
C# INTRO

Update 2023

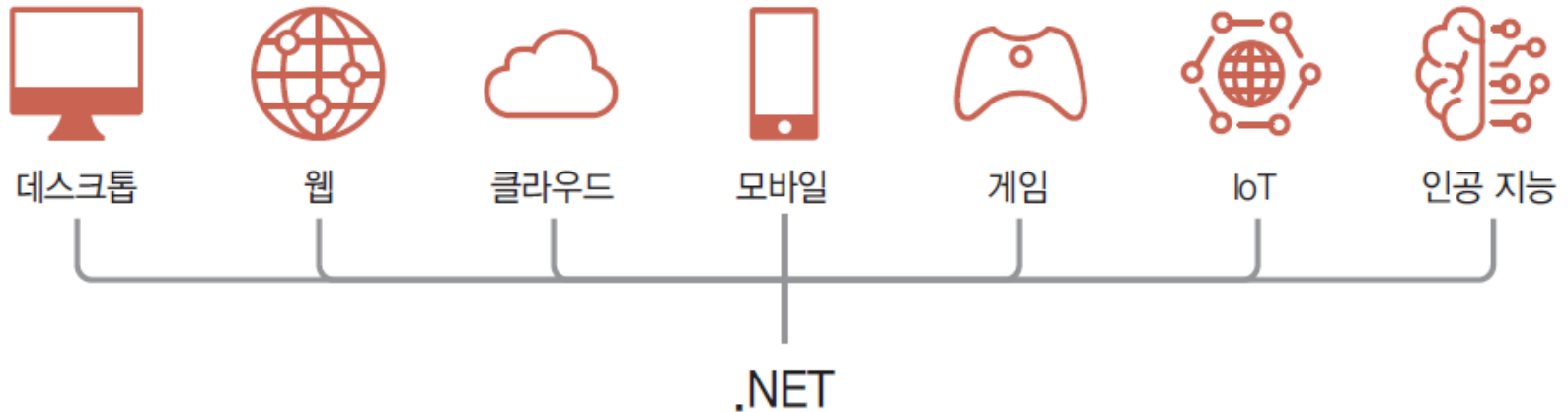
A solid green horizontal bar at the bottom of the slide.

01. C# 프로그래밍의 이해

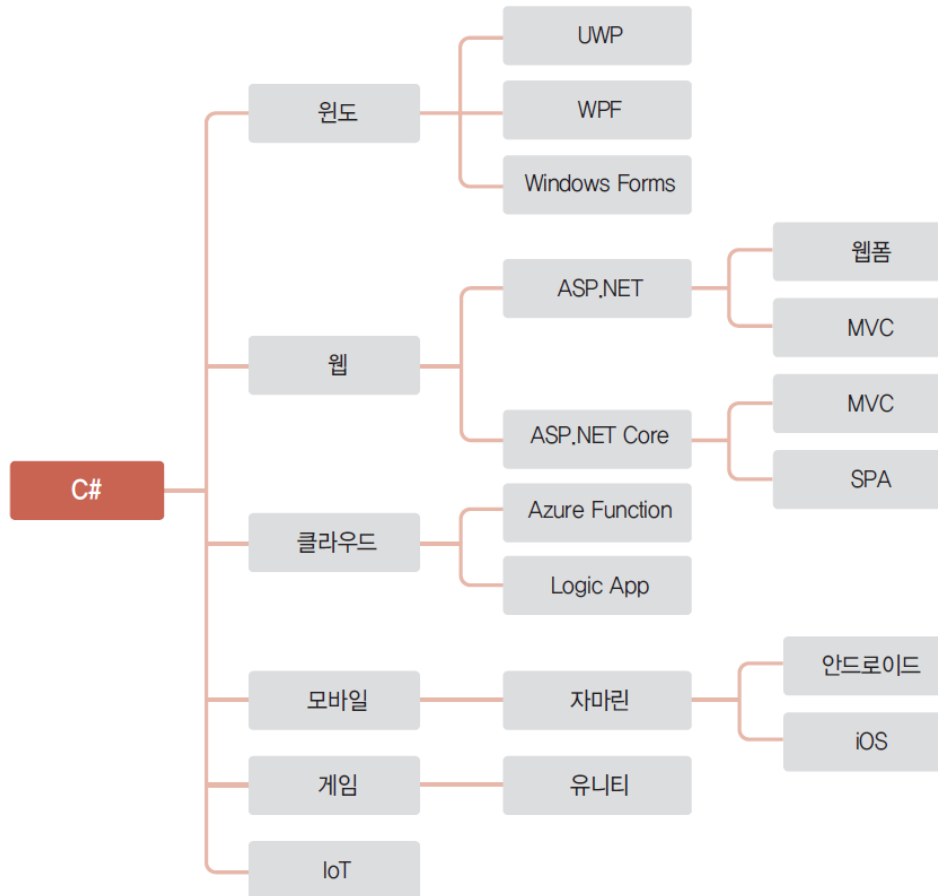
- 2000년 7월에 마이크로소프트가 발표한 객체 지향 프로그래밍 언어
- 마이크로소프트의 닷넷 플랫폼 기반
- C, C++, 자바, 자바스크립트와 기초 문법등이 제공되어짐
- C#의 활동 범위
 - 데스크톱 응용 프로그램
 - 웹 응용 프로그램
 - 모바일 응용 프로그램
 - 데이터베이스 응용 프로그램
 - 게임 프로그램
 - 클라우드 프로그램
 - IoT 프로그램

01. C# 프로그래밍의 이해

닷넷 생태계



01. C# 프로그래밍의 이해



02. C# 프로그래밍의 역사

버전	발표	특징
1.0	2002년 2월 13일	<ul style="list-style-type: none">• C#의 첫 번째 버전• 닷넷 프레임워크(.NET Framework) 1.0• 간결하고 현대화된 언어• 관리된 코드(managed code)• 자동화된 가비지 컬렉션(garbage collection)
1.1	2003년	비주얼 스튜디오 도구 기능 향상
2.0	2005년	<ul style="list-style-type: none">• 제네릭(generic)• 부분(partial) 클래스• 무명 메서드(anonymous method)• 이터레이터(반복기, iterator)• null 가능 형식(nullable type)• Static 클래스

02. C# 프로그래밍의 역사

버전	발표	특징
3.0	2006년	<ul style="list-style-type: none">• 암시적으로 형식화된 변수(implicitly typed local variables)• 개체 이니셜라이저(object initializer)• 컬렉션 이니셜라이저(collection initializer)• 무명 형식(anonymous types, 익명 형식)• 확장 메서드(extension methods)• 람다 식(lambda expression)• 자동 구현 속성(auto-implemented properties)• 쿼리 식(query expressions)• 익스프레션 트리(expression trees)
3.5	2007년	LINQ(Language INtegrated Query)
4.0	2010년	<ul style="list-style-type: none">• 다이나믹 바인딩(dynamic binding)• 명명된 또는 선택적 인수(named & optional arguments)
4.5	2012년	

02. C# 프로그래밍의 역사

버전	발표	특징
5.0	2013년	<ul style="list-style-type: none">• 비동기(async와 await)• 비동기 메서드(asynchronous methods)
6.0	2014년	<ul style="list-style-type: none">• 문자열 보간법(string interpolation)• 정적 멤버를 위한 using static 구문• 자동 속성 이니셜라이저(auto-property initializers)• null 조건부 연산자(null-conditional operator)• 식 본문 멤버(expression-bodied members)• nameof 연산자
7.0	2016년	<ul style="list-style-type: none">• 튜플(tuple)과 튜플 해체(deconstruction)• 패턴 매칭(pattern matching)• 숫자 구분자(digit separator)와 이진 리터럴(binary literals)• 로컬 함수(local functions)• out 키워드 기능 향상(out var)
8.0	2019년	<ul style="list-style-type: none">• nullable 참조 형식• 비동기 스트림

03. 관련 키워드

- **코드(code):**

텍스트로 된 소프트웨어를 만드는 명령 집합으로, 소스(source)

- **코딩(coding):**

프로그래밍 언어의 코드로 프로그램을 만드는 과정

코딩은 컴퓨터 프로그래밍과 개념이 비슷함

- **컴파일(compile):**

프로그램 소스 코드를 컴퓨터 등 하드웨어가 실행할 수 있는 기계 코드로 변환하는 프로그램을 컴파일러(compiler)라고 하며 변환하는 과정을 컴파일이라고 함

프로그램 소스 코드를 기계 코드로 실행 C, C++ 등이 컴파일 언어에 해당함

- **인터프리터(interpreter):**

따로 컴파일 과정을 거치지 않고 소스 코드를 바로 해석해서 실행하는 것
소스 코드를 인터프리터에서 실행






자바스크립트, 파이썬, PHP 등이 인터프리터 언어에 해당함

04. C VS C++ VS C#

- C => 모든 언어의 기본
- C++ => C + 객체지향
- C#
=> C, C++, JAVA의 장점을 모아 MS사에서 만든 언어

05. 프로그래밍 언어 순위

- 티오베(TIOBE) : <https://www.tiobe.com/tiobe-index>

Mar 2023	Mar 2022	Change	Programming Language		Ratings	Change
1	1			Python	14.83%	+0.57%
2	2			C	14.73%	+1.67%
3	3			Java	13.56%	+2.37%
4	4			C++	13.29%	+4.64%
5	5			C#	7.17%	+1.25%

C# 개발 환경

Update 2023

A solid green horizontal bar at the bottom of the slide.

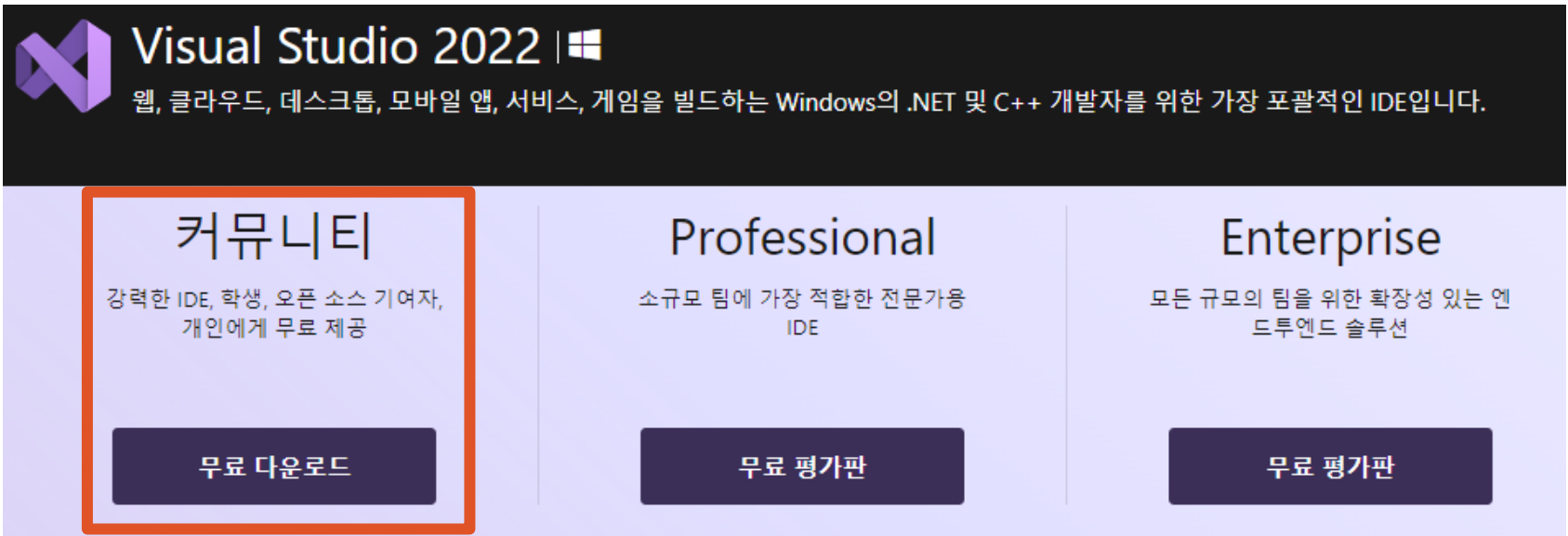
01. 비주얼 스튜디오

- 마이크로소프트에서 만든 통합 개발 환경
- C#, ASP.NET, C, C++, Node.js, 파이썬, 자바스크립트, 타입스크립트 등 주요 프로그래밍 개발 환경 제공
- 종류 : 커뮤니티용, 프로페셔널용, 엔터프라이즈용

02. 비주얼 스튜디오 설치

비주얼 스튜디오 커뮤니티 다운로드 및 설치

<https://www.visualstudio.com/downloads>



The image shows the Visual Studio 2022 download page. At the top, there is a dark header with the Visual Studio logo and the text 'Visual Studio 2022 | Windows'. Below the header, there is a description of the IDE. The main content area is divided into three columns: '커뮤니티' (Community), 'Professional', and 'Enterprise'. The '커뮤니티' column is highlighted with a red border and contains a '무료 다운로드' (Free Download) button. The 'Professional' and 'Enterprise' columns contain '무료 평가판' (Free Trial) buttons.

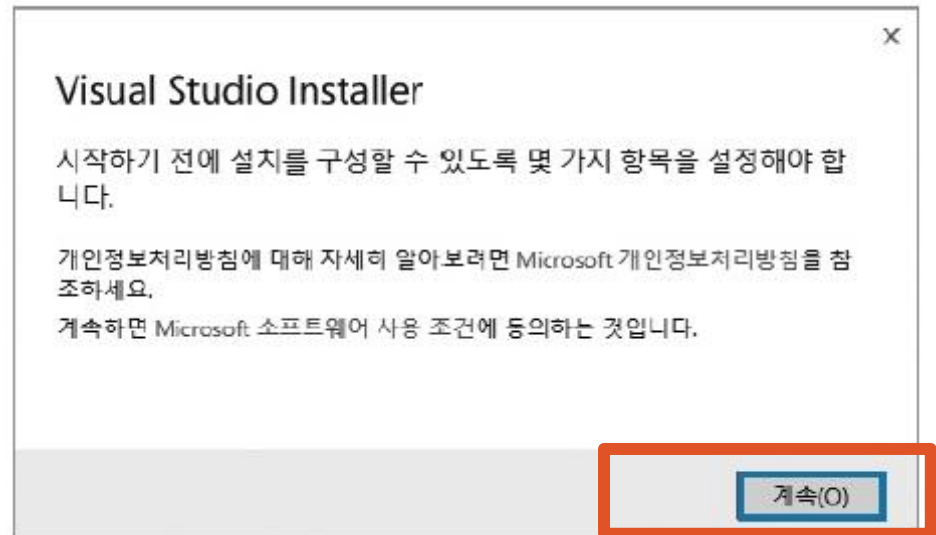
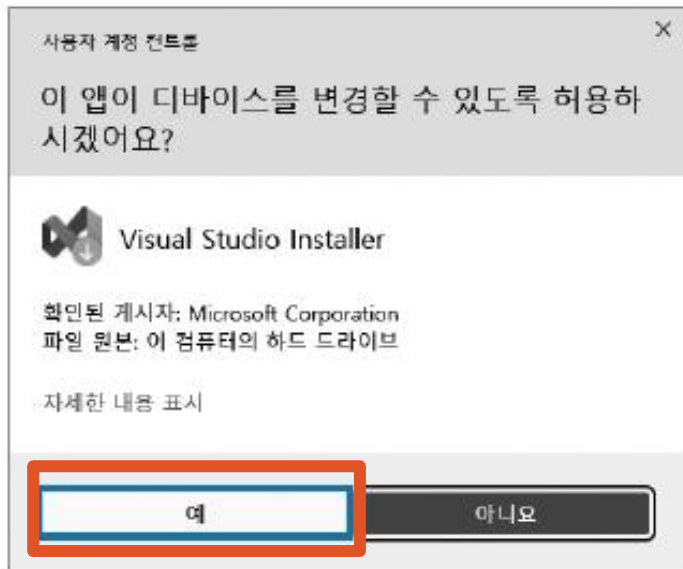
Visual Studio 2022 | Windows

웹, 클라우드, 데스크톱, 모바일 앱, 서비스, 게임을 빌드하는 Windows의 .NET 및 C++ 개발자를 위한 가장 포괄적인 IDE입니다.

커뮤니티	Professional	Enterprise
강력한 IDE, 학생, 오픈 소스 기여자, 개인에게 무료 제공	소규모 팀에 가장 적합한 전문가용 IDE	모든 규모의 팀을 위한 확장성 있는 엔드루엔드 솔루션
무료 다운로드	무료 평가판	무료 평가판

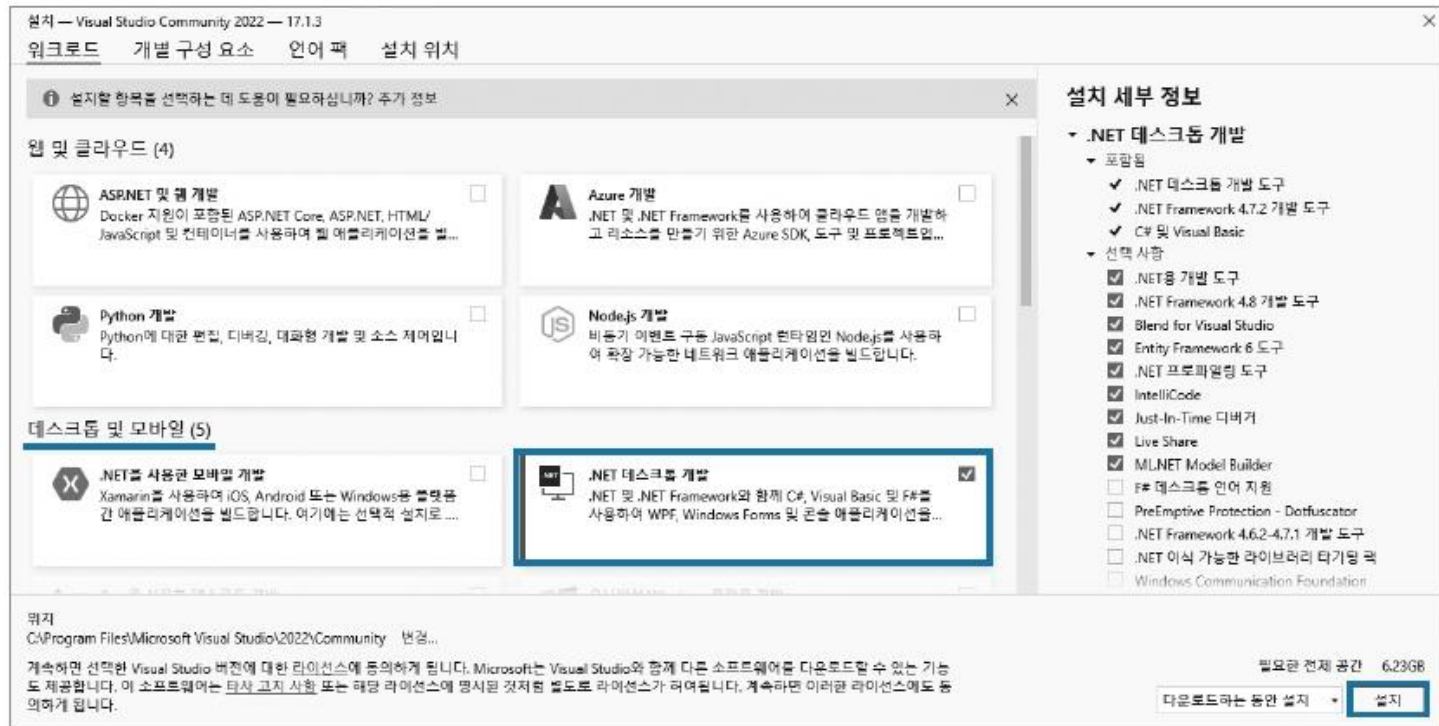
02. 비주얼 스튜디오 설치

[사용자 계정 컨트롤] 메시지 창이 나오면 <예>를 클릭한다.



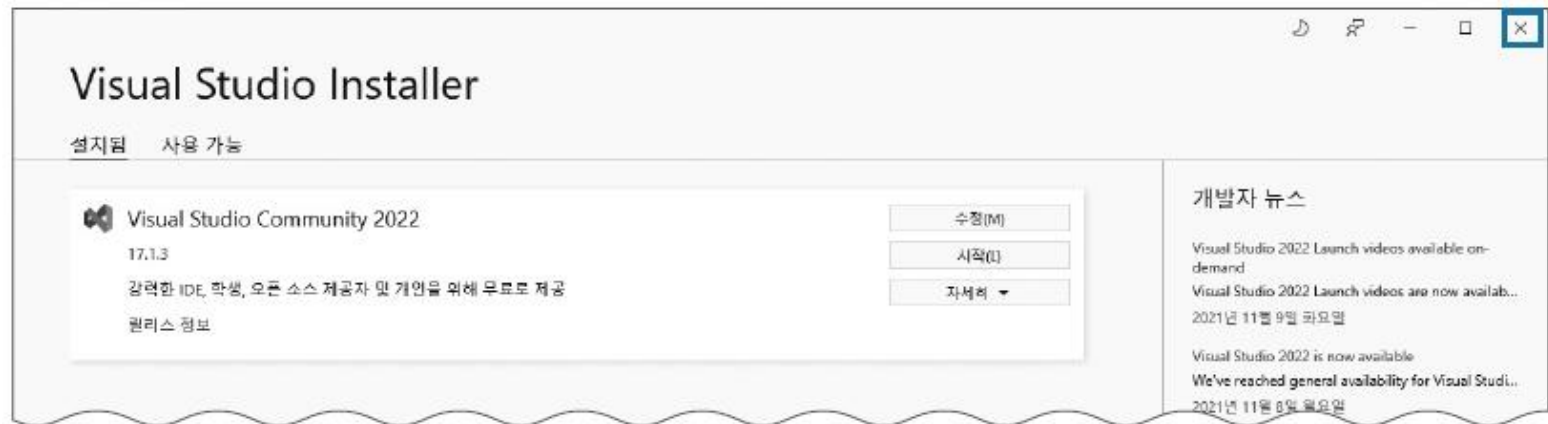
02. 비주얼 스튜디오 설치

[워크로드] 창이 뜨면 [데스크톱 및 모바일] 아래의
[.NET 데스크톱 개발]만 체크하고 [설치]를 클릭한다.



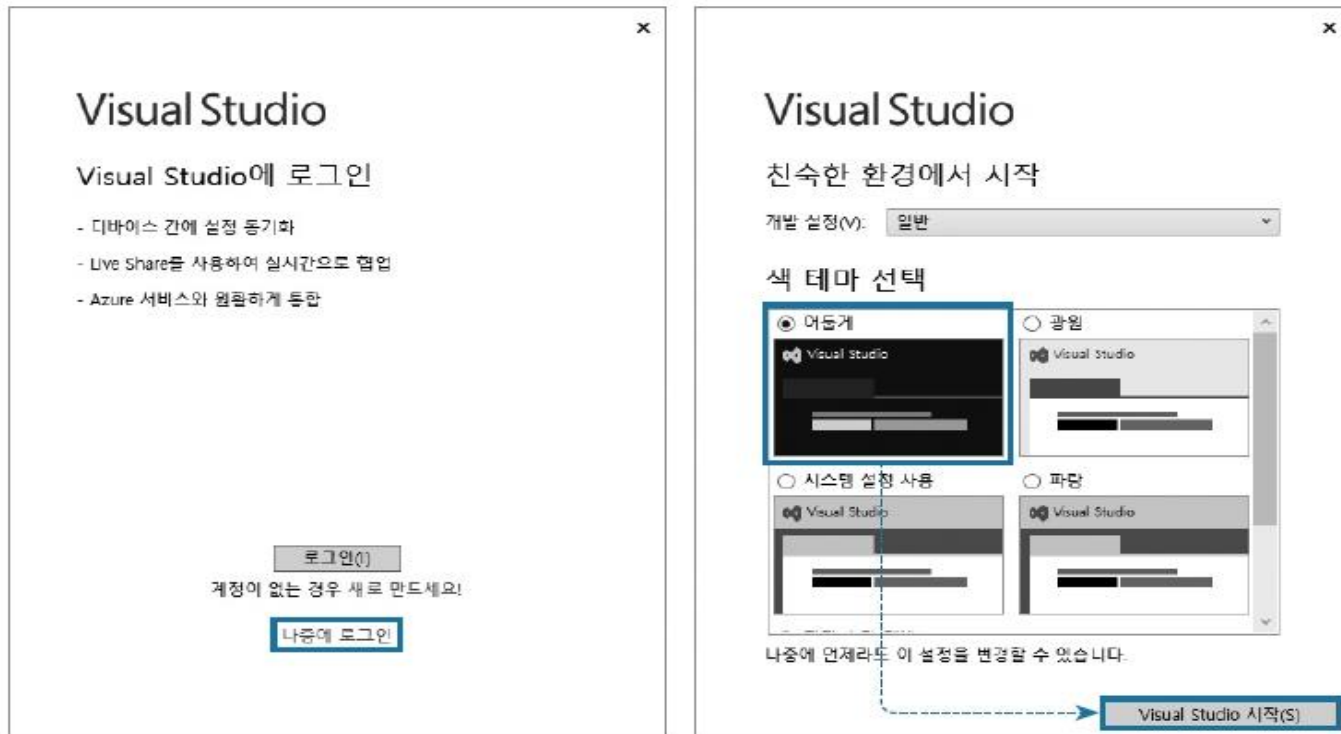
02. 비주얼 스튜디오 설치

설치가 완료되면 자동으로 Visual Studio가 실행된다. 설치가 완료된 [Visual Studio Installer] 창은 오른쪽 상단의 <X>를 클릭해서 닫는다.



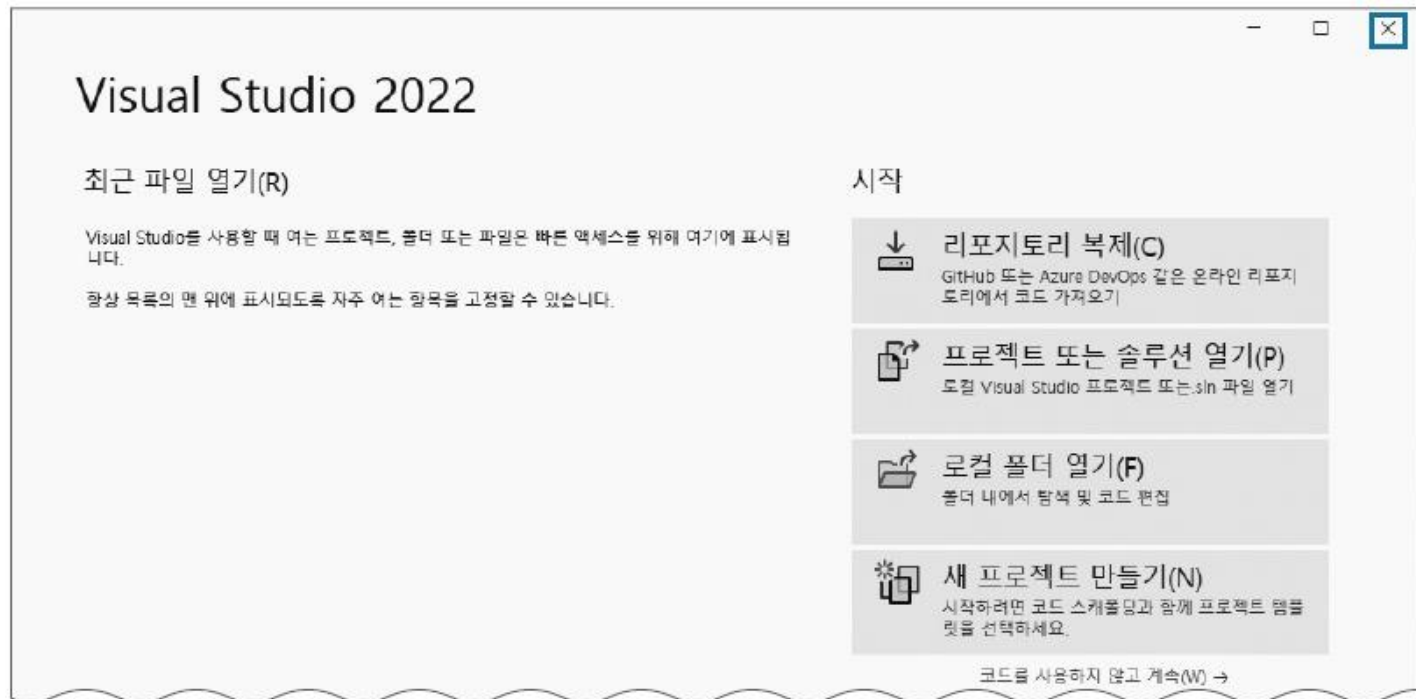
02. 비주얼 스튜디오 설치

로그인 창이 나오면 계정이 없다면 [나중에 로그인]을 클릭한다.
화면 환경 테마 선택하고 <Visual Studio 시작>을 클릭한다.



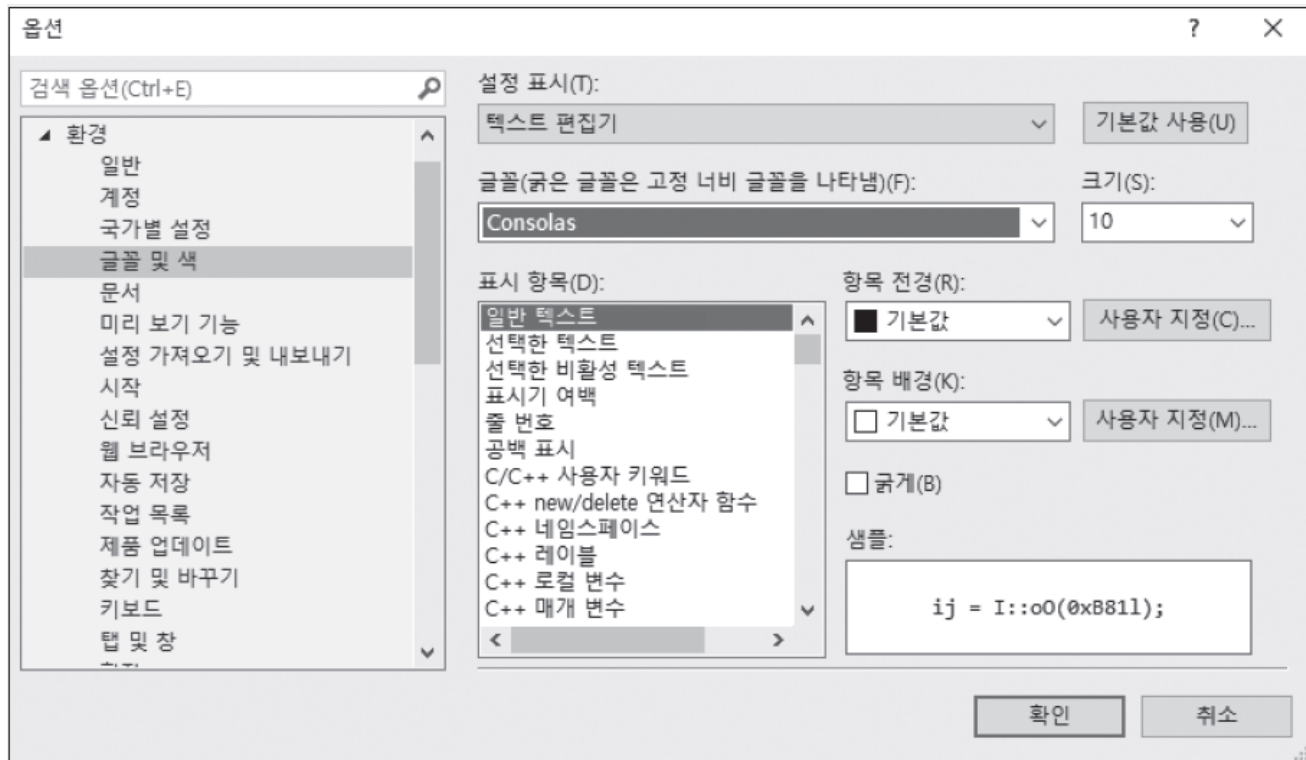
03. 비주얼 스튜디오 시작

Visual Studio [시작 페이지]가 나오면 오른쪽 위 <X>를 눌러 창을 닫는다



04. 비주얼 스튜디오 환경 구축

- 도구 > 옵션 > 환경 > 글꼴 및 색 > 글꼴에서 글꼴 스타일 변경



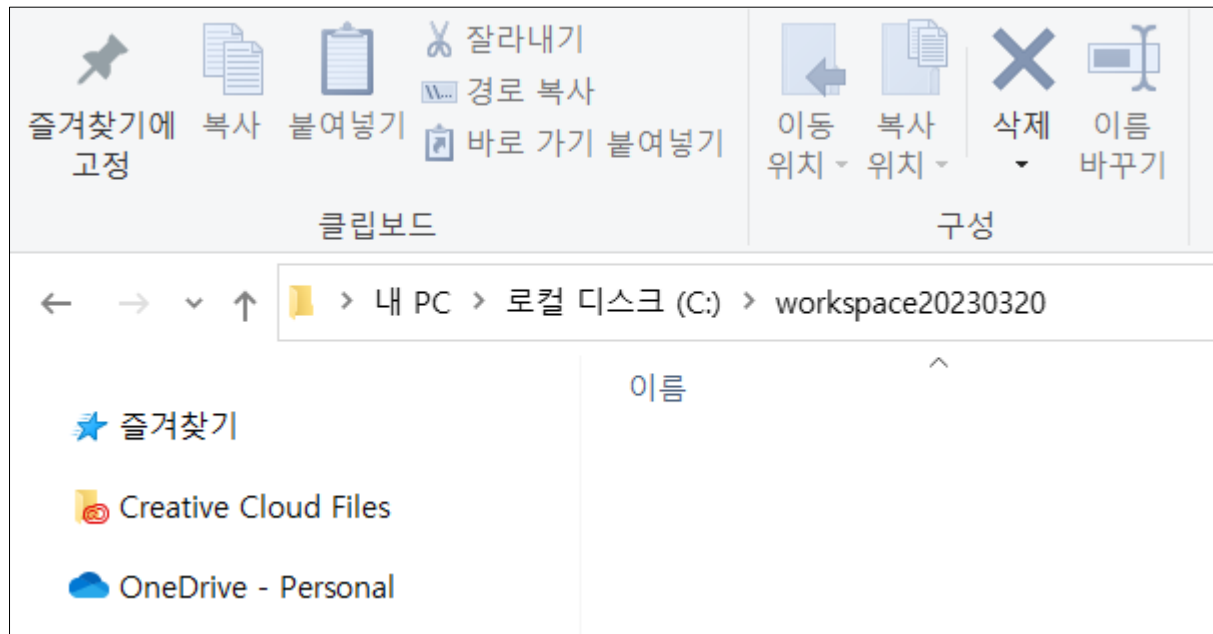
C# 프로그램 작성 및 실행

Update 2023

A solid green horizontal bar at the bottom of the slide.

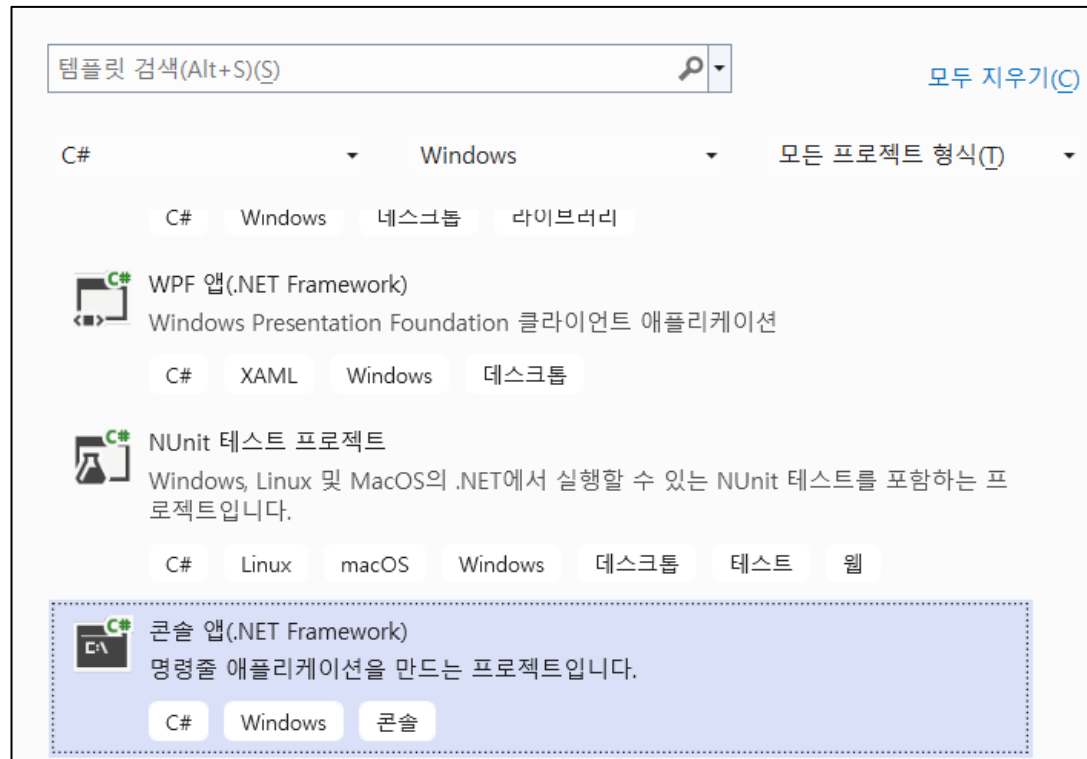
01. 프로젝트 생성

윈도우 탐색기를 실행한 후 프로젝트가 생성될 작업 폴더를 생성한다



01. 프로젝트 생성

[파일]-[새로만들기]-[새 프로젝트 만들기] 메뉴 클릭 후 [새 프로젝트 만들기] 창에서 언어를 [C#]으로 선택한 후 [콘솔 앱(.NET Framework)]을 다시 선택 한다. <다음>을 클릭한다.



01. 프로젝트 생성

[새 프로젝트 구성] 에서 프로젝트 이름, 위치 지정 후
<솔루션 및 프로젝트를 같은 디렉터리에 배치>를 체크한다.
[만들기] 클릭

새 프로젝트 구성

콘솔 앱(.NET Framework) C# Windows 콘솔

프로젝트 이름(I)
HelloWorld

위치(L)
C:\workspace20230320 ...

솔루션(S)
새 솔루션 만들기

솔루션 이름(M) ⓘ
HelloWorld

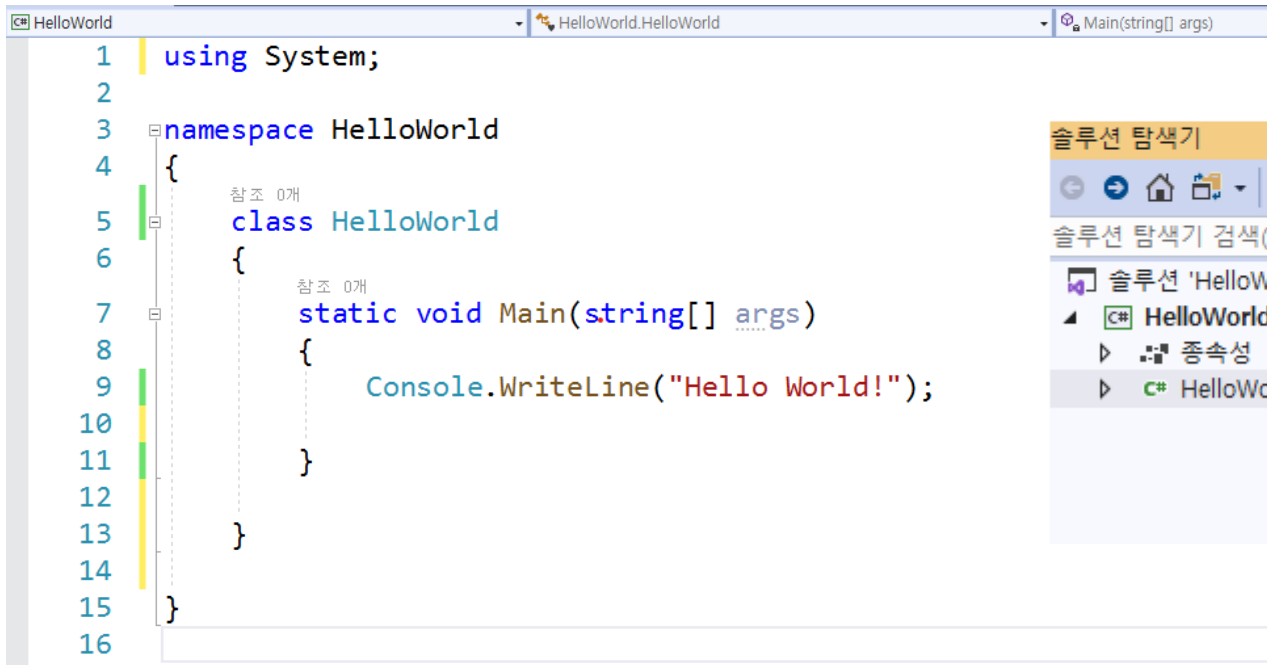
☒ 솔루션 및 프로젝트를 같은 디렉터리에 배치(D)

프레임워크(F)
.NET Framework 4.8

"C:\workspace20230320\HelloWorld"에 프로젝트이(가) 만들어집니다.

02. HelloWorld

[솔루션 탐색기]에서 Program.cs 마우스 우측 버튼 => 이름 변경 => HelloWorld.cs로 변경



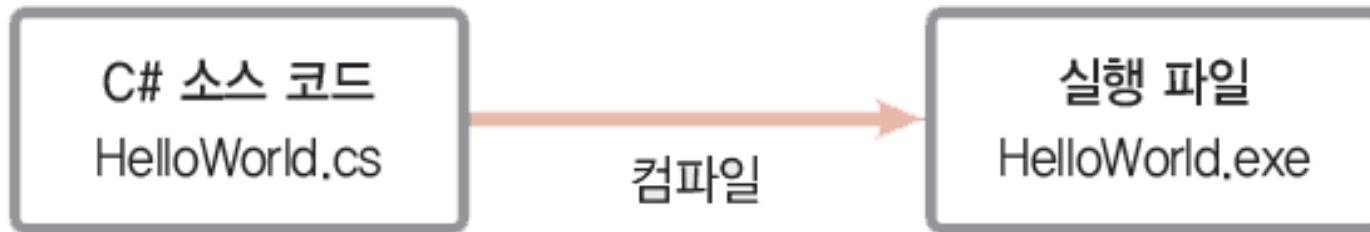
```
1  using System;
2
3  namespace HelloWorld
4  {
5      class HelloWorld
6      {
7          static void Main(string[] args)
8          {
9              Console.WriteLine("Hello World!");
10         }
11     }
12 }
13
14
15
16
```

The screenshot shows the Visual Studio IDE. The top toolbar includes icons for navigation and development. The Solution Explorer on the right shows the project structure: 'HelloWorld' (1/1개 프로젝트) containing 'HelloWorld' (C#) with sub-items '종속성' and 'HelloWorld.cs'. The code editor displays the contents of HelloWorld.cs, which includes a namespace declaration, a class declaration, and a static Main method that writes 'Hello World!' to the console.

03. 실행 1

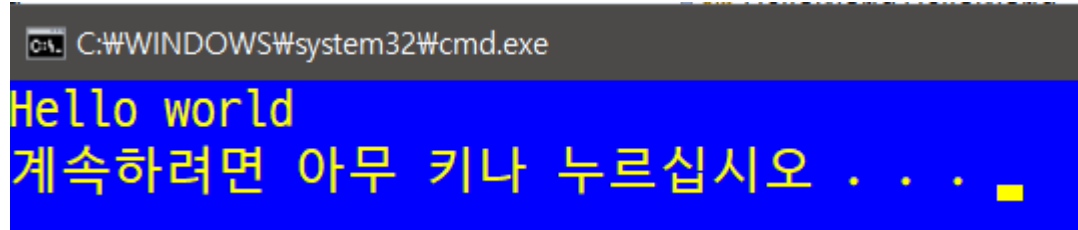
// 소스 코드 컴파일

C# 파일은 HelloWorld.cs처럼 확장자가 cs 인데, 컴파일 과정을 거치면 실행 가능한 exe 파일이 생성된다



03. 실행 1

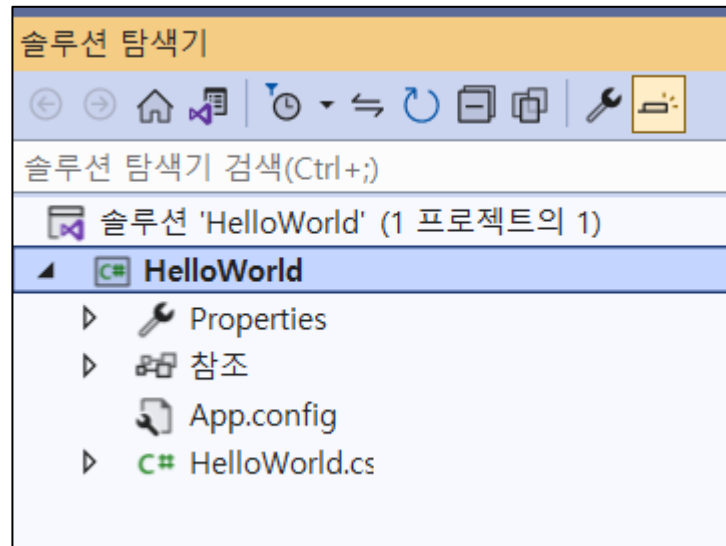
- [빌드]-[솔루션 빌드] (단축키 Ctrl+Shift+B 또는 F6)
- [디버그]-[디버그 시작]
- [디버그]-[디버그 하지 않고 시작] (단축키 Ctrl+F5)



```
C:\WINDOWS\system32\cmd.exe
Hello world
계속하려면 아무 키나 누르십시오 . . .
```

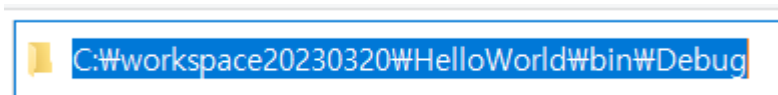
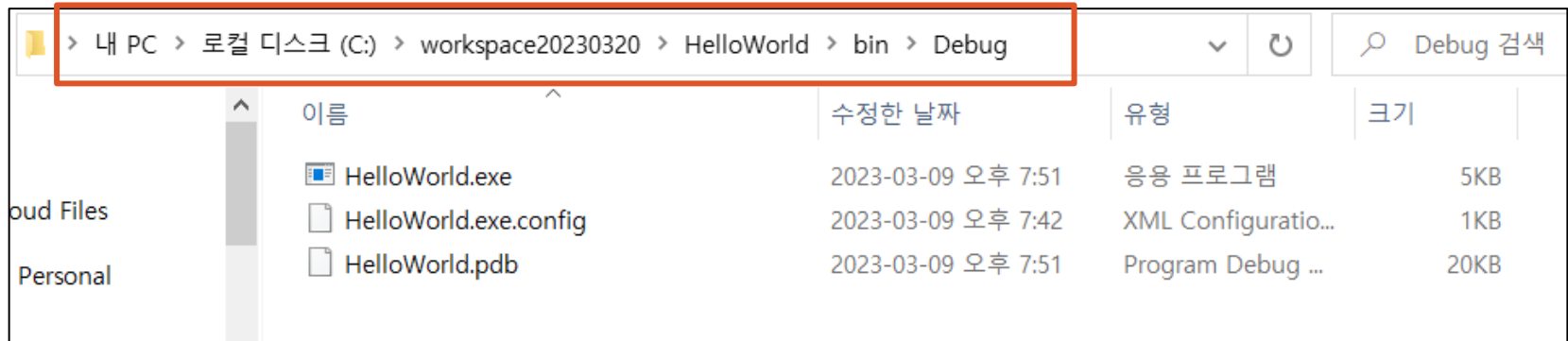
04. 실행 2

- [솔루션 탐색기] 창에서 솔루션 마우스 우측 버튼 => [파일 탐색기에서 폴더 열기]



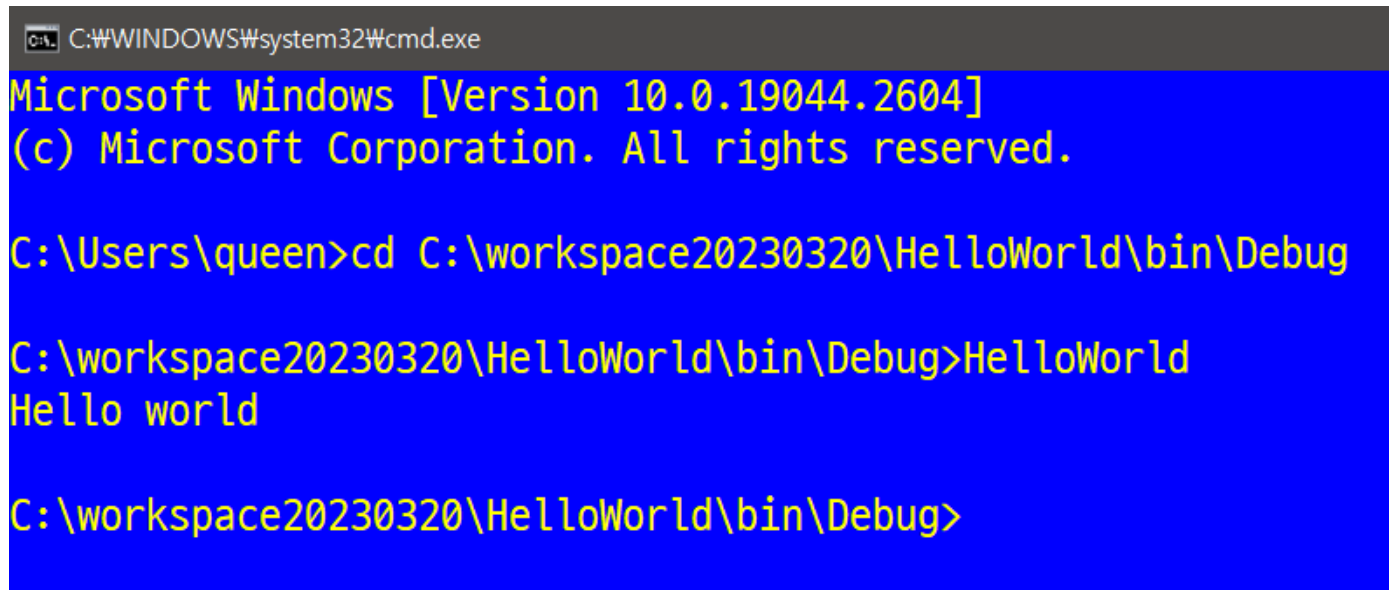
04. 실행 2

- 'bin\Debug\netcoreapp3.1' 폴더 차례로 클릭하여 경로를 변경한 후 노란색 폴더에 표시된 경로 뒷부분 클릭 후 [Ctrl]+[C] 경로 복사



04. 실행 2

- [윈도우] 키 + [R] 키 => cmd => 터미널 창
- cd 입력 후 [Ctrl]+[V] Helloworld.exe 파일이 있는 경로로 이동(다른 드라이브의 경우 드라이브 명 입력 후 진행)
- HelloWorld 입력 후 [Enter]



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\queen>cd C:\workspace20230320\HelloWorld\bin\Debug

C:\workspace20230320\HelloWorld\bin\Debug>HelloWorld
Hello world

C:\workspace20230320\HelloWorld\bin\Debug>
```

05. C# 인터렉티브

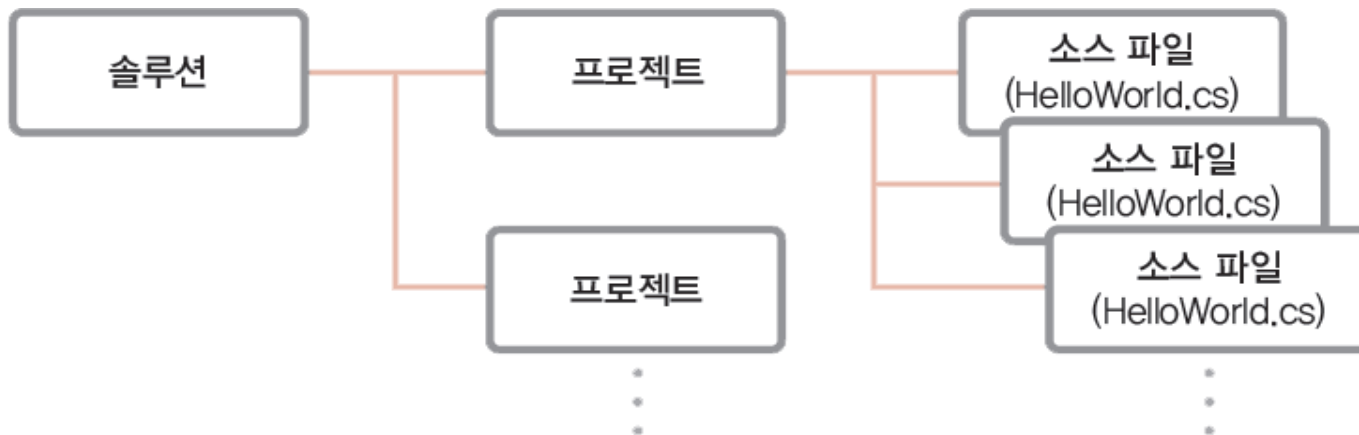
[보기]-[다른창]-[C# 대화형]

- C# 인터렉티브(interactive)(대화형) : 한 줄씩 코드를 실행하면서 C#의 여러 명령을 학습할 수 있는 도구

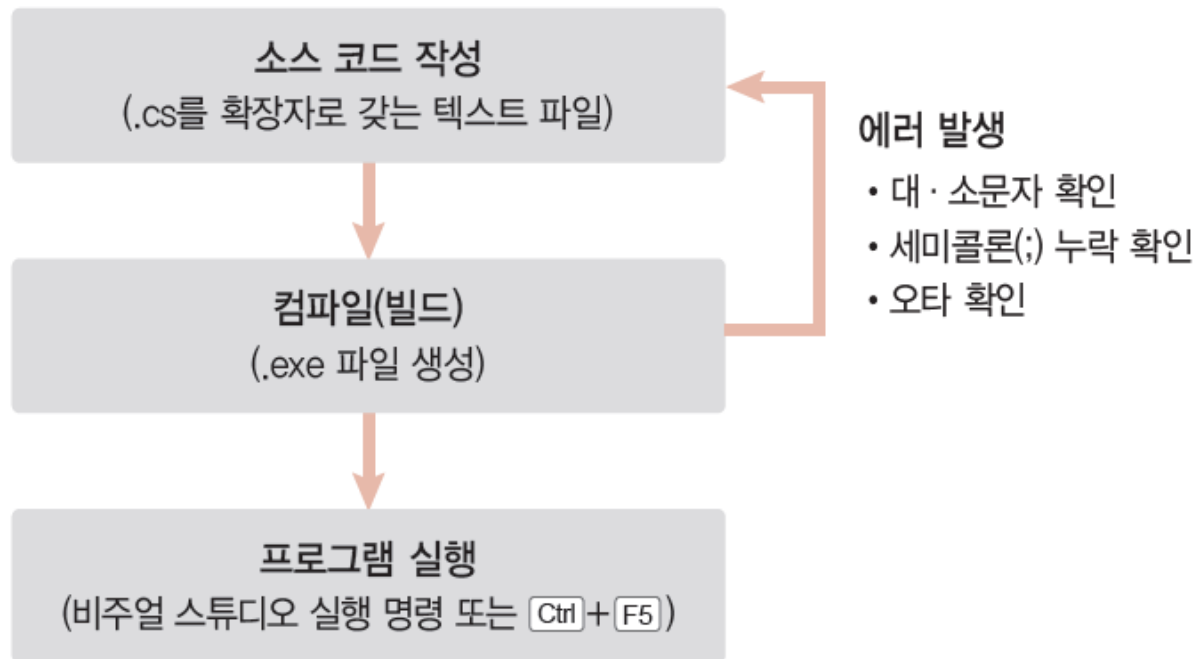
```
C# Interactive
↺ ⌵ ⬆ ⬇
Microsoft (R) Visual C# 대화형 컴파일러 버전 4.5.0-6.23109.5 ()
'CSharpInteractive.rsp'에서 컨텍스트를 로드하는 중입니다.
자세한 내용을 보려면 "#help"를 입력하세요.
> Console.WriteLine("Hello C#!!!");
Hello C#!!!
> Console.WriteLine("Hello world");
Hello world
> |
```

06. 솔루션 VS 프로젝트

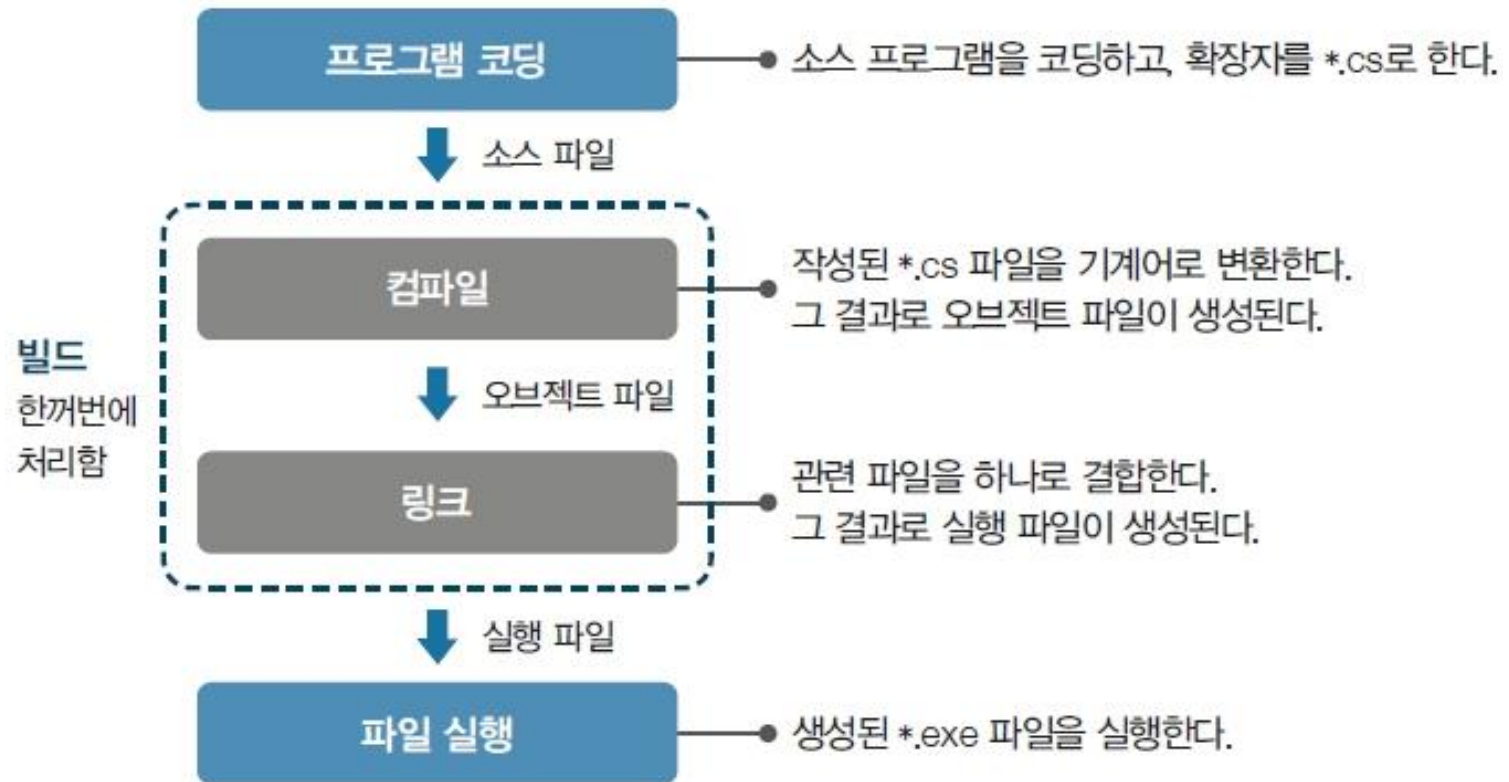
- **프로젝트(project):**
프로그램 하나를 이루는 가장 작은 단위가 되는 프로그램을 의미
- **솔루션(solution):**
하나 이상의 프로젝트를 모아서 만든 프로그램. 확장자는 sln (솔루션의 약자)



07. C# 프로그램 작성 및 실행단계



07. C# 프로그램 작성 및 실행단계



C# 문법 특징

Update 2023

A solid green horizontal bar at the bottom of the slide.

01. 기본 코드 구조

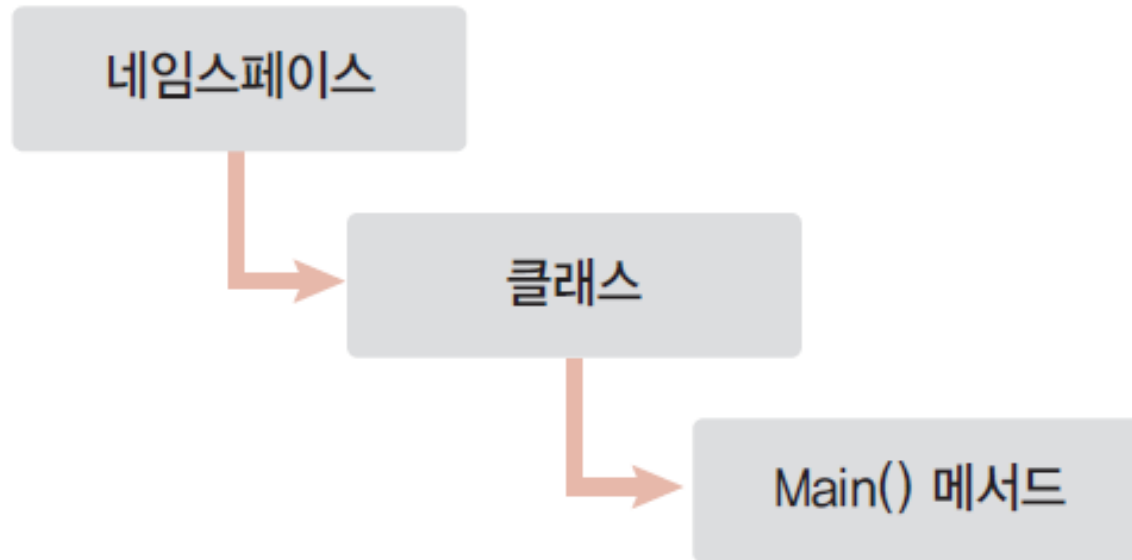
```
using System;

// 네임스페이스 선언부
namespace HelloWorld
{
    // 클래스 선언부
    참조 0개
    class HelloWorld
    {
        // 메인 메서드 선언부
        참조 0개
        static void Main(string[] args)
        {
            // 출력문
            Console.WriteLine("Hello World!");
        }
    }
}
```

- C# 프로그램은 class와 Main() 메서드가 반드시 있어야 하고, 하나 이상의 명령문이 있어야 한다.
- 명령문은 대소문자를 구별하여 세미콜론(;) 으로 종료한다.

- 네임스페이스 : 성격이나 일이 비슷한 클래스, 구조체, 인터페이스, 대리자등을 하나의 이름아래 묶는 기능

01. 기본 코드 구조



02. Main() 메소드 코드 조각

• **svm** + Tab Tab 입력

```
static void Main(string[] args)
{

}
```

• **cw** + Tab Tab 입력

```
Console.WriteLine();
```

- 비주얼 스튜디오에서 svm을 입력한 후 Tab 을 두 번 누르면 자동으로 `static void Main() {}` 코드 블록을 작성함
- `Console.WriteLine()` 메서드도 워낙 많이 사용하므로 cw를 입력한 후 Tab 을 두 번 누르면 자동으로 `Console.WriteLine();` 코드를 입력함

03. 중괄호 스타일

- 중괄호의 시작과 끝을 맞추는 형태인 Allman 스타일과 줄의 맨 마지막에 시작 중괄호를 넣는 K&R 스타일을 가장 많이 사용함

//Allman 스타일: 시작과 끝을 맞추는 형태

```
class BraceLocation
```

```
{
```

```
    static void Main()
```

```
    {
```

```
    }
```

```
}
```

//K&R 스타일: 줄의 맨 마지막에 시작 중괄호를 넣는 형태

```
class BraceLocation {
```

```
    static void Main() {
```

```
    }
```

```
}
```

C# 주식과 출력문

Update 2023

A solid green horizontal bar at the bottom of the slide.

01. 주석

- 주석문(comment)이란?

실행에 영향을 주지 않는 코드 설명문으로 프로그램 설명이나 작성자, 작성일 등 필요한 내용을 기록하는 용도로 사용된다.

- 한 줄 주석 : // 주석문
- 여러줄 주석

```
/*
```

```
    주석문1
```

```
    주석문2
```

```
*/
```


02. 출력문

// using 키워드 O

using System;

Console.WriteLine("줄바꿈 있음");

Console.Write("줄바꿈 없음1 ");

Console.Write("줄바꿈 없음2 ");

/*

줄바꿈 있음

줄바꿈 없음1 줄바꿈 없음2

*/

// using 키워드 X

System.Console.WriteLine("줄바꿈 있음");

System.Console.Write("줄바꿈 없음1 ");

System.Console.Write("줄바꿈 없음2 ");

/*

줄바꿈 있음

줄바꿈 없음1 줄바꿈 없음2

*/

03. 이스케이프(Escape)

- 이스케이프 문자란?
역슬래시(\) 기호와 특정 문자를 조합하여 특별한 기능을 제공하는 문자

종류	설명
\n	한 줄 내리기(다음 행으로 이동), newline
\t	Tab 들여쓰기(Tab 크기만큼 들여쓰기), tab
\r	캐리지 리턴(줄의 시작으로 이동), carriage-return
\'	작은따옴표 문자 하나 출력
\"	큰따옴표 문자 하나 출력

03. 이스케이프(Escape)


```
Console.WriteLine("\n\n철수 :\t' 영희야! 같이가자 \'");
```

```
/*  
*  
철수 :   '영희야! 같이가자'  
*/
```


04. 자리표시자 { index }

- 출력문안에 {0}, {1} 식으로 뒤에 이어 나올 값에 대한 자리 번호(인덱스)를 지정해 놓는 방법을 자리 표시자(place holder) 또는 서식 지정자(format specifier)라고 한다. => 순번 및 서식
- 인덱스 번호는 0 부터 시작한다.

```
Console.WriteLine("{0}, {1}", "Hello", "C#");
```



```
Console.WriteLine("{1}, {0}", "Hello", "C#");
```



04. 자리표시자 { index }

```
Console.WriteLine("C#1");
Console.WriteLine("C#2");
Console.WriteLine("C#3");
Console.WriteLine("C#4");

Console.WriteLine("\n\n");
Console.Write("아름다운 우리나라");
Console.Write("아름다운 우리나라");
Console.Write("아름다운 우리나라");
Console.Write("아름다운 우리나라");
Console.WriteLine();
Console.WriteLine("\n\t\t우리 강아지");
```

04. 자리표시자 { index }

```
Console.WriteLine("1위 {0}, 2위 {1}, 3위 {2}", "C", "JAVA", "PYTHON");  
Console.WriteLine("1위 {1}, 2위 {2}, 3위 {0}", "C", "JAVA", "PYTHON");
```

```
1위 C, 2위 JAVA, 3위 PYTHON  
1위 JAVA, 2위 PYTHON, 3위 C  
계속하려면 아무 키나 누르십시오 . . . ■
```

퀴즈

- 아래와 같은 결과 화면이 출력되도록 프로그래밍 하여라.

출력 화면

Hello World의 유래는?

C 와 UNIX를 개발한
브라이언 커니핸과 데니스 리치가
쓴 THE C Program Language 교재에
첫 예제가 Hello World! 출력 이었다.

해당 교재가 유명해지면서
모든 프로그래밍 첫 예제가
Hello World 출력으로 시작하게 되었다.

퀴즈

- 아래와 같은 결과 화면이 출력되도록 프로그래밍 하시오.
- `Console.Write` 출력문은 1개만 사용하고 이스케이프문자를 이용한다.

출력 화면

```
*  
* *  
* * *  
* * * *  
* * * * *
```


퀴즈

- 자리표시자 {}을 이용하여 아래와 같은 결과 화면이 출력되도록 프로그래밍 하시오.

출력 화면

2019

JAVA, C, Python, C++, C#

2050

Python, C#, C, C++, JAVA,

변수와 데이터형

Update 2023.03

A solid green horizontal bar at the bottom of the slide.

01. 변수

- 변수(Variable)란?

데이터를 메모리에 잠시 보관해 놓고 사용할 수 있는 임시 저장 공간

- 변수를 만들 때는 데이터 형식, 변수 이름, 값 등이 필요하다



02. 변수명

- 대소문자 구분.
- 문자, 숫자, 언더바(_)를 포함할 수 있음.
- 숫자로 변수명을 시작하면 안됨.
- 예약어는 변수명으로 사용할 수 없음.

```
bool, break, byte, case, catch, char, continue, default, do, else, false, float, for, if,  
int, null, short, sizeof, switch, try, void, while ...
```

03. 변수 선언1

- 데이터 형식 + 변수명

데이터 형식 문장의 끝

↓ ↓

int number;

 ↑

 변수 이름

메모리

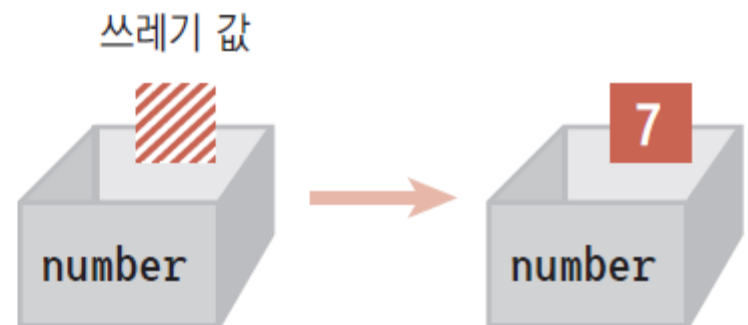
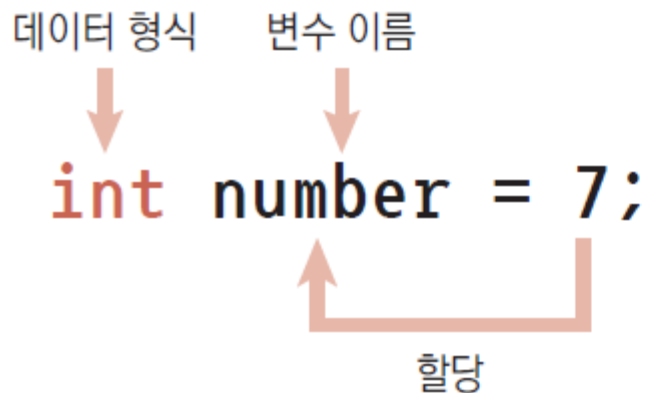
int number; →

The diagram illustrates the memory layout for the variable 'number'. It shows a vertical stack of three rectangular boxes representing memory cells. The middle box is highlighted with a thick red border, indicating it is the memory location allocated for the variable. An arrow points from the text 'int number;' to this highlighted box.

04. 변수 선언2

- 변수 선언과 동시에 초기값 지정하기

변수를 선언하면 변수 이름으로 메모리 공간이 만들어지는데, 처음에는 쓰레기 값이 저장되어 있으며 실제 사용할 값을 저장하는 것을 초기화라고 한다.



05. 변수 선언3

- 형식이 같은 변수 여러 개를 한 번에 선언하기

데이터 형식이 같은 변수를 콤마(,) 기호로 구분해서 여러 개 선언한다.

선언하려는 변수의 데이터 형식이 다르다면 따로 선언해야한다.

```
int a;
```

```
int b;    =    int a, b, c;
```

```
int c;
```

06. 변수 선언4

- 형식이 같은 변수 여러 개를 선언하고 동일한 값으로 초기화
형식이 같은 변수 여러 개를 동일한 값으로 초기화할 때는 좀 더 편하게
= 기호를 사용하여 다음 코드처럼 `a = b = c = 10;` 형태로 초기화할 수
있다.

```
int a, b, c;  
a = b = c = 10;
```


07. 리터럴

- 리터럴(literal) 이란?
변수에는 직접 정수형 또는 문자열 값을 저장할 수 있는데, 이 값을 자체를 리터럴이라고 한다.
- 리터럴 중에는 값 자체를 가지지 않는 널(null) 리터럴도 있다.
- 정수 리터럴, 실수 리터럴, 문자열 리터럴 등을 저장하여 사용할 수 있으며 정수는 숫자 그대로 표현하고, 실수는 대문자 F 또는 소문자 f를 접미사를 이용한다.
- 문자는 작은 따옴표로 묶어야 하고 문자열은 큰따옴표로 묶어야 한다.

```
Console.WriteLine(1234);    //정수 리터럴
Console.WriteLine(3.14F);    //실수 리터럴
Console.WriteLine('A');      //문자 리터럴
Console.WriteLine("HELLO");  //문자열 리터럴
```

08. 데이터 형식

데이터 형식	설명	크기	담을 수 있는 값의 범위
byte	부호 없는 정수	1 (8bit)	0 ~ 255
sbyte	signed byte 정수	1 (8bit)	-128 ~ 127
short	정수	2 (16bit)	-32,768 ~ 32,767
ushort	unsigned short 부호 없는 정수	2 (16bit)	0 ~ 65535
int	정수	4 (32bit)	-2,147,483,648 ~ 2,147,483,647
uint	unsigned int 부호 없는 정수	4 (32bit)	0 ~ 4294967295
long	정수	8 (64bit)	-922337203685477508 ~ 922337203685477507
ulong	unsigned long 부호 없는 정수	8 (64bit)	0 ~ 18446744073709551615

08. 데이터 형식

데이터 형식	설명	크기(byte)	범위
float	단일 정밀도 부동 소수 점 형식 (7개의 자릿수만 다를 수 있음.)	4 (32bit)	-3.402823e38 ~ 3.402823e38
double	복수 정밀도 부동 소수 점 형식 (15~16개의 자릿수를 다룰 수 있음.)	8 (64bit)	-1.79769313486232e308 ~ 1. 79769313486232e308

08. 데이터 형식

데이터 형식	설명	크기(byte)	범위
bool	논리 형식	1 (8bit)	true, false
char	유니코드 문자. 1글자 문자, 작은 따옴표 이용		
string	문자열. 큰 따옴표 이용		

```
Console.WriteLine('가');
```

```
Console.WriteLine("Hello C#");
```

```
Console.WriteLine(true);
```

GetType()

Update 2023

A solid green horizontal bar at the bottom of the slide.

GetType() 메소드

- GetType() 메소드는 정의된 변수의 데이터 형식을 알아볼 때 사용한다.
- 변수.GetType()
- 반환값은 닷넷형식의 데이터 형식으로 표시된다.

GetType() 메소드

```
int v1 = 100;
float v2 = 100f;
string v3 = "2002년 10월 31일";
bool v4 = true;

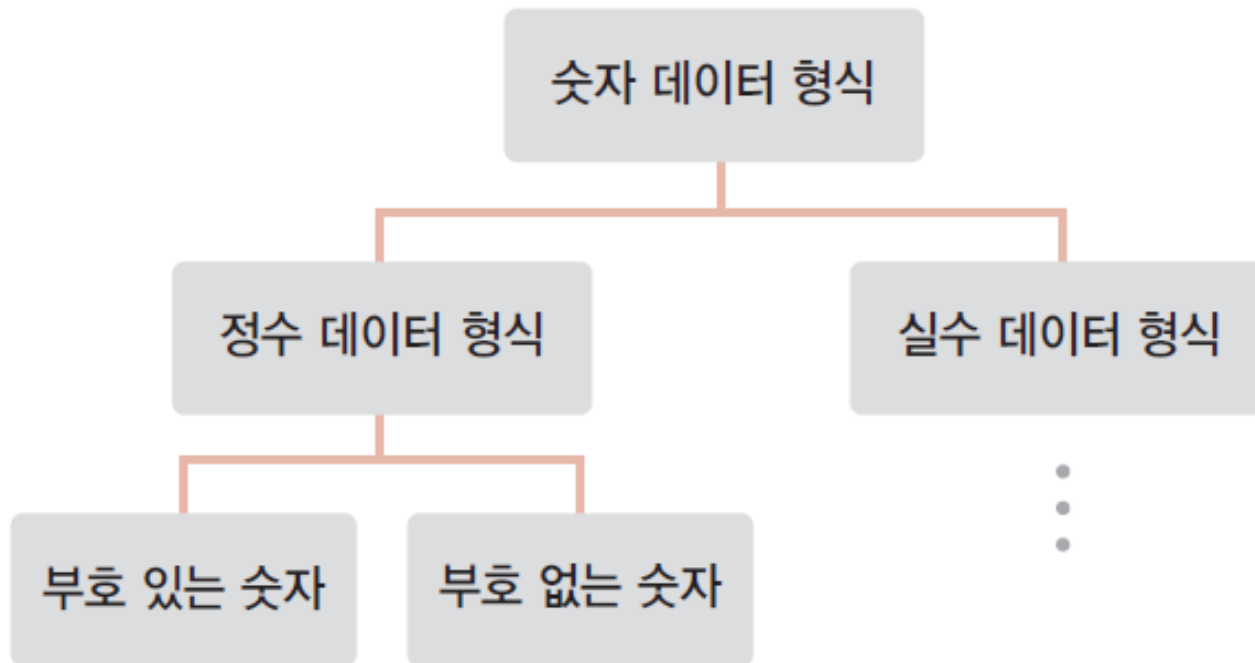
Console.WriteLine("\n\n\n데이터형 확인하기");
Console.WriteLine($"v1의 값은 {v1} 데이터형은 {v1.GetType()}");
Console.WriteLine($"v2의 값은 {v2} 데이터형은 {v2.GetType()}");
Console.WriteLine($"v3의 값은 {v3} 데이터형은 {v3.GetType()}");
Console.WriteLine($"v4의 값은 {v4} 데이터형은 {v4.GetType()}");
```

숫자 데이터 형식

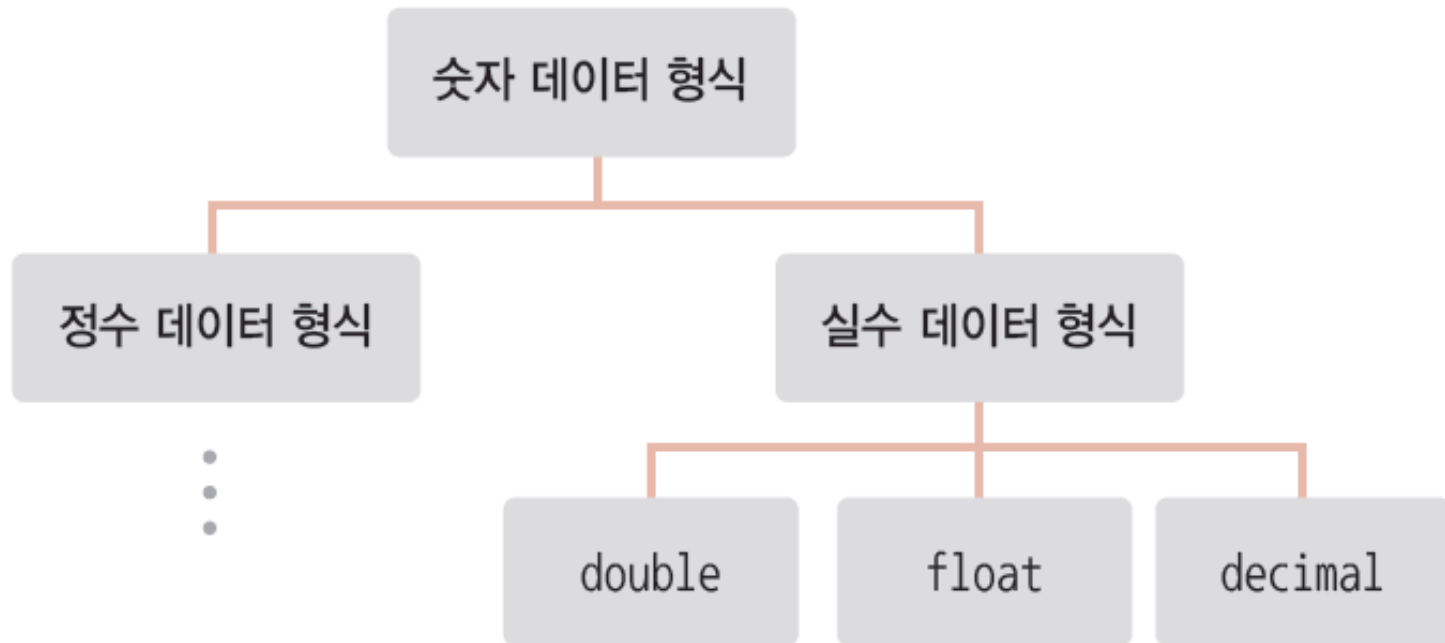
Update 2023

A solid green horizontal bar spanning the width of the slide at the bottom.

01. 숫자 데이터 형식 종류



01. 숫자 데이터 형식 종류



02. 정수 데이터 형식

종류	데이터 형식	닷넷 형식
부호 있는 정수(+, -)	sbyte	System.SByte
	short	System.Int16
	int	System.Int32
	long	System.Int64
부호 없는 정수(+)	byte	System.Byte
	ushort	System.UInt16
	uint	System.UInt32
	ulong	System.UInt64

- 접두사 s/u
s: signed
u : unsigned
- 부호 있는(signed)
: +, -. 양수와 음수를 모두 지원.
- 부호 없는(unsigned)
: 부호 없이 + 값만 다루는 정수형. 양수만 지원.

02. 정수 데이터 형식

데이터 형식	설명	크기	담을 수 있는 값의 범위
byte	부호 없는 정수	1 (8bit)	0 ~ 255
sbyte	signed byte 정수	1 (8bit)	-128 ~ 127
short	정수	2 (16bit)	-32,768 ~ 32,767
ushort	unsigned short 부호 없는 정수	2 (16bit)	0 ~ 65535
int	정수	4 (32bit)	-2,147,483,648 ~ 2,147,483,647
uint	unsigned int 부호 없는 정수	4 (32bit)	0 ~ 4294967295
long	정수	8 (64bit)	-922337203685477508 ~ 922337203685477507
ulong	unsigned long 부호 없는 정수	8 (64bit)	0 ~ 18446744073709551615

02. 정수 데이터 형식

- 각 정수 데이터 형식의 키워드는 그와 동일한 역할을 하는 닷넷(.NET) 형식을 제공하고 있다.
- 예) int 형식과 동일한 닷넷 데이터 형식은 System.Int32
- 변수를 선언할 때도 마찬가지로 int 대신 System.Int32를 사용할 수 있다.

int = System.Int32

02. 정수 데이터 형식

```
// int a;  
// 닷넷 형식의 정수 선언  
System.Int32 a;  
// long b;  
// 닷넷 형식의 long 정수 선언  
System.Int64 b;  
a = -100;  
b = -300;  
Console.WriteLine("a = {0}, b= {1}", a, b);  
  
// 변수의 데이터형 표시  
// 변수.GetType()  
Console.WriteLine(a.GetType()); // System.Int32  
Console.WriteLine(b.GetType()); // System.Int64
```

02. 정수 데이터 형식

```
// 부호가없는 정수 선언
// uint c;
System.UInt32 c;
// c = -500; 오류
c = 500;
Console.WriteLine(" c= {0}, c의 데이터형 = {1}", c, c.GetType());
```

03. 언더스코어(_) 문자 이용

- C# 7.0 버전부터는 언더스코어(_) 문자를 사용하는 숫자 구분자(digit separator)를 제공하여 세 자리마다 콤마로 구분되는 긴 숫자 형태를 표현할 수 있다.
- 숫자 구분자를 사용하면 숫자를 표시할 때 가독성이 높아진다.

```
// 숫자 구분자(digit separator)  
// 언더바(_)를 이용한 값 할당. 3자리마다 언더바 삽입  
// long d = 1000000;  
long d = 1_000_000;  
Console.WriteLine(" d= {0}, d의 데이터형 = {1}", d, d.GetType());
```


04. MinValue와 MaxValue

- 부호 있는 정수형 데이터 형식의 최솟값과 최댓값은 자료형.MinValue와 자료형.MaxValue 속성으로 출력할 수 있다.


```
// 데이터형에서 지원하는 최댓값과 최소값 속성
// 데이터형.MinValue , 데이터형.MaxValue
Console.WriteLine(" int의 최솟값 : {0}", int.MinValue);
Console.WriteLine(" int의 최댓값 : {0}", int.MaxValue);
Console.WriteLine(" long의 최솟값 : {0}", long.MinValue);
Console.WriteLine(" long의 최댓값 : {0}", long.MaxValue);
```

04. MinValue와 MaxValue

// 정수형 범위를 넘어서서 값을 할당한다면 ?
// 에러 => 오버플로우

```
int e1 = int.MinValue - 1;
```

```
int e2 = int.MaxValue + 1;
```

 int int.operator +(int left, int right)

CS0220: checked 모드에서 컴파일하면 작업이 오버플로됩니다.

```
int e1 = int.MaxValue;
```

```
long e2 = e1 + 1;
```

```
Console.WriteLine(e1);
```

```
Console.WriteLine(e2);
```

05. 실수형 데이터

- 실수 데이터는 부동소수점 방식과 10진 방식을 다루는 세 가지 데이터 형식 키워드인 double, float, decimal을 제공한다.
- double, float는 부동소수점 방식이다.
- 실수 데이터는 지수 표기법으로 표현가능하다. 예로 float 데이터 형식의 최댓값은 약 3.4×10^{38} 정도로 볼 수 있다.

데이터 형식	비트	범위	닷넷 형식
float	32비트	-3.402823E+38~+3.402823E+38	System.Single
double	64비트	-1.79769313486232E+308~ +1.79769313486232E+308	System.Double
decimal	128비트	-79228162514264337593542950335~ +79228162514264337593542950335	System.Decimal

05. 실수형 데이터 : float

- float 키워드(System.Single)를 이용한다.
- float 데이터 형식은 32비트 부동소수점 방식을 사용한다.
- Float 키워드로 실수 데이터를 직접 입력할 때는 대문자 F 또는 소문자 f를 접미사로 사용하여야 한다.

```
float f1 = 3.14F;
```

05. 실수형 데이터 : double

- double 키워드(System.Double)
- double 키워드로 실수 데이터를 직접 입력할 때는 대문자 D 또는 소문자 d를 접미사로 사용하며 생략 가능하다.
- 64비트 부동소수점 방식을 사용한다.

`double f2 = 3.14D;`

05. 실수형 데이터 : decimal

- decimal 키워드(System.Decimal)
- 십진수로 표시되며 128비트의 숫자를 표현한다.
- 소수점 28자리까지는 정확도(유효 자릿수)가 높기에 세금과 환율 계산 등 주로 금융 프로그램을 만들 때 사용한다.
- M 또는 m 으로 접미사를 이용하여 리터럴에 표시하여야 한다.

decimal f3 = 3.14M;

05. 실수형 데이터

```
float n1 = 3.14f;  
double n2 = 3.1435262726d;  
decimal n3 = 3.1435262726m;  
Console.WriteLine("실수형 데이터");  
Console.WriteLine(" n1 = {0}, {1}", n1, n1.GetType());  
Console.WriteLine(" n2 = {0}, {1}", n2, n2.GetType());  
Console.WriteLine(" n3 = {0}, {1}", n3, n3.GetType());
```

05. 실수형 데이터

```
Console.WriteLine("실수형 데이터 최대와 최소");  
Console.WriteLine(" float 최대값 {0}, 최소값 {1}",  
    float.MaxValue, float.MinValue);  
Console.WriteLine(" double 최대값 {0}, 최소값 {1}",  
    double.MaxValue, double.MinValue);  
Console.WriteLine(" decimal 최대값 {0}, 최소값 {1}",  
    decimal.MaxValue, decimal.MinValue);  
Console.WriteLine("=====");
```

```
float 최대값 3.402823E+38, 최소값 -3.402823E+38  
double 최대값 1.79769313486232E+308, 최소값 -1.79769313486232E+308  
decimal 최대값 79228162514264337593543950335, 최소값 -79228162514264337593543950335
```


퀴즈

- 원의 반지름과 파이값을 변수로 지정한 후 아래와 같은 결과 화면이 출력되도록 프로그래밍 하시오.
- 원의 넓이 = $3.14 * \text{반지름} * \text{반지름}$
- 원의 둘레 = $2 * 3.14 * \text{반지름}$

출력 화면

원의 반지름 = 5

원의 넓이 = ?

원의 둘레 = ?

퀴즈

- MinValue와 MaxValue 코드를 이용하여 byte, sbyte, short, ushort, int, uint, long, ulong, float, double 숫자형 데이터 중 4개를 선택하여 최댓값과 최솟값을 출력하도록 프로그래밍 하시오.

출력 화면

? 자료형 최댓값 : ? 최솟값 : ?
? 자료형 최댓값 : ? 최솟값 : ?
? 자료형 최댓값 : ? 최솟값 : ?
? 자료형 최댓값 : ? 최솟값 : ?

문자 데이터 형식

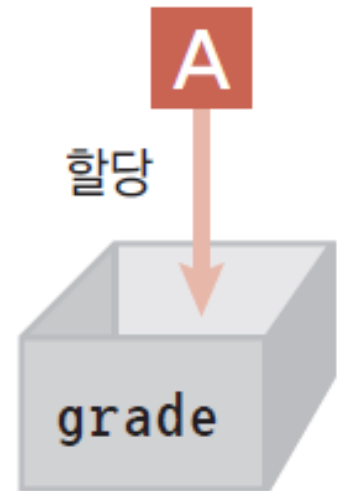
Update 2020

A solid green horizontal bar at the bottom of the slide.

01. 문자 char

- char 문자형은 2바이트, 16비트 공간에 문자 하나를 저장한다.
- 값을 초기화할 때는 작은따옴표 2개를 사용하여 문자 하나를 묶어준다.
- 닷넷 데이터 형식은 System.Char 이다.

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



02. 문자열 string

- string 문자형은 여러 문자를 담을 수 있는 데이터형이다.
- 값을 초기화할 때는 큰따옴표 2개를 사용하여 문자열을 묶어준다.
- 닷넷 데이터 형식은 System.String 이다.
- @을 문자열 앞에 삽입하면 여러 줄의 문자열을 변수에 저장할 수 있다.

```
string s1 = "Hello World";  
string s2 = @"  
오늘도  
당신을  
지원합니다.  
";
```

03. 문자와 문자열

```
Console.WriteLine("\n\n문자와 문자열 데이터형");
char c1 = 'A';
string s1 = "Hello World";
//@을 이용한 여러줄 문자열 변수 저장
string s2 = @"
    오늘도
    당신을
    응원합니다.
    ";
Console.WriteLine(" c1 = {0}, {1}", c1, c1.GetType());
Console.WriteLine(" s1 = {0}, {1}", s1, s1.GetType());
Console.WriteLine(" s2 = {0}", s2);
```

```
c1 = A, System.Char
s1 = Hello World, System.String
s2 =
    오늘도
    당신을
    응원합니다.
```

04. 문자열 보간법(string interpolation)

- 문자열을 그룹화하여 변수로 지정하거나 출력문에 사용한다.
- C# 6.0 버전부터 사용되었으며 문자열 템플릿(string template) 또는 템플릿 문자열(template string)이라고도 한다.
- String.Format() 메서드를 이용하여 변수에 값을 할당할 수 있다.
String 변수 = String.Format("{index}, variable")
String 변수 = String.Format("\${variable}")

04. 문자열 보간법(string interpolation)

```
string user_name = "김철수";  
int user_age = 23;  
string user_info = $"고객명 = {user_name}, 나이 = {user_age}";  
Console.WriteLine($"고객명 = {user_name}, 나이 = {user_age}");  
Console.WriteLine(user_info);  
Console.WriteLine(user_info.GetType());  
Console.WriteLine("=====");
```


04. 문자열 보간법(string interpolation)

```
string user1 = "Maria", user2 = "John";  
string msg = "Good Night";  
string msg1 = String.Format("{0}, {1}", user1, msg);  
string msg2 = String.Format("{0}, {1}", user2, msg);  
string msg3 = String.Format($"{user1} {msg},\n {user2} {msg}");  
Console.WriteLine(msg1);  
Console.WriteLine(msg2);  
Console.WriteLine(msg3);  
Console.WriteLine("=====");
```

```
Maria, Good Night  
John, Good Night  
Maria Good Night,  
John Good Night
```

퀴즈

- 오늘의 운세(<http://todayunse.co.kr>) 사이트에서 띠별 운세로 문자열 데이터를 변수로 저장한 후 아래와 같은 결과 화면으로 출력되도록 프로그래밍 하시오.

출력 화면

?년 ?월 ?일 ? 띠 운세

신중하게 보내고 싶을 때입니다.

약속없어도 갑자기 친구나 지인이 찾아 오거나

많은 사람과 떠들썩한 하루를 보낼 수 있을 것 같습니다.

금전운 : 금전운은 오후부터 회복을 기대할 수 있는 날.

사업운 : 쓸데 없는 수다는 평판을 떨어뜨릴 것 같다.

논리 데이터 형식

Update 2023

A solid green horizontal bar at the bottom of the slide.

bool

- 참(true) 또는 거짓(false) 값을 저장하며 bool 키워드를 사용한다.
- 1비트의 저장 공간을 차지하며 true와 false 키워드로 값을 지정할 수 있다.
- 닷넷 데이터 형식은 System.Boolean 이다.
- 출력문을 이용한 결과값은 첫글자가 대문자로 True, False로 표시된다.

```
bool b1 = true;
```

```
System.Boolean b2 = false;
```

bool

```
bool b1 = true; // 참
bool b2 = false; // 거짓
Console.WriteLine($"b1={b1}, b2={b2}, {b1.GetType()}");
Console.WriteLine("=====");
```

b1=True, b2=False, System.Boolean

상수

Update 2023

A solid green horizontal bar spanning the width of the slide at the bottom.

const

- 변수에 const 키워드를 붙이면 상수로 선언된다.
- const 데이터형 변수명 = 리터럴;
- 상수로 지정된 변수는 값을 변경할 수 없다.
- 상수로 지정된 변수명은 통상적으로 대문자로 지정한다.

`const double PI = 3.14;`

const

```
Console.WriteLine("\t\t상수");
const float PI = 3.14f;
const string MYNAME = "홍길동";
Console.WriteLine($"PI 값은? {PI}");
Console.WriteLine($"작성자 이름 : {MYNAME}");
MYNAME = "고길동";
Console.WriteLine($"(지역 상수) string MYNAME = "홍길동" =====");
```

CS0131: 할당식의 왼쪽은 변수, 속성 또는 인덱서여야 합니다.

퀴즈

- float 자료형의 최댓값과 최솟값을 MAX_FLOAT, MIN_FLOAT 상수로 정의한 후 아래와 같은 결과 화면으로 출력되도록 프로그래밍 하시오.

출력 화면

float 자료형의 최댓값 = ?

float 자료형의 최솟값 = ?

입력문

Update 2023

A solid green horizontal bar at the bottom of the slide.

01. 문자열 입력 관련 메소드

- 키보드로 입력받고 화면을 통해 출력하는 일반적인 내용을 표준 입출력 (standard input/output)이라고 한다.
- **Console.ReadLine()**: 콘솔에서 한 줄을 입력받는다. 콘솔에 대기하고 있다 사용자가 한 줄을 입력한 후 [Enter]키를 누르면 해당 문자열을 입력받아 그결과값을 반환한다.
- **Console.Read()**: 콘솔에서 입력키의 유니코드 숫자값이 변수로 저장되고 출력된다. 입력 변수의 데이터형은 정수이어야 한다.
- **Console.ReadKey()**: 콘솔에서 다음 문자나 사용자가 누른 키를 반환한다. 입력키의 변수 선언은 ConsoleKeyInfo를 이용해야하며 입력키의 정보 출력은 Key속성값을 사용한다.

02. Console.ReadLine()

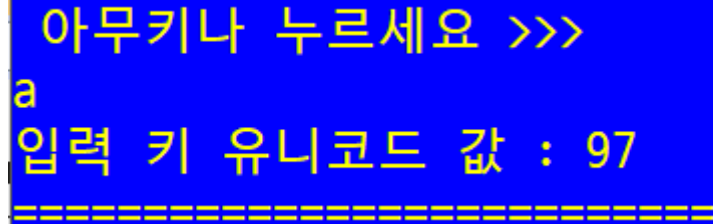
```
Console.Write(" 이름을 입력하세요 => ");  
string userName = Console.ReadLine();  
Console.WriteLine($" {userName} 님 환영합니다. ");  
Console.WriteLine("=====");
```

```
이름을 입력하세요 => 고길동  
고길동 님 환영합니다.
```

```
=====
```

03. Console.Read()

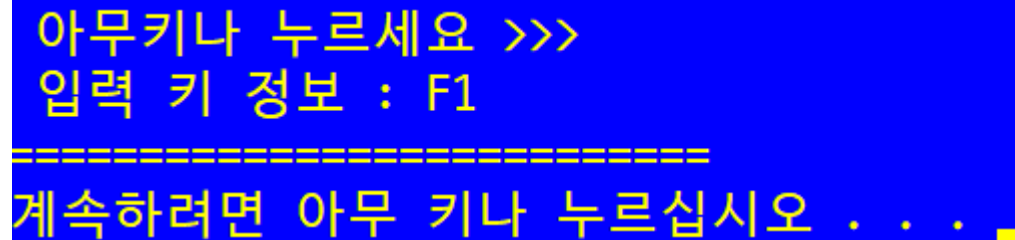
```
Console.WriteLine("\n 아무키나 누르세요 >>>");  
int user_key = Console.Read();  
Console.WriteLine($"입력 키 유니코드 값 : {user_key}");  
Console.WriteLine("=====");
```



```
아무키나 누르세요 >>>  
a  
입력 키 유니코드 값 : 97  
=====
```

04. Console.ReadKey()

```
ConsoleKeyInfo userkey;  
Console.WriteLine("\n 아무키나 누르세요 >>>");  
userkey = Console.ReadKey();  
Console.WriteLine($"입력 키 정보 : {userkey.Key}");  
Console.WriteLine("=====");
```



```
아무키나 누르세요 >>>  
입력 키 정보 : F1  
=====  
계속하려면 아무 키나 누르십시오 . . .
```

데이터 형식 변환

Update 2023

A solid green horizontal bar at the bottom of the slide.

01. 형 변환의 종류

- 암시적 형(implicit) 변환

변환 형식이 안전하게 유지되며 데이터가 손실되지 않아 특수한 구문이 필요 없다.

예) int 형 변수 => long 형 변수

- 명시적 형(explicit) 변환

캐스팅(casting)이라고도 한다.

별도의 문법이 필요하며 데이터가 손실될 수 있다

예) long 형 변수 => int 형 변수

변수명앞에 (데이터형) 을 이용하거나 Convert 클래스를 이용하여 변환해야 한다.

02. 암시적 데이터형 변환

```
int vv1 = 100;
long vv2 = vv1;
Console.WriteLine("\n\n\n암시적 형변환 테스트");
Console.WriteLine($"\\t vv1의 값은 {vv1}, 데이터형은 {vv1.GetType()}");
Console.WriteLine($"\\t vv2의 값은 {vv2}, 데이터형은 {vv2.GetType()}");

// 오류 발생. 오버플로우
long vv3 = long.MaxValue;
int vv4 = vv3;
```

03. 명시적 데이터형 변환

```
long vv3 = long.MaxValue;  
int vv4 = (int)vv3;  
double vv5 = (double)vv3;  
Console.WriteLine($"\\t vv3의 값은 {vv3}, 데이터형은 {vv3.GetType()}");  
Console.WriteLine($"\\t vv4의 값은 {vv4}, 데이터형은 {vv4.GetType()}");  
Console.WriteLine($"\\t vv5의 값은 {vv5}, 데이터형은 {vv5.GetType()}");
```

```
vv3의 값은 9223372036854775807, 데이터형은 System.Int64  
vv4의 값은 -1, 데이터형은 System.Int32  
vv5의 값은 9.22337203685478E+18, 데이터형은 System.Double
```

04. Convert 클래스

메서드	설명
<code>Convert.ToString()</code>	숫자 데이터 형식을 문자열로 변경합니다.
<code>Convert.ToInt32()</code>	숫자 데이터 형식을 정수 형식으로 변경합니다.
<code>Convert.ToDouble()</code>	숫자 데이터 형식을 실수 형식으로 변경합니다.
<code>Convert.ToChar()</code>	입력받은 숫자 또는 문자열 하나를 문자로 변경합니다.

04. Convert 클래스

```
int x1 = int.MinValue;
double x2 = Convert.ToDouble(x1);
bool x3 = Convert.ToBoolean(x1);
string x4 = Convert.ToString(x1);
Console.WriteLine($"\\t x1의 값은 {x1} 데이터형은 {x1.GetType()}");
Console.WriteLine($"\\t x2의 값은 {x2} 데이터형은 {x2.GetType()}");
Console.WriteLine($"\\t x3의 값은 {x3} 데이터형은 {x3.GetType()}");
Console.WriteLine($"\\t x4의 값은 {x4} 데이터형은 {x4.GetType()}");
```

```
x1의 값은 -2147483648 데이터형은 System.Int32
x2의 값은 -2147483648 데이터형은 System.Double
x3의 값은 True 데이터형은 System.Boolean
x4의 값은 -2147483648 데이터형은 System.String
```

퀴즈

- 데이터를 Console.ReadLine() 명령문을 이용하여 입력받아 변수에 저장한 후 아래와 같은 결과 화면으로 출력되도록 프로그래밍 하시오.
(데이터형 변경 필요)

입력 화면

사각형의 가로 길이를 숫자로 입력하여 주세요 10
사각형의 세로 길이를 숫자로 입력하여 주세요 5

출력 화면

사각형의 가로 길이 = 10
사각형의 세로 길이 = 5
사각형의 넓이 = 50

진법

Update 2020

A solid green horizontal bar at the bottom of the slide.

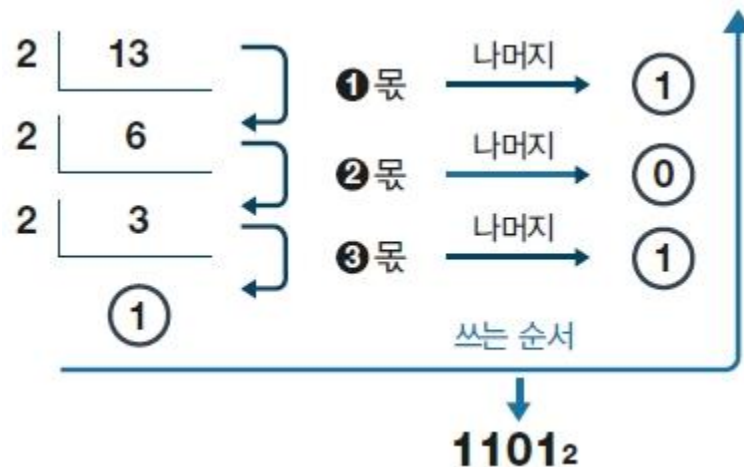
01. 비트와 바이트

- 컴퓨터에서 표현 가능한 제일 작은 단위는 비트(Bit)이다. 비트는 컴퓨터에서 표현 가능한 가장 작은 단위로, 0과 1만 존재한다.
- 비트 8개가 모이면 바이트(Byte)가 된다.
- 비트를 진수로 표현한다면 2진수(Binary)의 숫자 표기와 일치한다.

전기 스위치				
의미	꺼짐 , 꺼짐	꺼짐 , 켜짐	켜짐 , 꺼짐	켜짐 , 켜짐
2진수	00	01	10	11
10진수	0	1	2	3

02. 진법

- 10진수는 10개의 숫자(0~9)로 모든 숫자를 표현한다.
- 2진수는 2개의 숫자(0, 1)로만 모든 수를 표현한다.
- 16진수는 0~F까지 총 16가지의 숫자로 표현된다.



02. 진법

- 16진수, 2진수 변환표

16진수	2진수	16진수	2진수
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

03. Convert.ToString()와 PadLeft()

- Convert.ToString(문자열숫자, 2) : 2진수 문자열
- Convert.ToString(문자열숫자, 8) : 8진수 문자열
- Convert.ToString(문자열숫자, 16) : 16진수 문자열
- PadLeft(전체자릿수, 채움문자) : 왼쪽에 여백을 0으로 채운다

03. Convert.ToString()와 PadLeft()

```
int y1 = 100;
String y2 = Convert.ToString(y1, 2);
String y3 = Convert.ToString(y1, 8);
String y4 = Convert.ToString(y1, 16);
Console.WriteLine($"\\t 10진수 = {y1}, 데이터형은 {y1.GetType()} ");
Console.WriteLine($"\\t 2진수 = {y2}, 데이터형은 {y2.GetType()} ");
Console.WriteLine($"\\t 2진수 = {y2.PadLeft(8, '0')}, 데이터형은 {y2.GetType()} ");
Console.WriteLine($"\\t 8진수 = {y3}, 데이터형은 {y3.GetType()} ");
Console.WriteLine($"\\t 8진수 = {y3.PadLeft(8, '0')}, 데이터형은 {y3.GetType()} ");
Console.WriteLine($"\\t 16진수 = {y4}, 데이터형은 {y4.GetType()} ");
Console.WriteLine($"\\t 16진수 = {y4.PadLeft(4, '0')}, 데이터형은 {y4.GetType()} ");
```

```
10진수 = 100, 데이터형은 System.Int32
2진수 = 1100100, 데이터형은 System.String
2진수 = 01100100, 데이터형은 System.String
8진수 = 144, 데이터형은 System.String
8진수 = 00000144, 데이터형은 System.String
16진수 = 64, 데이터형은 System.String
16진수 = 0064, 데이터형은 System.String
```

04. 이진수 리터럴

- 이진수 리터럴(binary literal) 은 0b나 0B 접두사를 붙여 이진수를 직접 코드로 표현하는 방법이다.

예) 0b0010

- 출력문으로 이진수 리터럴값을 출력하면 10진수 형태로 자동변환되어 표시된다.

```
byte z = 0b0010;  
Console.WriteLine($"\\t z = {z} {z.GetType()}");  
Console.WriteLine($"\\t z = {Convert.ToString(z, 2)}");  
Console.WriteLine($"\\t z = {Convert.ToString(z, 2).PadLeft(4, '0')}");
```

```
z = 2 System.Byte  
z = 10  
z = 0010
```

05. 16진수 리터럴

- 16진수 리터럴(binary literal) 은 0x나 0X 접두사를 붙여 16진수를 직접 코드로 표현하는 방법이다.

예) 0xA0

- 출력문으로 16진수 리터럴값을 출력하면 10진수 자동변환되어 표시된다.

```
byte z16 = 0xF0;  
Console.WriteLine($"\\t z16 = {z16} {z16.GetType()}");  
Console.WriteLine($"\\t z16 = {Convert.ToString(z16, 16)}");  
Console.WriteLine($"\\t z16 = {Convert.ToString(z16, 16).PadLeft(8, '0')}");
```

```
z16 = 240 System.Byte  
z16 = f0  
z16 = 000000f0
```

퀴즈

- 입력받은 10진수를 2, 8, 16 진수로 출력되도록 프로그래밍하여라.

```
입력 >>69
=====
                               출력
=====
                2진수 = 1000101
                8진수 = 105
                16진수 = 45
```

var 키워드

Update 2023

A solid green horizontal bar at the bottom of the slide.

Var 키워드를 이용한 변수 설정

- 변수 선언시 var 키워드를 사용하면 데이터값에 따라 자동으로 데이터 형식이 지정된다.
- var 변수명 = 데이터값;
- 암시적 변수 지정이라고 한다.

var 키워드를 이용한 변수 설정

```
var q1 = "닷넷프레임워크";  
var q2 = 10;  
var q3 = 123.567;  
var q4 = '옹';  
var q5 = false;  
Console.WriteLine($"\\t q1의 값은 {q1} , 데이터형은 {q1.GetType()}");  
Console.WriteLine($"\\t q2의 값은 {q2} , 데이터형은 {q2.GetType()}");  
Console.WriteLine($"\\t q3의 값은 {q3} , 데이터형은 {q3.GetType()}");  
Console.WriteLine($"\\t q4의 값은 {q4} , 데이터형은 {q4.GetType()}");  
Console.WriteLine($"\\t q5의 값은 {q5} , 데이터형은 {q5.GetType()}");
```

```
q1의 값은 닷넷프레임워크 , 데이터형은 System.String  
q2의 값은 10 , 데이터형은 System.Int32  
q3의 값은 123.567 , 데이터형은 System.Double  
q4의 값은 옹 , 데이터형은 System.Char  
q5의 값은 False , 데이터형은 System.Boolean
```

default 키워드

Update 2023

A solid green horizontal bar at the bottom of the slide.

default 키워드를 이용한 변수 설정

- 변수 선언시 default 키워드를 사용하여 데이터 값을 지정하면 해당 변수의 데이터 형식으로 초기값이 지정된다.
- 숫자 데이터 형식은 0, char는 ₩0로 기본값이 지정된다.
- 데이터형 변수명 = default;
- 데이터형 변수명 = default(데이터형);
- String 변수의 경우 default 키워드를 이용할 수 없다

default 키워드를 이용한 변수 설정

```
int p1 = default;
long p2 = default(long);
float p3 = default;
bool p4 = default;
char p5 = default(char);
string p6 = default(string);
```

```
Console.WriteLine($"\\t p1의 값은 {p1} 데이터형은 {p1.GetType()}");
Console.WriteLine($"\\t p2의 값은 {p2} 데이터형은 {p2.GetType()}");
Console.WriteLine($"\\t p3의 값은 {p3} 데이터형은 {p3.GetType()}");
Console.WriteLine($"\\t p4의 값은 {p4} 데이터형은 {p4.GetType()}");
Console.WriteLine($"\\t p5의 값은 {p5} 데이터형은 {p5.GetType()}");
Console.WriteLine($"p6의 값은 {p6} 데이터형은 {p6.GetType()}");
```

default 키워드를 이용한 변수 설정

```
p1의 값은 0 데이터형은 System.Int32  
p2의 값은 0 데이터형은 System.Int64  
p3의 값은 0 데이터형은 System.Single  
p4의 값은 False 데이터형은 System.Boolean  
p5의 값은   데이터형은 System.Char
```

처리되지 않은 예외: System.NullReferenceException: 개체 참조가 개체의 인스턴스로 설정되지 않았습니다.

String 메소드

Update 2023

A solid green horizontal bar at the bottom of the slide.

01. 탐색 메소드

- `IndexOf()` : 현재 문자열에서 찾고자 하는 문자나 문자열의 위치를 숫자 형태로 반환한다.
- `LastIndexOf()` : 현재 문자열에서 찾고자 하는 문자나 문자열의 위치를 뒤에서부터 찾아 숫자 형태로 반환한다.
- `StartWith()` : 현재 문자열이 특정 문자열이나 문자로 시작하는지를 `True`, `False`로 반환다.
- `EndsWith()` : 현재 문자열이 특정 문자열이나 문자로 끝나는지를 `True`, `False`로 반환다.
- `Replace(w1, w2)` : 현재 문자열에서 특정 문자열을 다른 문자열로 변경한 후 반환한다.

01. 탐색 메소드

```
string sampleTxt1 = "Good Morning";
string sampleTxt2 = "가나다라마바사아자차카타파하";
Console.WriteLine($"sampleTxt1 = {sampleTxt1}");
Console.WriteLine($"sampleTxt2 = {sampleTxt2}");
Console.WriteLine($"마지막 o의 인덱스 위치 = {sampleTxt1.LastIndexOf("o")}");
Console.WriteLine($"a로 시작하는가? {sampleTxt1.StartsWith("a")}");
Console.WriteLine($"G로 시작하는가? {sampleTxt1.StartsWith("G")}");
Console.WriteLine($"G로 끝나는가? {sampleTxt1.EndsWith("G")}");
Console.WriteLine($"ing로 끝나는가? {sampleTxt1.EndsWith("ing")}");
Console.WriteLine($"차가 포함되었는가? {sampleTxt2.Contains("차")}");
Console.WriteLine($"Morning 글자 교체 {sampleTxt1.Replace("Morning", "Night")}");
```


01. 탐색 메소드

```
sampleTxt1 = Good Morning  
sampleTxt2 = 가나다라마바사아자차카타파하  
마지막 o의 인덱스 위치 = 6  
a로 시작하는가? False  
G로 시작하는가? True  
G로 끝나는가? False  
ing로 끝나는가? True  
차가 포함되었는가? True  
Morning 글자 교체 Good Night
```

02. 변형 메소드

- ToLower() : 알파벳 소문자로 반환한다.
- ToUpper() : 알파벳 대문자로 반환한다.
- Insert(index, string) : 현재 문자열의 index 위치에 string을 삽입한다.
- Remove(index, n) : 현재 문자열에서 index 위치에서 n개의 문자열을 삭제한다.
- Trim() : 현재 문자열에서 앞과 뒤에 삽입된 공백을 삭제한다.
- TrimStart() : 현재 문자열에서 앞에 삽입된 공백을 삭제한다.
- TrimEnd() : 현재 문자열에서 뒤에 삽입된 공백을 삭제한다.

02. 변형 메소드

```
string sampleTxt3 = "hello C#";  
string sampleTxt4 = "abcdefghijklmnopqrstu";  
Console.WriteLine($"\\t sampleTxt3 = {sampleTxt3}");  
Console.WriteLine($"\\t sampleTxt4 = {sampleTxt4}");  
Console.WriteLine($"\\t ToLower = {sampleTxt3.ToLower()}");  
Console.WriteLine($"\\t ToUpper = {sampleTxt4.ToUpper()}");  
Console.WriteLine("\\t Insert = {0}", sampleTxt3.Insert(0, "***"));  
Console.WriteLine("\\t Remove = {0}", sampleTxt4.Remove(3, 5));
```

```
sampleTxt3 = hello C#  
sampleTxt4 = abcdefghijklmnopqrstu  
ToLower = hello c#  
ToUpper = ABCDEFGHIJKLMNOPQRSTU  
Insert = ***hello C#  
Remove = abcijklmnopqrstu
```

02. 변형 메소드

```
Console.WriteLine();  
Console.Write("####");  
Console.Write("Trim={0}", "    홍 길 동    ".Trim());  
Console.Write("####");  
Console.WriteLine();
```

```
####Trim=홍 길 동####
```

03.분할 메소드

- SubString(index) : index 위치로 부터 새문자열을 반환한다.
- Splite() : 공백을 기준으로 문자열을 분리하여 배열로 반환한다.

```
string sampleTxt5 = "C# JAVA PYTHON";  
Console.WriteLine($"\\t sampleTxt5 = {sampleTxt5}");  
Console.WriteLine($"\\t Substring(2) = {sampleTxt5.Substring(2)}");  
Console.WriteLine($"\\t Substring(8) = {sampleTxt5.Substring(8)}");
```

```
sampleTxt5 = C# JAVA PYTHON  
Substring(2) =  JAVA PYTHON  
Substring(8) = PYTHON
```

03. 분할 메소드

```
string[] arr = sampleTxt5.Split();  
Console.WriteLine($"\\t arr = {arr}");  
Console.WriteLine($"\\t arr = {arr[0]}, {arr[1]}, {arr[2]}");  
foreach (var item in arr)  
{  
    Console.WriteLine($"\\t {item}");  
}
```

```
arr = System.String[]  
arr = C#, JAVA, PYTHON  
C#  
JAVA  
PYTHON
```

출력 Formatting

Update 2023

A solid green horizontal bar at the bottom of the slide.

01. 문자열 포매팅

- 출력문안의 자리지정자 부분에 삽입한다.

n만큼 왼쪽이나 오른쪽에 공백을 삽입하고 문자열을 삽입한다.

{Index, (-)n}

```
string sample1 = "가나다라마바사";  
Console.WriteLine("\t*{0, 20}*", sample1);  
Console.WriteLine("\t*{0, -20}*", sample1);  
Console.WriteLine("\n=====");
```

```
*           가나다라마바사*  
*가나다라마바사           *
```


02. 숫자 포매팅

- 출력문안의 자리지정자 부분에 삽입한다.

{Index, 서식지정자 }

- 서식 지정자

D숫자 – 10진수, 숫자는 전체자릿수로 숫자 부분이 없는경우 0으로 채워진다.

X숫자 – 16진수, 숫자는 전체자릿수로 숫자 부분이 없는경우 0으로 채워진다

N – 콤마(,)를 이용하여 숫자를 표시한다.

F숫자 – 고정소숫점을 표시한다. 숫자를 입력하면 소숫점이하 전체 자릿수를 지정할 수 있다.

E – 지수 형식으로 숫자를 표시한다.

02. 숫자 포매팅

```
Console.WriteLine("\t 10진수 => {0:D10}", 12345);  
Console.WriteLine("\t 16진수 => {0}", 0X32A); // 810  
Console.WriteLine("\t 16진수 => 0X{0:X}", 0X32A);  
Console.WriteLine("\t 16진수 => 0X{0:X10}", 0X32A);  
Console.WriteLine("\t N => {0}", 10000000000);  
Console.WriteLine("\t N => {0:N}", 10000000000);  
Console.WriteLine("\t N => {0:N0}", 10000000000);
```

```
10진수 => 0000012345  
16진수 => 810  
16진수 => 0X32A  
16진수 => 0X000000032A  
N => 10000000000  
N => 10,000,000,000.00  
N => 10,000,000,000
```

02. 숫자 포매팅

```
Console.WriteLine("\t F => {0:F}", 1234.5678); // 1234.57
Console.WriteLine("\t F => {0:F3}", 1234); // 1234.000
Console.WriteLine("\t F => {0:F0}", 1234.5678); // 1235
Console.WriteLine("\t F => {0:F7}", 1234.5678); // 1234.5678000
Console.WriteLine("\t E => {0:E}", 1234.5678); // 1.234568E+003
Console.WriteLine("\t E => {0:E2}", 1234.5678); // 1.23E+003
Console.WriteLine("\t E => {0:E5}", 1234.5678); // 1.23457E+003
```

```
F => 1234.57
F => 1234.000
F => 1235
F => 1234.5678000
E => 1.234568E+003
E => 1.23E+003
E => 1.23457E+003
```