

# 응용소프트웨어실습

---

## C# 프로그래밍 기본 1

# .NET

---

## ☐ .NET Framework

- Microsoft가 제공하는 Windows 데스크톱 프로그램 개발 및 실행환경
- CLS (Common Language Spec)를 따르는 언어라면, .NET 환경에서 실행이 가능함.
- CLR (Common Language Runtime)이라는 가상 머신 위에서 작동하며, 플랫폼에 독립적임.

## ☐ .NET Core

- .NET Framework를 오픈소스로 개발하여 공개한 프레임워크
- Windows 데스크톱의 표준 GUI를 렌더링 하는 WinForms 또는 WPF는 지원하지 않음.

## ☐ Mono

- .NET Framework의 오픈소스 버전
- C# 컴파일러와 CLR을 포함하며, Ecma 표준을 준수함.
- 각종 크로스 플랫폼에서 .NET 기반 응용 프로그램을 실행을 하는 동시에, 리눅스 개발자에게 더 나은 개발 도구를 제공하기 위한 목적으로 개발되었음.

## ☐ Visual Basic, C#, F# 등의 프로그래밍 언어를 지원함.

## ☐ .NET의 용도

### ☒ ASP.NET

- ☐ 웹 애플리케이션 (Web application) 개발

### ☒ Xamarin

- ☐ Mono 프로젝트에서 시작한 프레임워크
- ☐ Android, iOS, macOS 등의 크로스 플랫폼 애플리케이션 개발

### ☒ Windows Forms (WinForms)

- ☐ Windows 데스크톱 GUI 애플리케이션 개발

### ☒ Windows Presentation Forms (WPF)

- ☐ Windows 데스크톱 GUI 애플리케이션 개발

### ☒ Universal Windows Platform (UWP)

- ☐ Windows 10을 지원하는 디바이스의 애플리케이션 개발

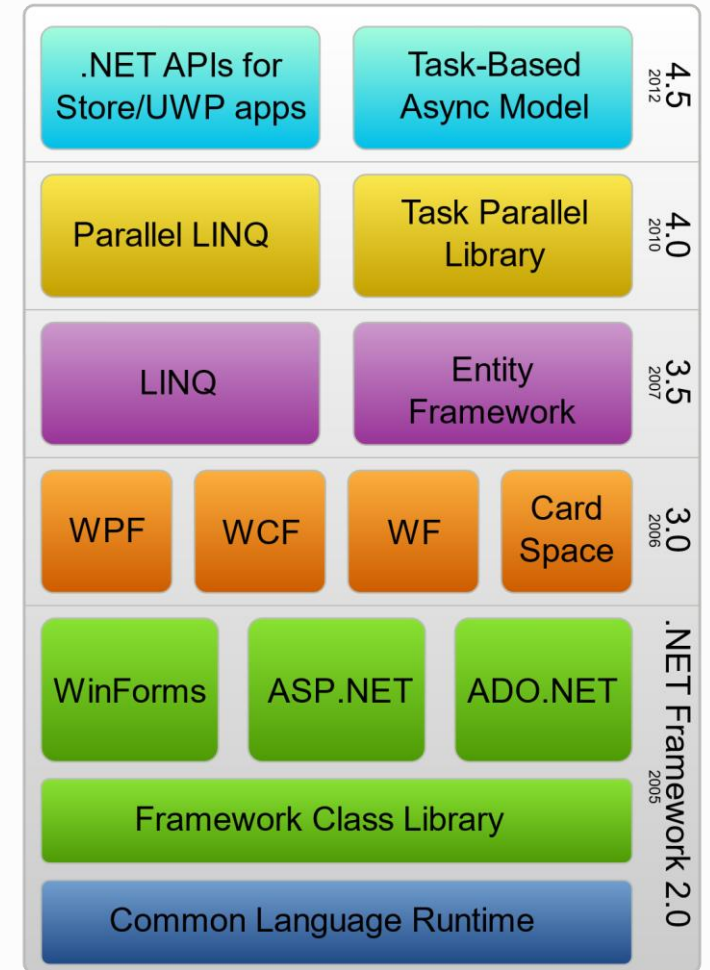
### ☒ Unity

- ☐ 게임 개발

### ☒ ...

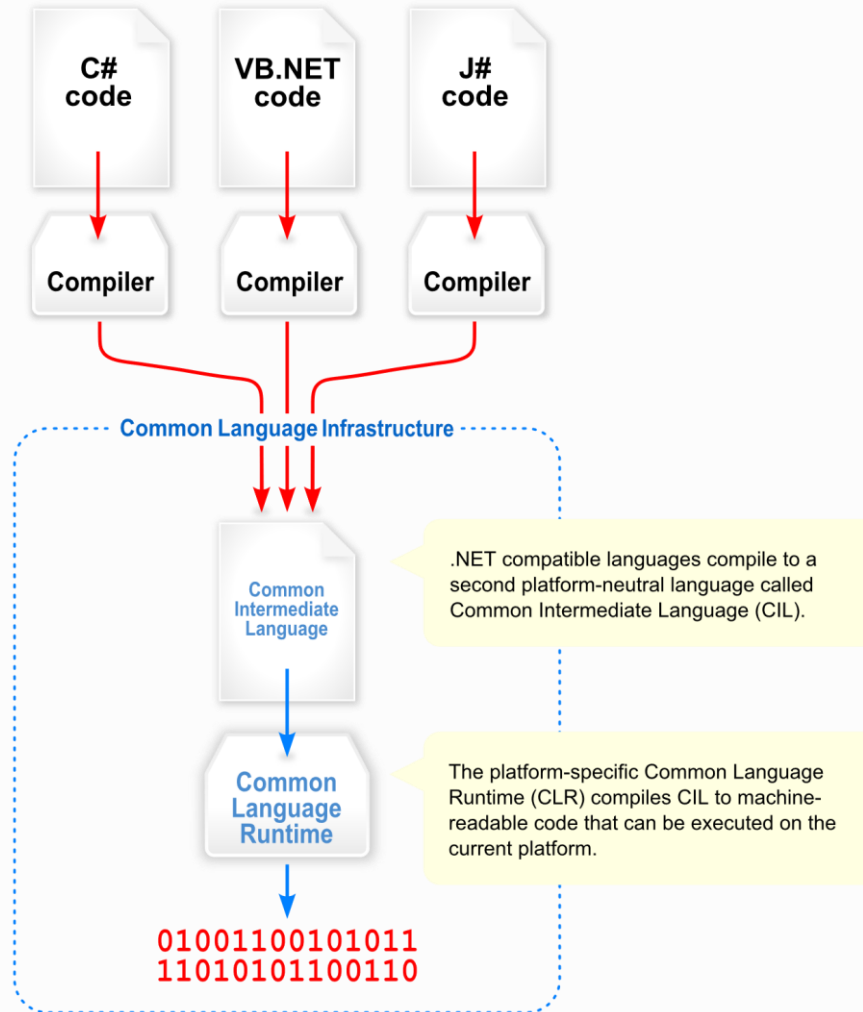
# .NET Framework의 버전

- ☐ 1.0
  - .NET Framework의 핵심 구성 요소 및 기본 프로그래밍 언어를 처음으로 완성한 버전
- ☐ 1.1
  - ADO.NET에 데이터베이스 지원을 추가하였고, ASP.NET의 기능을 강화한 버전
- ☐ 2.0
  - 제네릭 프로그래밍을 가능하게 하는 제네릭을 도입한 버전
- ☐ 3.0
  - WPF, WCF, WF, Card Space로 불리는 주요 기능을 추가한 버전
- ☐ 3.5
  - LINQ와 Entity Framework 등의 기능을 추가한 버전
- ☐ 4.0
  - 병렬 처리를 위한 Parallel LINQ 등의 기능을 추가한 버전
- ☐ 4.5
  - UWP 애플리케이션을 지원하고, 비동기 처리 기능을 추가한 버전
- ☐ 4.8
  - WinForm과 WPF에서 UWP를 컨트롤 하는 기능을 추가한 버전



- 베이스 클래스 라이브러리 (Base Class Library, BCL)
  - .NET Framework를 사용하는 모든 프로그래밍 언어가 사용할 수 있는 라이브러리
  - 파일 입출력, 그래픽 렌더링, 데이터베이스 조작 등의 공통된 기능을 해주는 클래스들을 제공함.
  
- 공통 언어 기반 (Common Language Infrastructure)
  - .NET Framework의 가장 중요한 요소임.
  - 애플리케이션의 개발과 실행 시에 언어에 종속적이지 않은 플랫폼을 제공함.
  - 예외 처리, 가비지 콜렉션, 보안, 호환 등을 위한 다양한 요소를 포함함.
  - 공통 언어 런타임 (Common Language Runtime, CLR)이라고 부르기도 함.
  - 다음의 주요 요소로 구성되어 있음.
    - 공통 타입 시스템 (Common Type System, CTS)
    - 공통 언어 스펙 (Common Language Spec, CLS)
    - 저스트 인 타임 컴파일러 (Just-In-Time, JIT)
    - 가상 실행 시스템 (Virtual Execution System, VES)

# .NET Framework의 컴파일



## ☐ ADO.NET

- 데이터베이스 등의 데이터 서비스에 접근할 수 있는 프레임워크
- ADO.NET을 통해 데이터 소스에 연결하여 검색, 처리 등의 작업을 할 수 있음.

## ☐ ASP.NET

- 강력한 웹 애플리케이션을 개발하기 위한 프레임워크
- 동적 웹 사이트, 웹 애플리케이션, 웹 서비스를 개발할 수 있음.
- ASP.NET Web Forms는 쉽고 편리한 방법으로 웹 사용자 인터페이스 (Web User Interface)를 생성할 수 있게 함.

## ☐ Windows User Interface

- Windows 데스크톱 GUI 애플리케이션을 개발할 수 있음.
- .NET은 WinForms, WPF, UWP 등의 프레임워크를 제공함.



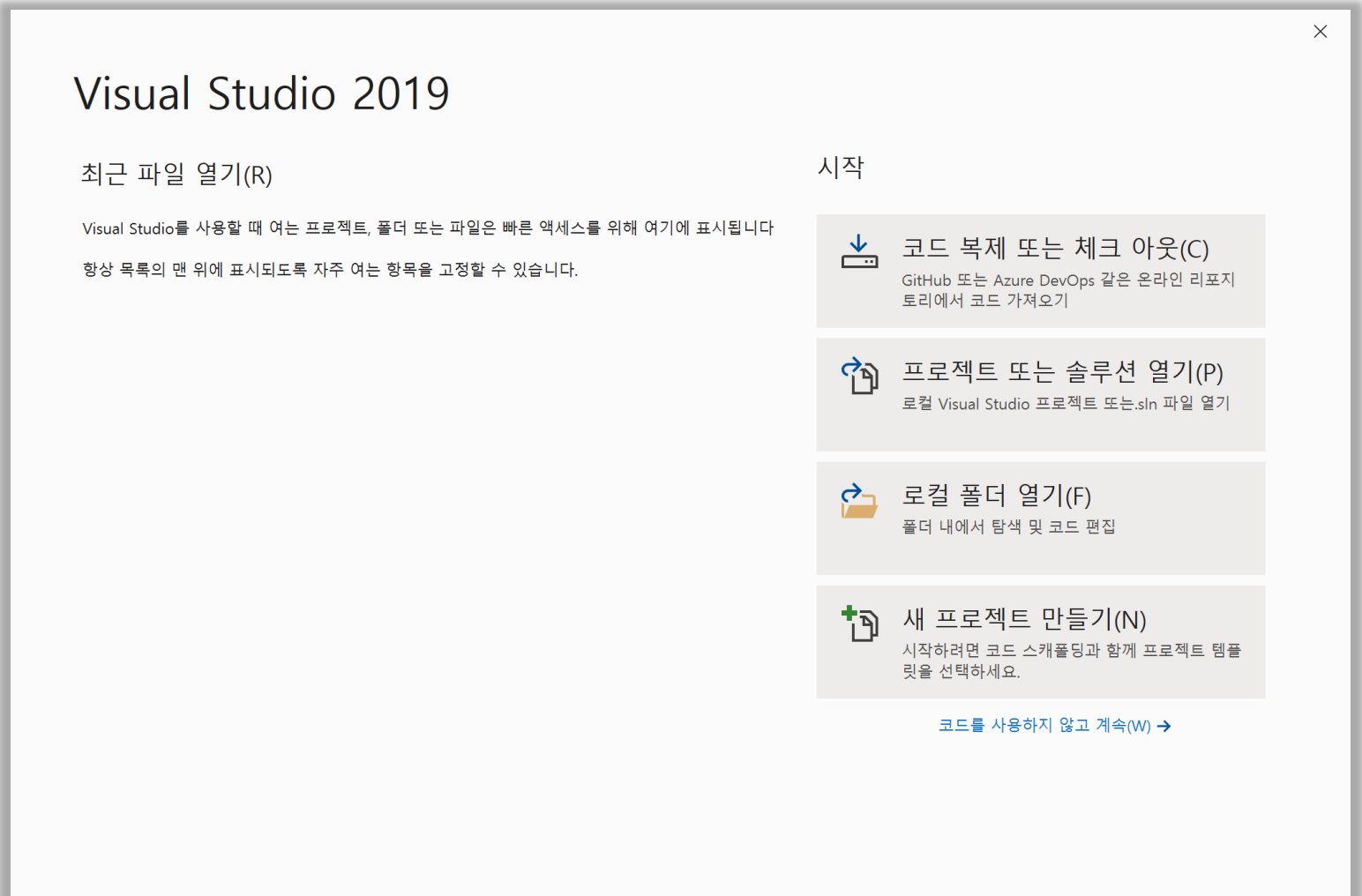
# C#

---

- .NET과 함께 발표된 프로그래밍 언어
  - .NET Core 또는 .NET Framework 환경을 지원하는 언어
- Microsoft가 개발하였으며, C++과 Java에서 영향을 받음.
  - 객체 지향 프로그래밍 언어
- 사건 기반 프로그래밍 (Event Driven)
  - 사용자의 명령, 마우스 클릭, 키보드 입력, 다른 프로그램의 메시지 등의 이벤트에 따라, 제어 흐름이 결정됨.
- 비주얼 프로그래밍 언어 (Visual Programming Language)
  - 사용자가 텍스트로 지정하는 대신에, 그래픽적으로 프로그램 요소를 조작하여 프로그램을 개발할 수 있음.
- 고속 응용프로그램 개발 (Rapid Application Development)
  - Visual Studio 등의 IDE를 통한 빠른 개발이 가능함.

- ☐ 객체 지향
  - 네임스페이스 (Namespace)
  - 클래스 (Class)
- ☐ 언어간 상호 운용성 (Language Interoperability)
  - .NET 기반의 다른 프로그래밍 언어로 개발된 컴포넌트들과 호환이 가능함.
- ☐ 적용 분야의 다양성
  - 콘솔 애플리케이션
  - GUI 애플리케이션
  - 웹 애플리케이션
  - ...
- ☐ 가비지 콜렉션 (Garbage Collection, GC)
- ☐ 멀티스레딩 지원 (Multi Thread)
- ☐ 비동기 처리 지원 (Async)

## □ 시작 페이지



# Visual Studio와 C# (cont`d)

## □ 새 프로젝트 만들기

### ■ 언어

□ C#

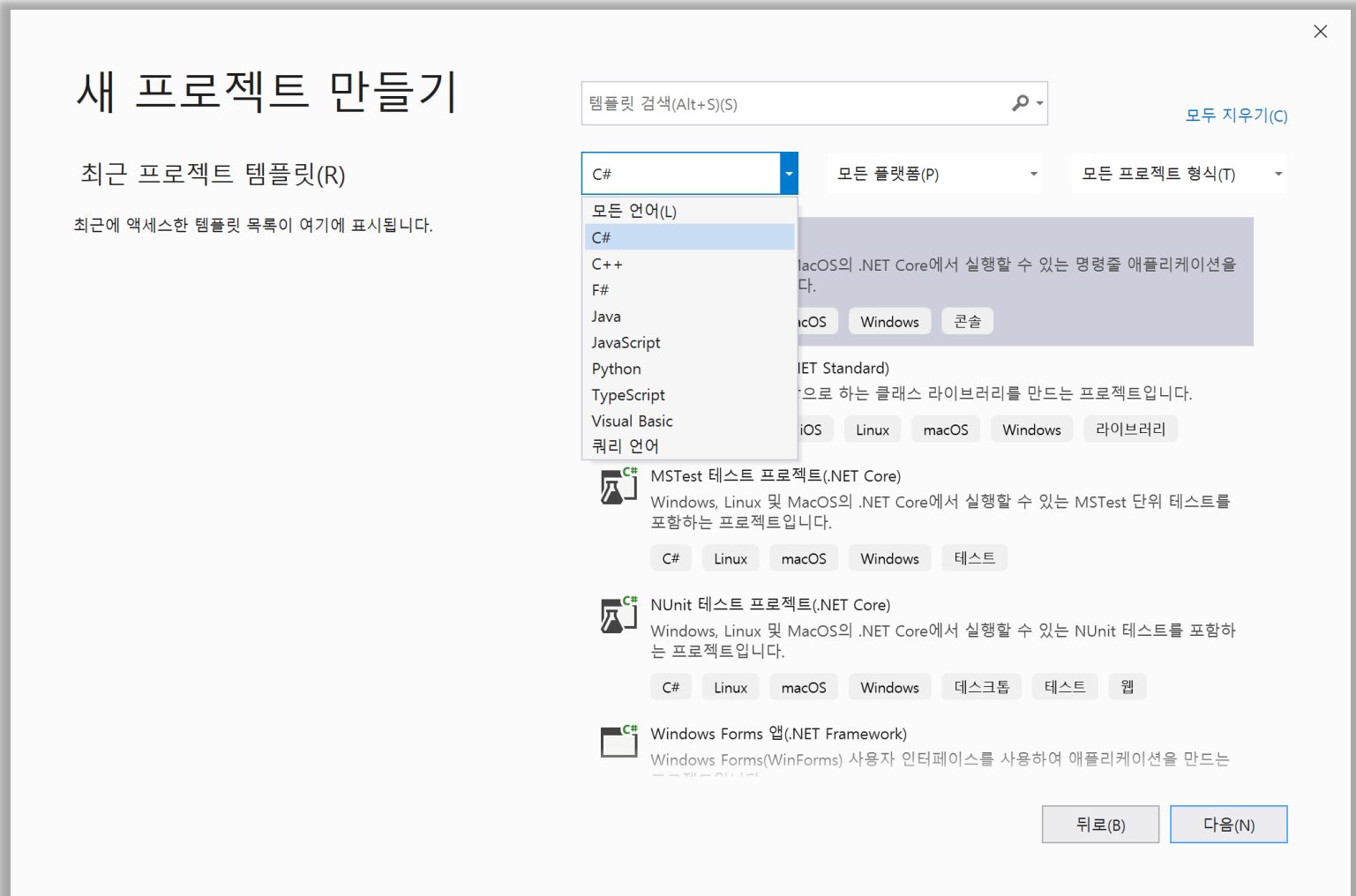
### ■ 프로젝트 형식

□ 콘솔 앱

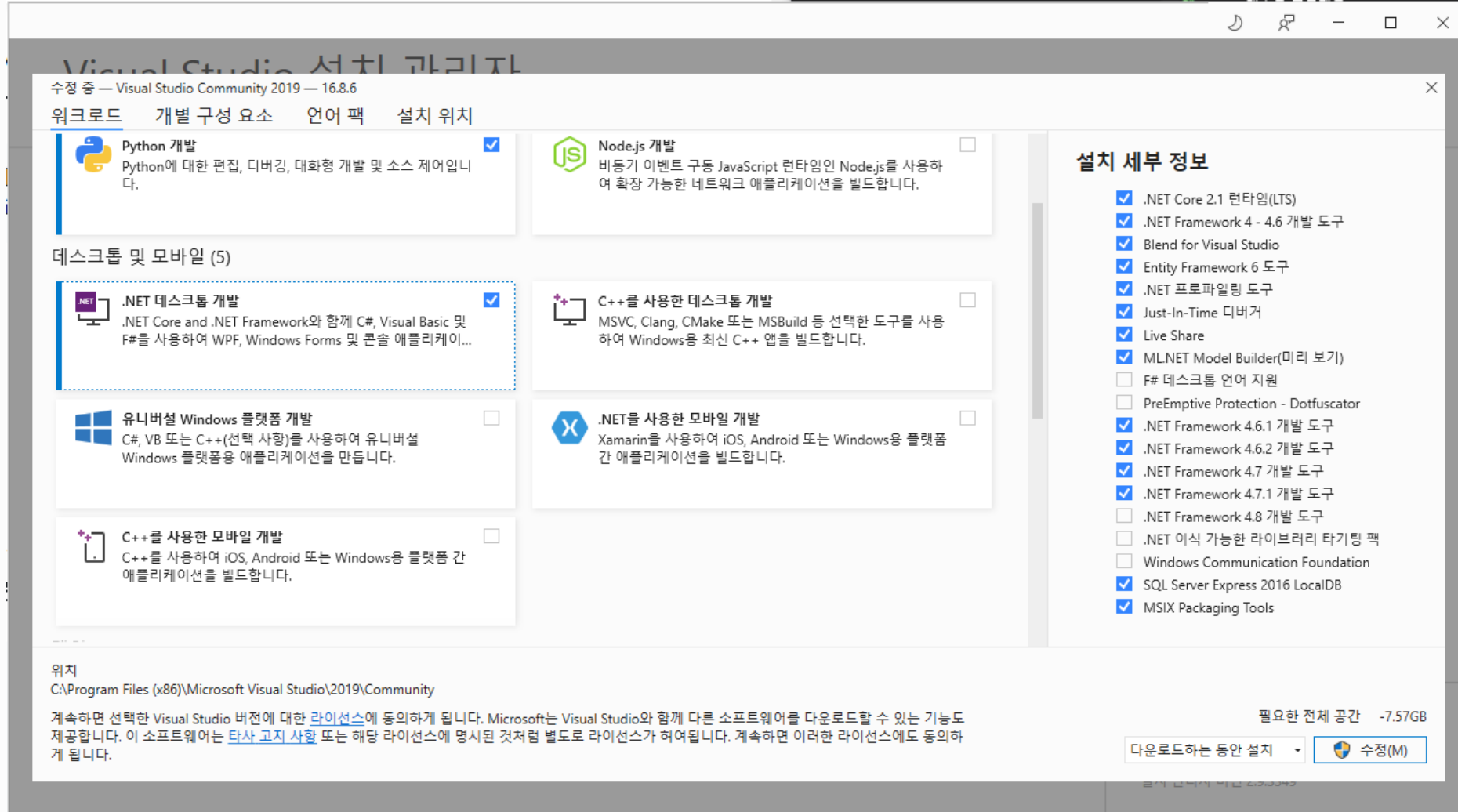
□ Windows Forms 앱

□ 클래스 라이브러리

□ ...



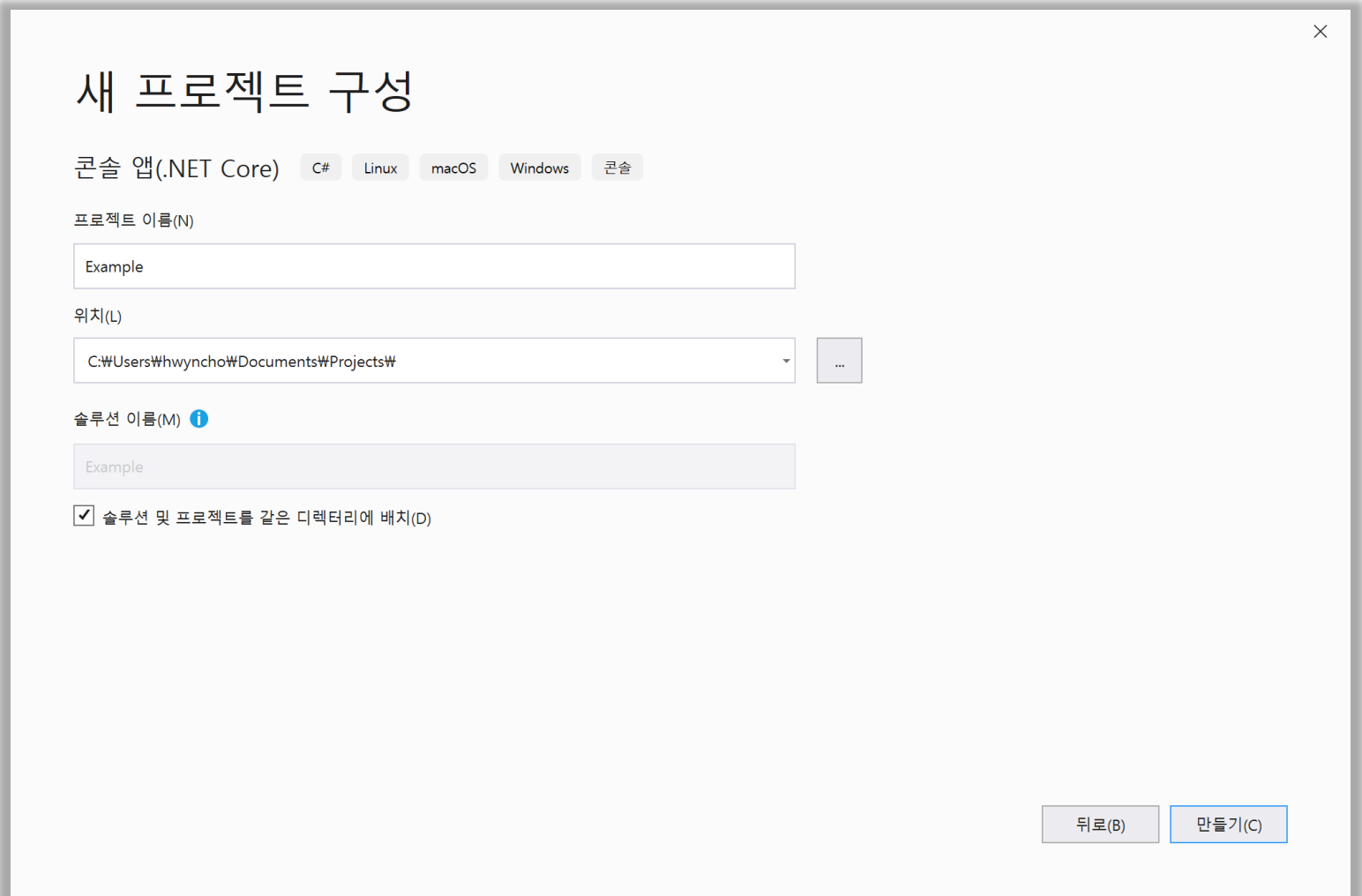
# Visual Studio와 C# (cont`d)



# Visual Studio와 C# (cont`d)

## □ 새 프로젝트 구성

- 프로젝트 이름
- 위치
- 솔루션 이름



새 프로젝트 구성

콘솔 앱(.NET Core) C# Linux macOS Windows 콘솔

프로젝트 이름(N)

Example

위치(L)

C:\Users\hwyncho\Documents\Projects\

솔루션 이름(M) ⓘ

Example

☒ 솔루션 및 프로젝트를 같은 디렉터리에 배치(D)

뒤로(B) 만들기(C)

# Visual Studio와 C# (cont`d)

## □ 코드 작성

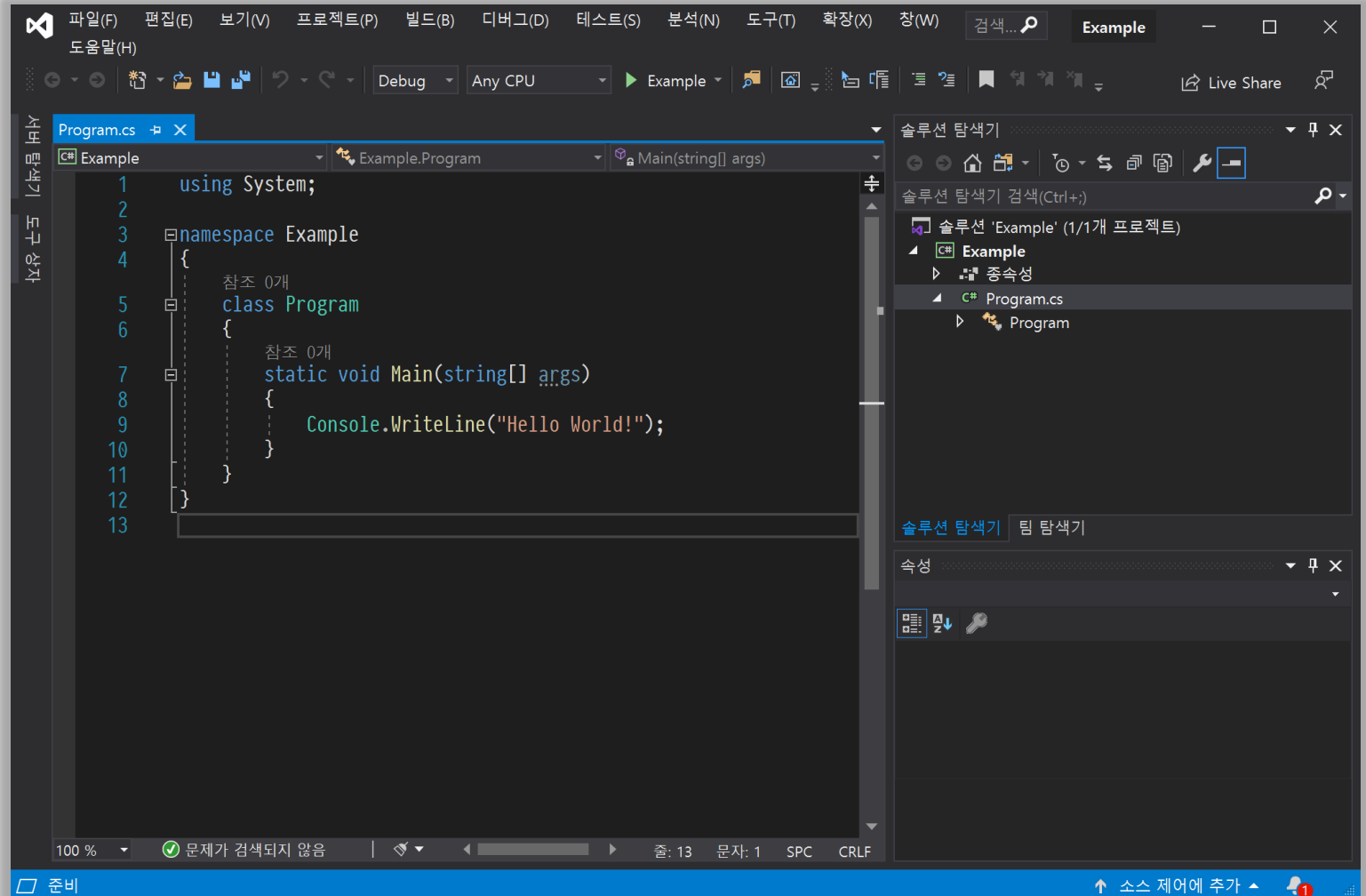
### ■ 코드 편집기

□ 구현하고자 하는 코드를 작성함.

### ■ 솔루션 탐색기, 클래스 뷰

□ 프로젝트의 구조를 보여줌.

□ 코드의 구조를 보여줌.



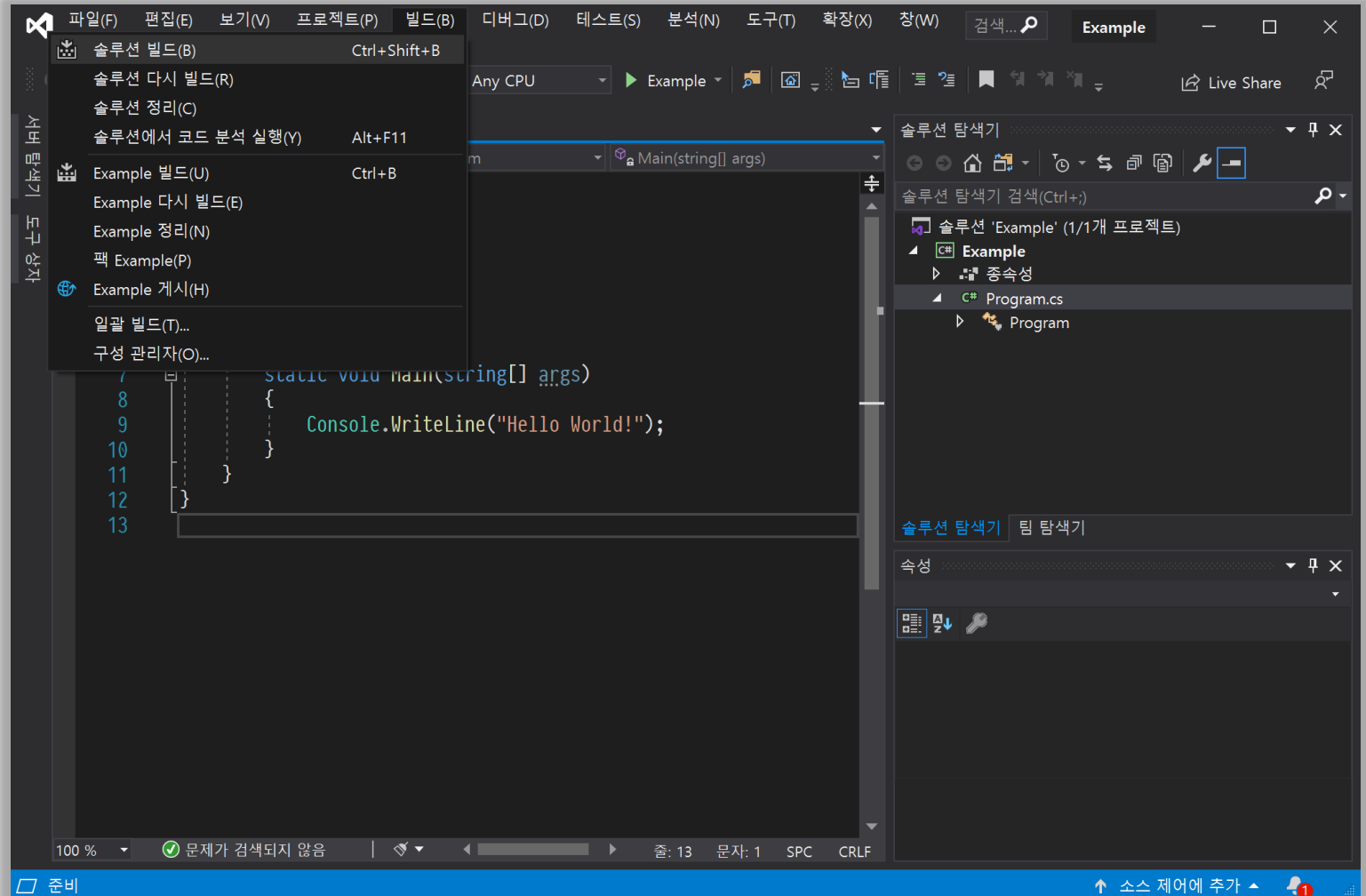


# Visual Studio와 C# (cont`d)

## ☐ 빌드 및 컴파일

### ☒ 빌드

#### ☐ 솔루션 빌드

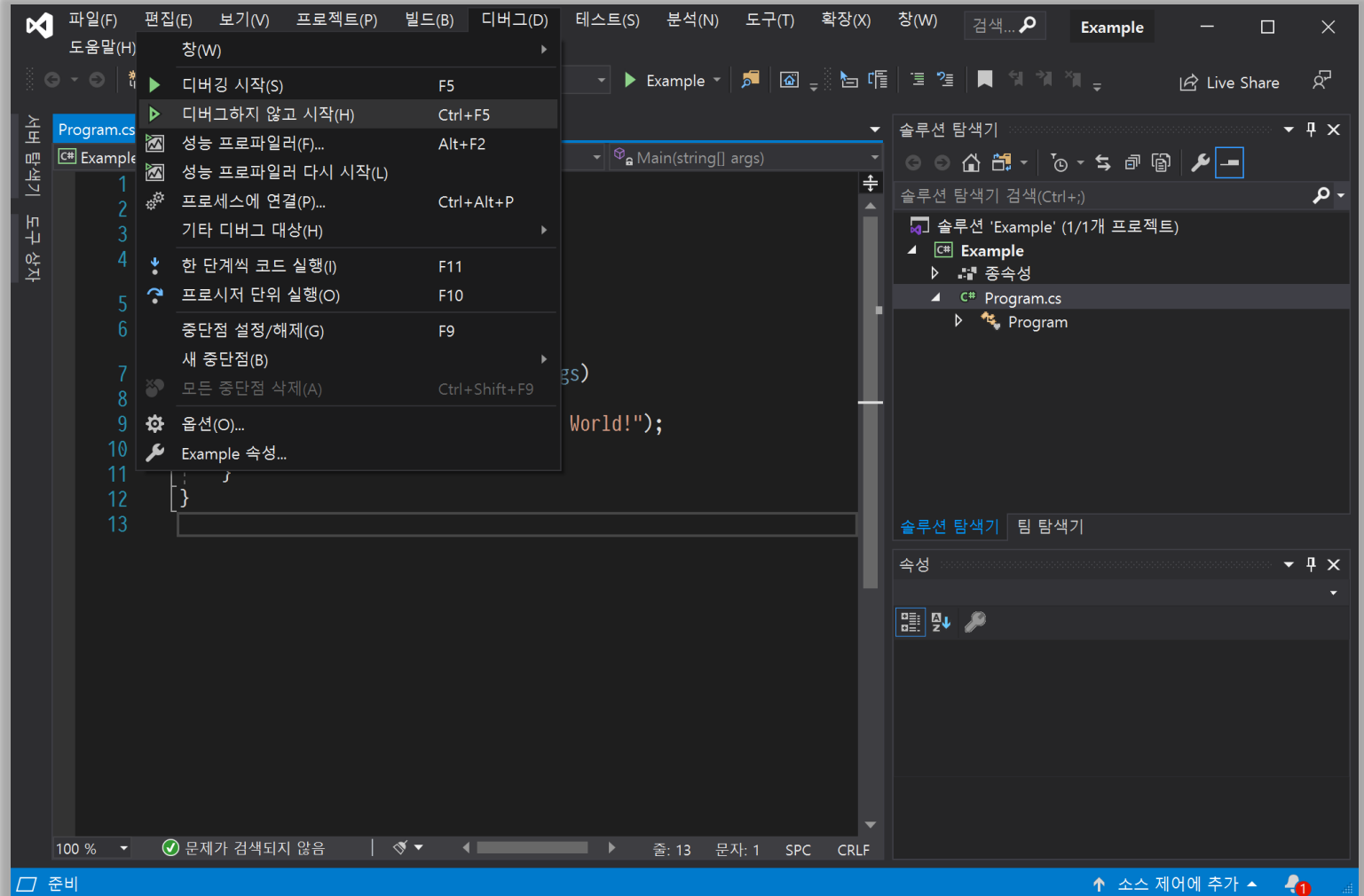


# Visual Studio와 C# (cont`d)

## □ 디버깅 및 실행

### ■ 디버그

- 디버깅 시작
- 디버그하지 않고 시작



# Visual Studio와 C# (cont`d)

## □ 프로젝트 디렉토리 및 파일

### ■ .sln 파일

- 솔루션의 정보를 보관하는 파일

### ■ .csproj 파일

- 프로젝트의 정보를 보관하는 파일

### ■ .cs 파일

- C# 코드를 보관하는 파일

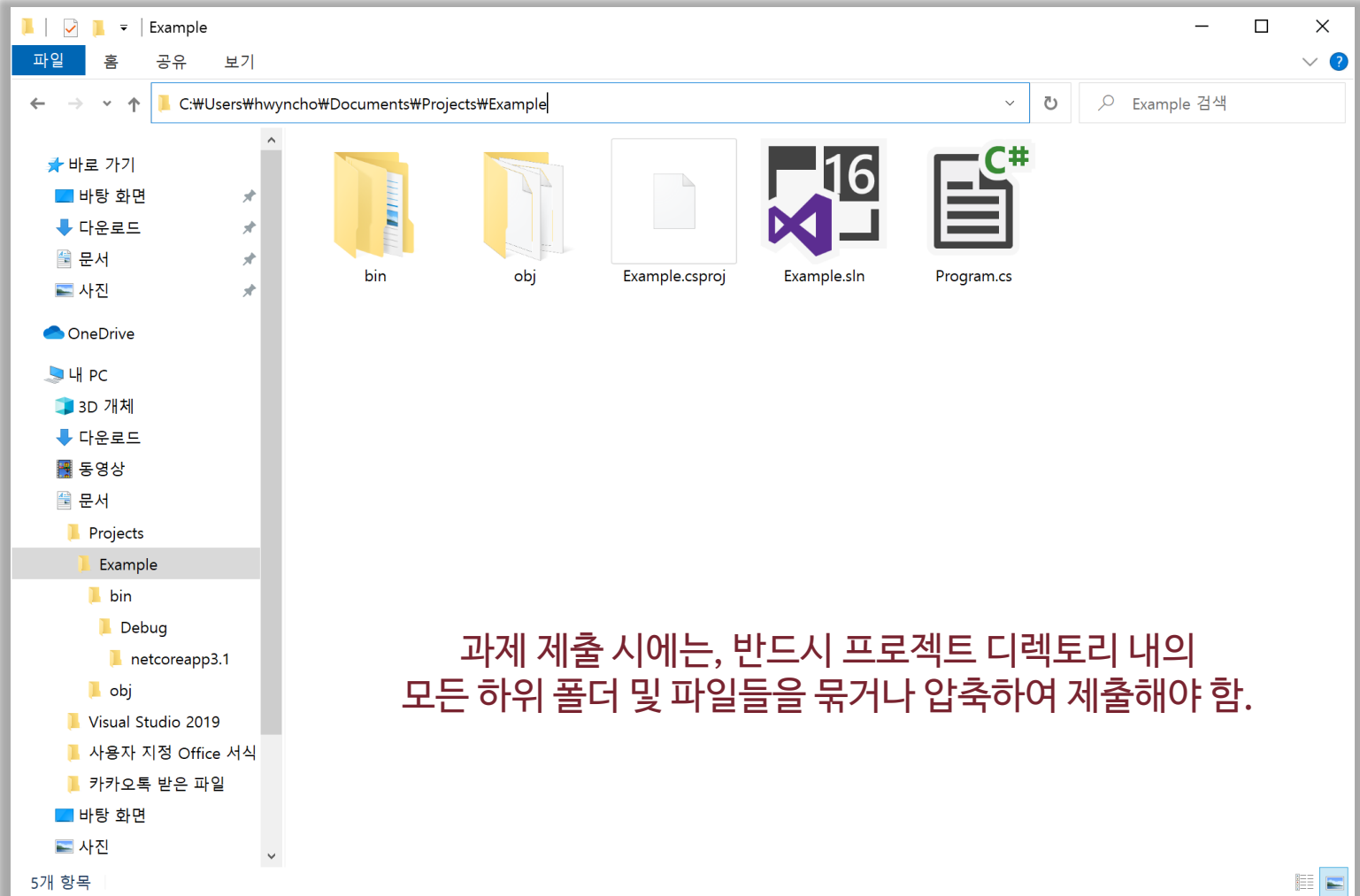
### ■ obj 폴더

- 컴파일 된 바이너리 파일을 보관하는 디렉토리

### ■ bin 폴더

- 링크 과정이 된 실행 파일을 보관하는 디렉토리

### ■ ...

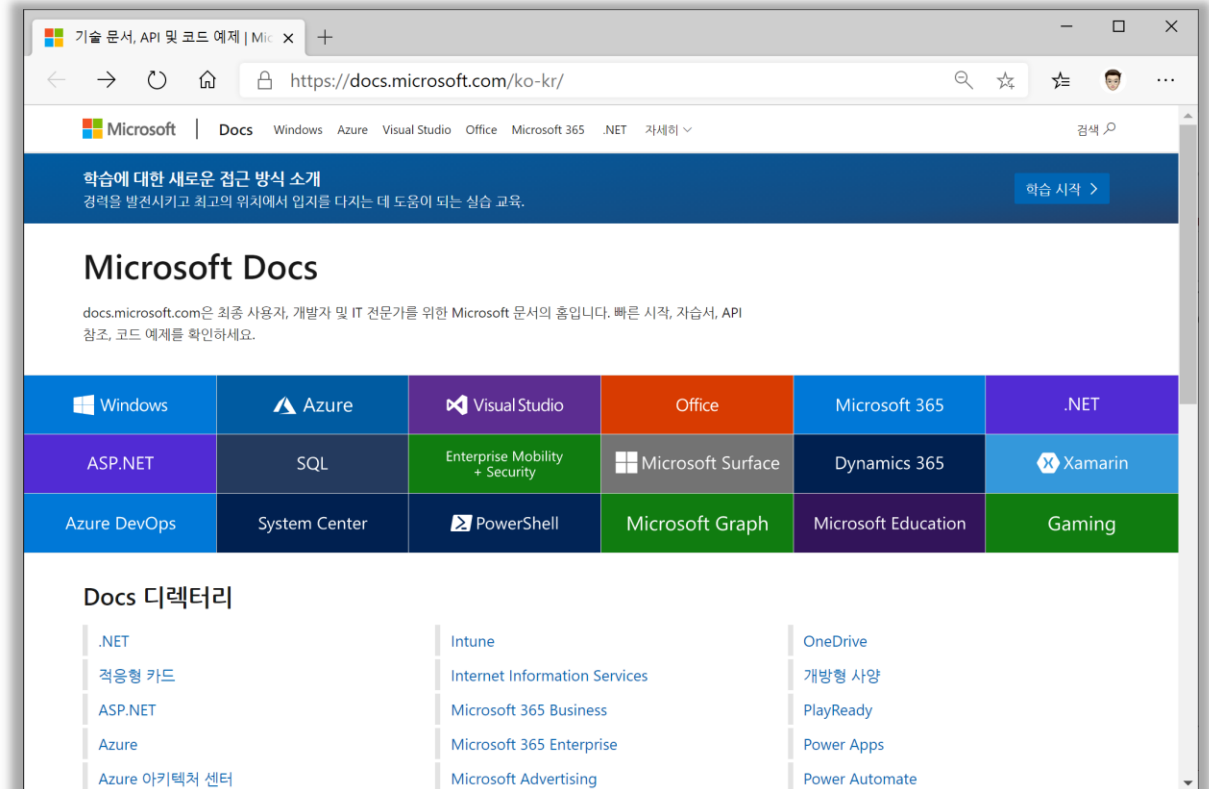


과제 제출 시에는, 반드시 프로젝트 디렉토리 내의 모든 하위 폴더 및 파일들을 묶거나 압축하여 제출해야 함.

# Visual Studio 도움말

## □ MSDN

- Microsoft Developer Network Library
- <https://msdn.microsoft.com/ko-kr>
- Visual Studio에서 F1 키를 입력하면, 검색이 가능함.

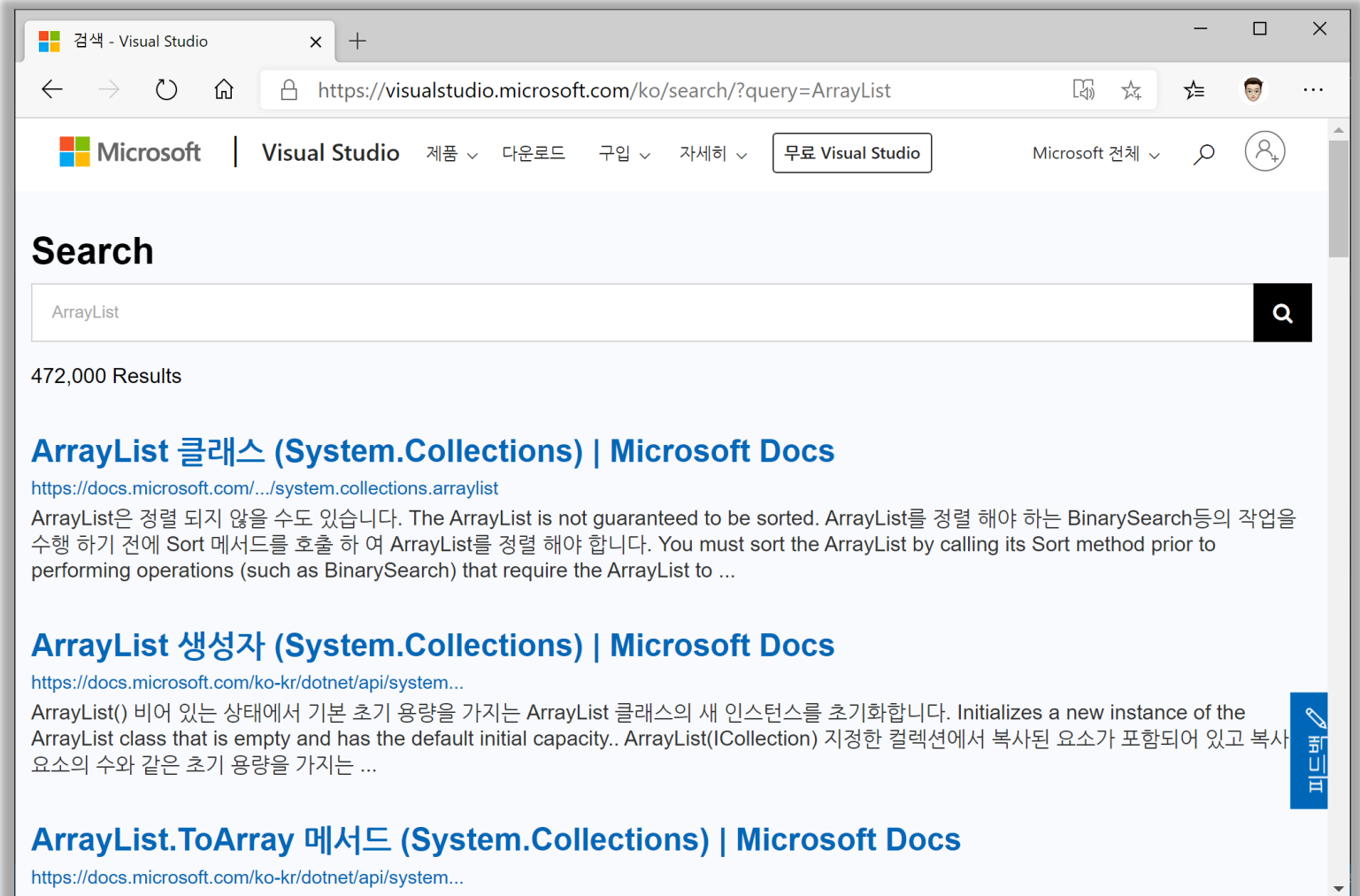


# Visual Studio 도움말 (cont`d)

## ☐ 검색

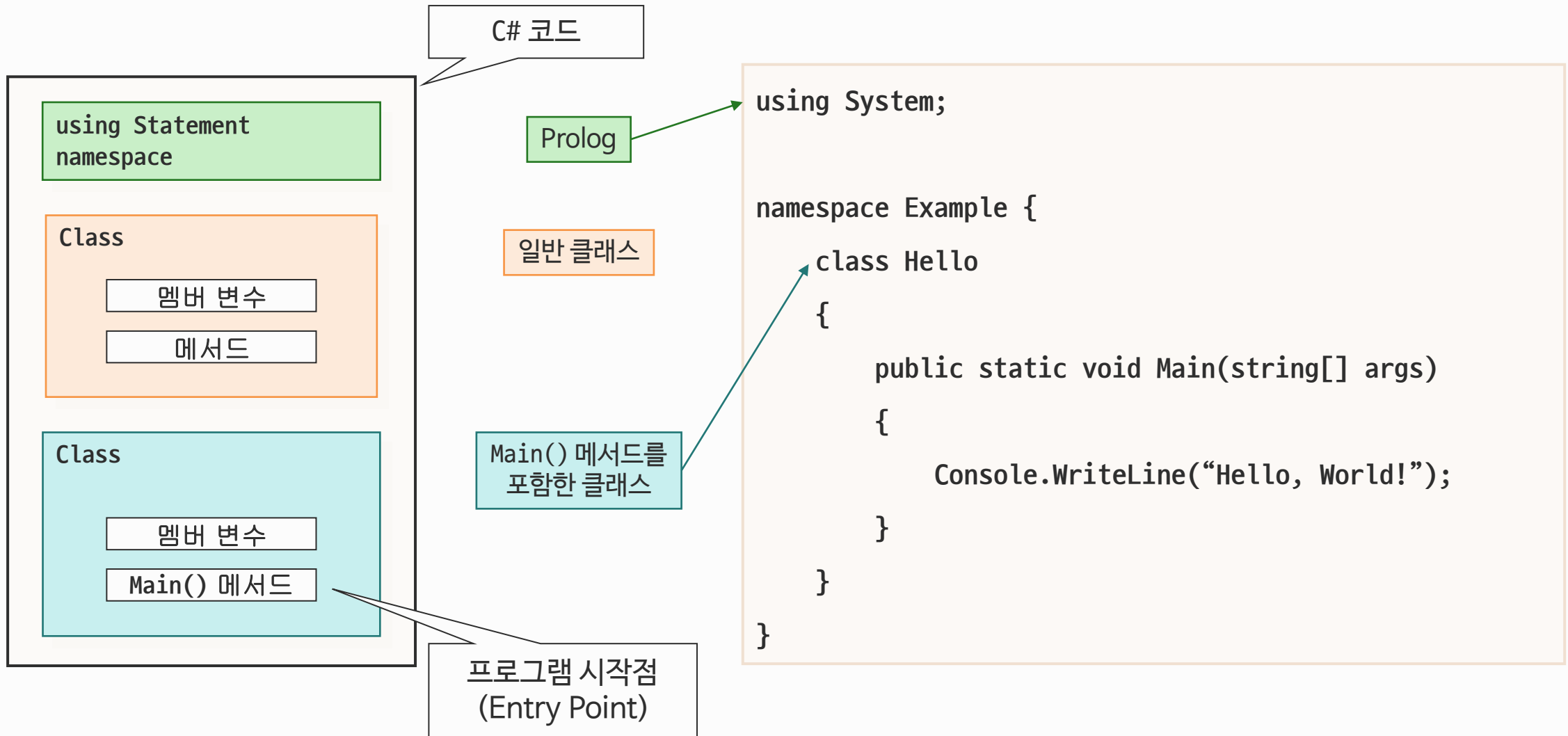
■ 우측 상단에 검색어를 입력

☐ 예시) ArrayList



# C# 프로그래밍 기본

---



## □ 클래스 (Class)

### ■ 기본 문법

```
class 클래스명  
{  
    /* 멤버 변수와 메소드 등 */  
}
```

- 하나의 C# 코드 파일 내에 하나 이상의 클래스가 존재할 수 있음.
- `partial` 키워드를 사용하여 하나의 클래스를 여러 개의 코드 파일에 걸쳐 정의할 수 있음.



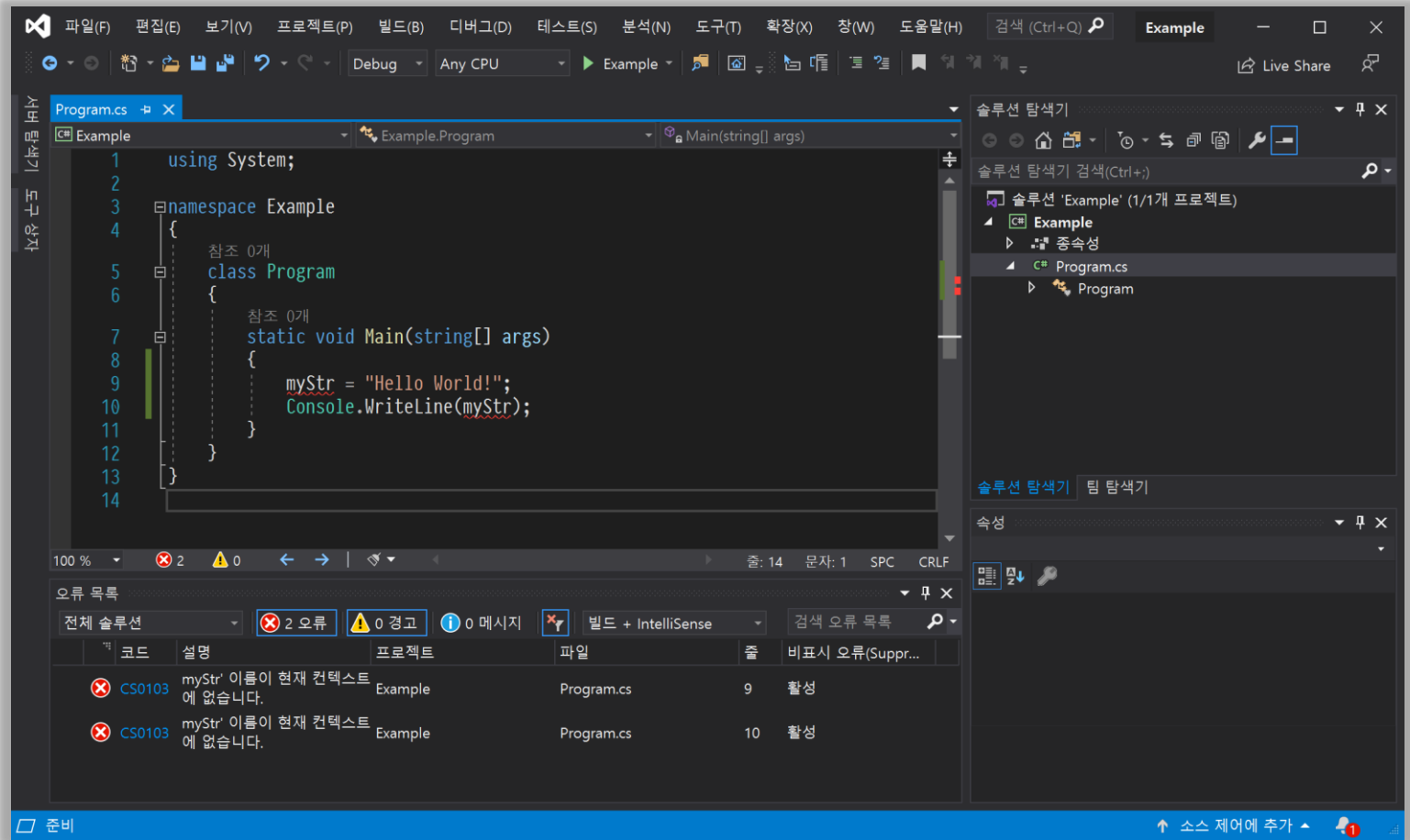
## ☐ Main() 함수

- 프로그램의 시작점이 되는 함수
- `public static void`로 선언해야 함.
- `main()`이 아닌 `Main()`임을 반드시 확인!

## ☐ Prolog

- `using Statements`와 `namespace` 부분을 포함
- `namespace`
  - ☐ 하나 이상의 클래스를 그룹화 한 단위를 `namespace`라고 함.
- 다른 `namespace`의 클래스에 접근하고자 하면, `using`을 사용하여 해당 `namespace`에 접근함.
  - ☐ 예시의 `Console.WriteLine()`은 `System` namespace를 `using System`으로 접근함으로써 사용 가능함.

- 구문 에러 (Syntax errors)
  - 기본 문법에 대한 규칙을 위반함.

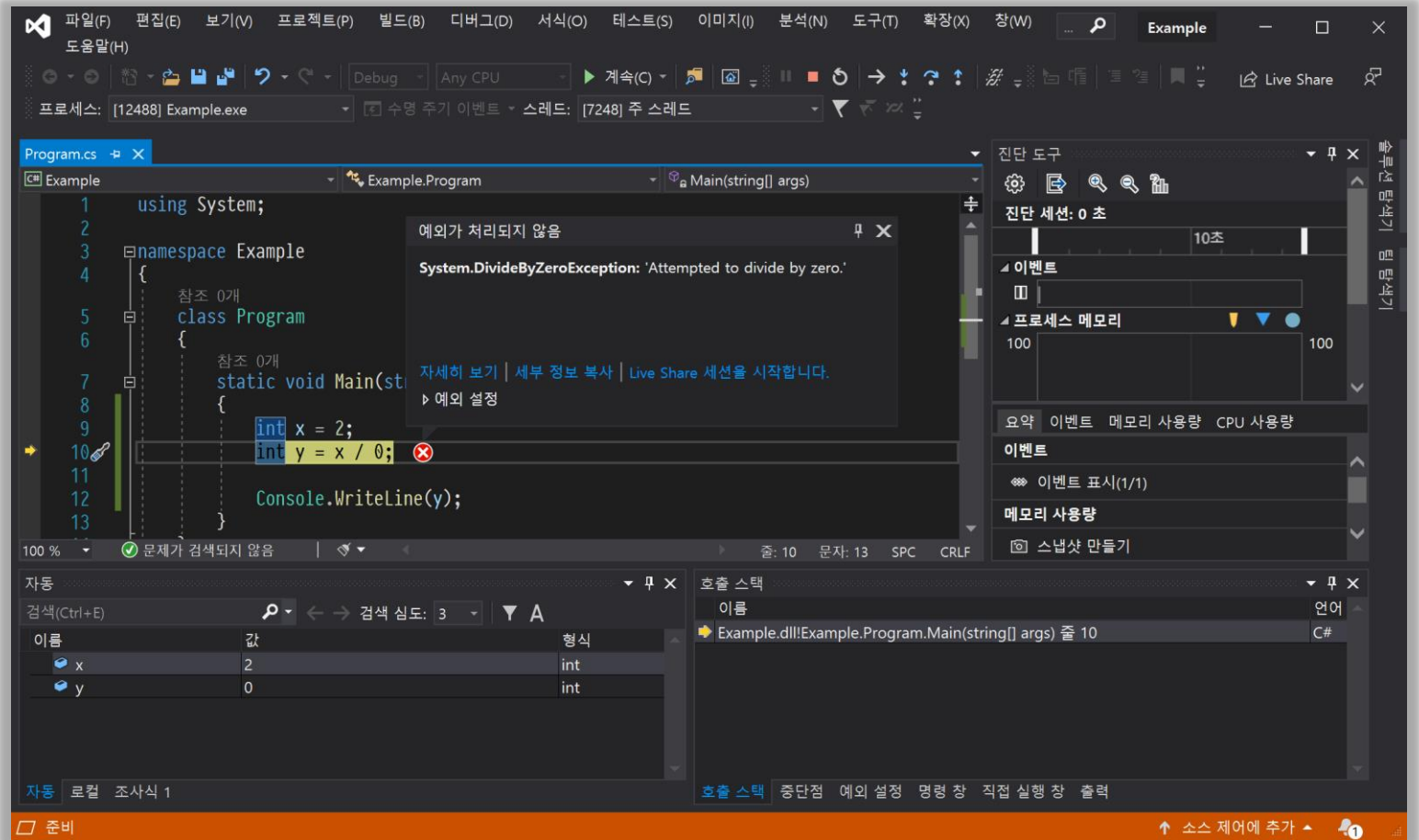


# 에러 (cont `d)

## □ 런타임 에러 (Runtime error)

■ 프로그램을 실행하는 도중에 발생함.

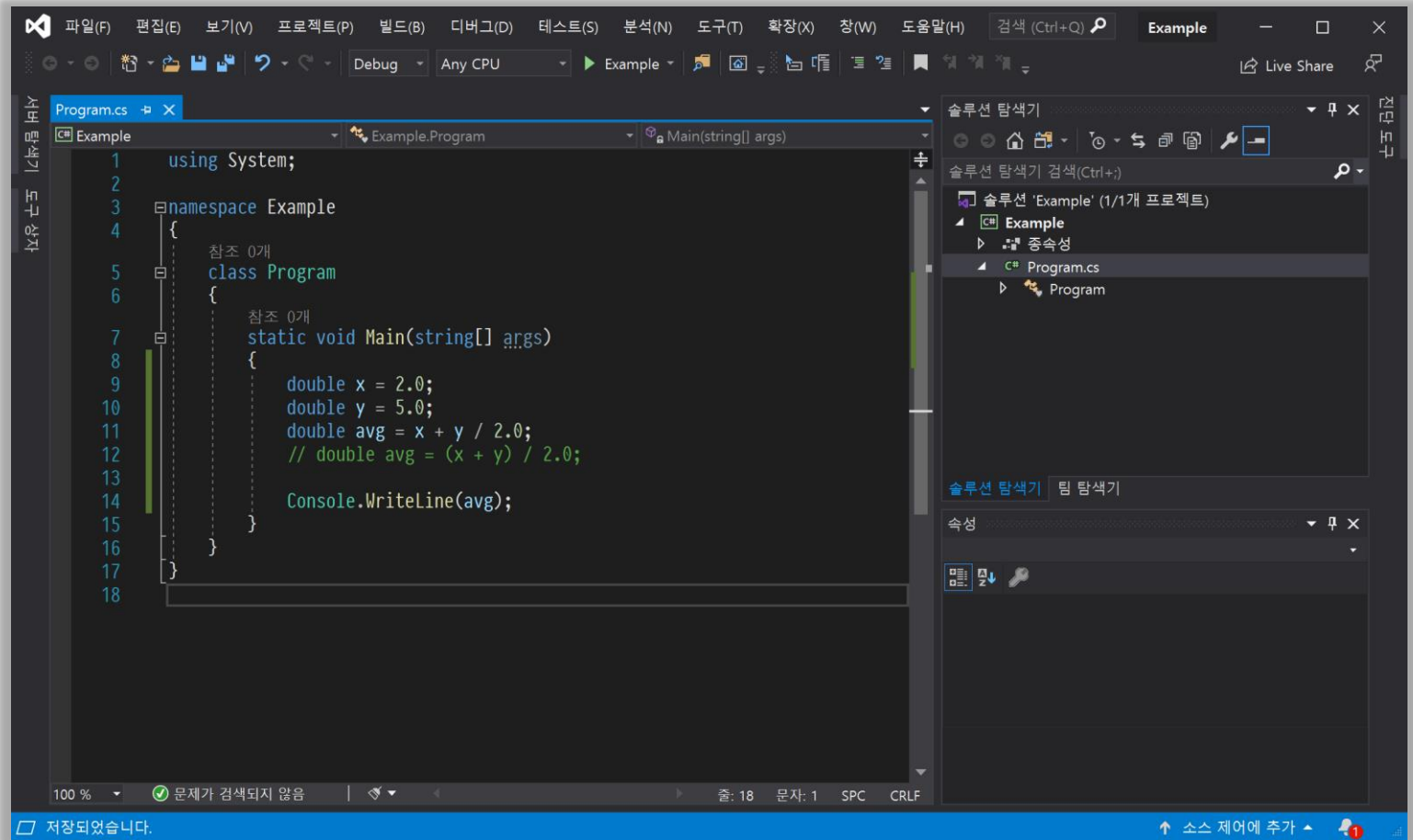
- 0으로 나누기
- 음수의 제곱근 구하기
- ...



# 에러 (cont `d)

## □ 로직 에러 (Logic error)

- 프로그램은 동작하지만, 실행 결과가 의도한 결과가 아닌 잘못된 결과가 나온 경우



## ☐ 디버그 (Debug)

- 프로그램의 다양한 오류를 찾고, 이를 수정하는 것
- 구문 오류는 Visual Studio 내의 코드 편집기에서 확인할 수 있음.
- 런타임 오류는 Visual Studio 내의 디버그 기능을 이용하여 확인할 수 있음.
- 로직 오류는 개발자 스스로 원인을 찾고 해결해야 함.
  - ☐ Visual Studio 내의 디버그 기능을 활용하면 변수의 현재 값, 함수의 반환 값 등을 확인할 수 있음.

## ☐ 변수, 함수, 객체 등의 명명 조건과 규칙

- 반드시 영문자 또는 밑줄(\_)로 시작해야 함.
- 영문자, 숫자, 밑줄을 포함할 수 있음.
- 공백( ) 또는 구두점(.)은 포함할 수 없음.
- 예약어는 사용할 수 없음.

## ☐ 보편적인 명명 규칙

### ■ Camel 표기법

#### ☐ 예시

- `youProbablyUseLowerCamelCase`
- `YouProbablyUseUpperCamelCase`

### ■ Snake 표기법

#### ☐ 예시

- `use_a_lot_of_underscores`
- `Typically_Use_A_Lot_Of_Underscores`

### ■ Hungarian 표기법

#### ☐ 예시

- `bFlag`
- `iNum`

# 명명 규칙 (Naming Rules) (cont`d)

## □ C#의 예약어

abstract	as	base	bool	break	byte	case	catch	char
checked	class	const	continue	decimal	default	delegate	do	double
else	enum	event	explicit	extern	false	finally	fixed	float
for	foreach	goto	if	implicit	in	int	interface	internal
is	lock	long	namespace	new	null	object	operator	out
override	params	private	protected	public	readonly	ref	return	sbyte
sealed	short	sizeof	stackalloc	static	string	struct	switch	this
throw	true	try	typeof	uint	ulong	unchecked	unsafe	ushort
using	virtual	void	while					

# 명명 규칙 (Naming Rules) (cont`d)

## ☐ 올바른 명명의 예시

- `_variable`
- `int_number`
- `name2`
- `doubleVar`
- `int2float`

## ☐ 잘못된 명명의 예시

- `max number`
  - ☐ 공백 사용
- `my.var`
  - ☐ 구두점 사용
- `2name`
  - ☐ 숫자로 시작

## ☐ 의미가 있는 명명의 예시

- `chCharacter`
- `iNumber`
- `pVar`
- `strName`
- `formMain`
- `formSub`
- `labelMessage`
- `btnExit`



## ☐ C#에서의 자료형

- System이라는 namespace에 정의되어 있음.
- System.object에서 파생된 System.Type을 상속함.

## ☐ 정수 범위의 자료형

타입	실제 이름	범 위	크기
Sbyte	System.Sbyte	-128 이상 127 이하	1
byte	System.Byte	0 이상 255 이하	1
char	System.Char	하나의 유니코드 문자. U+0000 이상 U+FFFF 이하	2
short	System.Int16	-23,768 이상 32,767 이하	2
ushort	System.Uint16	0 이상 65,535 이하	2
int	System.Int32	-2,147,483,648 이상 2,147,483,647 이하	4
uint	System.Uint32	0 이상 4,294,967,295 이하	4
long	System.Int64	-9,223,372,036,854,775,808 이상 9,223,372,036,854,775,807 이하	8
ulong	System.Uint64	0 이상 18,446,744,073,709,551,615 이하	8

# 자료형 (Data Type) (cont'd)

## □ 실수 범위의 자료형

타입	실제 이름	정밀도	범 위	크기
float	System.Single	7개의 자릿수	$\pm 1.5 \times 10^{-45} \sim \pm 3.4 \times 10^{38}$	4
double	System.Double	15~16개의 자릿수	$\pm 5.0 \times 10^{-324} \sim \pm 1.7 \times 10^{308}$	8
decimal	System.Decimal	28~29개의 자릿수	$1.0 \times 10^{-28} \sim \pm 7.9 \times 10^{28}$	16

## □ 그 외의 자료형

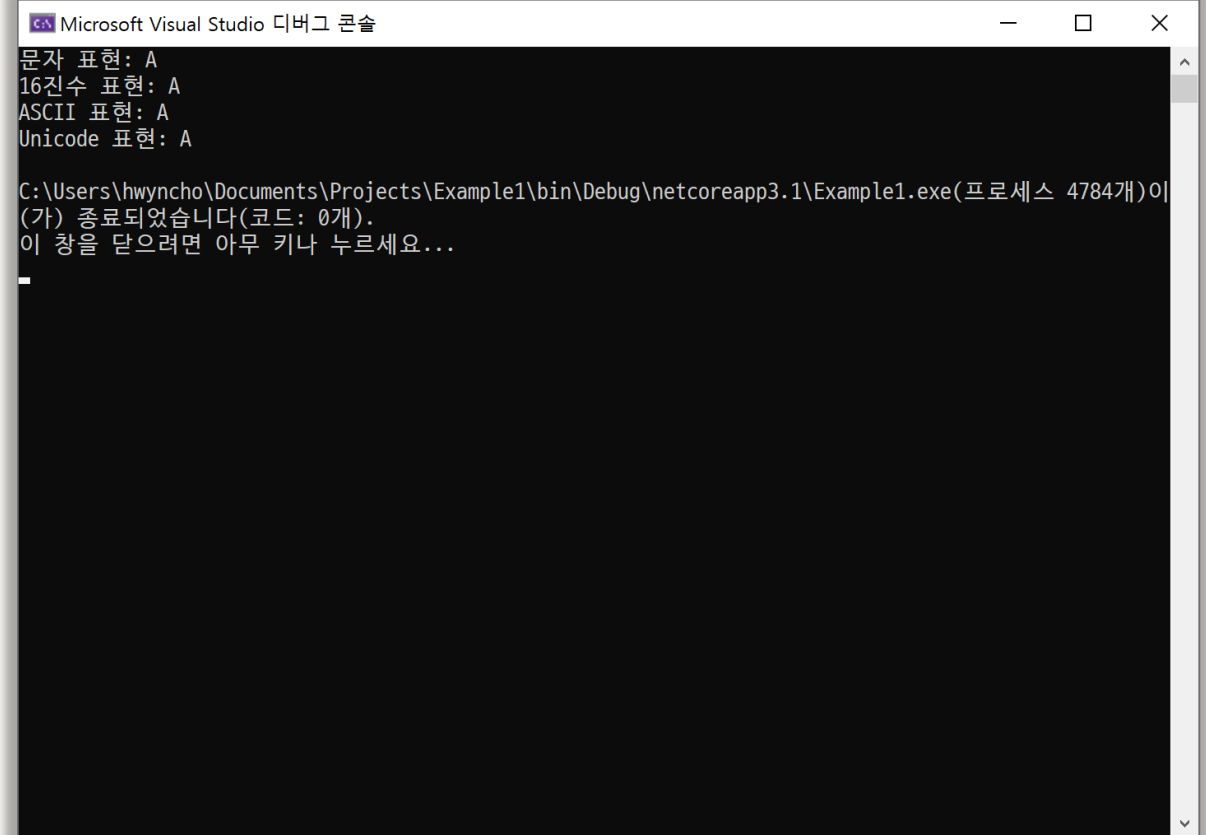
타입	실제 이름	범 위
Object	System.Object	모든 타입의 최상위 부모 클래스 C#에서 모든 객체들은 System.Object 클래스를 상속함.
string	System.String	문자열을 나타내는 타입
bool	System.Boolean	Boolean 값 참(true) 또는 거짓(false)을 나타냄

# 자료형 예시 1

```
using System;

namespace Example1
{
    class Program
    {
        static void Main(string[] args)
        {
            char chVar1 = 'A';
            char chVar2 = '\x0041';
            char chVar3 = (char)65;
            char chVar4 = '\u0041';

            Console.WriteLine("문자 표현: {0}", chVar1);
            Console.WriteLine("16진수 표현: {0}", chVar2);
            Console.WriteLine("ASCII 표현: {0}", chVar3);
            Console.WriteLine("Unicode 표현: {0}", chVar4);
        }
    }
}
```



Microsoft Visual Studio 디버그 콘솔

문자 표현: A  
16진수 표현: A  
ASCII 표현: A  
Unicode 표현: A

C:\Users\hwyncho\Documents\Projects\Example1\bin\Debug\netcoreapp3.1\Example1.exe(프로세스 4784개)이  
(가) 종료되었습니다(코드: 0개).  
이 창을 닫으려면 아무 키나 누르세요...

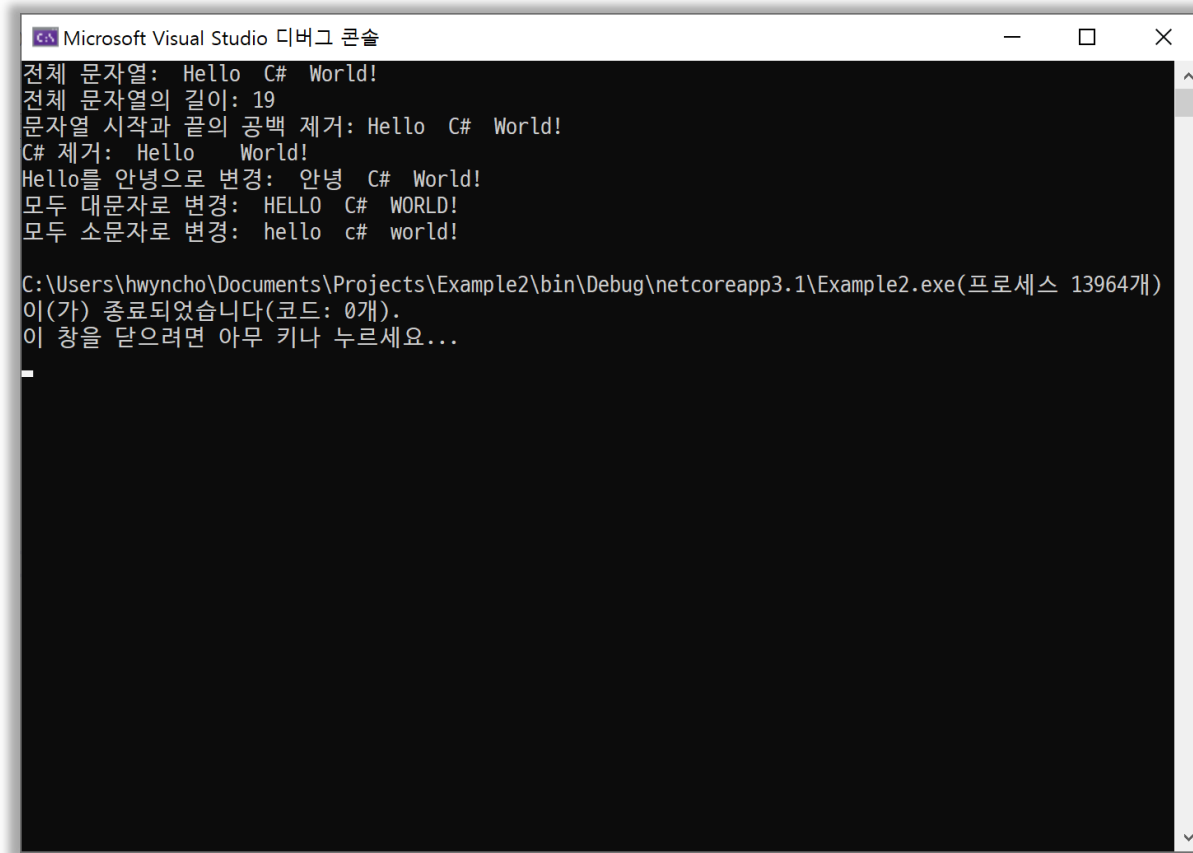
# 자료형 예시 2

```
using System;

namespace Example2
{
    class Program
    {
        static void Main(string[] args)
        {
            string strText1 = " Hello ";
            string strText2 = " C# ";
            string strText3 = " World! ";
            string strText4 = strText1 + strText2 + strText3;

            Console.WriteLine("전체 문자열: {0}", strText4);
            Console.WriteLine("전체 문자열의 길이: {0}", strText4.Length);
            Console.WriteLine("문자열 시작과 끝의 공백 제거: {0}", strText4.Trim());
            Console.WriteLine("C# 제거: {0}", strText4.Remove(8, 2));
            Console.WriteLine("Hello를 안녕으로 변경: {0}", strText4.Replace("Hello", "안녕"));
            Console.WriteLine("모두 대문자로 변경: {0}", strText4.ToUpper());
            Console.WriteLine("모두 소문자로 변경: {0}", strText4.ToLower());
        }
    }
}
```

# 자료형 예시 2 (cont`d)



```
Microsoft Visual Studio 디버그 콘솔
전체 문자열: Hello C# World!
전체 문자열의 길이: 19
문자열 시작과 끝의 공백 제거: Hello C# World!
C# 제거: Hello World!
Hello를 안녕으로 변경: 안녕 C# World!
모두 대문자로 변경: HELLO C# WORLD!
모두 소문자로 변경: hello c# world!

C:\Users\hwyncho\Documents\Projects\Example2\bin\Debug\netcoreapp3.1\Example2.exe(프로세스 13964개)
이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

## □ 변수

- 데이터의 저장에 관련된 것
- 데이터를 읽거나 쓰는 것이 가능하고, 참조만 하는 것도 역시 가능함.

## □ 변수의 선언 방법

- `[modifiers] datatype identifier;`

```
// ...  
int a = 11; // System.Int32 a = 11; 하고 동일  
public static int b = 10, c = 20;  
  
i = 11; // i 선언이 없으므로 에러 발생  
int j = 10; int j = 20; // 두 번 선언으로 인한 에러  
// ...
```

# 변수 (Variable) (cont'd)

## □ 변수의 기본값

■ 변수를 선언할 때 초기값을 지정하지 않으면, 기본값으로 초기화 됨.

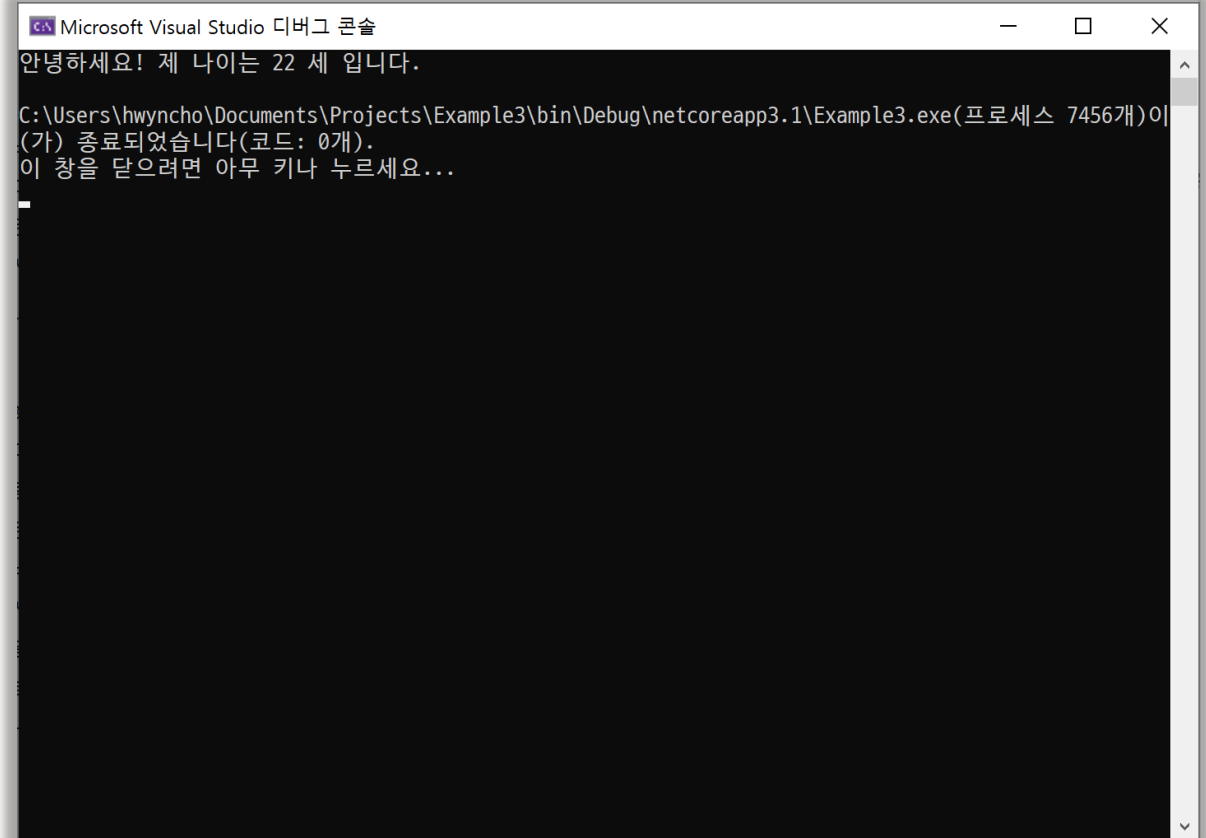
자료형	기본값	자료형	기본값
bool	false	byte	0
char	'\0'	decimal	0.0M
double	0.0D	enum	0
float	0.0F	int	0
long	0L	sbyte	0
short	0	struct	구조체 내의 모든 멤버는 해당 자료형의 기본값으로 초기화
uint	0	ulong	0
ushort	0		

# 변수 예시

```
using System;

namespace Example3
{
    class Program
    {
        static void Main(string[] args)
        {
            int iAge = 22;
            string strText1 = "안녕하세요!";
            string strText2 = "제 나이는";
            string strText3 = "세 입니다.";

            Console.WriteLine("{0} {1} {2} {3}", strText1,
                                strText2,
                                iAge,
                                strText3);
        }
    }
}
```



The screenshot shows the 'Microsoft Visual Studio 디버그 콘솔' (Microsoft Visual Studio Debugger Console) window. It displays the output of the program: '안녕하세요! 제 나이는 22 세 입니다.' (Hello! My age is 22 years old). Below this, it shows the file path 'C:\Users\hwyncho\Documents\Projects\Example3\bin\Debug\netcoreapp3.1\Example3.exe (프로세스 7456개)' and the status '(가) 종료되었습니다(코드: 0개)' (The program has ended with code 0). A prompt '이 창을 닫으려면 아무 키나 누르세요...' (Press any key to close this window...) is visible at the bottom.



## □ 연산자

- 변수의 값을 다양하게 조작할 수 있음.

## □ 산술 연산자

- 계산식을 처리하는 연산자

연산자	예시	결과
+	<code>a = b + c;</code>	b와 c 값의 합이 a에 배정
-	<code>a = b - c;</code>	b와 c 값의 차가 a에 배정
*	<code>a = b * c;</code>	b와 c 값의 곱이 a에 배정
/	<code>a = b / c;</code>	b를 c로 나눈 값이 a에 배정
%	<code>a = b % c;</code>	b를 c로 나눈 값의 나머지가 a에 배정
<<	<code>a = b &lt;&lt; c;</code>	b를 c bit만큼 왼쪽으로 이동시켜 a에 배정
>>	<code>a = b &gt;&gt; c;</code>	b를 c bit만큼 오른쪽으로 이동시켜 a에 배정

# 연산자 (Operators) (cont`d)

## □ 증가, 감소 연산자

### ■ 변수의 값을 증가 또는 감소하는 연산자

연산자	예시	결과
++	a = ++b;	b 값이 1 증가한 후, a에 배정
--	a = --b;	b 값이 1 감소한 후, a에 배정
++	a = b++;	b 값이 a에 배정된 후, b 값이 1 증가
--	a = b--;	b 값이 a에 배정된 후, b 값이 1 감소

## □ 배정 연산자

### ■ 식을 대입하여 변수에 결과를 저장하는 연산자

연산자	예시	결과
=	a = b;	b의 값을 a에 배정
+=	a += b;	a와 b의 합을 a에 배정 (a = a + b;)
-=	a -= b;	a와 b의 차를 a에 배정 (a = a - b;)
*=	a *= b;	a와 b의 곱을 a에 배정 (a = a * b;)
/=	a /= b;	a를 b로 나눈 결과가 a에 배정 (a = a / b;)
%=	a %= b;	a를 b로 나눈 나머지가 a에 배정 (a = a % b;)

## □ 논리 연산자

- 참, 거짓을 비교하거나 판단하는 연산자
- 연산의 결과는 true 또는 false임.

연산자	예시	결과
!	!a	a가 false이면 true a가 true이면 false
&	a & b	a와 b가 모두 true인 경우에만 true (a가 false이어도 b를 평가)
&&	a && b	a와 b가 모두 true인 경우에만 true (a가 false이면 b를 평가하지 않음.)
	a   b	a 또는 b 하나라도 true이면 true (a가 true이어도 b를 평가)
	a    b	a 또는 b 둘 중 하나라도 true이면 true (a가 true이면 b를 평가하지 않음)
^	a ^ b	a와 b 값이 다른 경우에만 true

# 연산자 (Operators) (cont`d)

## □ 비교 연산자

- 값의 크기를 비교하는 연산자
- 연산의 결과는 true 또는 false임.

연산자	예시	결과
==	a == b;	a가 b와 같으면 true, 같지 않으면 false 반환
!=	a != b;	a가 b와 같으면 false, 같지 않으면 true 반환
<	a < b;	a가 b보다 작으면 true, 크거나 같으면 false 반환
>	a > b;	a가 b보다 작거나 같으면 false, 크면 true 반환
<=	a <= b;	a가 b보다 작거나 같으면 true, 크면 false 반환
>=	a >= b;	a가 b보다 크거나 같으면 true, 작으면 false 반환

# 연산자 (Operators) (cont`d)

## □ 연산자의 우선순위

■ 하나의 계산식에 여러 연산자들이 사용되면, 우선순위에 따라 연산자의 처리 순서를 판별함.

우선순위	범주	연산자
1	주 연산자	(x) xy f(x) a[x] x++ x- new typeof sizeof checked unchecked
2	단항 연산자	+ - ! ~ ++x --x (T)x
3	곱하기, 나누기	* / %
4	더하기, 빼기	+ -
5	shift	<< >>
6	관계 연산자	< > <= >= is
7	동등 연산자	== !=
8	논리곱 비트 연산자	&
9	배타적 논리합 비트 연산자	^
10	논리합 비트 연산자	
11	조건 논리합 연산자	&&
12	조건 논리곱 연산자	
13	삼항(ternary) 조건 연산자	?:
14	지정 연산자	= *= /= %= += -= <<= >>= &= ^=  =

## ☐ 연산자의 우선순위

### ■ 연산자가 동등한 우선순위에 있는 경우

- ☐ 대입 연산자는 오른쪽에서 왼쪽으로 연산 순서를 맞춤.
- ☐ 대입 연산자를 제외한 모든 연산자는 왼쪽에서 오른쪽으로 연산 순서를 맞춤.
- ☐ 산술 연산자는 대수학(algebra)의 규칙을 따름.

### ■ 연산자가 낮은 우선순위에 있는 경우

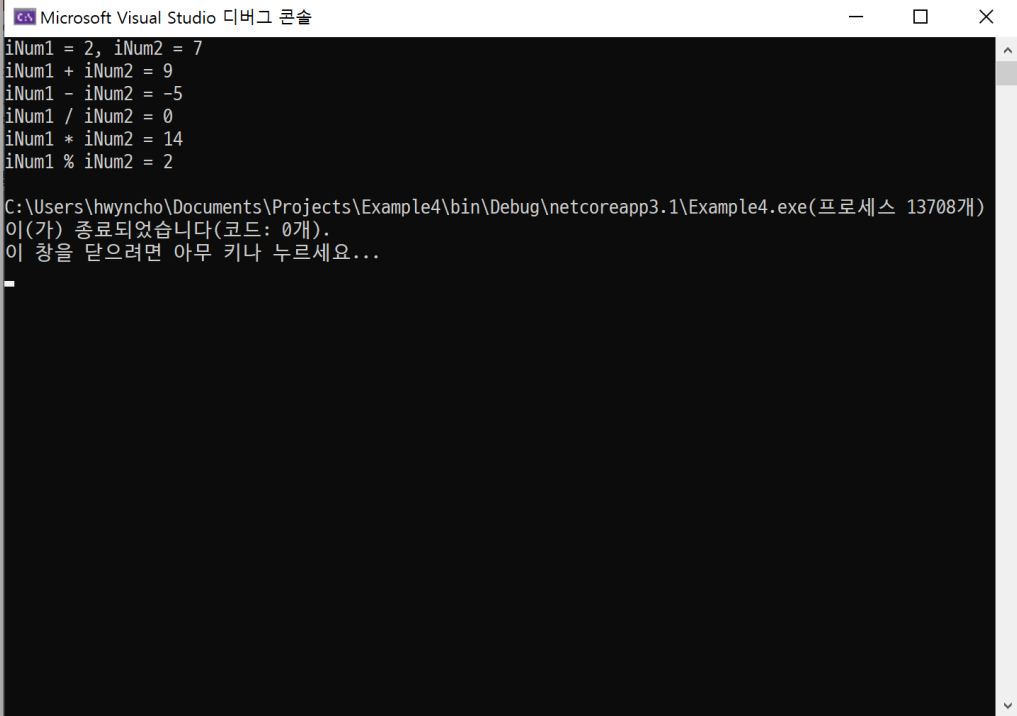
- ☐ 우선순위가 낮은 연산자를 먼저 처리하기 위해서는, ( )를 사용해야 함.

# 연산자 예시 1

```
using System;

namespace Example4
{
    class Program
    {
        static void Main(string[] args)
        {
            int iNum1 = 2;
            int iNum2 = 7;

            Console.WriteLine("iNum1 = {0}, iNum2 = {1}", iNum1, iNum2);
            Console.WriteLine("iNum1 + iNum2 = {0}", iNum1 + iNum2);
            Console.WriteLine("iNum1 - iNum2 = {0}", iNum1 - iNum2);
            Console.WriteLine("iNum1 / iNum2 = {0}", iNum1 / iNum2);
            Console.WriteLine("iNum1 * iNum2 = {0}", iNum1 * iNum2);
            Console.WriteLine("iNum1 % iNum2 = {0}", iNum1 % iNum2);
        }
    }
}
```



```
Microsoft Visual Studio 디버그 콘솔

iNum1 = 2, iNum2 = 7
iNum1 + iNum2 = 9
iNum1 - iNum2 = -5
iNum1 / iNum2 = 0
iNum1 * iNum2 = 14
iNum1 % iNum2 = 2

C:\Users\hwyncho\Documents\Projects\Example4\bin\Debug\netcoreapp3.1\Example4.exe(프로세스 13708개)
이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

# 연산자 예시 2

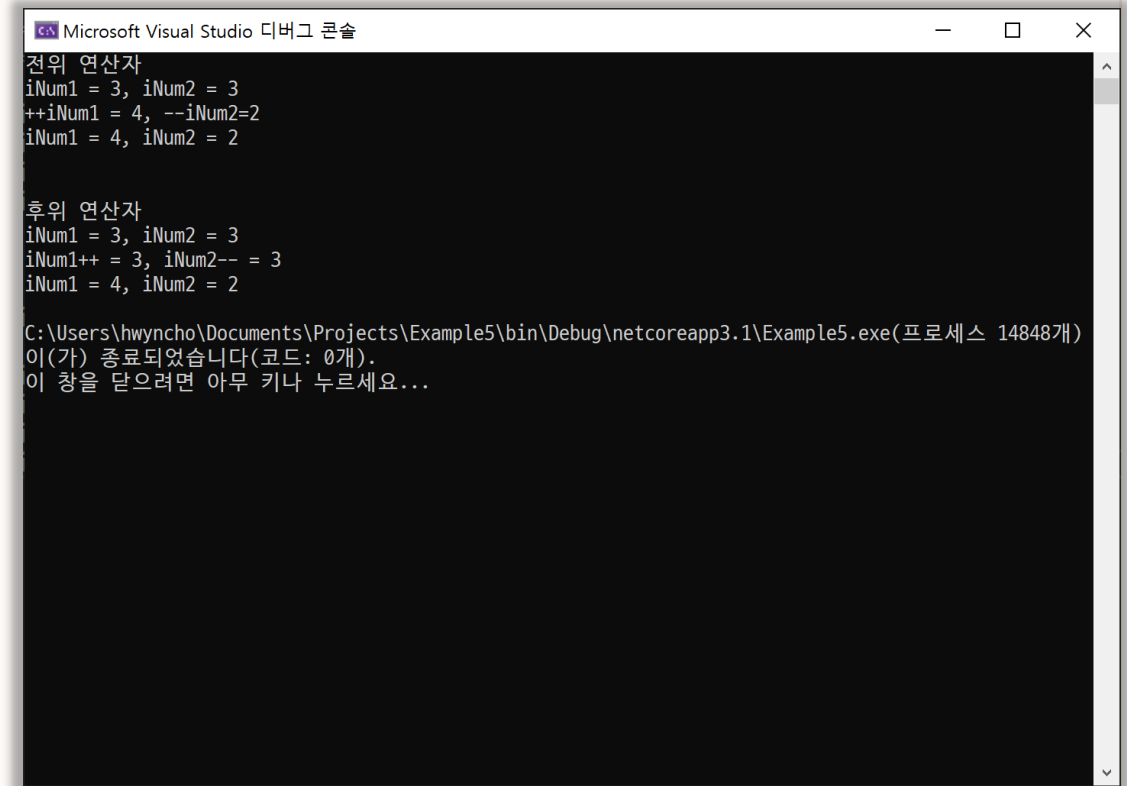
```
using System;

namespace Example5
{
    class Program
    {
        static void Main(string[] args)
        {
            int iNum1 = 3;
            int iNum2 = 3;

            Console.WriteLine("전위 연산자");
            Console.WriteLine("iNum1 = {0}, iNum2 = {1}", iNum1, iNum2);
            Console.WriteLine("++iNum1 = {0}, --iNum2 = {1}", ++iNum1, --iNum2);
            Console.WriteLine("iNum1 = {0}, iNum2 = {1}", iNum1, iNum2);

            iNum1 = 3;
            iNum2 = 3;
            Console.WriteLine("\n");

            Console.WriteLine("후위 연산자");
            Console.WriteLine("iNum1 = {0}, iNum2 = {1}", iNum1, iNum2);
            Console.WriteLine("iNum1++ = {0}, iNum2-- = {1}", iNum1++, iNum2--);
            Console.WriteLine("iNum1 = {0}, iNum2 = {1}", iNum1, iNum2);
        }
    }
}
```



```
Microsoft Visual Studio 디버그 콘솔

전위 연산자
iNum1 = 3, iNum2 = 3
++iNum1 = 4, --iNum2=2
iNum1 = 4, iNum2 = 2

후위 연산자
iNum1 = 3, iNum2 = 3
iNum1++ = 3, iNum2-- = 3
iNum1 = 4, iNum2 = 2

C:\Users\hwyncho\Documents\Projects\Example5\bin\Debug\netcoreapp3.1\Example5.exe(프로세스 14848개)
이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

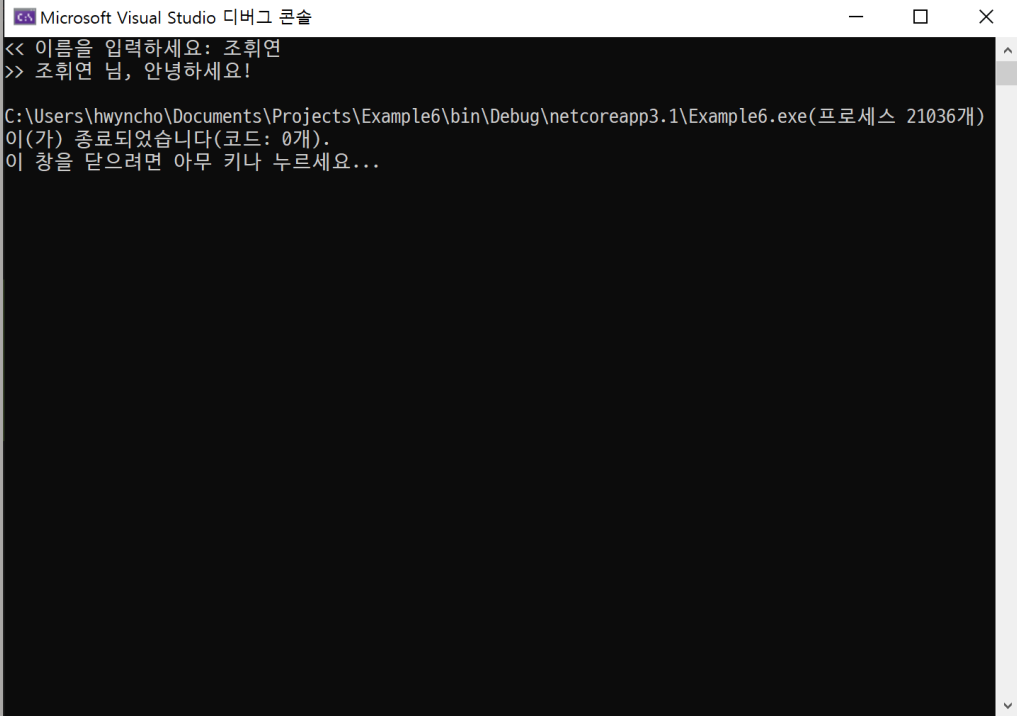


# 표준 입출력 예시

```
using System;

namespace Example6
{
    class Program
    {
        static void Main(string[] args)
        {
            string strName;

            Console.Write("<< 이름을 입력하세요: ");
            strName = Console.ReadLine();
            Console.WriteLine(">> {0} 님, 안녕하세요!", strName);
        }
    }
}
```



```
Microsoft Visual Studio 디버거 콘솔
<< 이름을 입력하세요: 조휘연
>> 조휘연 님, 안녕하세요!

C:\Users\hwyncho\Documents\Projects\Example6\bin\Debug\netcoreapp3.1\Example6.exe(프로세스 21036개)
이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

# 사용자 정의 클래스, 배열, 연산자 예시

```
using System;

namespace Example7
{
    public class Student
    {
        private int mINumber;
        private int mIScore;

        public Student(int iNumber, int iScore)
        {
            mINumber = iNumber;
            mIScore = iScore;
        }

        public int getNumber() { return mINumber; }

        public int getScore() { return mIScore; }
    }
}
```

# 사용자 정의 클래스, 배열, 연산자 예시 (cont`d)

```
class Program
{
    static void Main(string[] args)
    {
        const int NUM_STUDENTS = 5;
        Array arrStudents = Array.CreateInstance(typeof(Student), NUM_STUDENTS);

        int iScoreSum = 0;
        double dScoreMean = 0.0;

        Console.WriteLine("<< {0} 명의 학생의 점수를 입력하세요.", NUM_STUDENTS);
        for (int i = 0; i < NUM_STUDENTS; i++)
        {
            int iNumber = i + 1;
            string strScore = Console.ReadLine();
            int iScore = Convert.ToInt32(strScore);

            Student student = new Student(iNumber, iScore);
            arrStudents.SetValue(student, i);
        }
    }
}
```

# 사용자 정의 클래스, 배열, 연산자 예시 (cont`d)

```
for (int i = 0; i < arrStudents.Length; i++)
{
    int iNumber = ((Student)arrStudents.GetValue(i)).getNumber();
    int iScore = ((Student)arrStudents.GetValue(i)).getScore();

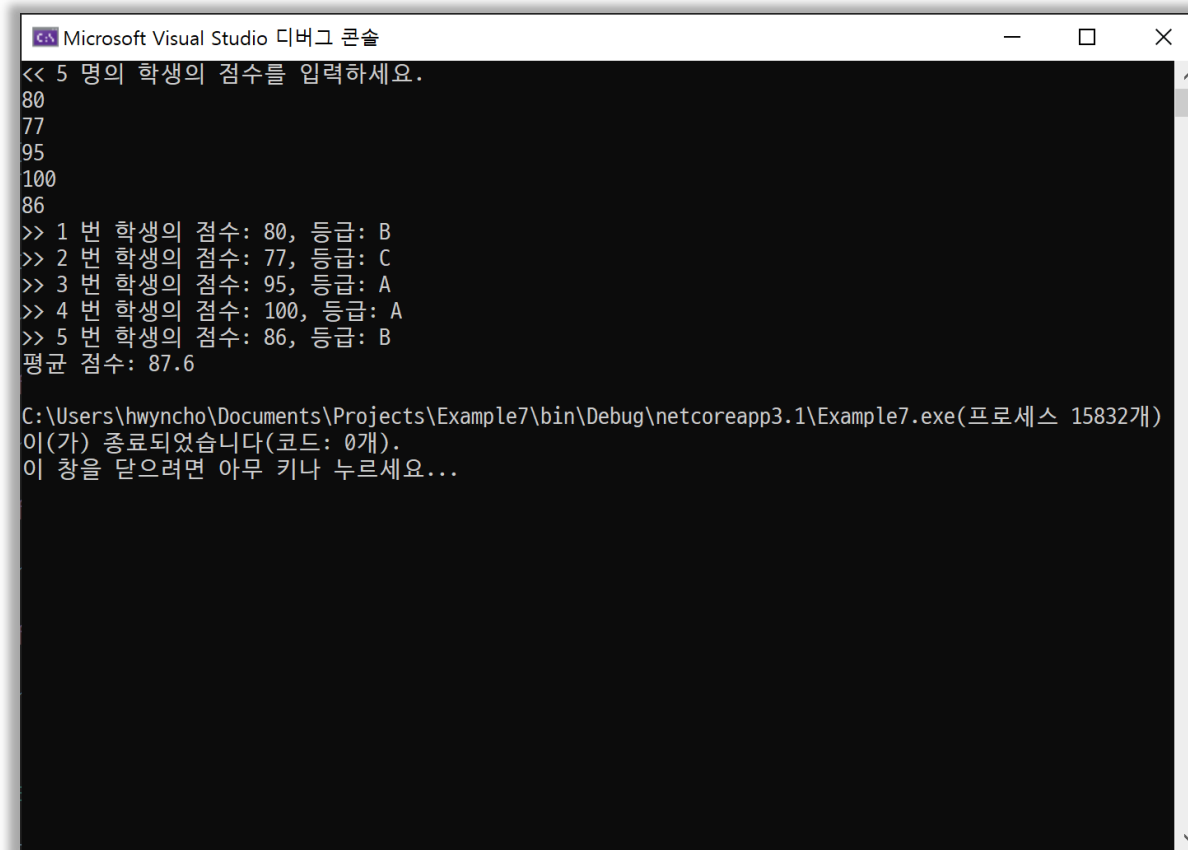
    string strGrade = "F";
    if (iScore >= 90) { strGrade = "A"; }
    else if (iScore >= 80) { strGrade = "B"; }
    else if (iScore >= 70) { strGrade = "C"; }
    else if (iScore >= 60) { strGrade = "D"; }

    Console.WriteLine(">> {0} 번 학생의 점수: {1}, 등급: {2}", iNumber, iScore, strGrade);

    iScoreSum += iScore;
}

dScoreMean = (double)iScoreSum / NUM_STUDENTS;
Console.WriteLine("평균 점수: {0}", dScoreMean);
}
}
```

# 사용자 정의 클래스, 배열, 연산자 예시 (cont`d)



The screenshot shows a Microsoft Visual Studio debugger console window. The title bar reads "Microsoft Visual Studio 디버그 콘솔". The console output is as follows:

```
<< 5 명의 학생의 점수를 입력하세요.  
80  
77  
95  
100  
86  
>> 1 번 학생의 점수: 80, 등급: B  
>> 2 번 학생의 점수: 77, 등급: C  
>> 3 번 학생의 점수: 95, 등급: A  
>> 4 번 학생의 점수: 100, 등급: A  
>> 5 번 학생의 점수: 86, 등급: B  
평균 점수: 87.6  
  
C:\Users\hwyncho\Documents\Projects\Example7\bin\Debug\netcoreapp3.1\Example7.exe(프로세스 15832개)  
이(가) 종료되었습니다(코드: 0개).  
이 창을 닫으려면 아무 키나 누르세요...
```