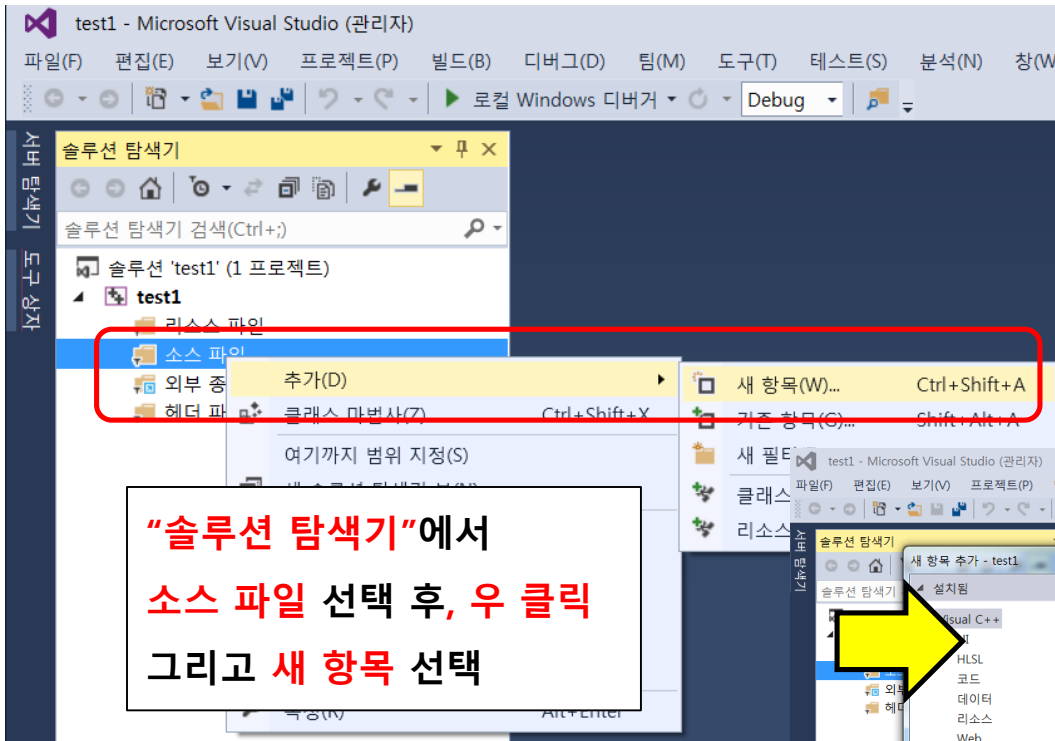
2. Visual studio 실행하기 Visual Studio 2013

2.2 Visual Studio 2013 실행



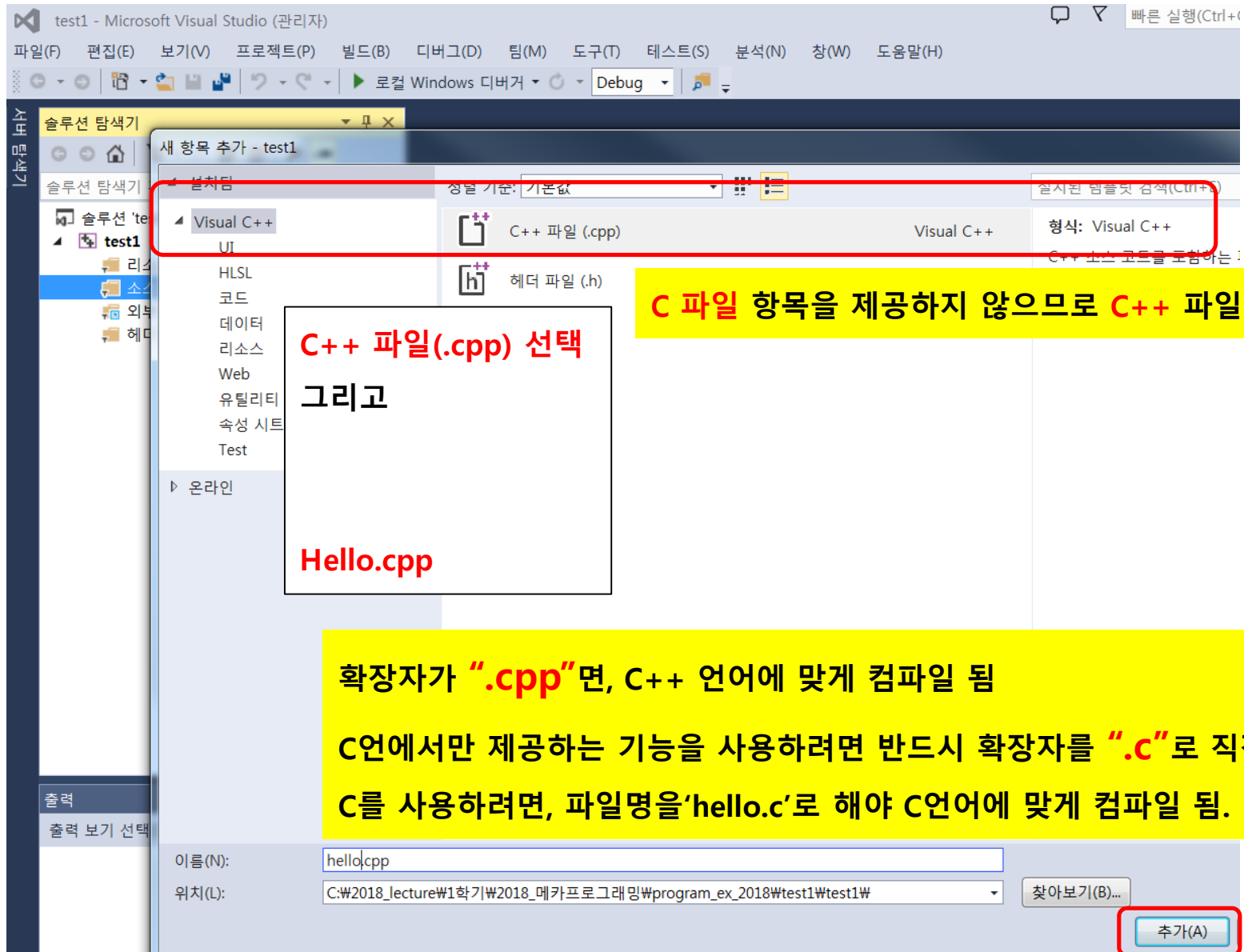
2. Visual studio 실행하기 Visual Studio 2013

2.2 Visual Studio 2013 실행



2. Visual studio 실행하기 Visual Studio 2013

2.2 Visual Studio 2013 실행



2. Visual studio 실행하기

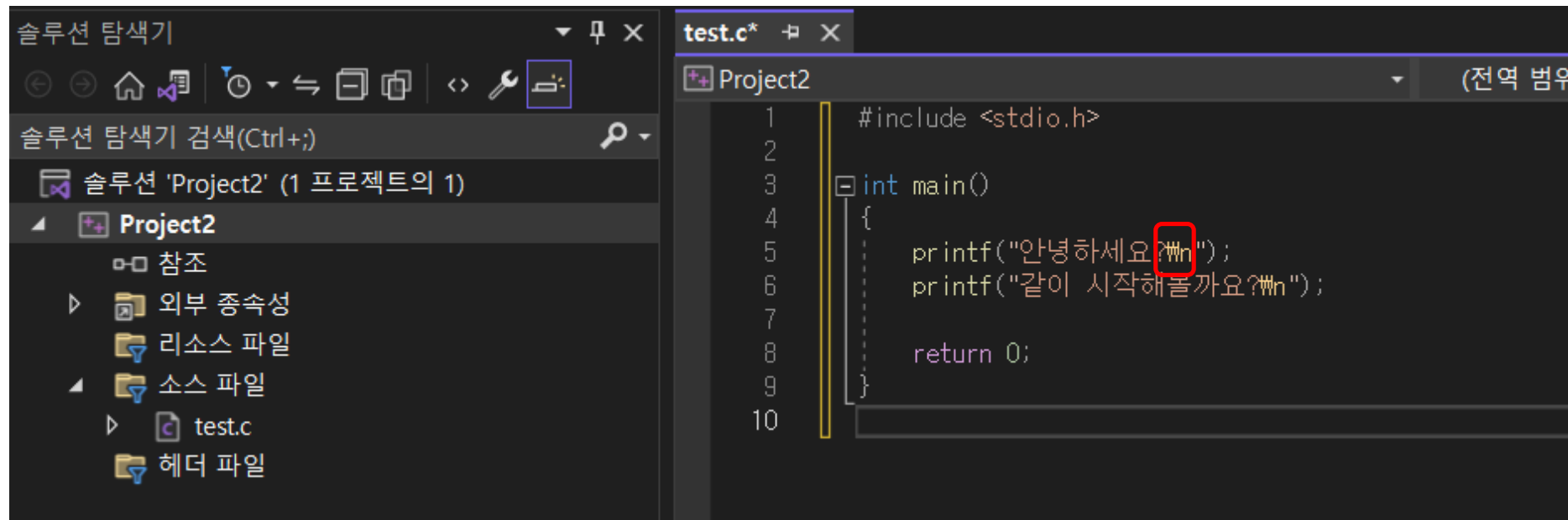
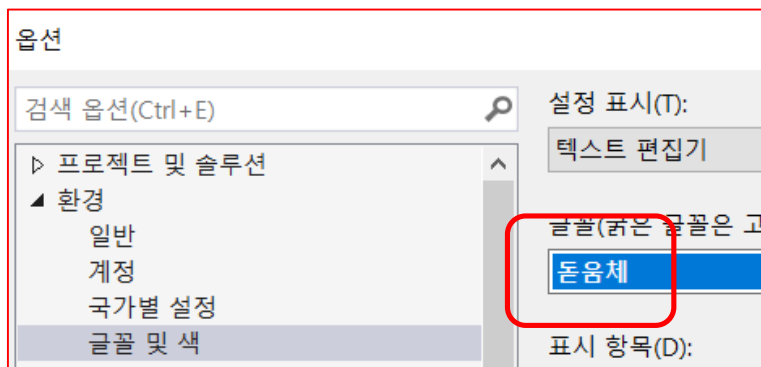
2.3 프로젝트에 Source 파일 추가하기 (아래 입력)

```
#include <stdio.h>

int main(void)
{
    printf("안녕하세요?\n");
    printf("같이 시작해볼까요?\n");

    return 0;
}
```

: 도구 - 옵션 - 환경 - 글꼴 및 색



2. Visual studio 실행하기

2.3 프로젝트에 Source 파일 추가하기(아래 입력)

```
#include <stdio.h>

int main(void)
{
    printf("안녕하세요?\\n");
    printf("같이 시작해볼까요?\\n");

    return 0;
}
```

: 도구 - 옵션 - 환경 - 글꼴 및 색

옵션

검색 옵션(Ctrl+E)

설정 표시(T):

텍스트 편집기

▶ 프로젝트 및 솔루션

▲ 환경

일반

계정

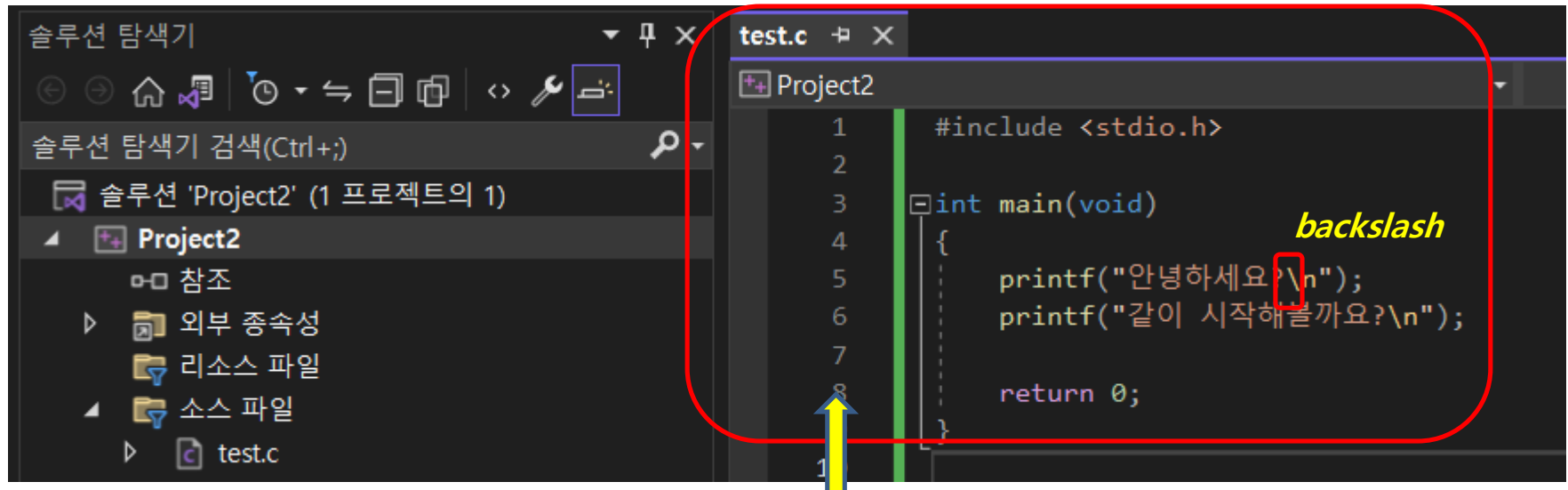
국가별 설정

글꼴 및 색

글꼴(굵은 글꼴)

Consoles

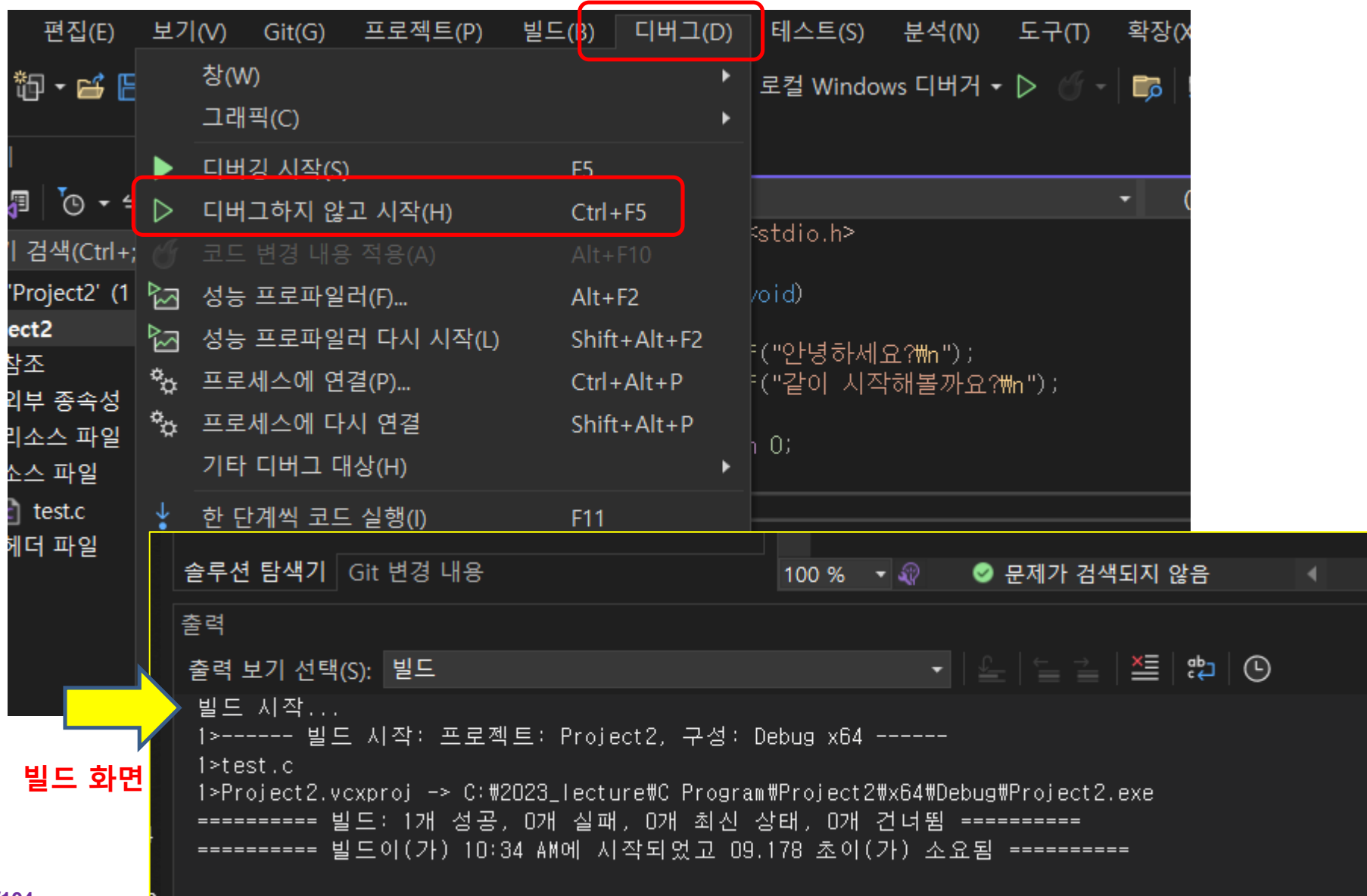
표시 항목(D):



줄 번호 생성 : 도구 - 옵션 - 텍스트 편집기 - 모든언어 - 줄번호

2. Visual studio 실행하기

2.4 프로젝트 Build(Compile & Link) 실행(디버그 - 디버그하지 않고 시작)



The screenshot shows the Visual Studio interface. The '디버그(D)' menu is open, and the '디버그하지 않고 시작(H)' option is highlighted with a red box. A yellow arrow points from this option to the '출력' (Output) window. The '출력' window shows the build output for 'Project2'.

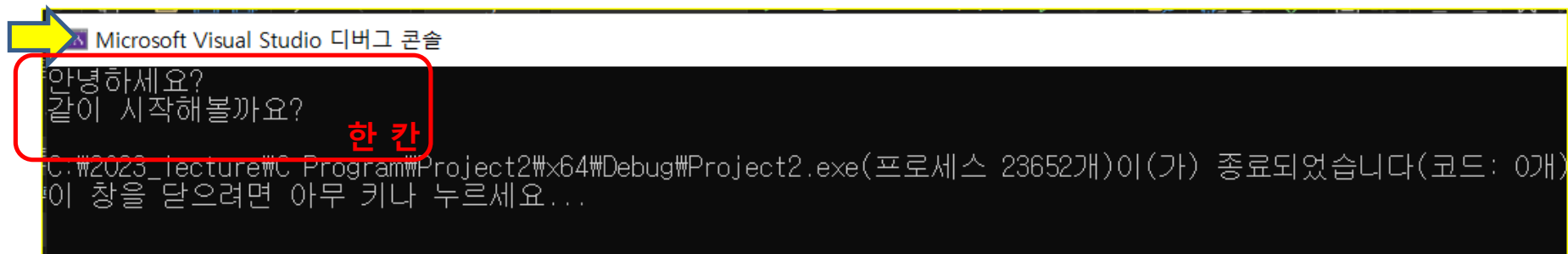
출력 보기 선택(S): 빌드

```
빌드 시작...
1>----- 빌드 시작: 프로젝트: Project2, 구성: Debug x64 -----
1>test.c
1>Project2.vcxproj -> C:\2023_lecture\C Program\Project2\x64\Debug\Project2.exe
===== 빌드: 1개 성공, 0개 실패, 0개 최신 상태, 0개 건너뛴 =====
===== 빌드이(가) 10:34 AM에 시작되었고 09.178 초이(가) 소요됨 =====
```

빌드 화면

2. Visual studio 실행하기

2.4 프로젝트 Build(Compile & Link) 실행 화면 생성



실행이 정상적으로 끝나면 '계속하려면 아무 키나 누르십시오...'가 나타나며,
아무 키나 누르면 다시 돌아간다

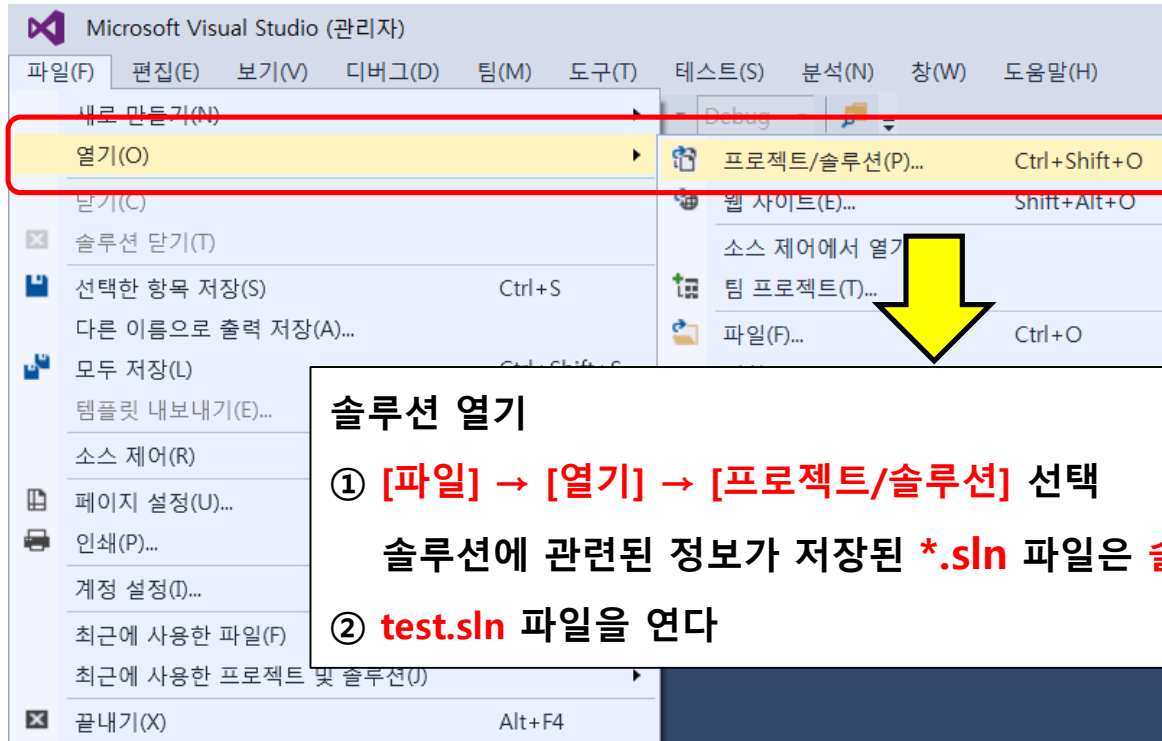
프로젝트 저장: [파일] → [모두 저장] / 솔루션 닫기: [파일] → [솔루션 닫기]

보안 프로그램(안랩, 알약, avast 등)에 의해 실행
안될 경우, 보안프로그램을 잠시 중단



2. Visual studio 실행하기 Visual Studio 2013

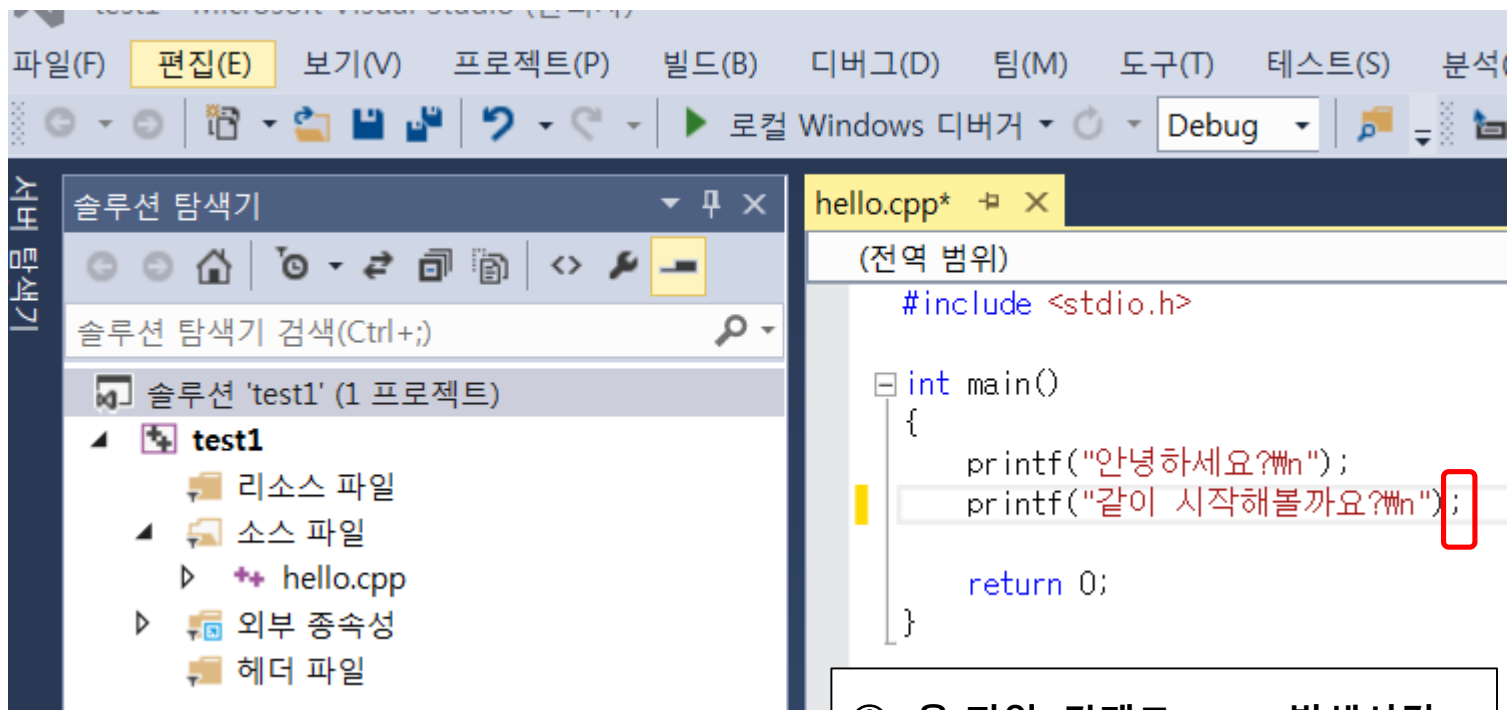
2.5 프로젝트 열기



이름	수정한 날짜	유형	크기
Debug	2018-02-19 오후 1...	파일 폴더	
test1	2018-02-19 오후 1...	파일 폴더	
test1.sln	2018-02-19 오전 8...	Microsoft Visual ...	1KB

2. Visual studio 실행하기 Visual Studio 2013

2.6 컴파일 에러 수정하기(1)



- Ⓐ ;을 지워, 강제로 error 발생시킴
- Ⓑ [빌드]-[솔루션 빌드] 메뉴를 선택

오류 목록					
오류 2개 경고 0개 메시지 0개					
	설명	파일	줄	열	프로젝트
1	error C2143: 구문 오류: ';'이(가) 'return' 앞에 없습니다.	hello.cpp	8	1	test1
2	IntelliSense: ';'가 필요합니다.	hello.cpp	8	2	test1

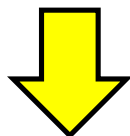
Error 메시지를 더블클릭 하면 error 위치로 커서 이동함

2. Visual studio 실행하기 Visual Studio 2013

2.6 컴파일 에러 수정하기(2)

오류 목록					
오류 2개 경고 0개 메시지 0개					
	설명	파일	줄	열	프로젝트
1	error C2143: 구문 오류 : ';'이(가) 'return' 앞에 없습니다.	hello.cpp	8	1	test1
2	IntelliSense: ';'가 필요합니다.	hello.cpp	8	2	test1

Error 메시지를 더블클릭 하면 error 위치로 커서 이동함



```
#include <stdio.h>

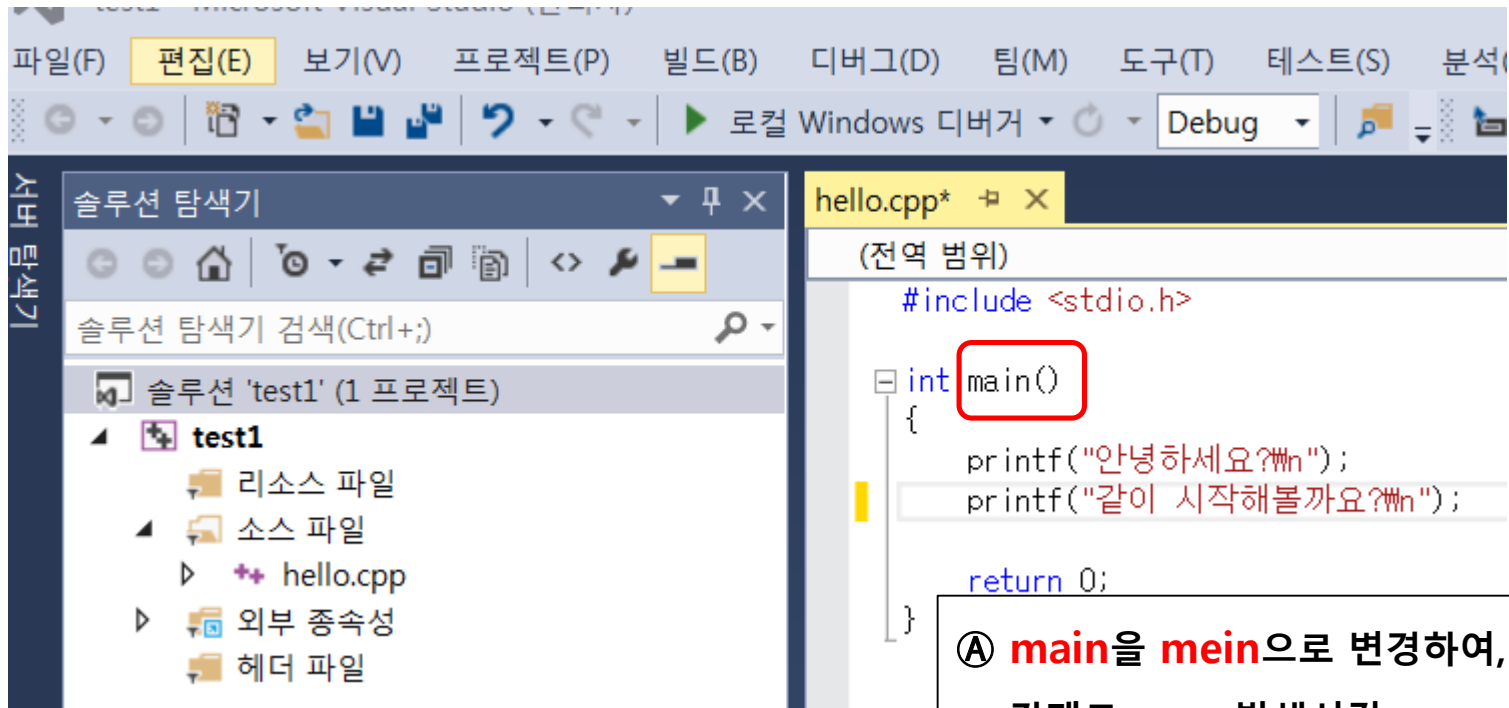
int main()
{
    printf("안녕하세요?\n");
    printf("같이 시작해볼까요?\n");
    return 0;
}
```

커서의 위아래 줄을 확인하며, error 수정

* 에러는 가능한 앞에 있는 것부터 수정한다

2. Visual studio 실행하기 Visual Studio 2013

2.6 컴파일 에러 수정하기(3)



- ① **main**을 **mein**으로 변경하여, 강제로 error 발생시킴
- ② [빌드]-[솔루션 빌드] 메뉴를 선택

오류 목록		
오류 2개 경고 0개 메시지 0개		
	설명	파일
1	error LNK2019: _main 외부 기호(참조 위치: __tmainCRTStartup 함수)에서 확인하지 못했습니다.	MSVCRTD.lib(crtexe.obj)
2	error LNK1120: 1개의 확인할 수 없는 외부 참조입니다.	test1.exe

Link error 발생 (startup에서 mein 함수를 확인 못함)

2. Visual studio 실행하기 Visual Studio 2013

2.7 프로그램 실습 (아르바이트 급여 구하기, 하루 근무시간, 1달 근무일수, 시간당 급여)

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int hours, days, hourly_rate, pay;
```

```
    hours = 8;
```

```
    days = 25;
```

```
    hourly_rate = 4700;
```

```
    pay = hours * days * hourly_rate;
```

```
    printf("아르바이트 총 시간 %d시간 \n", hours * days);
```

```
    printf("총 아르바이트 급여 %d원 \n", pay);
```

```
    return 0;
```

```
}
```

[참고]

총 근무 시간을 계산?

급여 계산?

그외 값은 임의로 할 것

→ 실행 결과

아르바이트 총 시간 200시간

총 아르바이트 급여 940000원

계속하려면 아무 키나 누르십시오 ...

식을 계산한 결과 값이 변환 명세에 맞게 출력된다.

```
printf ( " ... %d ... %lf ... " , 표현식1, 표현식2);
```

이 내용은 모니터에 그대로 출력된다.

→ 빌드 - 솔루션빌드

→ 디버그 - 디버깅하지 않고 시작

2. Visual studio 실행하기 Visual Studio 2013

2.8 프로그램 실습 (원의 넓이와 둘레 구하기)

```
#include <stdio.h>

int main()
{
    double pi = 3.14;
    int radius = 3;
    double area, circle;

    area = radius * radius * pi;
    circle = 2 * radius * pi;

    printf("반지름: %d\n", radius);
    printf("원의 둘레: %.2lf\n", circle);
    printf("원의 넓이: %.2lf\n", area);

    return 0;
}
```

→ 실행 결과

반지름: 3

원의 둘레: 18.84

원의 넓이: 28.26

계속하려면 아무 키나 누르십시오 ...

[참고]

정수형으로 **int** (4Byte)

실수형으로 **float**(4바이트), **double**(8바이트)

%d : 십진수 표현

%lf : 소수점 6자리까지 출력(double형) / %f float형

%.2lf : %.2(l)f 의 의미는 세째자리에서

반올림해서 소수점 이하 2 자리로 출력하는 뜻

식을 계산한 결과 값이 변환 명세에 맞게 출력된다.

```
printf ( " ... %d ... %lf ... " , 표현식1, 표현식2);
```

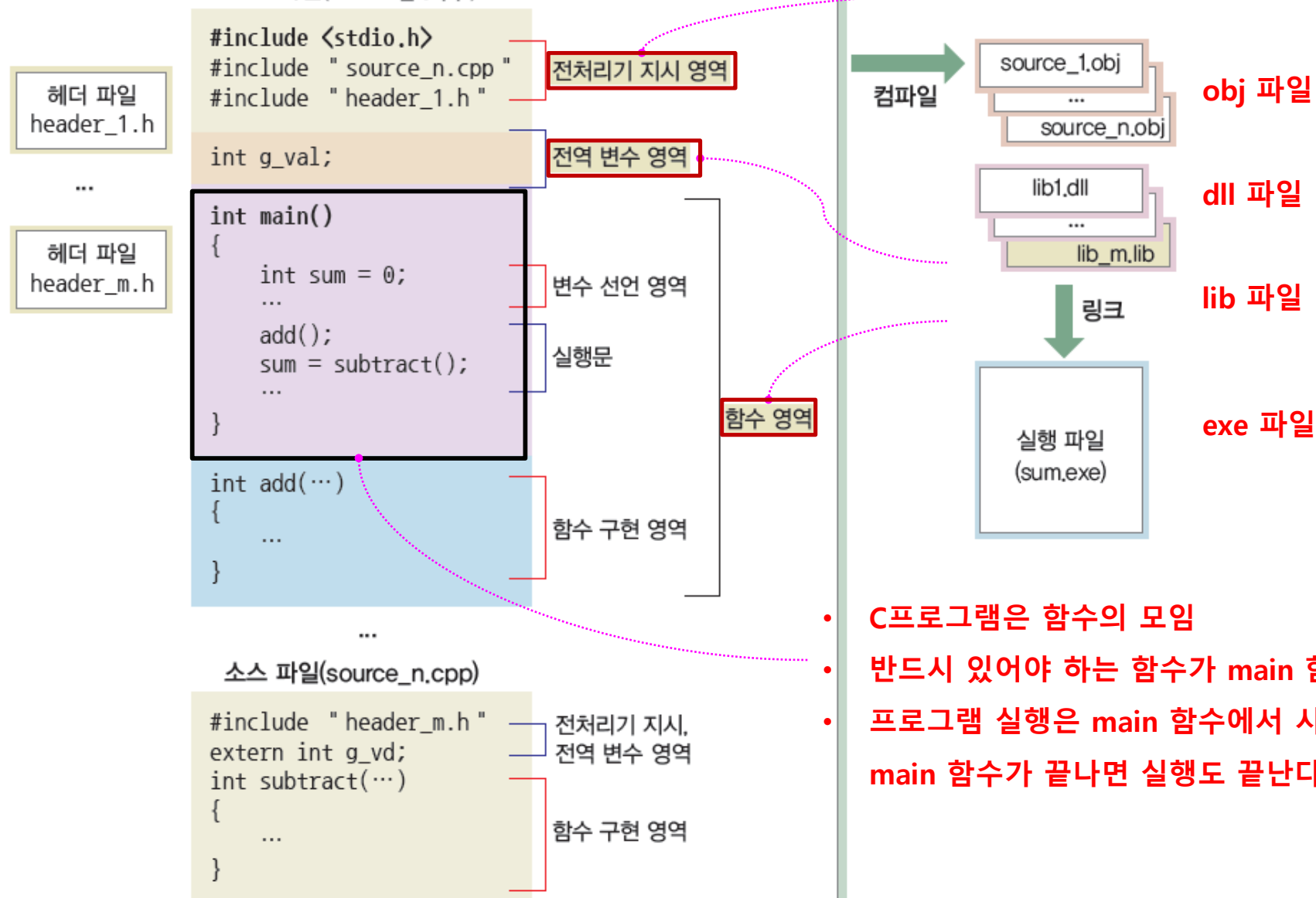
이 내용은 모니터에 그대로 출력된다.

→ 빌드 - 솔루션빌드

→ 디버그 - 디버깅하지 않고 시작

3. 상수, 변수와 자료형

3.1 C응용 프로그램 소스 파일(source_1.cpp)



3. 상수, 변수와 자료형

3.2 프로그램 기본 구조

```
1  #include <stdio.h> // 전처리기 지시 영역
2  /*
3   C 프로그램의 기본 구조를 보여주는 프로그램.
4   밑변이 4이고 높이가 5인 사각형의 면적 구하기 */
5  int main()
6  {
7      // 변수 선언
8      int area, width, height;
9
10
11
12     // 자료 처리
13     width = 4;
14     height = 5;
15     area = width * height;
16
17     // 결과 출력
18     printf("면적 = %d Wn", area);
19
20     return 0; // 함수의 결과값 반환
21 }
```

main 함수

프로그램을 실행하면 처음으로 실행되는 함수
반드시 존재해야 하며 프로젝트에 하나만 존재

함수의 본체(body)

{ }로 묶은 내용을 블록(block)이라 함
{ }는 함수의 본체 외에도 한 개 이상의 문장
(statement)을 묶을 때 사용

선언부

프로그램에서 사용할 변수를 선언하는 곳
C 언어는 반드시 실행부 이전에 변수 선언

실행부

프로그램에서 처리할 명령문 및 처리 구문

주석(comment)문

한 줄 주석: // 로 시작

여러 줄 주석: /* */ 로 묶기

3. 상수, 변수와 자료형

3.3 변수와 상수

- 변수 : 프로그램이 실행되는 동안 **계속 변해 가는 값을** 저장
→ Global variables, Local variables
- 상수 : 프로그램이 수행되는 동안 변하지 않고 사용되는 값을 저장
→ **const** (constant),
- 변수의 필요성
 - 프로그램은 데이터를 처리하는 과정을 기술한 것
 - 데이터를 처리하기 위해서는 데이터를 저장할 있는 그릇(변수)가 필요



냄비 속에서 라면은 계속 변한다(물, 파, 달걀을 넣고 끓임).
= 변수 속에서 데이터는 계속 변한다(더하거나 빼는 등).

[변수 규칙]

알파벳, 숫자, 밑줄로 구성.

공백은 쓸 수 없음.

첫 문자로 숫자는 쓸 수 없음.

대소문자를 구분.

3. 상수, 변수와 자료형

3.3 변수와 상수

- 변수선언 : 프로그램이 실행되는 동안 계속 변해 가는 값을 저장

➔ Global variables, Local variables

```
char c = 'A';  
int sum;  
int width, height;
```

```
unsigned char c = 'A';  
unsigned int sum;  
unsigned int width, height;
```

[변수 규칙]

알파벳, 숫자, 밑줄로 구성.

공백은 쓸 수 없음.

첫 문자로 숫자는 쓸 수 없음.

대소문자를 구분.

- 변수선언금지 : 예약어

ISO C(C99) 예약어

_Bool, _Complex, _Imaginary, auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, inline, int, long, register, restrict, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while

마이크로소프트 C 예약어

__asm, __based, __cdecl, __declspec, __except, __fastcall, __finally, __inline, __int16, __int32, __int64, __int8, __leave, __stdcall, __try, dllexport, dllimport, naked, thread

3. 상수, 변수와 자료형

3.3 변수 선언 예제

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int a1, a2, a3;
```

// 정수를 저장할 int형 변수 선언

```
    a1 = 10;
```

```
    a2 = 20;
```

```
    a3 = a1 + a2;
```

// a1과 a2에 저장된 값을 더한 결과를 a3에 저장

```
    printf("a1 = %d, a2 = %d, a3 = %d\n", a1, a2, a3);
```

```
    a3 = a2 / 2;
```

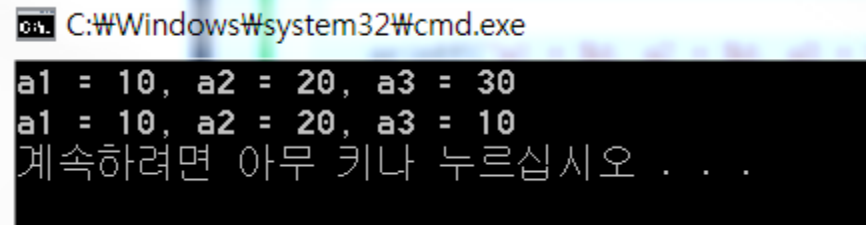
// a2에 저장된 값을 2로 나눈 값(몫)을 다시 a3에 저장

```
    printf("a1 = %d, a2 = %d, a3 = %d\n", a1, a2, a3);
```

```
    return 0;
```

```
}
```

→ 디버그 - 디버깅하지 않고 시작



```
C:\Windows\system32\cmd.exe
```

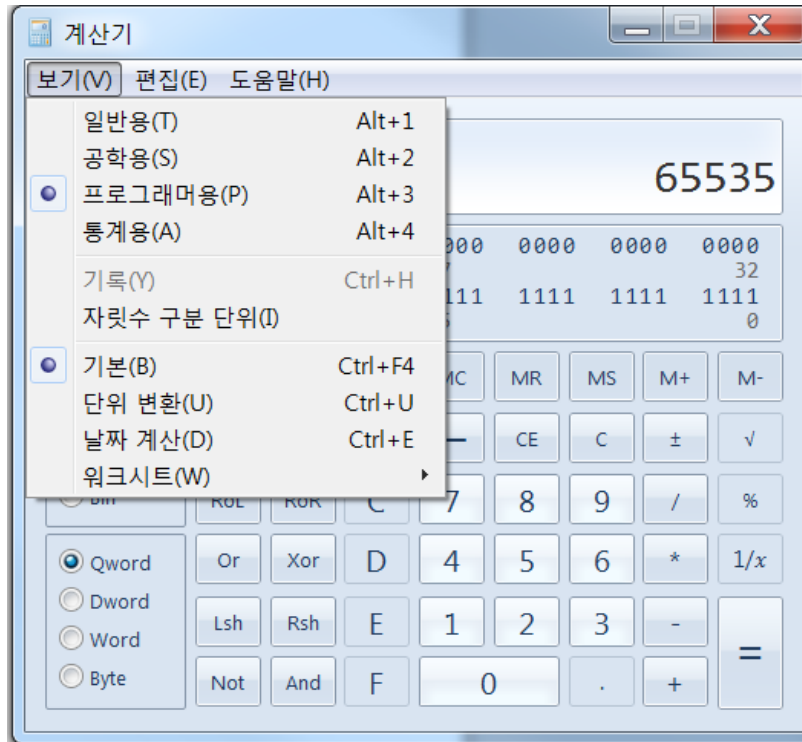
```
a1 = 10, a2 = 20, a3 = 30
```

```
a1 = 10, a2 = 20, a3 = 10
```

```
계속하려면 아무 키나 누르십시오 . . .
```


3. 상수, 변수와 자료형

3.3 변수와 상수

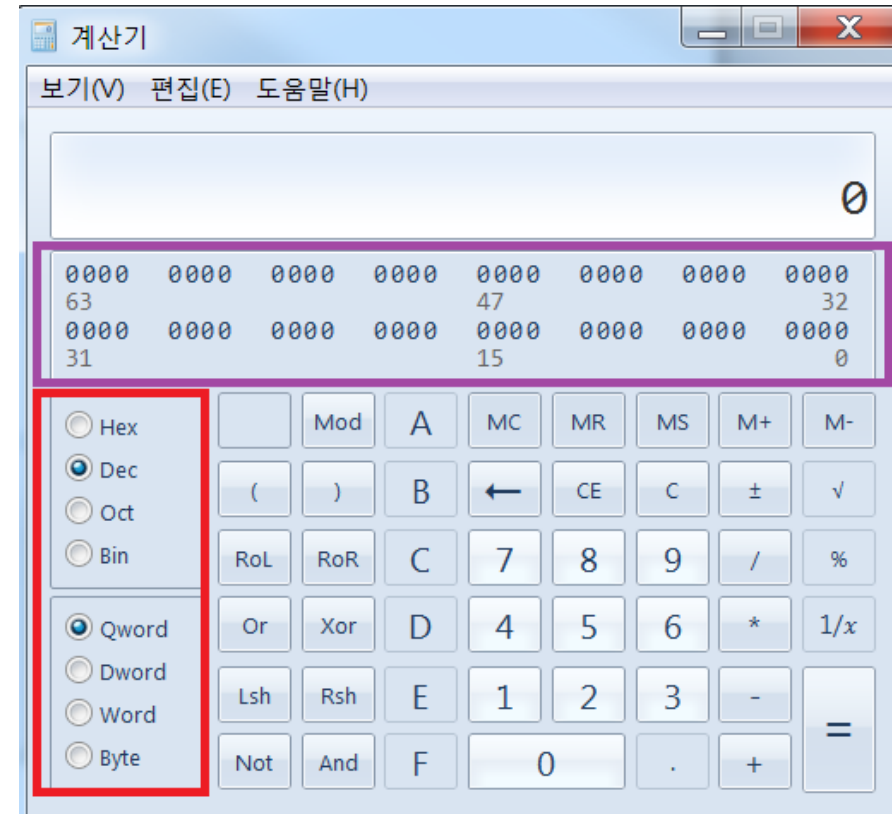


1	0	1	0	1	0	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1×128	$+ 0 \times 64$	$+ 1 \times 32$	$+ 0 \times 16$	$+ 1 \times 8$	$+ 0 \times 4$	$+ 1 \times 2$	$+ 0 \times 1 = 170$

3. 상수, 변수와 자료형

3.3 변수와 상수

■	2진수	10진수	16진수
■	00000000	0	00
■	00000001	1	01
■	00000010	2	02
■	00000011	3	03
■	00001010	10	0A
■	00001111	15	0F
■	00010000	16	10
■	10101010	170	AA
■	11111111	255	FF
■	000011111111	255	0FF
■	0000000111111111	255	00FF
■	0000111111111111	4095	0FFF
■	1111111111111111	6535	FFFF

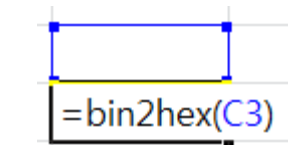
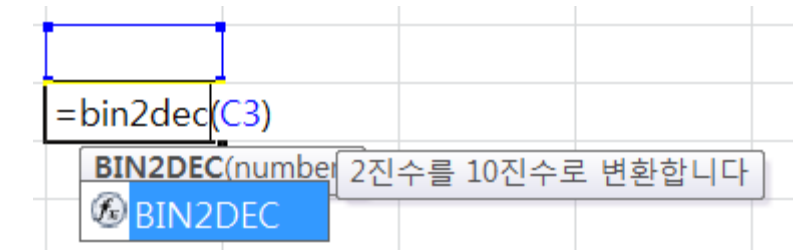
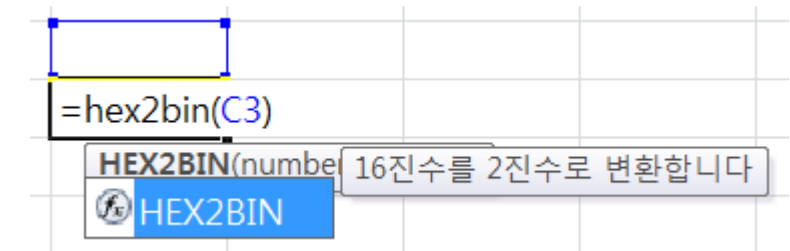
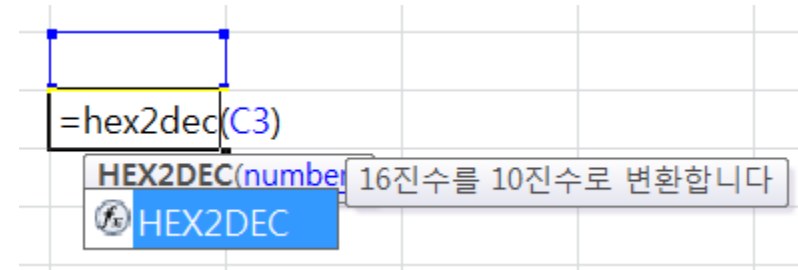
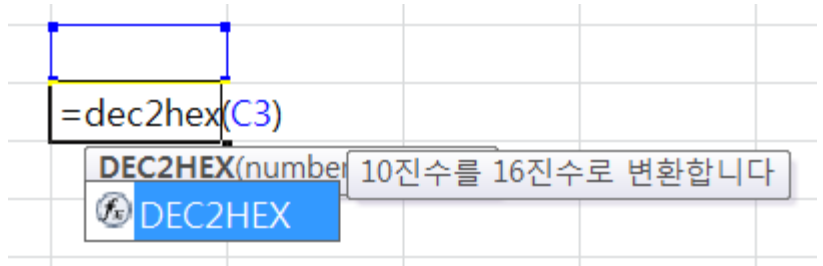
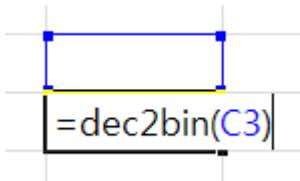


8bit의 최대 표현값 0 ~ 255 / 0xFF

9	512
10	1024
11	2048
12	4096

3. 상수, 변수와 자료형

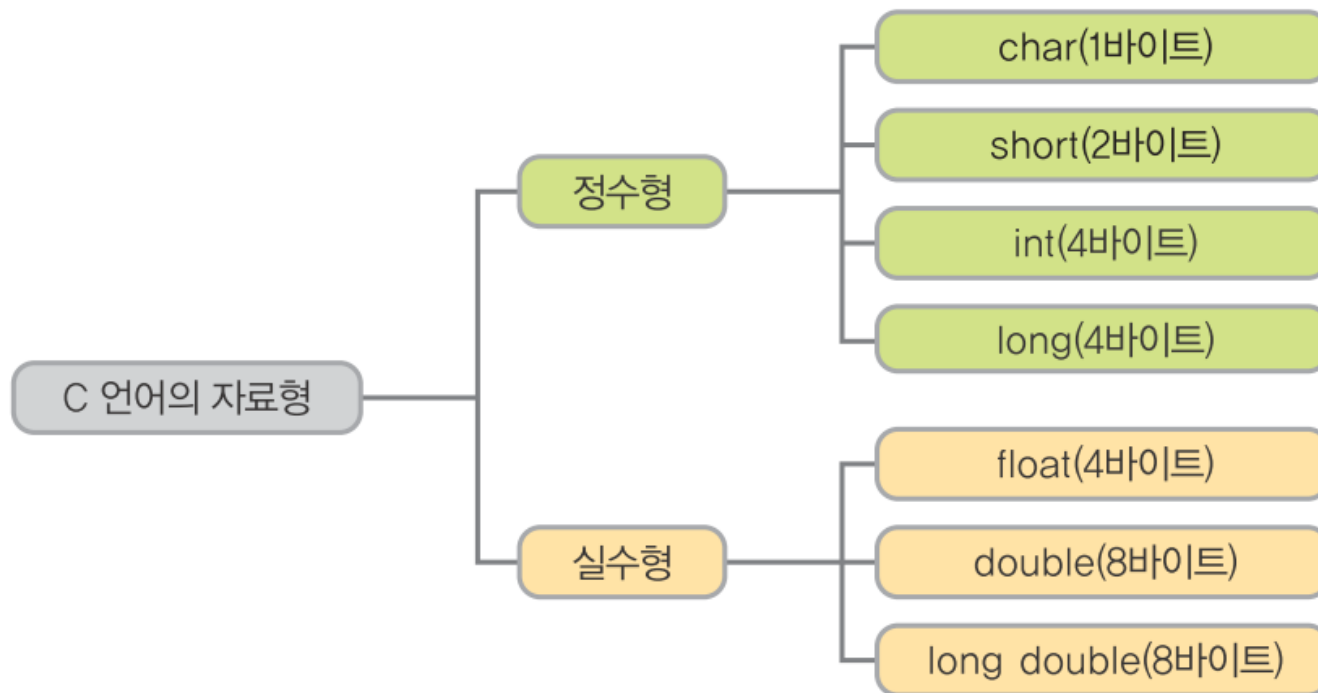
3.3 변수와 상수 (excel 연습)



3. 상수, 변수와 자료형

3.4 자료형

- C 언어에는 정수형 **integer type** 과 실수 형 **floating point type** 의 두 가지 자료형이 있으며, 다시 정수형은 네 가지 자료 형으로, 실수 형은 세 가지 자료 형으로 구분

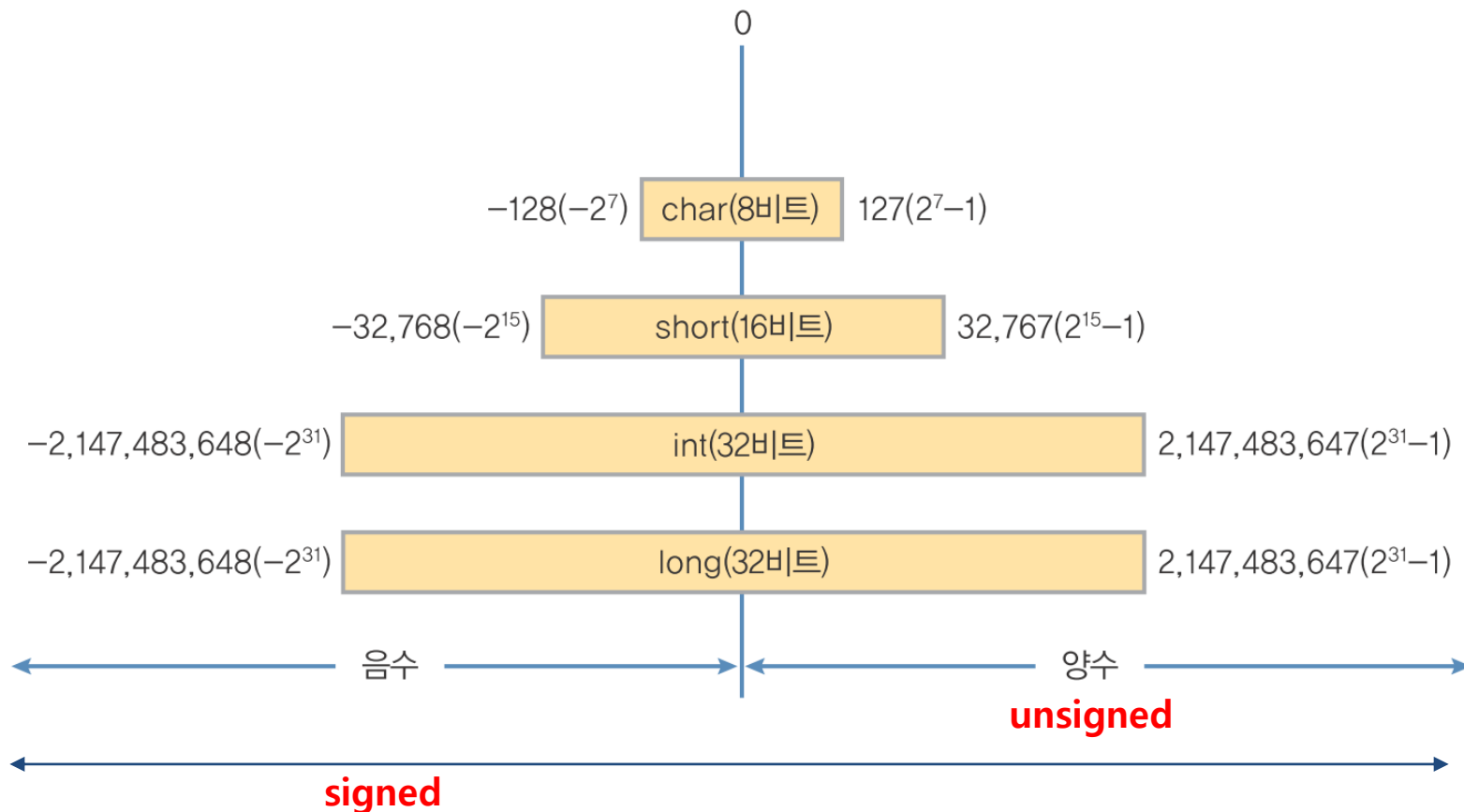


3. 상수, 변수와 자료형

3.4 자료형

■ 정수형 : 4가지 형태

- char형은 문자를 나타내지만 정수형으로 사용
- 정수의 표현 범위 : -2^{n-1} 에서 $+2^{n-1} - 1$

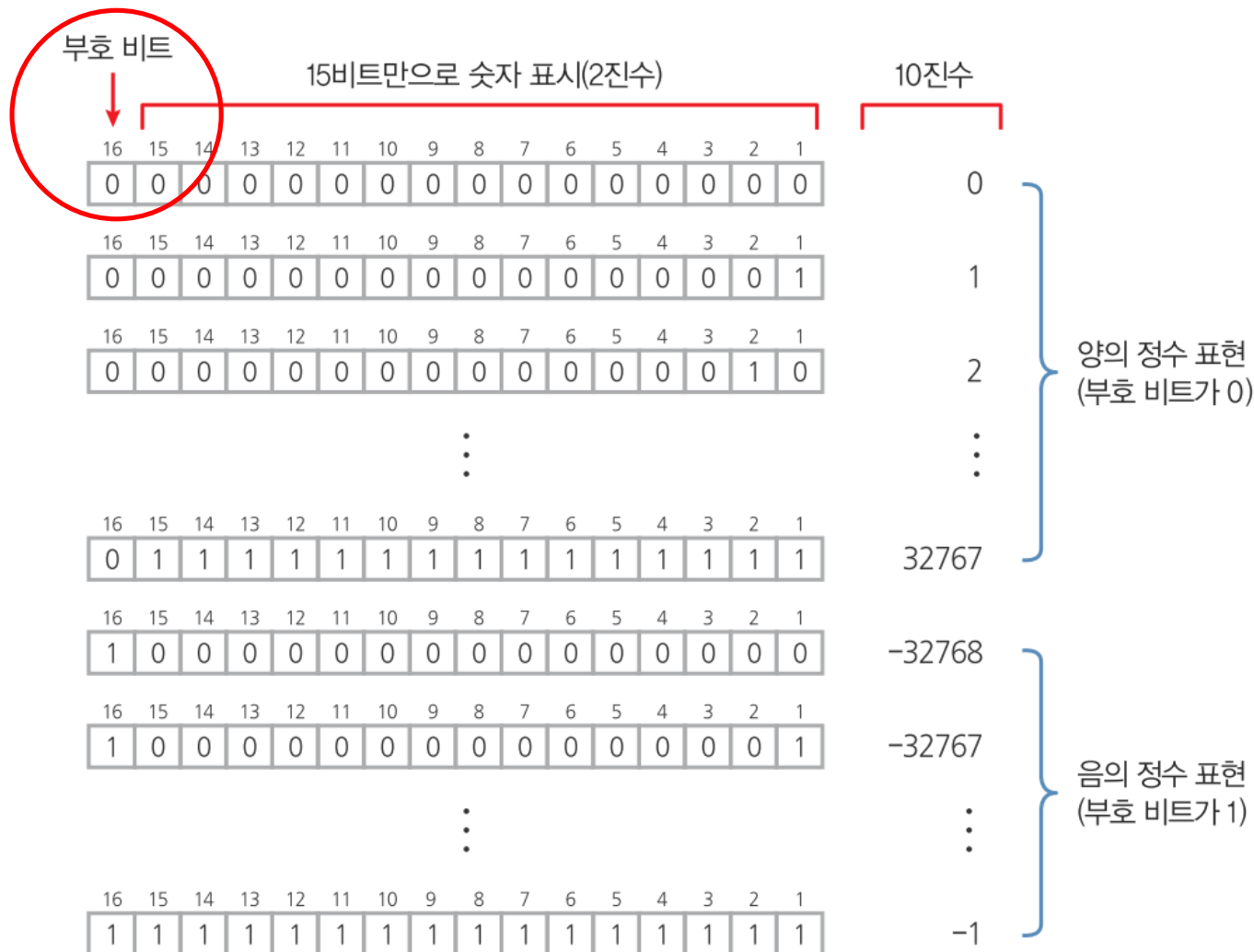


3. 상수, 변수와 자료형

3.4 자료형

■ 정수형 : 음수 표현

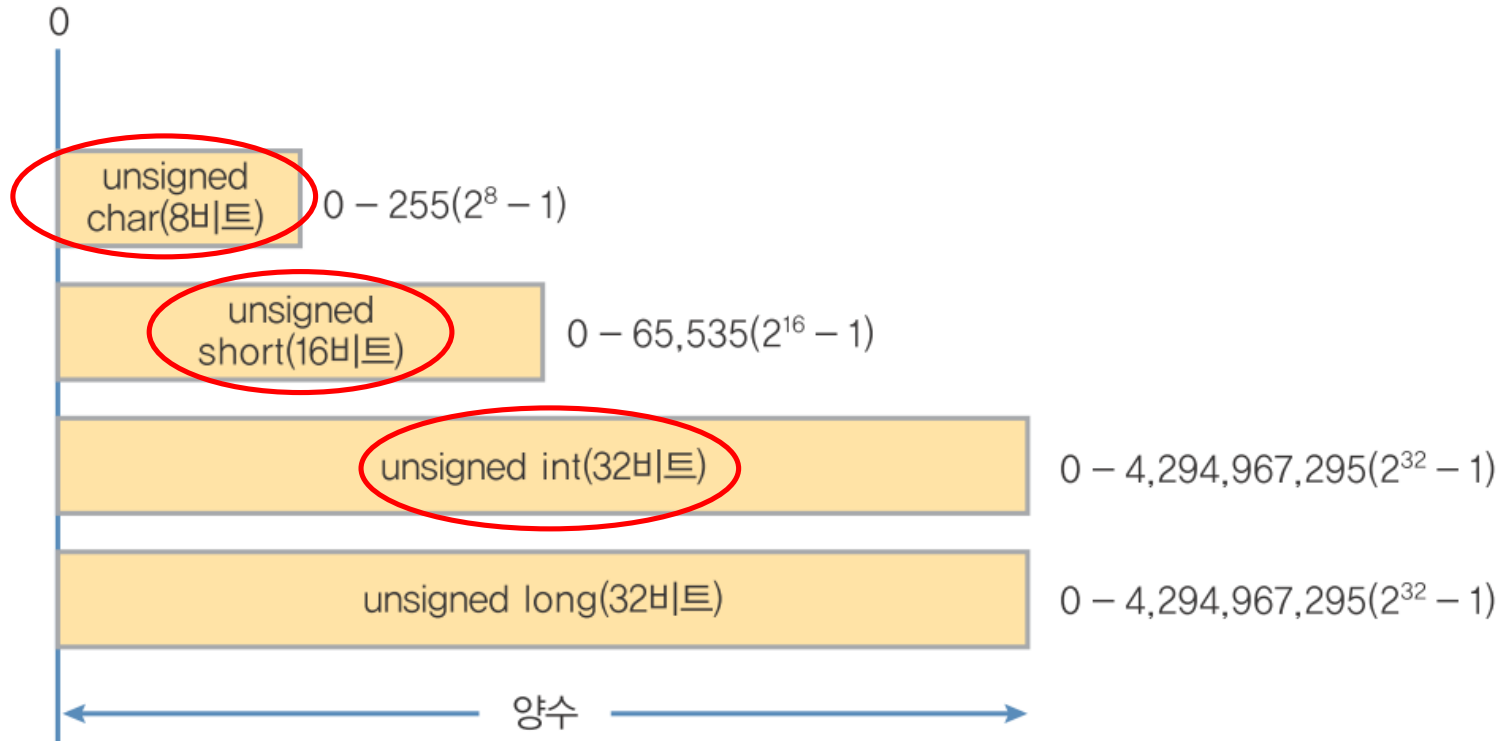
- 정수 자료형은 최상위 비트(MSB: Most significant Bit)를 부호 비트로 사용



3. 상수, 변수와 자료형

3.4 자료형

- 정수형 : 양수만 표현하기 위한 **unsigned** 형 제공



3. 상수, 변수와 자료형

3.4 자료형

■ 정수형 : 문자정수형(char)

- 미국표준협회^{ANSI}에서 **아스키** ASCII, American Standards Committee for Information Interchange

이름으로 제정

- **아스키** 코드는 8비트 중에서 **7비트만을** 사용해서 **128개의** 문자를 표현

2진법	10진법	문자	2진법	10진법	문자
010 1010	042	*	101 0101	085	U
010 1011	043	+	101 0110	086	V
010 1100	044	,	101 0111	087	W
010 1101	045	-	101 1000	088	X
010 1110	046	.	101 1001	089	Y
010 1111	047	/	101 1010	090	Z
011 0000	048	0	101 1011	091	[
011 0001	049	1	101 1100	092	\
011 0010	050	2	101 1101	093]
011 0011	051	3	101 1110	094	^
011 0100	052	4	101 1111	095	_
011 0101	053	5	110 0000	096	`
011 0110	054	6	110 0001	097	a
011 0111	055	7	110 0010	098	b
011 1000	056	8	110 0011	099	c
011 1001	057	9	110 0100	100	d

3. 상수, 변수와 자료형

3.4 자료형

■ ASCII

10	HEX	문자	10	HEX	문자	10	HEX	문자	10	HEX	문자	10	HEX	문자	10	HEX	문자
0	0x00	NULL	22	0x16	STN	44	0x2C	,	66	0x42	B	88	0x58	X	110	0x6E	n
1	0x01	SOH	23	0x17	ETB	45	0x2D	-	67	0x43	C	89	0x59	Y	111	0x6F	o
2	0x02	STX	24	0x18	CAN	46	0x2E	.	68	0x44	D	90	0x5A	Z	112	0x70	p
3	0x03	ETX	25	0x19	EM	47	0x2F	/	69	0x45	E	91	0x5B	[113	0x71	q
4	0x04	EOT	26	0x1A	SUB	48	0x30	0	70	0x46	F	92	0x5C	₩	114	0x72	r
5	0x05	ENQ	27	0x1B	ESC	49	0x31	1	71	0x47	G	93	0x5D]	115	0x73	s
6	0x06	ACK	28	0x1C	FS	50	0x32	2	72	0x48	H	94	0x5E	^	116	0x74	t
7	0x07	BEL	29	0x1D	GS	51	0x33	3	73	0x49	I	95	0x5F	_	117	0x75	u
8	0x08	BS	30	0x1E	RS	52	0x34	4	74	0x4A	J	96	0x60	`	118	0x76	v
9	0x09	HT	31	0x1F	US	53	0x35	5	75	0x4B	K	97	0x61	a	119	0x77	w
10	0x0A	₩n	32	0x20	SP	54	0x36	6	76	0x4C	L	98	0x62	b	120	0x78	x
11	0x0B	VT	33	0x21	!	55	0x37	7	77	0x4D	M	99	0x63	c	121	0x79	y
12	0x0C	FF	34	0x22	"	56	0x38	8	78	0x4E	N	100	0x64	d	122	0x7A	z
13	0x0D	₩r	35	0x23	#	57	0x39	9	79	0x4F	O	101	0x65	e	123	0x7B	{
14	0x0E	SO	36	0x24	\$	58	0x3A	:	80	0x50	P	102	0x66	f	124	0x7C	
15	0x0F	SI	37	0x25	%	59	0x3B	;	81	0x51	Q	103	0x67	g	125	0x7D	}
16	0x10	DLE	38	0x26	&	60	0x3C	<	82	0x52	R	104	0x68	h	126	0x7E	~
17	0x11	DC1	39	0x27	'	61	0x3D	=	83	0x53	S	105	0x69	i	127	0x7F	DEL
18	0x12	DC2	40	0x28	(62	0x3E	>	84	0x54	T	106	0x6A	j			
19	0x13	DC3	41	0x29)	63	0x3F	?	85	0x55	U	107	0x6B	k			
20	0x14	DC4	42	0x2A	*	64	0x40	@	86	0x56	V	108	0x6C	l			
21	0x15	NAK	43	0x2B	+	65	0x41	A	87	0x57	W	109	0x6D	m			

3. 상수, 변수와 자료형

3.4 자료형(문자, 문자열 상수)

문자 상수

단일 인용 부호로 **한 개의 문자를 묶어서 표현**

예) 'A' , 'a'

문자열(string) 상수

이중 인용부호로 **여러 문자를 묶어서 표현**

예) "Hello World" , "C Language" , "A"

문자열의 **마지막에는 NUL 문자**가 내포됨. NULL (무효의, 의미 없는, 존재하지 않는)

10	HEX	문자
0	0x00	NULL

3. 상수, 변수와 자료형

3.4 자료형

■ 실수형

- 소수점 이하의 부분을 가진 실수를 저장할 수 있는 변수
- 실수형으로 **float**형(4바이트), **double**형(8바이트), **long double**형(8바이트)을 제공
- 실수형의 표현
 - 고정 소수점 **fixed point** 표현
 - **부동 소수점 floating point** 표현 : 소수점 이하를 나타내는 가수mantissa 부분과 승수를 나타내는 지수exponential 부분으로 구성



C에서 표현

0.341e1 = 0.314 x 10¹

3. 상수, 변수와 자료형

3.4 자료형

```
01 #include <stdio.h>
```

```
02
```

```
03 int main()
```

```
04 {
```

```
05     char c1 = 100;
```

```
06     char c2 = 27;
```

```
07     char c3 = 28;
```

```
08     char c4 = c1+c2;
```

```
09     char c5 = c1+c3;
```

```
10
```

```
11     float f1 = 3.14159;
```

```
12     float f2 = f1*f1;
```

```
13     double d1 = 3.14159;
```

```
14     double d2 = d1*d1;
```

Local variables



-- char형 변수에 값 저장(char형 변수는 127까지 표현 가능)

-- float형 변수에 값 저장

-- double형 변수에 값 저장

3. 상수, 변수와 자료형

3.5 정수의 표현

Ⓐ 2의 보수 구하기: 방법 1

- ① 2진수를 구한다.
- ② ①의 결과에서 각 비트를 반전하여
1의 보수를 구한다.
- ③ ②의 결과에 1을 더한다.

5 : 0000 0101
1111 1010
1111 1011

Ⓑ 2의 보수 구하기: 방법 2

- ① 2진수를 구한다.
- ② ①의 결과에서 오른쪽 비트부터 첫 번째 1
이 나올 때까지 그대로 옮겨 적는다.
- ③ 나머지 비트들을 반전시킨다.

12 : ① 0000 1100
③ ↓ ② ↓
1111 0100

3. 상수, 변수와 자료형

3.6 제어 문자

→ Printf 문에서 **출력의 위치를 조정하거나 특수한 문자를 표현하는 데 사용**

/ 와 다름

Keyboard의 ₩ 를 의미

제어 문자	기능
\a	‘삐’ 경고음 발생
\b	<u>한 칸 뒤로 옮겨 출력(backspace)</u>
\f	새 페이지의 처음으로 옮겨 출력(form feed)
\n	<u>다음 행의 처음으로 옮겨 출력(new line)</u>
\r	현재 행의 처음으로 옮겨 출력(carriage return)
\t	<u>수평으로 탭만큼 옮겨 출력</u>
\\	역빗금(\) 출력
\‘	작은따옴표(‘) 출력
\“	큰따옴표(“) 출력

3. 상수, 변수와 자료형

3.6 제어 문자

출력 값	변환명세	자료형	출력 형식
정수	<u>%d</u> , %i	int형	정수를 10진수 형태로 출력
	%u	unsigned int형	부호가 없는 정수를 10진수 형태로 출력
	%o	정수형	정수를 8진수 형태로 출력
	%x	정수형	정수를 16진수 형태로 출력
실수	<u>%f</u>	float형	실수를 소수점 아래 6자리까지 출력
	<u>%lf</u>	double형	실수를 소수점 아래 6자리까지 출력
	%e	float형	'가수×10 ^{지수} '에 해당하는 float 형 실수를 '가수e지수'와 같이 지수 형식(과학적 표기 형식)으로 출력
	%le	double형	double형 실수를 '가수e지수' 형식으로 출력
문자	<u>%c</u>	char형	문자 한 개만 출력
문자열	<u>%s</u>	문자열	문자열 출력

Fixed
point

Floating
point

(%d, %lf, %c는 정수형, 실수형, 문자형 중 기본 자료형의 변환명세)

3. 상수, 변수와 자료형

3.6 제어 문자 연습

```
#include <stdio.h>
```

'W141' : 8진수 141

'Wx61' : 16진수 61

```
int main()
```

```
{
```

```
    printf("%cWn", 'a');
```

// 문자 상수 'a' 출력

```
    printf("%cWn", 97);
```

// ASCII 코드 값이 97인 문자 출력

```
    printf("%cWn", 'W141');
```

// ASCII 코드 값이 8진수로 141인 문자 출력

```
    printf("%cWn", 'Wx61');
```

// ASCII 코드 값이 16진수로 61인 문자 출력

```
    printf("빠음 : %cWn", 'Wa');
```

// 이스케이프 문자 빠음을 출력

```
    printf("단일 인용부호 : %cWn", 'W');
```

// 이스케이프 문자 '를 출력

```
    printf("abcdefWbWbWbWb");
```

// 이스케이프 문자 백스페이스를 출력

```
    printf("%sWn", "ghijk");
```

// 문자열 상수 출력

```
    return 0;
```

```
}
```

→ 디버그 - 디버깅하지 않고 시작

C:\Windows\system32\cmd.exe

a

a

a

a

빠

음

:

단

일

:

.

abghijk

계속하려면 아무 키나 누르십시오 . . .

3. 상수, 변수와 자료형

3.7 sizeof

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("%d %dWn", sizeof 2013, sizeof 2013L);
```

```
    printf("%d %dWn", 2013, 0x7DD);
```

```
    printf("%d %xWn", 2013, 2013);
```

```
    return 0;
```

```
}
```

뒤의 값이 차지하는 공간 크기

C에서 표현 ?????

unsigned int : U

long int : L

unsigned long int : UL

→ 디버그 - 디버깅하지 않고 시작

```
4 4
2013 2013
2013 7dd
계속하려면 아무 키나 누르십시오 . . .
```

3. 상수, 변수와 자료형

3.8 프로그램 연습

: 삼각형의 넓이 구하기

➔ 밑변(width) 3cm, 높이(height) 5cm인 삼각형의 넓이(area) 구하기

➔ 삼각형의 넓이 = 밑변 * 높이 / 2

➔ 변수형은 어떻게 선언할까 ??

3. 상수, 변수와 자료형

3.8 프로그램 연습

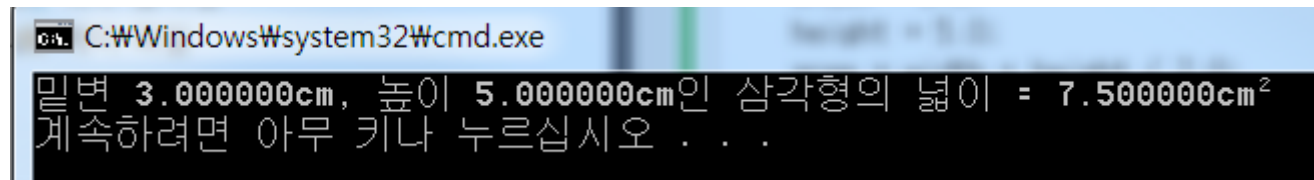
: 삼각형의 넓이 구하기 (1)

```
#include <stdio.h>
int main()
{
    double width, height;    // 변수 선언
    double area;

    width = 3.0;
    height = 5.0;
    area = width * height / 2.0;

    printf("밑변 %lfcm, 높이 %lfcm인 삼각형의 넓이 = ", width, height);
    printf("%lfcm²\n", area);
    return 0;
}
```

→ 디버그 - 디버깅하지 않고 시작



```
C:\Windows\system32\cmd.exe
밑변 3.000000cm, 높이 5.000000cm인 삼각형의 넓이 = 7.500000cm²
계속하려면 아무 키나 누르십시오 . . .
```

3. 상수, 변수와 자료형

3.9 열거형(Enumeration)

- : 기본 변수보다 더 복잡한 파생형 타입으로,
- : 열거형은 정수와 동일한 타입이나 **제한된 값(일정한 값)만 가질 수 있는 기본 타입**.

enum { 멤버, 멤버, ... } 변수명;



- : enum 다음의 { } 괄호 안에 변수가 가질 수 있는 값의 종류를 나열한 후 변수명을 적는다.
- : **대입 가능한 값이 정해져 있어** 안정성이 향상된다.
- : 열거 **멤버는 0부터 시작한다. 초기값을 줄 수 있다.**

`enum { SUN = 1, MON, TUE, WED, THR, FRI, SAT } day;`

- : 열거 **멤버의 중복은 안되지만 값의 중복은 가능하다.**

`enum { MAN, WOMAN, MAN } human;`

`enum { MAN = 1, WOMAN = 2, GIRL = 2 } human;`

3. 상수, 변수와 자료형

3.9 열거형

```
enum { EAST, WEST, SOUTH, NORTH } mark;  
|  
mark= SOUTH;  
if(mark==EAST) ...
```

3.10 사용자 정의형(**typedef**)

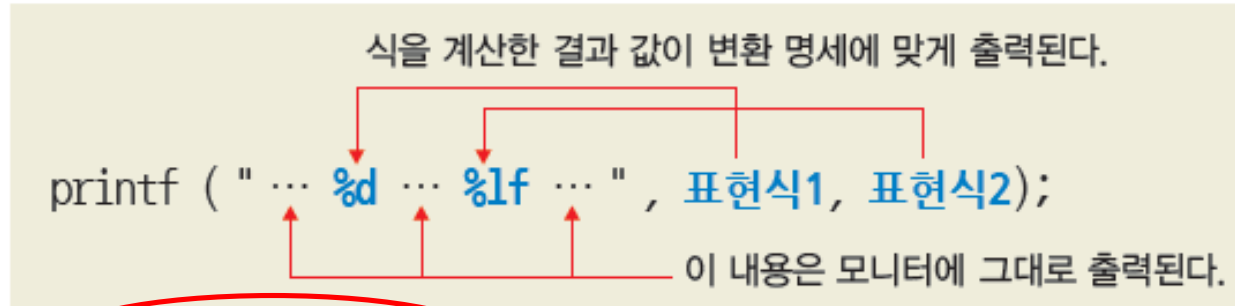
typedef int mecha

정의

int 를 mecha로 정의

4. 표준 입출력과 형식 지정자

4.1 printf 함수 (표준 입출력과 형식 지정자)



```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("%cWn", 'a');
```

// 문자 상수 'a' 출력

```
    printf("%cWn", 97);
```

// ASCII 코드 값이 97인 문자 출력

```
    printf("%cWn", 'W141');
```

// ASCII 코드 값이 8진수로 141인 문자 출력

```
    printf("%cWn", 'Wx61');
```

// ASCII 코드 값이 16진수로 61인 문자 출력

```
    printf("빠음 : %cWn", 'Wa');
```

```
    printf("단일 인용부호 : %cWn", 'W');
```

```
    printf("abcdefWbWbWbWb");
```

```
    printf("%sWn", "ghijk");
```

```
    return 0;
```

```
}
```

'W141' : 8진수 141

'Wx61' : 16진수 61

[참고]

정수형으로 **int** (4Byte)

실수형으로 **float**(4바이트), **double**(8바이트)

%d : 십진수 표현

%lf : 소수점 6자리까지 출력(double형)

%.2lf : 소수점 아래 몇자리까지 출력할지?

4. 표준 입출력과 형식 지정자

4.1 printf 함수 (표준 입출력과 형식 지정자)

➔ 제어문자 : Printf 문에서 출력의 위치를 조정하거나 특수한 문자를 표현하는 데 사용

\ (백슬래시)로 시작 문자
폰트에 따라 'w'로 표시
백슬래시로 읽음
(Keyboard의 ₩ 를 의미)

/ 와 다름

제어 문자	기능
\a	'뽀' 경고음 발생
\b	한 칸 뒤로 옮겨 출력(backspace)
\f	새 페이지의 처음으로 옮겨 출력(form feed)
\n	다음 행의 처음으로 옮겨 출력(new line)
\r	현재 행의 처음으로 옮겨 출력(carriage return)
\t	수평으로 탭만큼 옮겨 출력
\\	역빗금(\) 출력
\'	작은따옴표(') 출력
\“	큰따옴표(“) 출력

4. 표준 입출력과 형식 지정자

4.1 printf 함수 (변환명세 **conversion specification**)

printf("변환명세를 n개 포함한 형식 문자열", 인수1, 인수2, ..., 인수n)

이 자리에는 인수의 값을 10진수 정수 형태로 출력하라는 뜻

printf(" ... **%d** ... **%lf** ... **%c** ... ", 정수형 인수, 실수형 인수, 문자형 인수)

출력 값	변환명세	자료형	출력 형식
정수	%d , %i	int형	정수를 10진수 형태로 출력
	%u	unsigned int형	부호가 없는 정수를 10진수 형태로 출력
	%o	정수형	정수를 8진수 형태로 출력
	%x	정수형	정수를 16진수 형태로 출력
실수	%f	float형	실수를 소수점 아래 6자리까지 출력
	%lf	double형	
	%e	float형	'가수×10 ^{지수} '에 해당하는 float 형 실수를 '가수e지수'와 같이 지수 형식(과학적 표기 형식)으로 출력
	%le	double형	
문자	%c	char형	문자 한 개만 출력
문자열	%s	문자열	문자열 출력


Fixed
point

Floating
point

4. 표준 입출력과 형식 지정자

4.2 표준 입력과 표준 출력



- 출력 : 컴퓨터 내부의 내용을 사람이 인식할 수 있는 형태로 출력
모니터, 프린터, 스피커 등에 표시해 주는 과정
- 입력 : 사용자로부터 마우스, 터치스크린, 조이스틱 등 입력 장치를 통하여
프로그램의 변수에 자료를 전달하는 과정
- 표준 입력과 표준 출력
: 표준 입력 장치는 키보드, 표준 출력 장치는 모니터
- 라이브러리 함수
: `scanf` , `printf` 함수 등  `#include <stdio.h>`

4. 표준 입출력과 형식 지정자

4.3 표준 출력

예제 1. 단순 특정 문자열을 출력하기

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Hello! ");
```

```
    printf("My name is 'C'. ");
```

```
    printf("Nice to meet you. ");
```

```
    return 0;
```

```
}
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Hello! ");
```

```
    printf("My name is 'Yongseok'. \n");
```

```
    printf("You said W>Hello!W". \n\n");
```

```
    printf("Yes!!Wb ");
```

```
    printf("I said W>Hello!W" \n\n");
```

```
    return 0;
```

```
}
```

위 두개의 printf 함수 비교?

4. 표준 입출력과 형식 지정자

4.3 표준 출력

예제 1. 단순 특정 문자열을 출력하기

```
#include <stdio.h>
```

```
int main()  
{
```

```
    printf("Hello! ");
```

```
    printf("My name is 'C'. ");
```

```
    printf("Nice to meet you. ");
```

```
    return 0;
```

```
}
```



```
Hello! My name is 'C'. Nice to meet you. 계속하려면 아무 키나 누르십시오 . . .
```

```
#include <stdio.h>
```

```
int main()  
{
```

```
    printf("Hello! ");
```

```
    printf("My name is 'Yongseok'. \n");
```

```
    printf("You said \b\"Hello!\b". \n\n");
```

```
    printf("Yes!!\b ");
```

```
    printf("I said \b\"Hello!\b" \n\n");
```

```
    return 0;
```

```
}
```



```
Hello! My name is 'Yongseok'.  
You said "Hello!".
```

```
Yes! I said "Hello!"
```

```
계속하려면 아무 키나 누르십시오 . . .
```

4. 표준 입출력과 형식 지정자

먼저 결과값에 대해 예상하기

4.3 표준 출력

예제 2. 변환명세(**conversion specification**) 형식 출력하기

```
#include <stdio.h>

int main()
{
    int age = 26;
    double height = 175.7;
    char grade = 'A';

    printf("나이 %d세\n", age);
    printf("키 %lfcm의 표준체중: %lf\n", height, (height - 100) * 0.9);
    printf("학점: %c\n", grade);
    printf("국적: %s\n\n", "대한민국");
    printf("1억: %e\n", 1.0e8);
    printf("10진수 %d = 8진수 %o = 16진수 %x\n", age, age, age);

    return 0;
}
```

→ 디버그 – 디버깅하지 않고 시작

4. 표준 입출력과 형식 지정자

4.3 표준 출력

예제 2. 변환명세(**conversion specification**) 형식 출력하기

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int age = 26;
```

```
    double height = 175.7;
```

```
    char grade = 'A';
```

```
    printf("나이 %d세\n", age);
```

```
    printf("키 %lfcm의 표준체중: %lf\n", height, (height - 100) * 0.9);
```

```
    printf("학점: %c\n", grade);
```

```
    printf("국적: %s\n\n", "대한민국");
```

```
    printf("1억: %e\n", 1.0e8);
```

```
    printf("10진수 %d = 8진수 %o = 16진수 %X\n", age, age, age);
```

```
    return 0;
```

```
}
```

```
나이 26세
키 175.700000cm의 표준체중: 68.130000
학점: A
국적: 대한민국

1억: 1.000000e+008
10진수 26 = 8진수 32 = 16진수 1A
계속하려면 아무 키나 누르십시오 . . .
```

4. 표준 입출력과 형식 지정자

4.4 문자와 문자열 전용 출력 함수 `putchar`

`putchar` 함수 : 문자 1개 출력

```
putchar('문자');  
putchar(문자형 변수);
```

```
#include <stdio.h>
```

```
int main()  
{  
    char grade = 'A';  
    putchar(grade);  
    putchar('+');  
    putchar('\n');  
  
    return 0;  
}
```

→ 디버그 – 디버깅하지 않고 시작

```
A+  
계속하려면 아무 키나 누르십시오 . . .
```

4. 표준 입출력과 형식 지정자

4.5 puts 함수

: 문자열 출력 후 무조건 개행 문자('\n') 출력 → 즉 줄이 바뀜

: 형식

```
puts("문자열");
```

```
puts(문자열 변수);
```

```
#include <stdio.h>
```

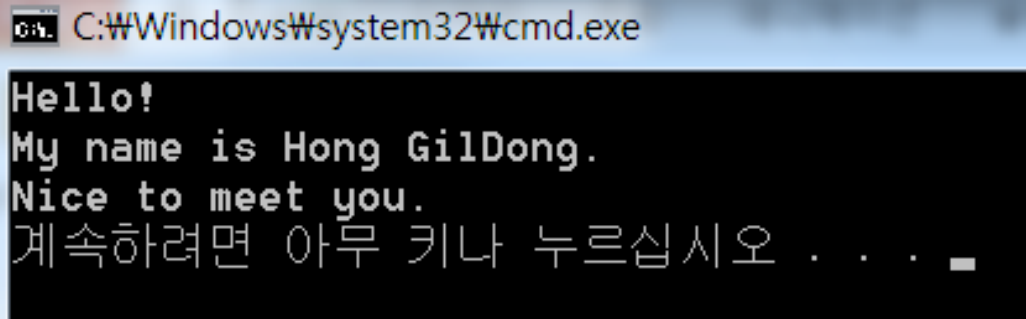
```
int main()  
{
```

```
    puts("Hello!");  
    puts("My name is Hong GilDong.");  
    puts("Nice to meet you.");
```

```
    return 0;
```

```
}
```

→ 디버그 – 디버깅하지 않고 시작



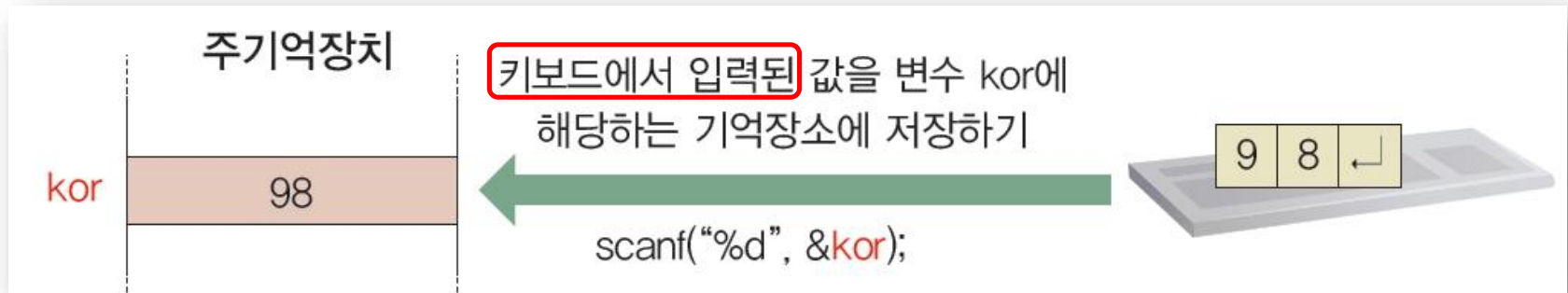
```
cmd C:\Windows\system32\cmd.exe  
Hello!  
My name is Hong GilDong.  
Nice to meet you.  
계속하려면 아무 키나 누르십시오 . . .
```

4. 표준 입출력과 형식 지정자

4.6 입력 함수 `scanf`

: 사용자가 직접 입력할 수 있는 함수

: `scanf`, `scanf_s`, `getchar`, `gets` 등의 함수를 사용



`scanf([형식 제어 문자], [&] 변수 또는 배열, 문자열 변수);`

```
int age;
```

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

“안에 다른 문자는 사용하지 않음

~~`scanf("나이는? %d", &age);`~~ [X]

`printf("나이는? "); scanf("%d", &age);` [O]

“로 닫기 전에 공백 문자(white character)나 `\n`을 넣지 않기

~~`scanf("%d", &age)`~~ ~~`scanf("%d\n", &age)`~~ [X]

4. 표준 입출력과 형식 지정자

4.7 입력 함수 scanf , **scanf_s**

- C 언어는 scanf() 함수와 scanf_s() 함수를 제공하는데 비주얼 스튜디오 2010 이상의 버전에서는 scanf() 함수보다 scanf_s() 함수의 사용을 권장하고 있습니다. scanf() 함수의 경우 문자열을 입력할 때 정의된 크기보다 큰 문자열이 입력되면 실행 시간 오류(오버플로)가 발생합니다. 반면 scanf_s() 함수는 입력된 문자열이 정의된 크기를 벗어나면 입력을 받지 않아 오류를 발생시키지 않습니다.
- 비주얼 스튜디오 2013 이전 버전에서는 scanf() 함수를 사용하는 경우 경고 메시지가 출력되지만 프로그램은 실행됩니다.

4. 표준 입출력과 형식 지정자

4.8 예제 `scanf`, `scanf_s`

```
#include <stdio.h>
```

```
int main()
{
    int age;
    char gender;
    double height;

    printf("성별은? (남자라면 M 여자라면 F) ");
    scanf("%c", &gender);
    printf("나이는? ");
    scanf("%d", &age);
    printf("키는? ");
    scanf("%lf", &height);

    printf("\n=====");
    printf("성별: %c\n", gender);
    printf("나이: %d세\n", age);
    printf("키: %.1lfcm\n", height);

    return 0;
}
```

→ 디버그 – 디버깅하지 않고 시작
(오류 발생)

4. 표준 입출력과 형식 지정자

4.9 예제 `scanf`, `scanf_s`

예제: `scanf` 를 `scanf_s` 로 변경하여 빌드 후 실행

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int age;
```

```
    char gender;
```

```
    double height;
```

```
    printf("성별은? (남자라면 M 여자라면 F) ");
```

```
    scanf_s("%c", &gender);
```

```
    printf("나이는? ");
```

```
    scanf_s("%d", &age);
```

```
    printf("키는? ");
```

```
    scanf_s("%lf", &height);
```

```
    printf("Wn=====Wn");
```

```
    printf("성별: %cWn", gender);
```

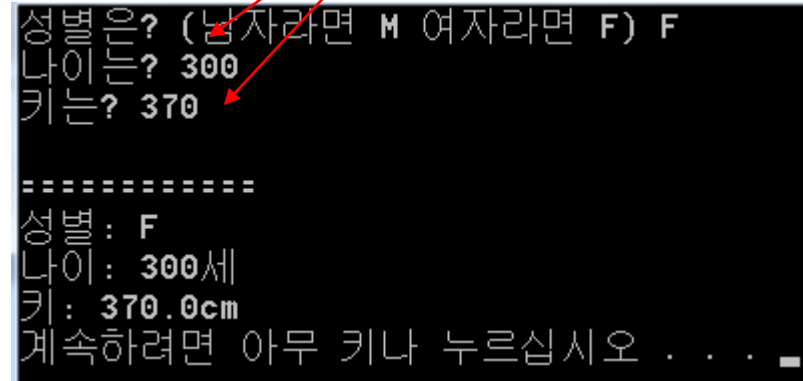
```
    printf("나이: %d세Wn", age);
```

```
    printf("키: %.1lfcmWn", height);
```

```
    return 0;
```

```
}
```

대문자 입력후 ENTER



```
성별은? (남자라면 M 여자라면 F) F
나이는? 300
키는? 370

=====
성별: F
나이: 300세
키: 370.0cm
계속하려면 아무 키나 누르십시오 . . .
```

4. 표준 입출력과 형식 지정자

여러 개의 자료를 한꺼번에 입력하기

```
scanf("변환명세를 n개 포함한 형식 문자열", &변수명1, &변수명2, ..., &변수명n )  
scanf("%d%d", &변수명1, &변수명2)
```

반드시 개수가 같아야 함

- 키보드 입력 방법

- 입력 자료 간 구분은 공백 문자(스페이스바, 탭키, 엔터키 입력 문자)를 사용
- 입력의 끝은 [Enter]키 사용
- 예) 입력 방법 1: 10 20 ↵
 입력 방법 2: 10[Tab] 20 ↵
 입력 방법 3: 10 ↵
 20 ↵

✓ "%d %d"도 가능

✓ ~~"%d%d\n"~~, ~~"%d%d"~~ 처럼 "로 닫기 전에 'wn'이나 빈칸을 넣지 않아야 함

✓ "%d, %d" → 입력 시 반드시 ','를 눌러야 함

입력 방법: 10, 20↵

4. 표준 입출력과 형식 지정자

여러 개의 자료를 한꺼번에 입력하기

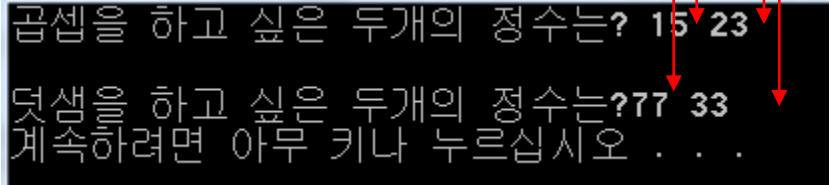
```
#include <stdio.h>

int main()
{
    unsigned int n1, n2, n3, n4;

    printf("곱셈을 하고 싶은 두개의 정수는? ");
    scanf("%d%d", &n1,&n2);
    printf("\n덧셈을 하고 싶은 두개의 정수는?");
    scanf("%d%d", &n3,&n4);

    return 0;
}
```

→ 디버그 - 디버깅하지 않고 시작



A terminal window showing the execution of the C program. The first prompt is "곱셈을 하고 싶은 두개의 정수는?". The user enters "15 23". Red arrows point from the text "Space key" to the space between 15 and 23, and from "enter" to the end of the line. The second prompt is "덧셈을 하고 싶은 두개의 정수는?". The user enters "77 33". A red arrow points from "enter" to the end of the line. The third line of output is "계속하려면 아무 키나 누르십시오 . . .".

4. 표준 입출력과 형식 지정자

4.10 문자 전용 입력 함수 `getchar`

```
변수 = getchar()
```

: `getchar` 함수 실행시 키보드에서 누른 문자가 `getchar()`의 결과 값이 됨
→ 이 결과 값(문자 1개)이 변수에 대입(저장)됨

```
#include <stdio.h>
```

```
int main()
{
    unsigned char grade;

    printf("C 언어의 학점은? ");

    grade = getchar();

    printf("C 언어의 학점은 %c입니다.\n", grade);

    return 0;
}
```

→ 디버그 - 디버깅하지 않고 시작

문자 1개만 해당함



```
c언어의 학점은?A+
c언어의 학점은 A입니다.
계속하려면 아무 키나 누르십시오...
```

4. 표준 입출력과 형식 지정자

4.11 문자열 전용 입력 함수 `gets`, `gets_s`

: `getchar`은 공백, 탭 등이 포함된 즉, 문자열을 한꺼번에 입력 받을 수 없음

→ “Hong GilDong”을 입력하면 “Hong”만 입력 됨

→ 행 단위 입력 함수 `gets` 또는 `gets_s`를 사용



```
gets(문자열을 저장할 변수);  
↳ 엔터키를 입력하기 전까지의 모든 문자가 변수에 저장
```

`#include <stdio.h>`

```
int main()  
{  
    char address[30];  
    printf("주소는? ");  
    gets_s(address);  
    printf("입력한 주소: %s", address);  
    return 0;  
}
```

최대 29개까지

```
주소는?부산시 사상구 주례로 47  
입력한 주소는 : 부산시 사상구 주례로 47계속하려면 아무 키나 누르십시오 . . .
```

4. 표준 입출력과 형식 지정자

Library

#include <stdio.h> 를 선언해야 아래의 다양한 함수를 사용

```
#include <stdio.h>
```

```
printf
```

```
putchar
```

```
puts
```

```
Scanf_s
```

```
getchar
```

```
gets_s
```

msdn.microsoft.com/en-us/library/d9x1s805.aspx

5. 연산자

5.1 연산자의 종류

: 피연산자의 개수에 따라 **단항**, **이항**, **삼항 연산자**로 분류

연산자 분류	예
산술 연산자	+ - * / %
부호 연산자	+ -
대입 연산자	= 복합 대입 연산자
증감 연산자	++ --
포인터 연산자	* &
구조체 연산자	. ->
관계 연산자	== != <= < >= >
논리 연산자	&& !
비트 연산자	& ~ >> <<
삼항 조건 연산자	? :
쉼표 연산자	,
sizeof 연산자	sizeof
캐스트 연산자	(type)
괄호 연산자	()
C++ 연산자	new delete :: .* ->*

5. 연산자

5.2 나머지 연산자

: 피연산자의 개수에 따라 단항, 이항, 삼항 연산자로 분류

/	나누기	5 / 2	2
		5 / 2.0	2.5
%	나머지 구하기(정수만 가능)	5 % 2	1

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int value = 386;
```

```
    int h = value / 100;
```

```
    int d = value / 10 % 10;
```

```
    int n = value % 10;
```

```
    printf("%d백%d십%d\n", h, d, n);
```

```
    return 0;
```

```
}
```

→ 디버그 - 디버깅하지 않고 시작

3백8십6

계속하려면 아무 키나 누르십시오 . . .

5. 연산자

5.3 복합 대입 연산자(compound assignment operator)

a += b	a = a + b
a -= b	a = a - b
a *= b	a = a * b
a /= b	a = a / b
a %= b	a = a % b

```
#include <stdio.h>
```

```
int main()  
{
```

```
    int x, y;
```

```
    x = y = 5;
```

```
    printf("x = %d, y = %d\n", x, y);  
    printf("x += y의 결과는 %d\n\n", x += y);
```

```
    printf("x = %d, y = %d\n", x, y);  
    printf("x -= y의 결과는 %d\n\n", x -= y);
```

```
    printf("x = %d, y = %d\n", x, y);  
    printf("x *= y의 결과는 %d\n\n", x *= y);
```

```
    printf("x = %d, y = %d\n", x, y);  
    printf("x /= y의 결과는 %d\n\n", x /= y);
```

```
    printf("x = %d, y = %d\n", x, y);  
    printf("x %%= y의 결과는 %d\n", x %= y);
```

```
    return 0;
```

```
}
```

→ 빌드 - 솔루션빌드

→ 디버그 - 디버깅하지 않고 시작

```
x = 5, y = 5  
x += y의 결과는 10
```

```
x = 10, y = 5  
x -= y의 결과는 5
```

```
x = 5, y = 5  
x *= y의 결과는 25
```

```
x = 25, y = 5  
x /= y의 결과는 5
```

```
x = 5, y = 5  
x %%= y의 결과는 0  
계속하려면 아무 키나 누르십시오
```

5. 연산자

5.4 관계 연산자(relational operator)

: 좌우 피연산자의 크기를 비교 (결과, 참/거짓)

관계 연산자의 종류와 연산 결과(x는 1, y는 2인 경우)

관계 연산자	의미	연산 결과
$x > y$	x가 y보다 큰가?	거짓(0)
$x >= y$	x가 y보다 크거나 같은가?	거짓(0)
$x < y$	x가 y보다 작은가?	참(1)
$x <= y$	x가 y보다 작거나 같은가?	참(1)
$x == y$	x가 y와 같은가?	거짓(0)
$x != y$	x가 y와 같지 않은가?	참(1)

(주) $x = y$ 는 대입문

5. 연산자

5.5 논리 연산자(logical operator)

: 논리 값(참과 거짓)에 대한 연산 (결과는 참/거짓)

: **조건 표현에 주로 사용**

예, if ((year % 4) == 0 **&&** (year % 100 != 0))

if ((average < 60) **||** (absence > 12))

x	y	x && y	x y	!x
거짓(0)	거짓(0)	거짓(0)	거짓(0)	참(1)
거짓(0)	참(1)	거짓(0)	참(1)	참(1)
참(1)	거짓(0)	거짓(0)	참(1)	거짓(0)
참(1)	참(1)	참(1)	참(1)	거짓(0)

예, 'x가 5보다 크고 10보다 작으면 참**이고** 아니면 거짓'을 표현하면?

5 < x < 10

×

(5 < x) && (x < 10)

5. 연산자

5.6 조건 연산자(conditional operator)

: 유일한 삼항 연산자

형식

조건 연산자

조건식 ? 수식1 : 수식2;

예

max = $a > b$? a : b;

조건식 $a > b$ 가 참일 때의 결과

조건식 $a > b$ 가 거짓일 때의 결과

→ 디버그 - 디버깅하지 않고 시작

5. 연산자

5.6 조건 연산자(conditional operator)

```
#include <stdio.h>
```

```
int main()  
{
```

```
    int x;
```

```
    printf("정수 입력 >> ");
```

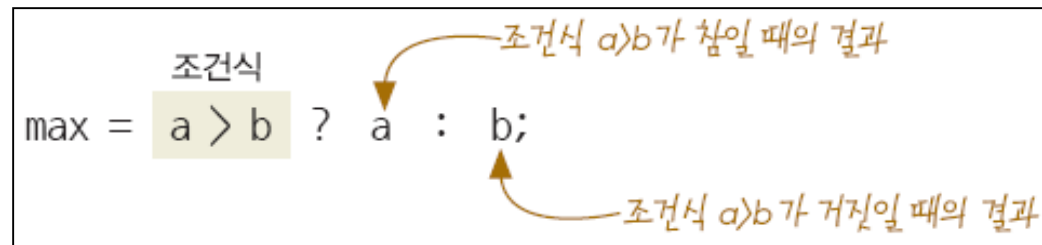
```
    scanf_s("%d", &x);
```

```
    (x % 2 == 0) ? printf("%d는 짝수입니다.\n", x) :
```

```
                printf("%d는 홀수입니다.\n", x);
```

```
    return 0;
```

```
}
```



디버그 - 디버깅하지 않고 시작

5. 연산자

5.6 조건 연산자(conditional operator)

```
#include <stdio.h>

int main()
{
    int x;

    printf("정수 입력 >> ");
    scanf_s("%d", &x);

    (x % 2 == 0) ? printf("%d는 짝수입니다.\n", x) :
                  printf("%d는 홀수입니다.\n", x);

    return 0;
}
```

정수 입력 >> 7_



정수 입력 >> 7
7는 홀수입니다.
계속하려면 아무 키나 누르십시오 . . .

5. 연산자

5.7 프로그래밍 실습

문제 : 두 수를 입력 받아 몫과 나머지 출력하기

해결 과정 :

- (1) 두 수 $n1, n2$ 입력
- (2) 두 수 중 큰 수는 max 에 작은 수는 min 에 저장하기
- (3) 큰 수 max 를 작은 수 min 으로 나눈 몫과 나머지 출력하기

5. 연산자

5.7 프로그래밍 실습

문제 : 두 수를 입력 받아 몫과 나머지 출력하기

해결 과정 :

(1) 두 수 n1, n2 입력

```
printf("두 정수 입력 : ");  
scanf_s("%d %d", &n1, &n2);
```

(2) 두 수 중 큰 수는 max에 작은 수는 min에 저장하기

```
(n1 > n2) ? (max = n1, min = n2) : (max = n2, min = n1);
```

(3) 큰 수 max를 작은 수 min으로 나눈 몫과 나머지 출력하기

```
printf("Wn>> 큰 수 / 작은 수 = %dWn", max / min);  
printf(">> 큰 수 % 작은 수 = %dWn", max % min);
```

5. 연산자

5.7 프로그래밍 실습

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n1, n2, max, min;
```

```
    printf("두 정수 입력 : ");
```

```
    scanf("%d %d", &n1, &n2);
```

```
    (n1 > n2) ? (max = n1, min = n2) : (max = n2, min = n1);
```

```
    printf("\n>> 큰 수 / 작은 수 = %d\n", max / min);
```

```
    printf(">> 큰 수 % 작은 수 = %d\n", max % min);
```

```
    return 0;
```

```
}
```

→ 디버그 - 디버깅하지 않고 시작

space keyboard 입력

두 정수 입력 : 7 5

>> 큰 수 / 작은 수 = 1

>> 큰 수 % 작은 수 = 2

계속하려면 아무 키나 누르십시오 . . .

5. 연산자

5.8 증감 연산자(increment/decrement operator)

: 변수에만 사용하며 변수의 값을 1 증가 또는 감소시킴

➔ 변수++

: 변수 값을 1 증가

: 변수 = 변수 + 1 ➔ 변수 += 1

➔ 변수--

: 변수 값을 1 감소

: 변수 = 변수 - 1 ➔ 변수 -= 1

종류	증감 연산자
전위형(prefix)	++X
	--X
후위형(postfix)	X++
	X--



```
int x = 5;  
int y = x++;  
printf("%d", x);  
printf("%d", y);
```

현재 x값을 사용하여
수식을 평가한 후
1증가

```
int x = 5;  
int y = ++x;  
printf("%d", x);  
printf("%d", y);
```

먼저 x값을 1증가한 후 이
증가된 값을 수식 평가에
사용

5. 연산자

5.8 증감 연산자(increment/decrement operator)

먼저 생각해 봅시다

```
#include <stdio.h>

int main()
{
    int x = 1, y = 2, z = 3;

    x = ++x * 5;
    y = y++ * 5;
    z = 5 - --z;

    printf("x = %d\n", x);
    printf("y = %d\n", y);
    printf("z = %d\n", z);

    return 0;
}
```

→ 빌드 - 솔루션빌드

→ 디버그 - 디버깅하지 않고 시작

5. 연산자

5.8 증감 연산자(increment/decrement operator)

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 1, y = 2, z = 3;
```

```
    x = ++x * 5;  x = x + 1;
                  x = x * 5;
```

```
    y = y++ * 5;  y = y * 5;
                  y = y + 1;
```

```
    z = 5 - --z;  z = z - 1;
                  z = 5 - z;
```

```
    printf("x = %d\n", x);
```

```
    printf("y = %d\n", y);
```

```
    printf("z = %d\n", z);
```

```
    return 0;
```

```
}
```

```
x = 10
```

```
y = 11
```

```
z = 3
```

```
계속하려면 아무 키나 누르십시오 . . .
```

5. 연산자

5.9 비트 연산자(bit operator)

: 비트 단위로 연산을 수행

AND(&), OR(|), XOR(^), NOT(~) 연산자

➔ 비트 논리 연산자 : & , | , ^ , ~

➔ 비트 이동 연산자 : << , >>

x(비트 값)	y(비트 값)	x & y	x y	x ^ y	~x
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

5. 연산자

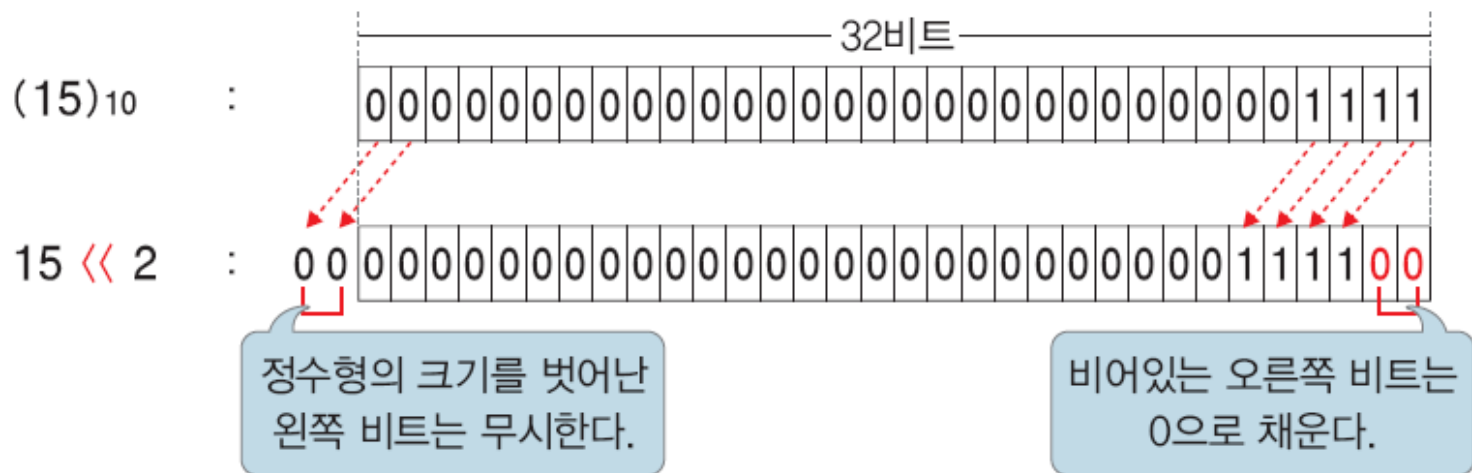
5.10 비트 이동 연산자

: 왼쪽 비트 이동 연산자(<<)

: 피연산자 << n

➔ 피연산자의 각 비트값을 왼쪽으로 한 비트씩 이동하기를 n번 반복

➔ 그 자리에는 0으로 채워짐



: 오른쪽 비트 이동 연산자(>>)

: 피연산자 >> n

➔ 피연산자의 각 비트값을 오른쪽으로 한 비트씩 이동하기를 n번 반복

5. 연산자

5.10 비트 이동 연산자

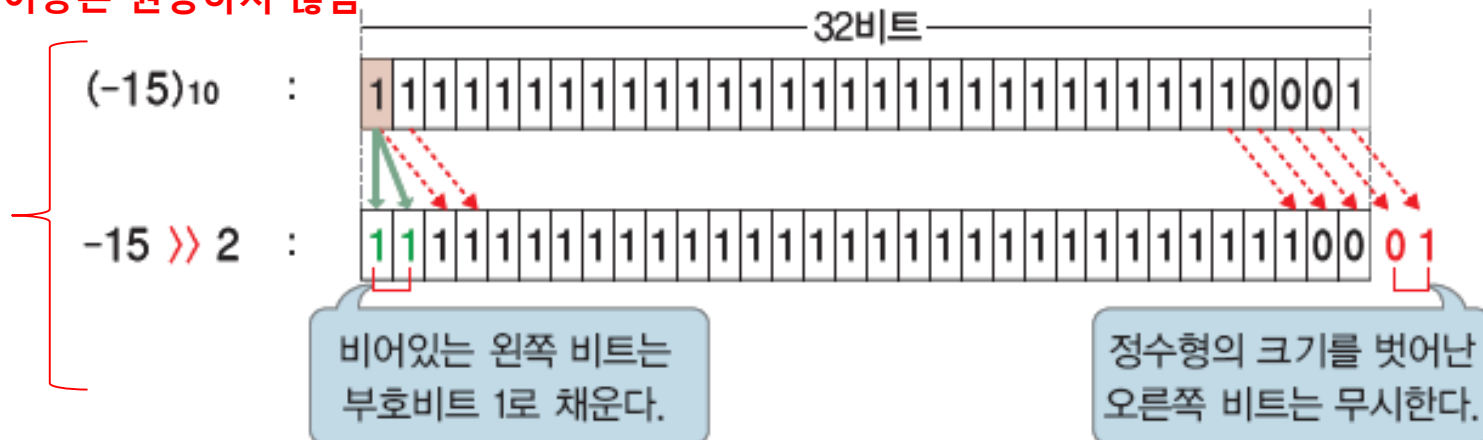
: 오른쪽 비트 이동 연산자(>>)

: 피연산자 >> n

➔ 피연산자의 각 비트값을 오른쪽으로 한 비트씩 이동하기를 n번 반복



음수의 비트 이동은 권장하지 않음.



5. 연산자

5.11 연습 문제

먼저 계산해 봅시다 : 엑셀 활용

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 5, y = 12, n = 1;
```

```
    printf("%d & %d = %d\n", x, y, x & y);
```

```
    printf("%d | %d = %d\n", x, y, x | y);
```

```
    printf("%d ^ %d = %d\n", x, y, x ^ y);
```

```
    printf("~%d = %d\n", x, ~x);
```

```
    x = 15;
```

```
    printf("%d >> %d = %d\n", x, n, x >> n);
```

```
    printf("%d >> %d = %d\n", x, n+1, x >> (n + 1));
```

```
    printf("%d >> %d = %d\n", x, n+2, x >> n + 2);
```

```
    printf("%d << %d = %d\n", x, n, x << n);
```

```
    printf("%d << %d = %d\n", x, n+1, x << (n + 1));
```

```
    printf("%d << %d = %d\n", x, n+2, x << (n + 2));
```

```
    return 0;
```

```
}
```

→ 디버그 – 디버깅하지 않고 시작

5. 연산자

5.11 연습 문제

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int x = 5, y = 12, n = 1;
```

```
    printf("%d & %d = %d\n", x, y, x & y);
```

```
    printf("%d | %d = %d\n", x, y, x | y);
```

```
    printf("%d ^ %d = %d\n", x, y, x ^ y);
```

```
    printf("~%d = %d\n\n", x, ~x);
```

```
    x = 15;
```

```
    printf("%d >> %d = %d\n", x, n, x >> n);
```

```
    printf("%d >> %d = %d\n", x, n+1, x >> (n + 1));
```

```
    printf("%d >> %d = %d\n", x, n+2, x >> n + 2);
```

```
    printf("%d << %d = %d\n", x, n, x << n);
```

```
    printf("%d << %d = %d\n", x, n+1, x << (n + 1));
```

```
    printf("%d << %d = %d\n", x, n+2, x << (n + 2));
```

```
    return 0;
```

```
}
```

```
5 & 12 = 4
5 | 12 = 13
5 ^ 12 = 9
~5 = -6
```

```
15 >> 1 = 7
15 >> 2 = 3
15 >> 3 = 1
15 << 1 = 30
15 << 2 = 60
15 << 3 = 120
```

계속하려면 아무 키나 누르십시오 . . .

5. 연산자

5.12 프로그래밍 실습

배경 : 많은 산업 분야에서 변수 단위가 아닌 **변수에 포함된 비트 단위의 값을 이용한다.**

특정 비트의 값이 1인지 0인지에 따라 로봇이 팔을 든 상태(1)인지 내린 상태(0)인지?

특정 스위치에 해당하는 비트의 **값이 1이면 스위치가 on 상태, 0이면 off**

문제 : 사용자에게서 입력받은 문자의 실제 비트열(8비트) 출력하기

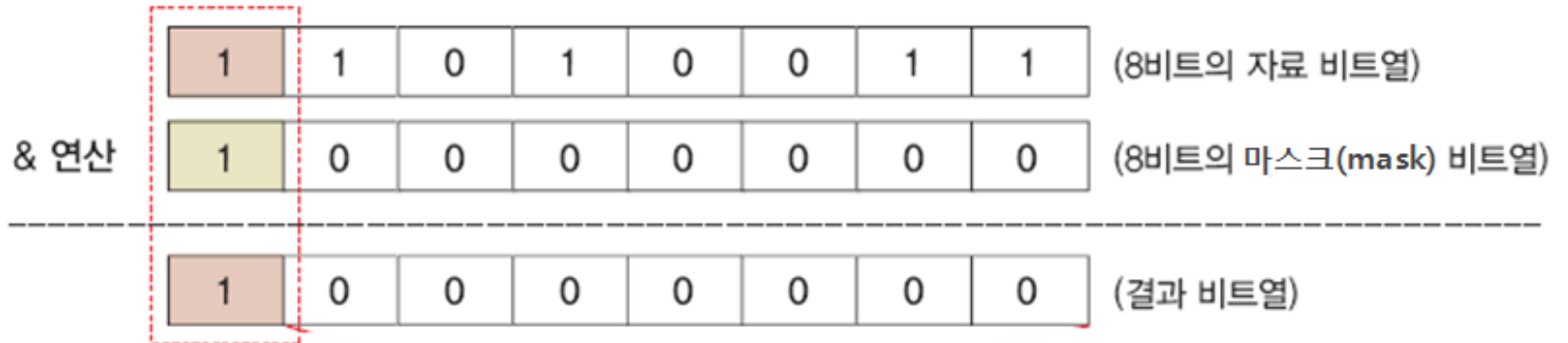
자료형

■ ASCII (16진수, char형만 이해)

10	HEX	문자	10	HEX	문자	10	HEX	문자	10	HEX	문자	10	HEX	문자	10	HEX	문자
0	0x00	NULL	22	0x16	STN	44	0x2C	,	66	0x42	B	88	0x58	X	110	0x6E	n
1	0x01	SOH	23	0x17	ETB	45	0x2D	-	67	0x43	C	89	0x59	Y	111	0x6F	o
2	0x02	STX	24	0x18	CAN	46	0x2E	.	68	0x44	D	90	0x5A	Z	112	0x70	p
3	0x03	ETX	25	0x19	EM	47	0x2F	/	69	0x45	E	91	0x5B	[113	0x71	q
4	0x04	EOT	26	0x1A	SUB	48	0x30	0	70	0x46	F	92	0x5C	\	114	0x72	r
5	0x05	ENQ	27	0x1B	ESC	49	0x31	1	71	0x47	G	93	0x5D]	115	0x73	s
6	0x06	ACK	28	0x1C	FS	50	0x32	2	72	0x48	H	94	0x5E	^	116	0x74	t
7	0x07	BEL	29	0x1D	GS	51	0x33	3	73	0x49	I	95	0x5F	_	117	0x75	u
8	0x08	BS	30	0x1E	RS	52	0x34	4	74	0x4A	J	96	0x60	`	118	0x76	v
9	0x09	HT	31	0x1F	US	53	0x35	5	75	0x4B	K	97	0x61	a	119	0x77	w
10	0x0A	\n	32	0x20	SP	54	0x36	6	76	0x4C	L	98	0x62	b	120	0x78	x
11	0x0B	VT	33	0x21	!	55	0x37	7	77	0x4D	M	99	0x63	c	121	0x79	y
12	0x0C	FF	34	0x22	"	56	0x38	8	78	0x4E	N	100	0x64	d	122	0x7A	z
13	0x0D	\r	35	0x23	#	57	0x39	9	79	0x4F	O	101	0x65	e	123	0x7B	{
14	0x0E	SO	36	0x24	\$	58	0x3A	:	80	0x50	P	102	0x66	f	124	0x7C	
15	0x0F	SI	37	0x25	%	59	0x3B	;	81	0x51	Q	103	0x67	g	125	0x7D	}
16	0x10	DLE	38	0x26	&	60	0x3C	<	82	0x52	R	104	0x68	h	126	0x7E	~
17	0x11	DC1	39	0x27	'	61	0x3D	=	83	0x53	S	105	0x69	i	127	0x7F	DEL
18	0x12	DC2	40	0x28	(62	0x3E	>	84	0x54	T	106	0x6A	j			
19	0x13	DC3	41	0x29)	63	0x3F	?	85	0x55	U	107	0x6B	k			
20	0x14	DC4	42	0x2A	*	64	0x40	@	86	0x56	V	108	0x6C	l			
21	0x15	NAK	43	0x2B	+	65	0x41	A	87	0x57	W	109	0x6D	m			

5. 연산자

5.12 프로그래밍 실습



INPUT ASCII **&** (1 << 7) **?** 1 : 0);

5. 연산자

5.12 프로그래밍 실습

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    char ch;
```

```
    printf("문자 입력 : ");
```

```
    scanf_s("%c", &ch);
```

```
    printf("%2d", ch & (1 << 7) ? 1 : 0);
```

```
    printf("%2d", ch & (1 << 6) ? 1 : 0);
```

```
    printf("%2d", ch & (1 << 5) ? 1 : 0);
```

```
    printf("%2d", ch & (1 << 4) ? 1 : 0);
```

```
    printf("%2d", ch & (1 << 3) ? 1 : 0);
```

```
    printf("%2d", ch & (1 << 2) ? 1 : 0);
```

```
    printf("%2d", ch & (1 << 1) ? 1 : 0);
```

```
    printf("%2d\n", ch & (1) ? 1 : 0);
```

```
    return 0;
```

```
}
```

→ 디버그 – 디버깅하지 않고 시작

%2d : 2자리보다 작으면 여백을 추가해서 2자리 확보

3 : ASCII 0x33 => 0011 0011

문자 입력 : 3

0 0 1 1 0 0 1 1

계속하려면 아무 키나 누르십시오 . . .

[참고]

%d 변환명세에서의 필드폭

"%**필드폭**d" "%**+****필드폭**d" "%**-****필드폭**d"

출력에 사용할 전체 칸

무조건 부호 출력
양수도 + 출력

왼쪽 정렬

[참고]

printf("%d\n",1234);	1 2 3 4	%d만 사용
printf("%+d\n",1234);	+ 1 2 3 4	+를 사용하여 부호 지정
printf("%10d\n",1234);	1 2 3 4	출력 폭 지정, 묵시적으로 오른쪽 정렬 10칸
printf("%010d\n",1234);	0 0 0 0 0 0 1 2 3 4	폭과 0 지정, 오른쪽 정렬을 하고 0으로 채움
printf("%-10d\n",1234);	1 2 3 4	폭과 왼쪽 정렬 지정
printf("%-+10d\n",1234);	+ 1 2 3 4	폭과 왼쪽 정렬, 부호 지정
printf("%+10d\n",1234);	+ 1 2 3 4	폭과 부호 지정
printf("%d\n",-1234);	- 1 2 3 4	음수 출력
printf("%10d\n",-1234);	- 1 2 3 4	폭을 지정하고 음수 출력 10칸
printf("%010d\n",-1234);	- 0 0 0 0 0 0 1 2 3 4	폭과 0 지정, 왼쪽을 0으로 채움
printf("%#10o\n",1234);	0 2 3 2 2	#를 지정해 8진수 표시 출력
printf("%#10X\n",1234);	0 X 4 D 2	#를 지정해 16진수(0X) 표시하여 대문자로 출력
printf("%#10x\n",1234);	0 x 4 d 2	#를 지정해 16진수(0x) 표시 출력