CHAPTER 01

C# 프로그래밍 첫걸음

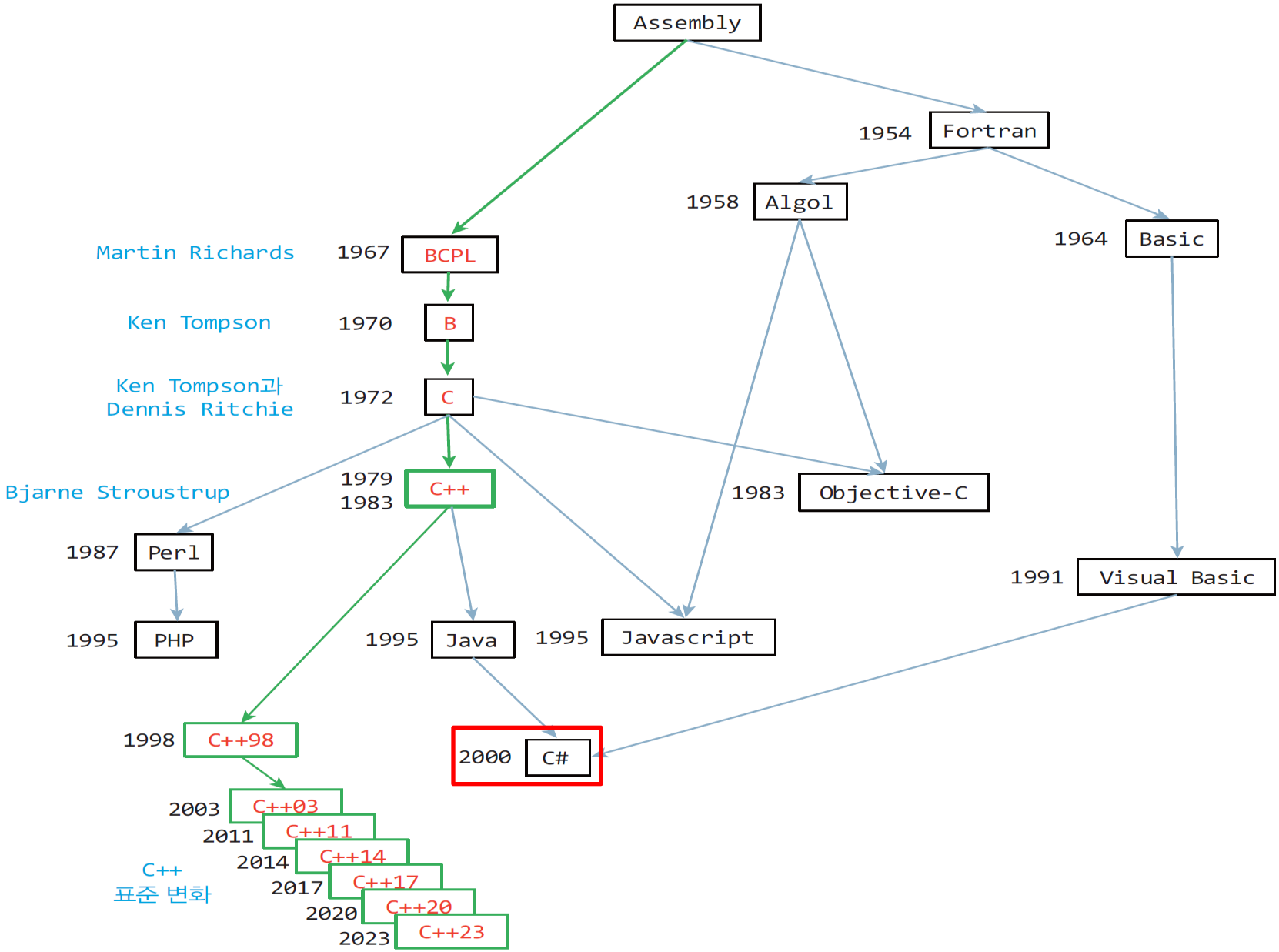
01 플랫폼과 프로그래밍 언어

02 라이브러리와 프레임워크

03 C#으로 할 수 있는 일

04 실습 환경 구축

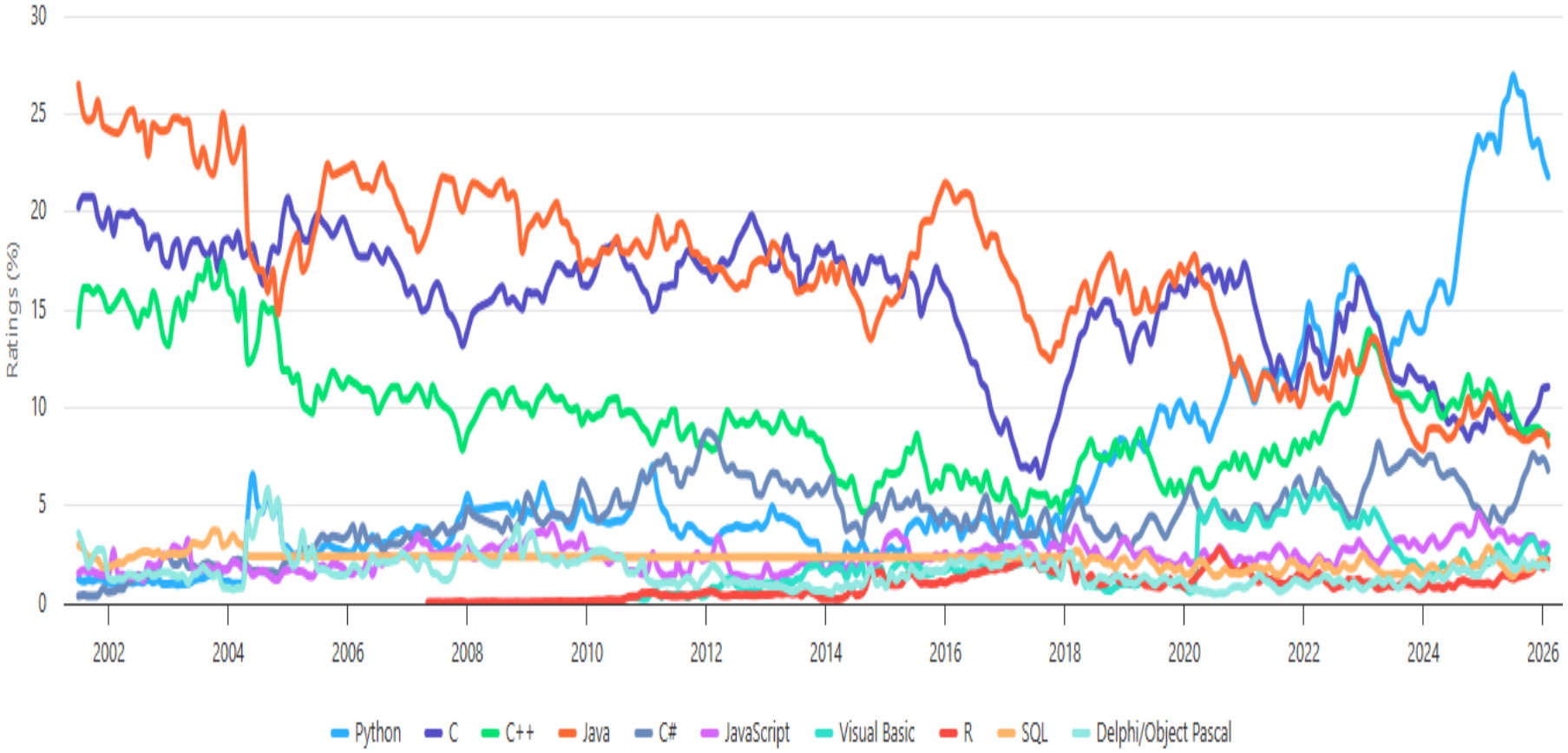
프로그래밍 언어의 진화


Assembly
1954 Fortran
1958 Algol
1964 Basic
Martin Richards
1967 BCPL
Ken Tompson
1970 B
Ken Tompson과
Dennis Ritchie
1972 C
Bjarne Stroustrup
1979
1983 C++
1983 Objective-C
1987 Perl
1991 Visual Basic
1995 PHP
1995 Java
1995 Javascript
1998 C++98
2000 C#
2003 C++03
2011 C++11
2014 C++14
2017 C++17
2020 C++20
2023 C++23
C++
표준 변화

프로그래밍 언어 순위 // 2026년 2월 기준

Feb 2026	Feb 2025	Change	Programming Language	Ratings
1	1		Python	21.81%
2	4	↑	C	11.05%
3	2	↓	C++	8.55%
4	3	↓	Java	8.12%
5	5		C#	6.83%
6	6		JavaScript	2.92%
7	10	↑	Visual Basic	2.85%
8	15	⇈	R	2.19%
9	7	↓	SQL	1.93%
10	9	↓	Delphi/Object Pascal	1.88%

프로그래밍 언어 순위 // 2026년 2월 기준



01 플랫폼과 프로그래밍 언어

■ C#의 특징

- 형식 안정(Type-Safe)적인 객체 지향(Object-Oriented) 언어

  -잘못된 타입 사용을 컴파일 또는 실행 시 막아준다는 의미

  ex) int a = “hello” // 컴파일 오류

  - 객체 지향은 프로그램을 **객체(Object)** 단위로 구성하는 방식

  - 객체는 상태(State)와 행동(Behavior)을 함께 묶어 하나의 독립적인 단위로 만든 것

- 기존 프로그래밍 언어의 생산성을 개선하여 성능이 높음

- 다양한 운영체제나 플랫폼에서 동작

```
자동차 객체
 ├── 상태(속성)
 │    - 색상
 │    - 속도
 │    - 연료량
 │
 └── 행동(메서드)
      - 시동걸기()
      - 가속하기()
      - 멈추기()
```

01 플랫폼과 프로그래밍 언어

- 다양한 운영체제나 플랫폼에서 동작

구분	운영체제	플랫폼
역할	기기를 작동시키는 핵심 시스템	앱·서비스가 돌아가는 환경
위치	가장 기본 단계	OS 위에서 동작하는 경우가 많음
예	Windows, Android	유튜브, 스팀, 앱스토어

■ 닷넷 플랫폼 활용: 프로그램이 모든 운영체제에서 동작

객체지향과 절차지향

	객체지향언어	절차지향언어
개념	프로그램을 객체(Object) 중심으로 구성 객체 = 데이터 + 기능(메서드) 현실 세계를 모델링하는 방식	프로그램을 함수(절차) 중심으로 구성 데이터와 함수가 분리되어 있음 위에서 아래로 순서대로 실행
특징	캡슐화 (Encapsulation) 상속 (Inheritance) 다형성 (Polymorphism) 추상화 (Abstraction)	순차적 실행 전역 변수 사용이 많음 구조가 비교적 단순 작은 프로그램에 적합
언어	Java C++ Python C#	C Pascal Fortran

■플랫폼

- 플랫폼(Platform): 소프트웨어 응용 프로그램 실행에 사용되는 하드웨어와 소프트웨어의 집합

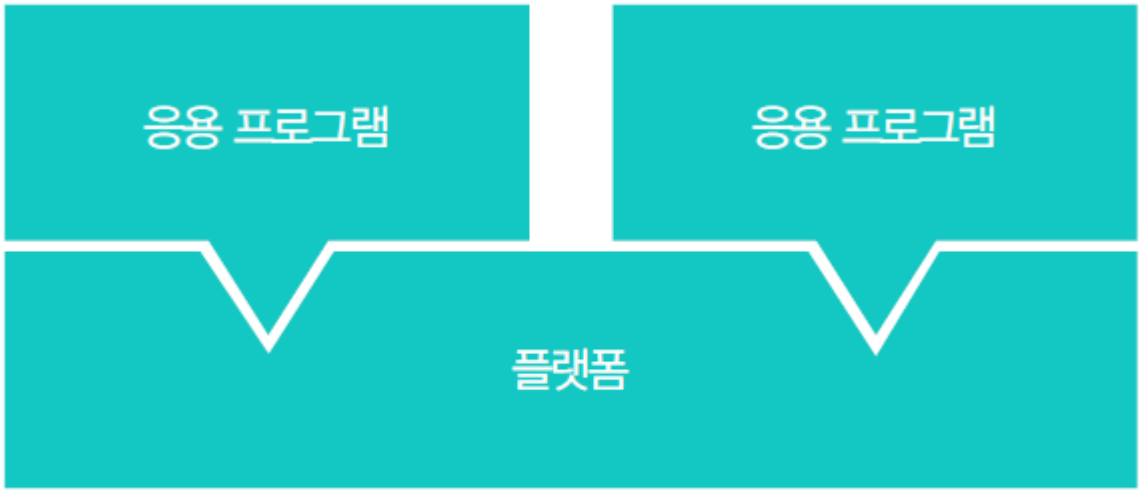
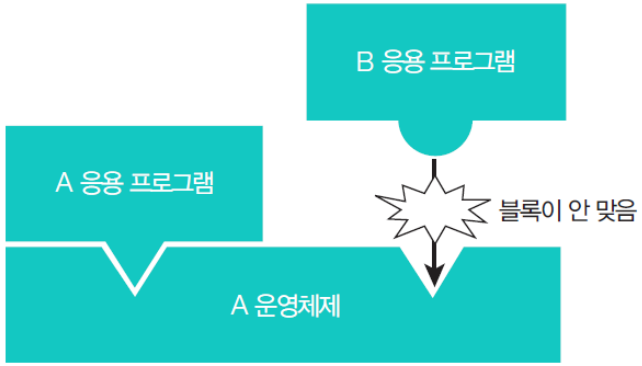


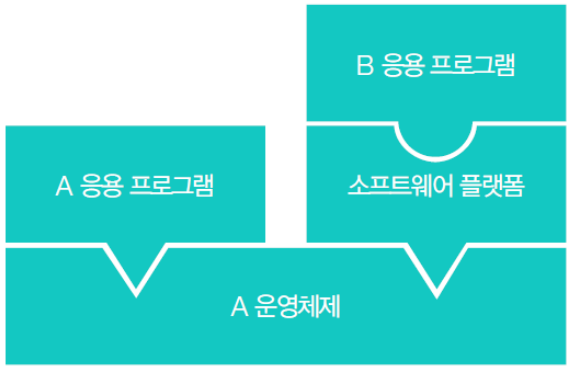
그림 1-1 플랫폼과 응용 프로그램

■소프트웨어 플랫폼

- [그림 1-2]의 (b)와 같이 중간에 변환할 수 있는 레고 블록이 소프트웨어 플랫폼



(a) 소프트웨어 플랫폼이 없는 경우: B 운영체제용으로 만들어진 B 응용 프로그램을 A 운영체제에서 실행 불가



(b) 소프트웨어 플랫폼이 있는 경우: B 운영체제용으로 만들어진 B 응용 프로그램을 A 운영체제에서 실행 가능

그림 1-2 소프트웨어 플랫폼의 역할

■닷넷 플랫폼

- 닷넷 플랫폼: 마이크로소프트 사가 만든 중간 레고 블록. 현재는 마이크로소프트 사가 활용할 수 있는 모든 프로그래밍 언어를 닷넷 플랫폼과 연결시킴

그림 1-3 초기의 닷넷 플랫폼

■ 닷넷 플랫폼

- 닷넷 플랫폼은 어떻게 보면 플랫폼의 기본적인 발전 형태

- 현재는 자바 플랫폼도 자바뿐만 아니라 Jython (자바에서 쓰는 파이썬), JRuby(자바에서 쓰는 루비), Rhino (자바에서 쓰는 자바스크립트),LuaJava(자바에서 쓰는 루아), Groovy 등의 언어를 모두 활용할 수 있음

■닷넷 프레임워크

- 마이크로소프트 사는 클래스 라이브러리를 함께 제공(닷넷 프레임워크)

  - 응용 프로그램 프레임워크: 윈도 폼, WPF, 윈도 유니버셜 응용 프로그램
  - 웹 응용 프로그램 프레임워크: 실버라이트
  - 웹 서버 프레임워크: ASP.NET, ASP.NET MVC
  - 웹 서비스 프레임워크: WCF

■닷넷 프레임워크

- 문라이트 프로젝트: 모노 프로젝트의 일환으로, 닷넷 프레임워크를 리눅스에서도 동작하도록 구현해서 C#이 리눅스에서도 동작함
- 2020년에는 .NET Core(모노) 플랫폼이 정식 플랫폼이 되어 대부분의 라이브러리가 대부분의 운영체제에서 동작할 수 있게 됨

그림 1-4 모노 프로젝트

그림 1-5 NET 5.0 플랫폼부터 지원하는 운영체제와 사용할 수 있는 프레임워크(또는 엔진)

- C#으로 할 수 있는 일
  - 게임 프레임워크(게임 엔진): 유니티
  - 머신러닝과 딥러닝: ML.NET

■ 라이브러리

```
• print("Hello World")
• printf("Hello World")
• Console.WriteLine("Hello World")
• System.out.println("Hellp World")
• puts("Hello World")
```

- 각 코드를 실행하면 모두 화면에 Hello World를 출력
- 라이브러리: print, printf, WriteLine, println, puts 식별자처럼 미리 만들어둔 코드
- 닷넷 프레임워크가 제공하는 라이브러리로 쉽게 프로그램을 만들 수 있음

■ 프레임워크 // 프로그램의 흐름을 스스로 관리하는 대규모 라이브러리 집합

- 프레임워크는 제어역전이 있는 대규모의 라이브러리
  - 라이브러리는 개발자가 사용해줘야만 함
  - 하지만 프레임워크는 프로그램의 초기화부터 종료까지의 흐름을 직접 관리

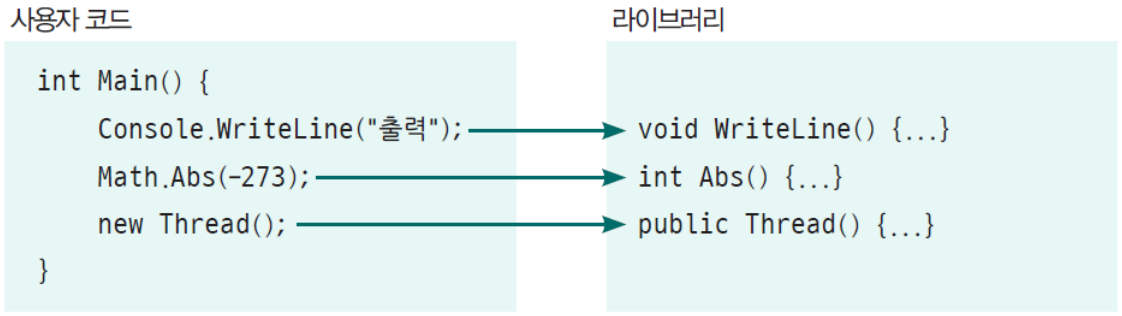


그림 1-6 라이브러리는 사용자 코드에게 호출되는 것

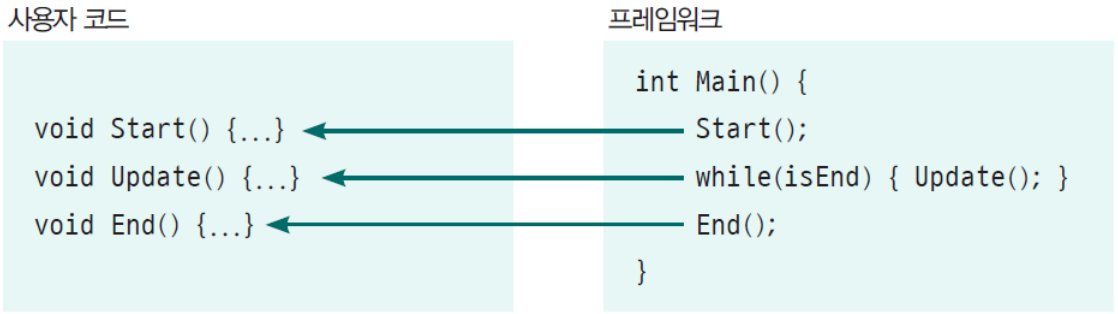


그림 1-7 프레임워크는 사용자 코드를 호출하는 것(제어 역전)

프레임워크

- 프레임워크는 제어역전이 있는 대규모의 라이브러리
- 제어역전(Inversion of Control)
  - 일반적인 프로그램 동작: 내 코드 -> 라이브러리 호출
  - 프레임워크 동작: 프레임워크 -> 내 코드를 호출
  - 누가 프로그램의 흐름을 통제하느냐가 바뀜

■ 프레임워크와 라이브러리

- 현재는 프레임워크와 라이브러리라는 두 용어가 너무 혼용되어 “대규모의 라이브러리 = 프레임워크”라고 의미하는 경우도 많음
- 하지만 지금은 배우는 시점이므로 어느 정도 구분

구분	라이브러리	프레임워크
제어권	내가 가짐	프레임워크가 가짐
호출 주체	내가 호출	프레임워크가 호출
구조	기능 모음	구조 + 기능
규모	비교적 작음	대규모
비유	공구 상자 내가 필요한 망치를 꺼내 씀	이미 지어진 건물 구조 내가 벽지나 가구만 채워 넣음

03 C#으로 할 수 있는 일

■ GUI 개발 // Graphic User Interface

- 윈도에서 작동하는 GUI 프로그램 개발을 위한 윈도 폼과 MPF 제공
- 윈도 폼(Windows Form)
  - C++를 사용한 윈도 개발을 C#으로 옮겨 놓은 형태
  - 개발자가 폼 디자이너 이용해 도구 상자에서 컨트롤을 윈도에 배치할 때마다 폼 디자이너가 프로그램의 UI를 표시하면서 C# 코드를 자동으로 만들어 줌

GUI 개발

- 윈도 폼(Windows Form)

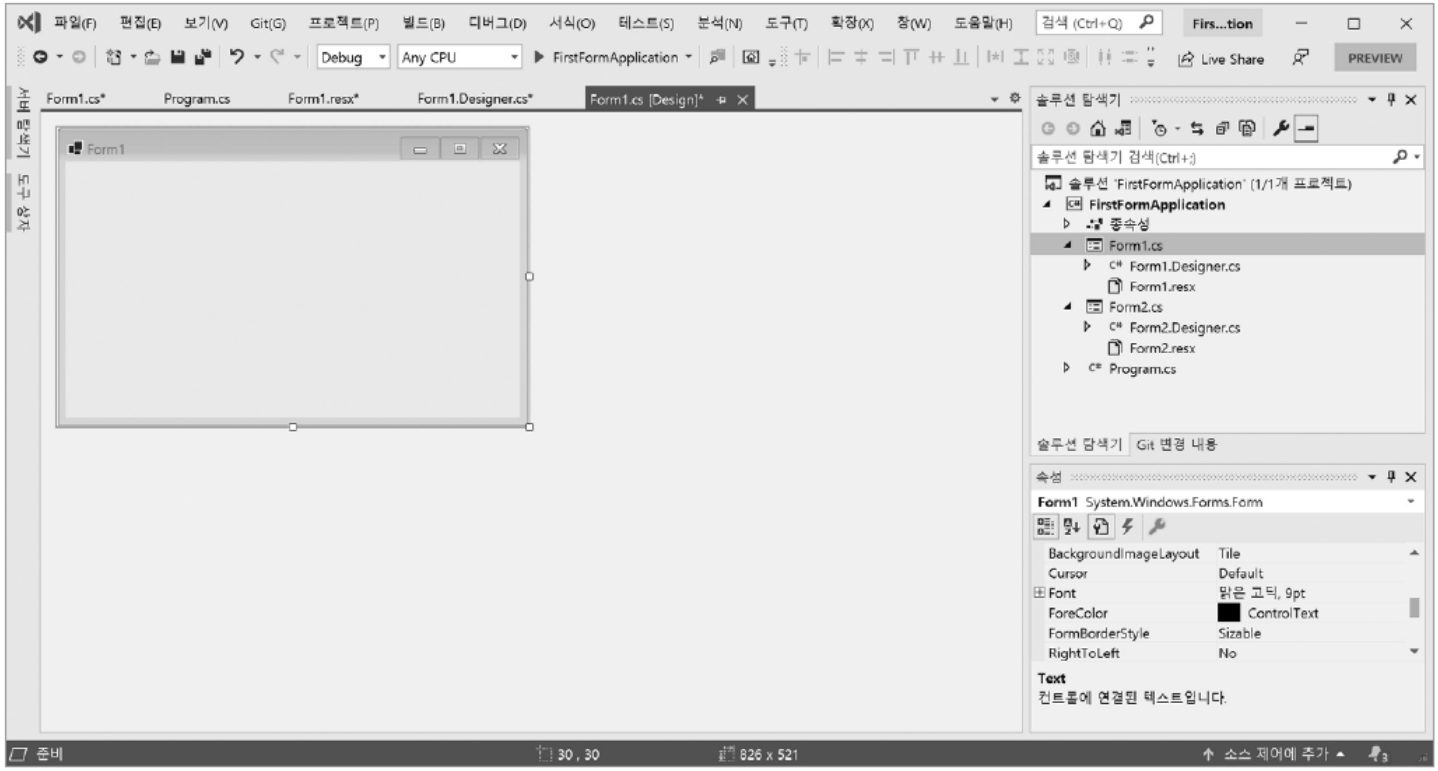


그림 1-8 윈도 폼 개발 화면

03 C#으로 할 수 있는 일

GUI 개발

- WPF(Windows Presentation Foundation)
  - 현대적인 개발 패턴 중에 MVC 패턴과 MVVM 패턴을 적용해 개발 생산성을 향상시킨 프레임워크
  - DirectX 등의 기능도 추가로 내장하여 3D 그래픽까지 자체적으로 처리 가능

GUI 개발

- WPF(Windows Presentation Foundation)

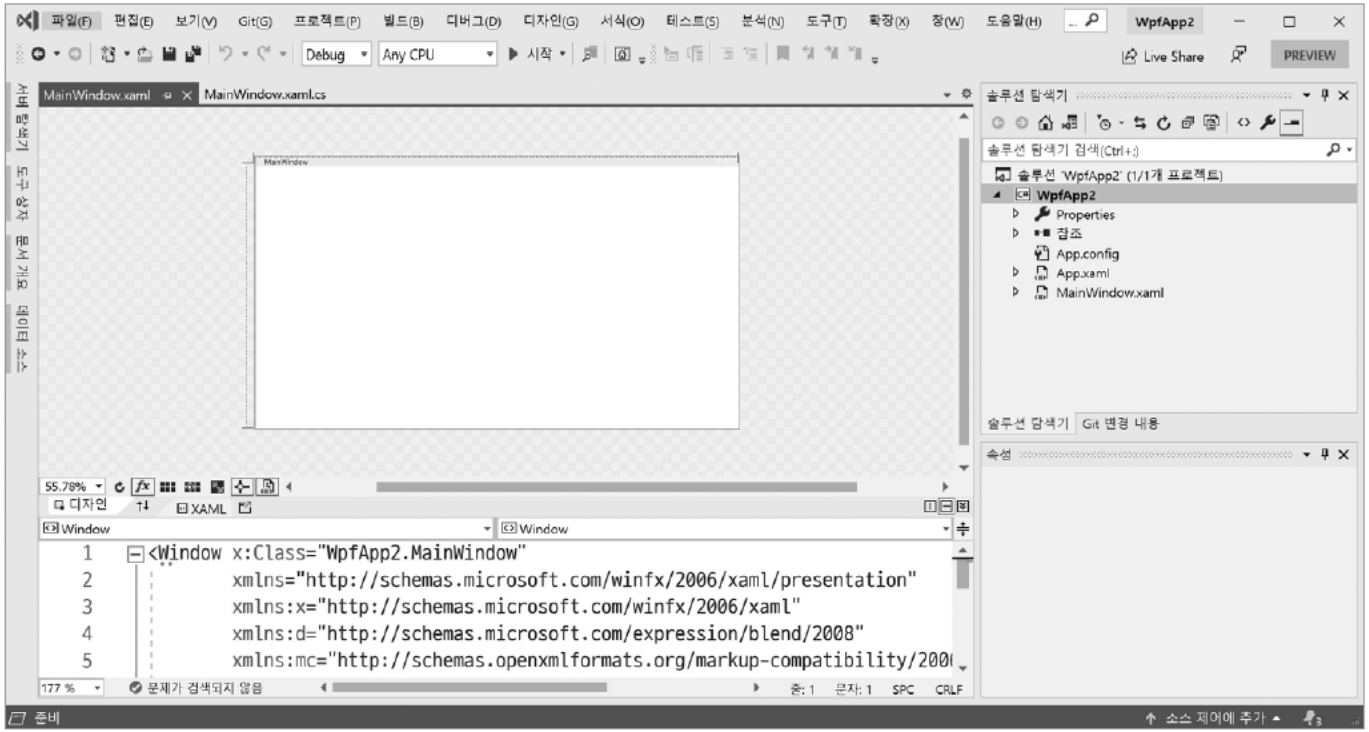


그림 1-9 WPF 개발 화면

03 C#으로 할 수 있는 일

■ 웹 개발

- C#이 사용되는 가장 대표적인 분야
- 마이크로소프트 사는 2가지 프레임워크 지원
  (ASP.NET 프레임워크, ASP.NET MVC)

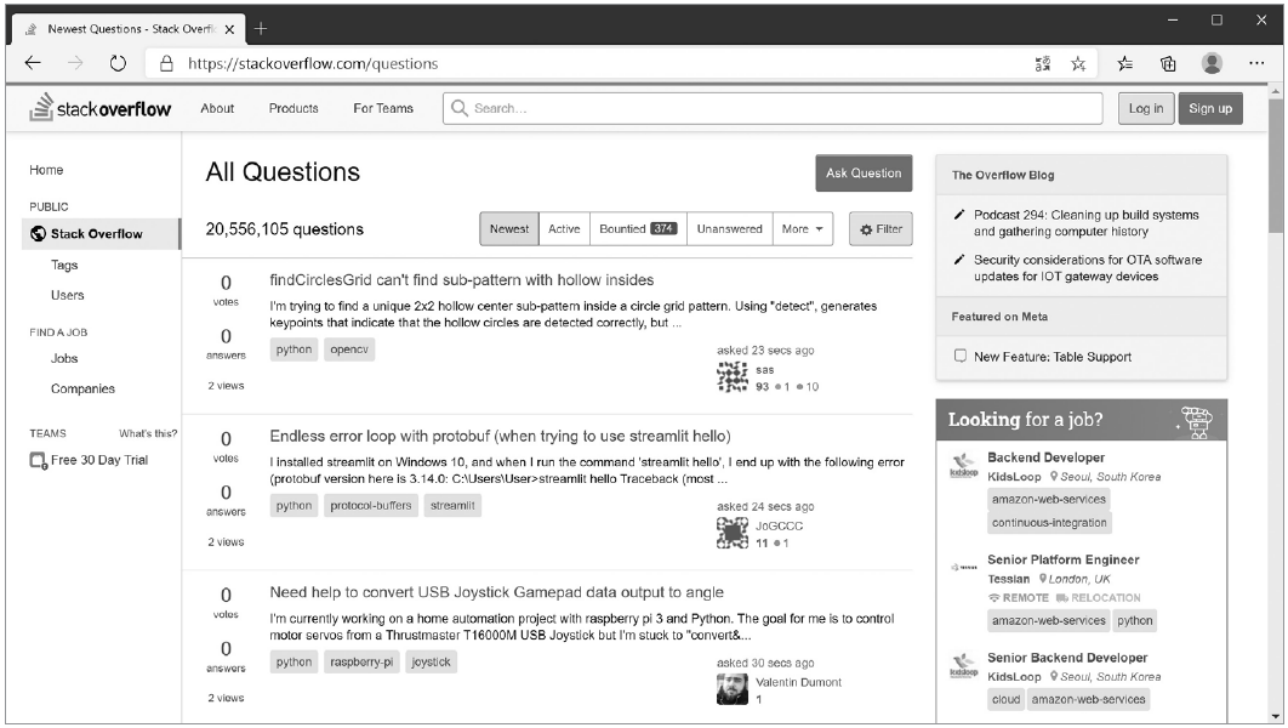


그림 1-10 ASP.NET MVC로 개발된 스택오버플로

03 C#으로 할 수 있는 일

■ 게임 개발

▪게임 클라이언트 개발

- 유니티 엔진 개발로 C#으로 게임 클라이언트 개발 활성화

▪게임 서버 개발

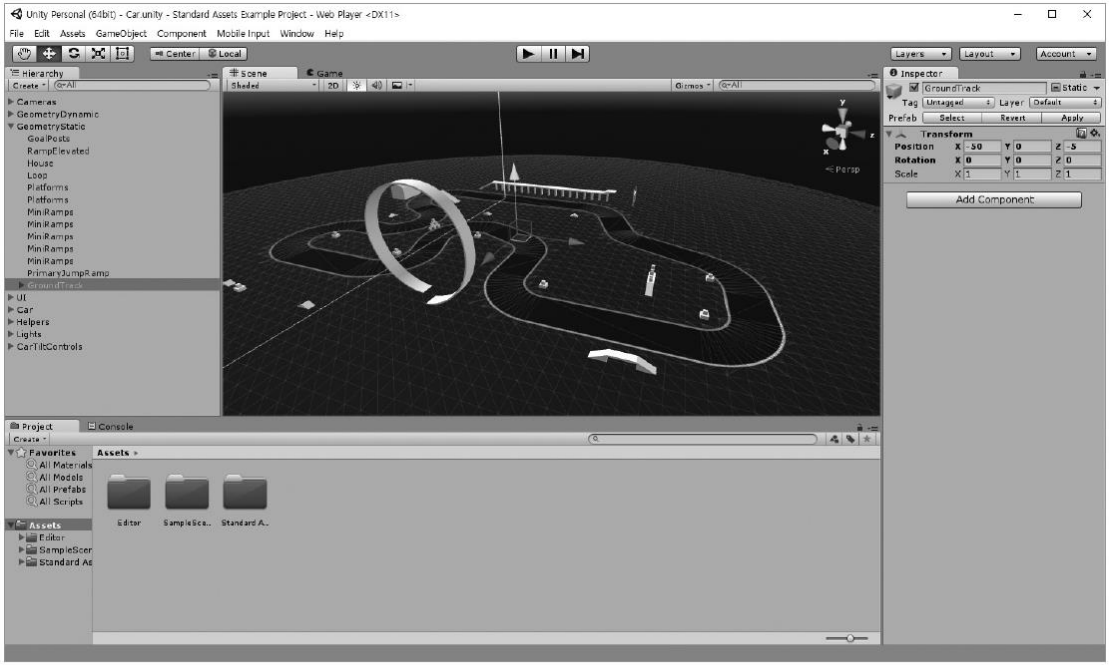


그림 1-11 유니티 엔진

IoT (사물 인터넷) 개발 // Internet Of Things

- C#을 사용해 다양한 IoT 개발 가능
  - IoT는 다양한 사물에 센서와 통신 기능을 내장하고 인터넷과 연결하는 기술

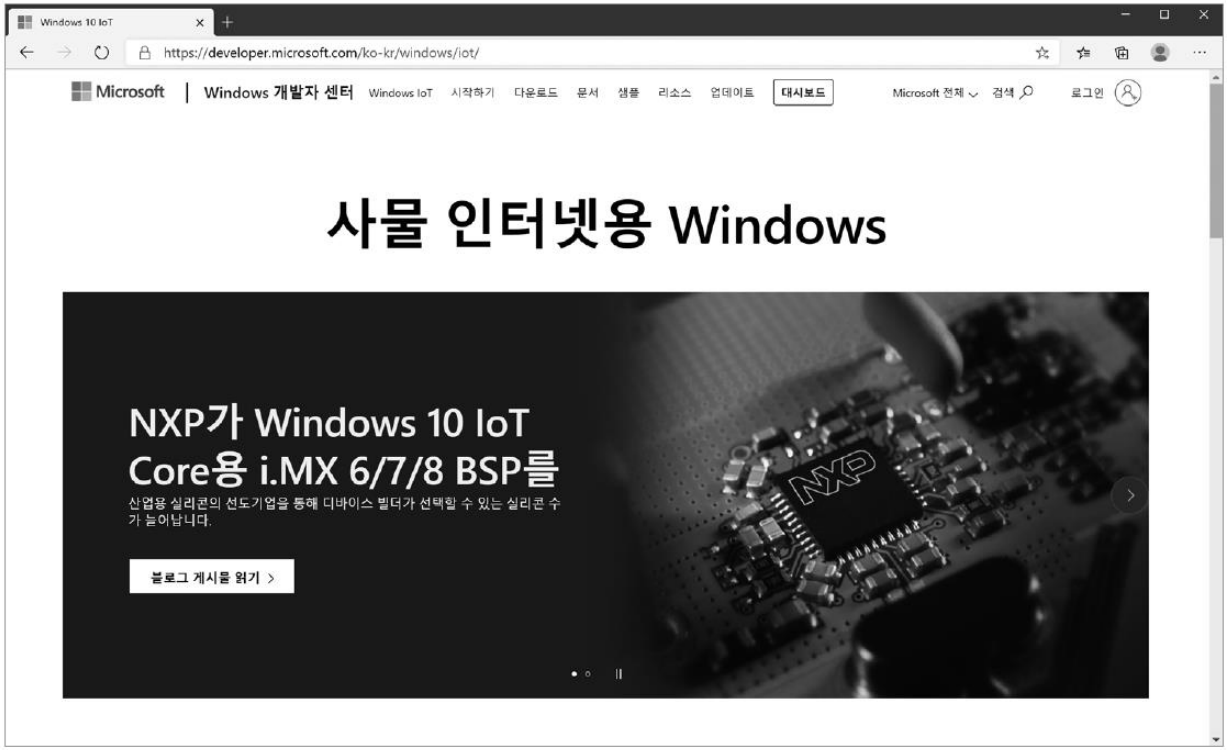
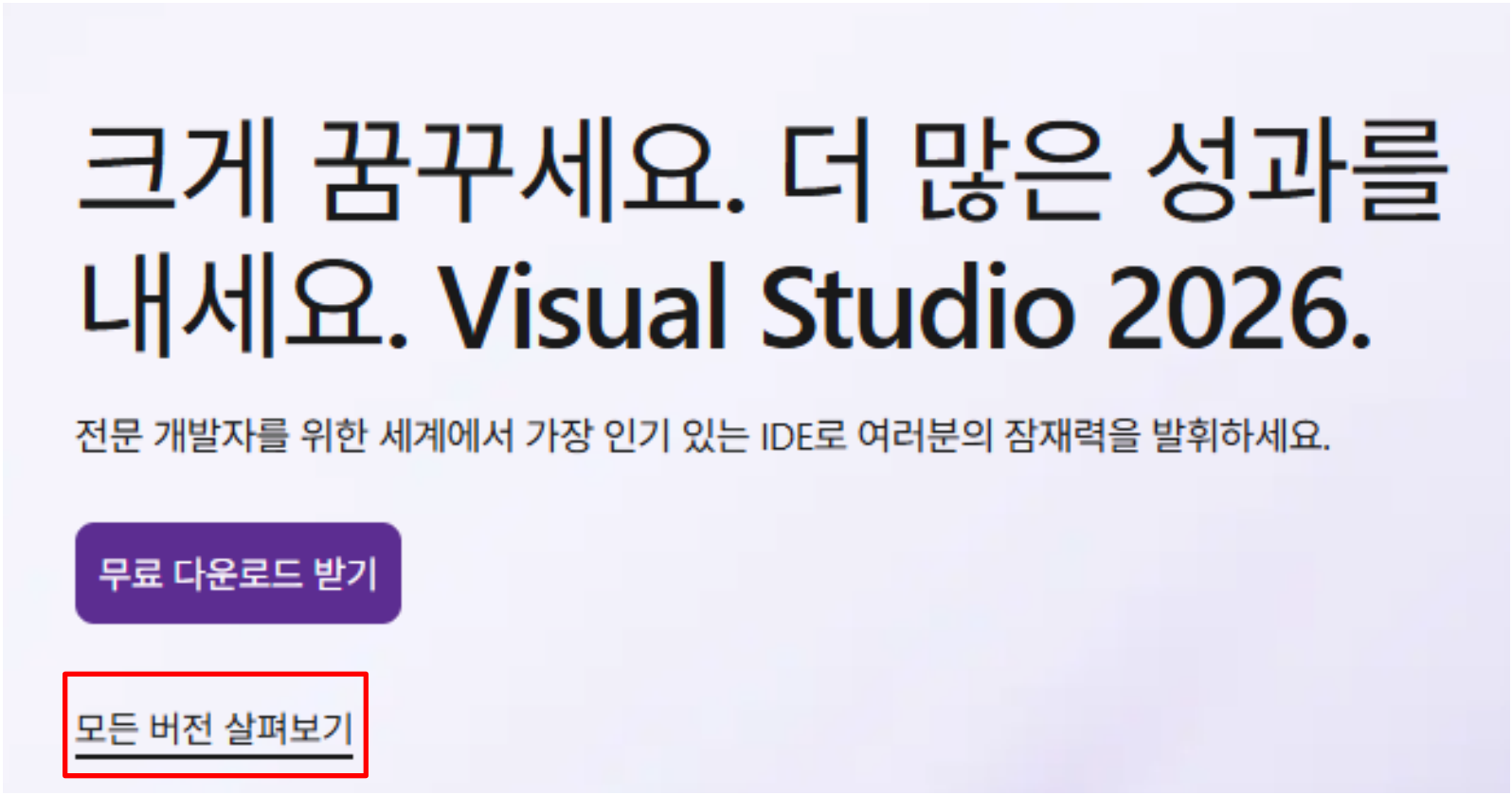


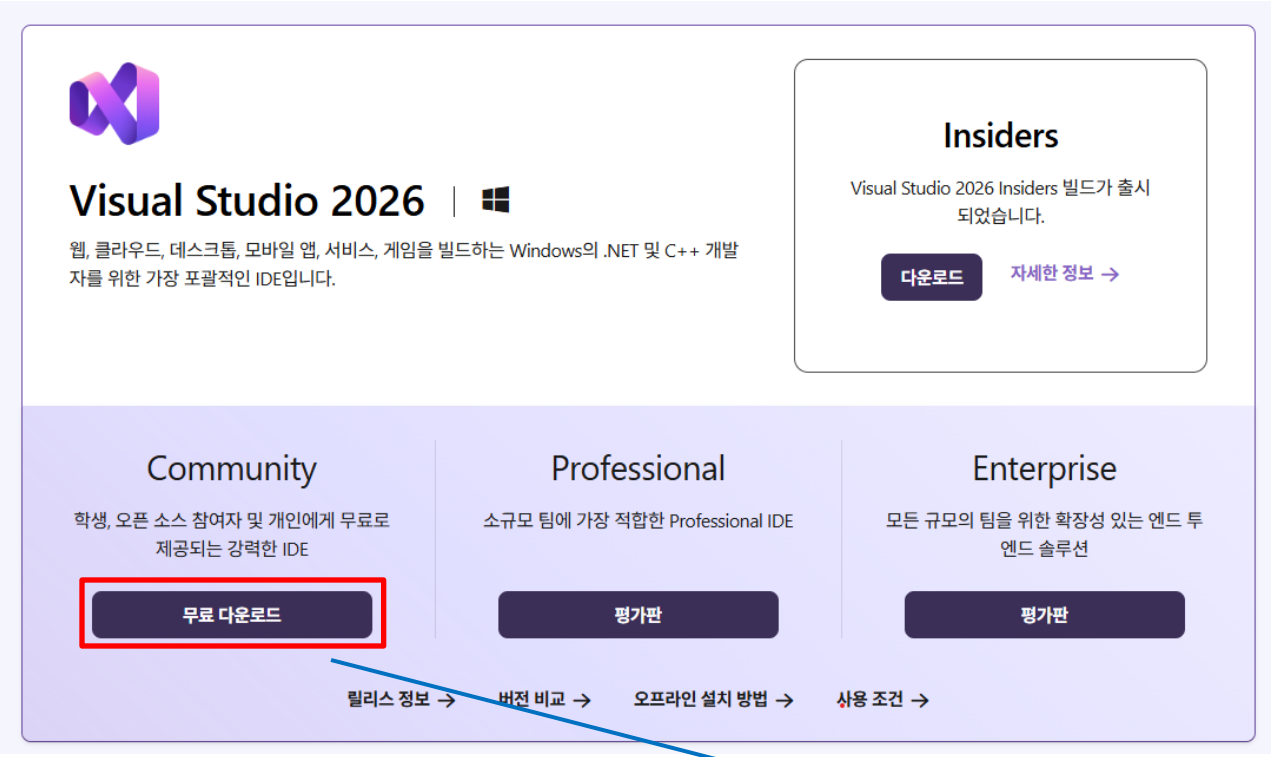
그림 1-12 윈도 10 IoT 코어

Visual Studio 시작

https://visualstudio.microsoft.com/ko/

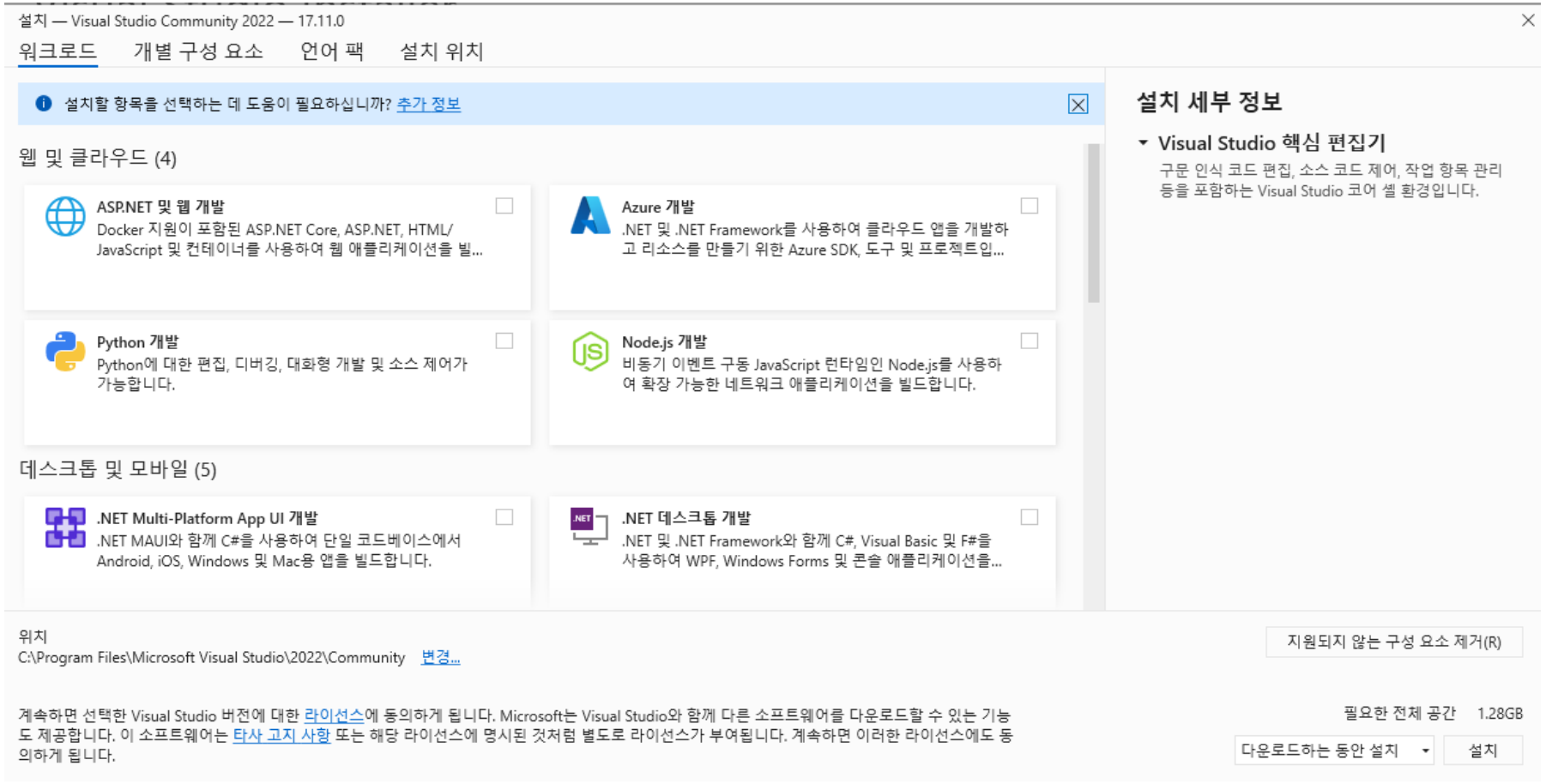


04 실습 환경 구축

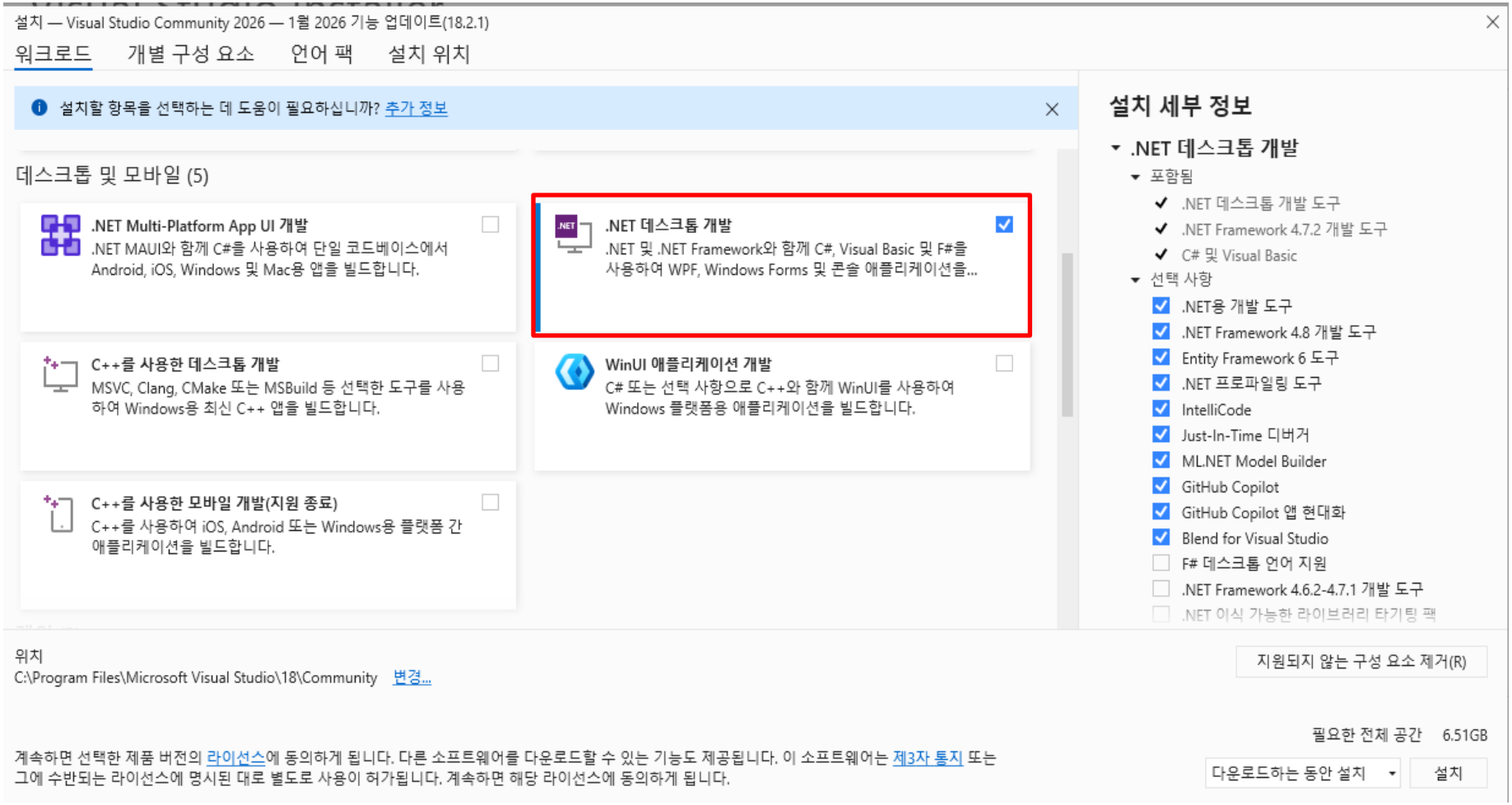

Visual Studio 2026
웹, 클라우드, 데스크톱, 모바일 앱, 서비스, 게임을 빌드하는 Windows의 .NET 및 C++ 개발자를 위한 가장 포괄적인 IDE입니다.
Insiders
Visual Studio 2026 Insiders 빌드가 출시되었습니다.
다운로드
자세한 정보 →
Community
학생, 오픈 소스 참여자 및 개인에게 무료로 제공되는 강력한 IDE
무료 다운로드
Professional
소규모 팀에 가장 적합한 Professional IDE
평가판
Enterprise
모든 규모의 팀을 위한 확장성 있는 엔드 투 엔드 솔루션
평가판
릴리스 정보 →
버전 비교 →
오프라인 설치 방법 →
사용 조건 →


VisualStudioSetup.exe

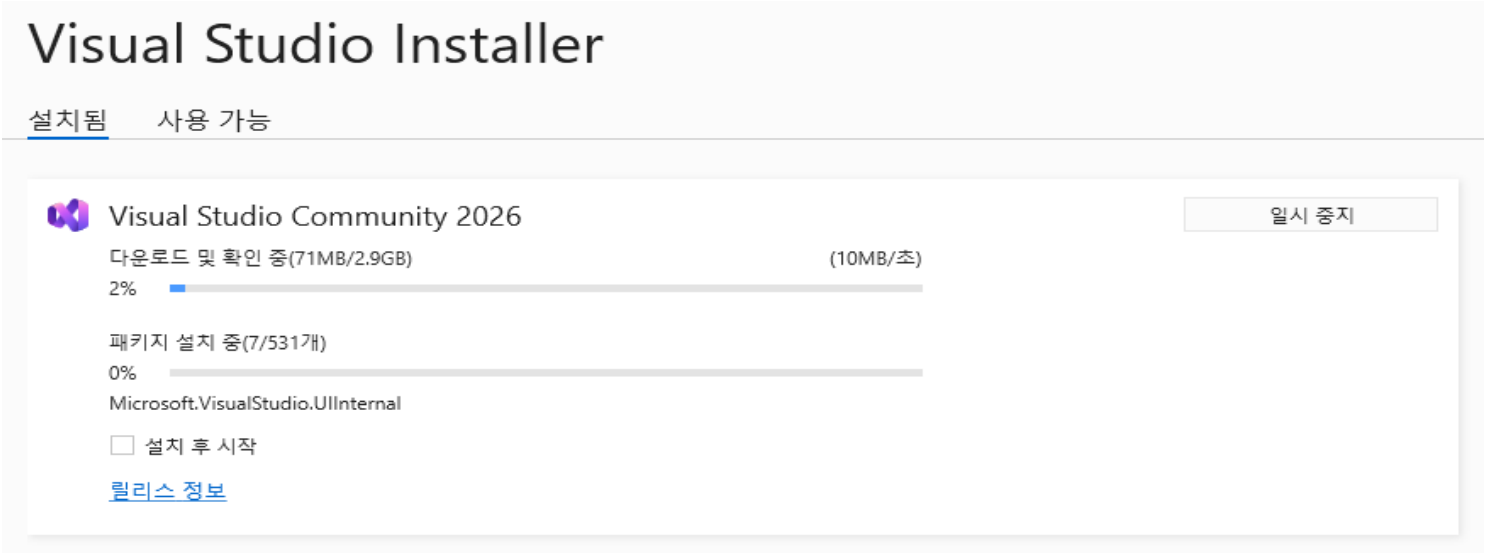
04 실습 환경 구축

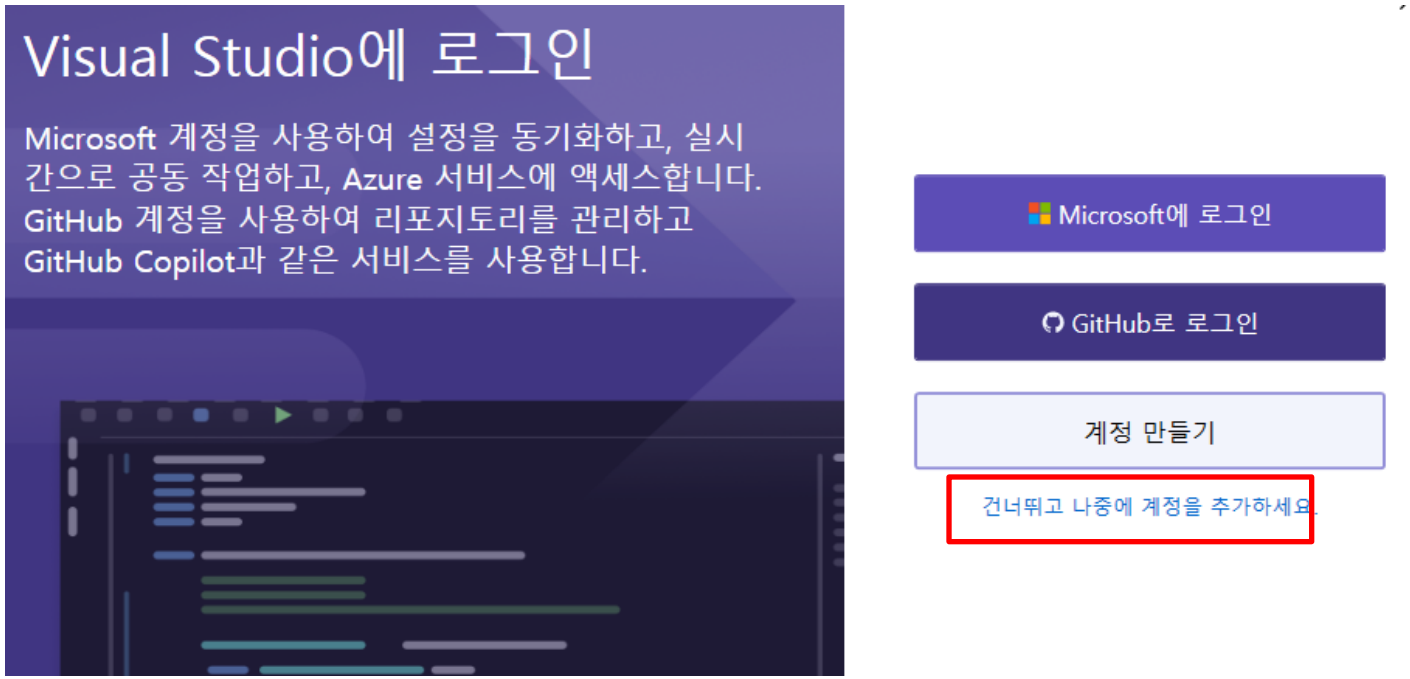

설치 — Visual Studio Community 2022 — 17.11.0
워크로드
개별 구성 요소
언어 팩
설치 위치
설치할 항목을 선택하는 데 도움이 필요하십니까? 추가 정보
웹 및 클라우드 (4)
ASP.NET 및 웹 개발
Docker 지원이 포함된 ASP.NET Core, ASP.NET, HTML/JavaScript 및 컨테이너를 사용하여 웹 애플리케이션을 빌...
Azure 개발
.NET 및 .NET Framework를 사용하여 클라우드 앱을 개발하고 리소스를 만들기 위한 Azure SDK, 도구 및 프로젝트입...
Python 개발
Python에 대한 편집, 디버깅, 대화형 개발 및 소스 제어가 가능합니다.
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.
.NET 데스크톱 개발
.NET 및 .NET Framework와 함께 C#, Visual Basic 및 F#을 사용하여 WPF, Windows Forms 및 콘솔 애플리케이션을...
설치 세부 정보
Visual Studio 핵심 편집기
구문 인식 코드 편집, 소스 코드 제어, 작업 항목 관리 등을 포함하는 Visual Studio 코어 셸 환경입니다.
위치
C:\Program Files\Microsoft Visual Studio\2022\Community
변경...
지원되지 않는 구성 요소 제거(R)
계속하면 선택한 Visual Studio 버전에 대한 라이선스에 동의하게 됩니다. Microsoft는 Visual Studio와 함께 다른 소프트웨어를 다운로드할 수 있는 기능도 제공합니다. 이 소프트웨어는 타사 고지 사항 또는 해당 라이선스에 명시된 것처럼 별도로 라이선스가 부여됩니다. 계속하면 이러한 라이선스에도 동의하게 됩니다.
필요한 전체 공간 1.28GB
다운로드하는 동안 설치
설치

04 실습 환경 구축

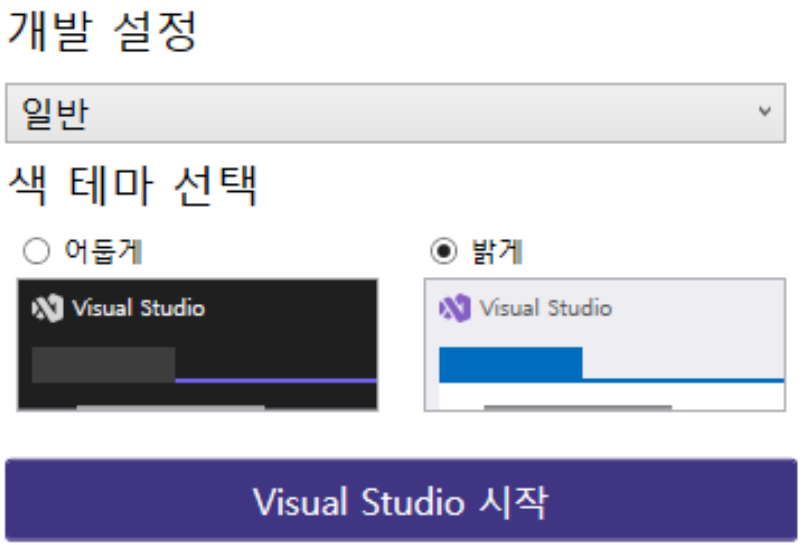

설치 — Visual Studio Community 2026 — 1월 2026 기능 업데이트(18.2.1)
워크로드
개별 구성 요소
언어 팩
설치 위치
설치할 항목을 선택하는 데 도움이 필요하십니까? 추가 정보
데스크톱 및 모바일 (5)
.NET Multi-Platform App UI 개발
.NET MAUI와 함께 C#을 사용하여 단일 코드베이스에서 Android, iOS, Windows 및 Mac용 앱을 빌드합니다.
.NET 데스크톱 개발
.NET 및 .NET Framework와 함께 C#, Visual Basic 및 F#을 사용하여 WPF, Windows Forms 및 콘솔 애플리케이션을...
C++를 사용한 데스크톱 개발
MSVC, Clang, CMake 또는 MSBuild 등 선택한 도구를 사용하여 Windows용 최신 C++ 앱을 빌드합니다.
WinUI 애플리케이션 개발
C# 또는 선택 사항으로 C++와 함께 WinUI를 사용하여 Windows 플랫폼용 애플리케이션을 빌드합니다.
C++를 사용한 모바일 개발(지원 종료)
C++를 사용하여 iOS, Android 또는 Windows용 플랫폼 간 애플리케이션을 빌드합니다.
설치 세부 정보
.NET 데스크톱 개발
포함됨
.NET 데스크톱 개발 도구
.NET Framework 4.7.2 개발 도구
C# 및 Visual Basic
선택 사항
.NET용 개발 도구
.NET Framework 4.8 개발 도구
Entity Framework 6 도구
.NET 프로파일링 도구
IntelliCode
Just-In-Time 디버거
ML.NET Model Builder
GitHub Copilot
GitHub Copilot 앱 현대화
Blend for Visual Studio
F# 데스크톱 언어 지원
.NET Framework 4.6.2-4.7.1 개발 도구
위치
C:\Program Files\Microsoft Visual Studio\18\Community 변경...
지원되지 않는 구성 요소 제거(R)
필요한 전체 공간 6.51GB
계속하면 선택한 제품 버전의 라이선스에 동의하게 됩니다. 다른 소프트웨어를 다운로드할 수 있는 기능도 제공됩니다. 이 소프트웨어는 제3자 통지 또는 그에 수반되는 라이선스에 명시된 대로 별도로 사용이 허가됩니다. 계속하면 해당 라이선스에 동의하게 됩니다.
다운로드하는 동안 설치
설치

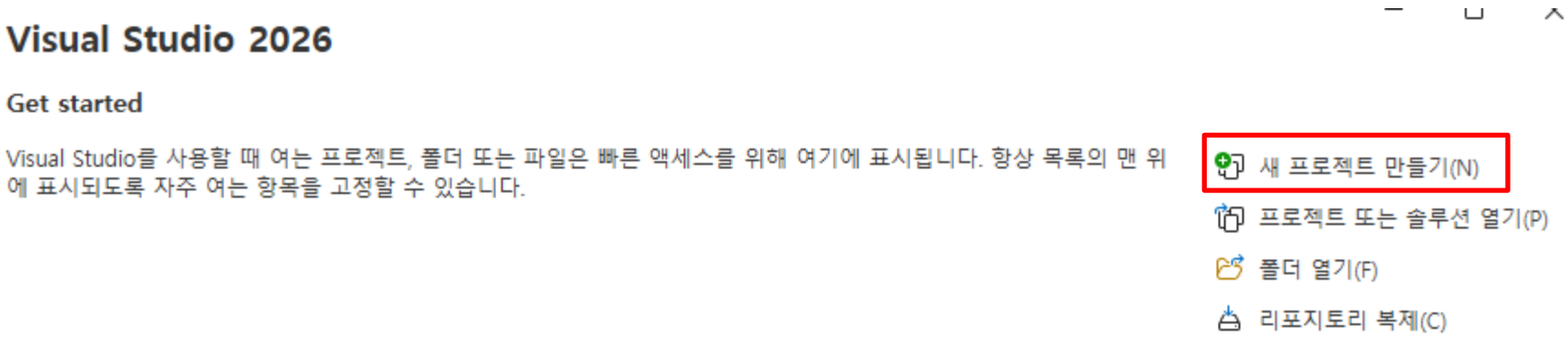
04 실습 환경 구축


Visual Studio Installer
설치됨
사용 가능
Visual Studio Community 2026
일시 중지
다운로드 및 확인 중(71MB/2.9GB)
(10MB/초)
2%
패키지 설치 중(7/531개)
0%
Microsoft.VisualStudio.UIInternal
설치 후 시작
릴리스 정보


Visual Studio에 로그인
Microsoft 계정을 사용하여 설정을 동기화하고, 실시간으로 공동 작업하고, Azure 서비스에 액세스합니다.
GitHub 계정을 사용하여 리포지토리를 관리하고 GitHub Copilot과 같은 서비스를 사용합니다.
Microsoft에 로그인
GitHub로 로그인
계정 만들기
건너뛰고 나중에 계정을 추가하세요.

04 실습 환경 구축


개발 설정
일반
색 테마 선택
어둡게
밝게
Visual Studio
Visual Studio
Visual Studio 시작


Visual Studio 2026
Get started
Visual Studio를 사용할 때 여는 프로젝트, 폴더 또는 파일은 빠른 액세스를 위해 여기에 표시됩니다. 항상 목록의 맨 위에 표시되도록 자주 여는 항목을 고정할 수 있습니다.
새 프로젝트 만들기(N)
프로젝트 또는 솔루션 열기(P)
폴더 열기(F)
리포지토리 복제(C)

04 실습 환경 구축


새 프로젝트 만들기
최근 프로젝트 템플릿(R)
최근에 액세스한 템플릿 목록이 여기에 표시됩니다.
C#
모든 언어(L)
C#
C++
F#
JavaScript
Python
TypeScript
Visual Basic
XAML
쿼리 언어
새로 만들기
템플릿 검색(Alt+S)(S)
모든 언어(L)
모든 플랫폼(P)
모든 프로젝트 형식(T)
콘솔 앱
Windows, Linux 및 macOS의 .NET에서 실행할 수 있는 명령줄 응용 프로그램을 만들기 위한 프로젝트
C# Linux macOS Windows 콘솔
클래스 라이브러리
.NET 또는 .NET Standard를 대상으로 하는 클래스 라이브러리를 만들기 위한 프로젝트
C# Android Linux macOS Windows 라이브러리
작업자 서비스
Worker Service를 만드는 데 사용하는 빈 프로젝트 템플릿입니다.
C# Linux macOS Windows 클라우드 서비스
MSTest 테스트 프로젝트
Windows, Linux 및 macOS의 .NET에서 실행할 수 있는 MSTest 테스트가 포함된 프로젝트
C# Linux macOS Windows 데스크톱 MSTest
2026년 1월 기능 업데이트
뒤로(B)
다음(N)

■ 프로그램 생성

▪ 프로젝트 생성

- [새 프로젝트 구성] 대화상자가 표시되면, [프로젝트 이름]을 “FirstProgram”이라고 변경

그림 1-23 [새 프로젝트 구성] 대화상자

■ 프로그램 생성

▪ 프로젝트 생성

- [추가 정보] 대화상자가 표시되면, 반드시 [최상위 문 사용 안 함]에 체크

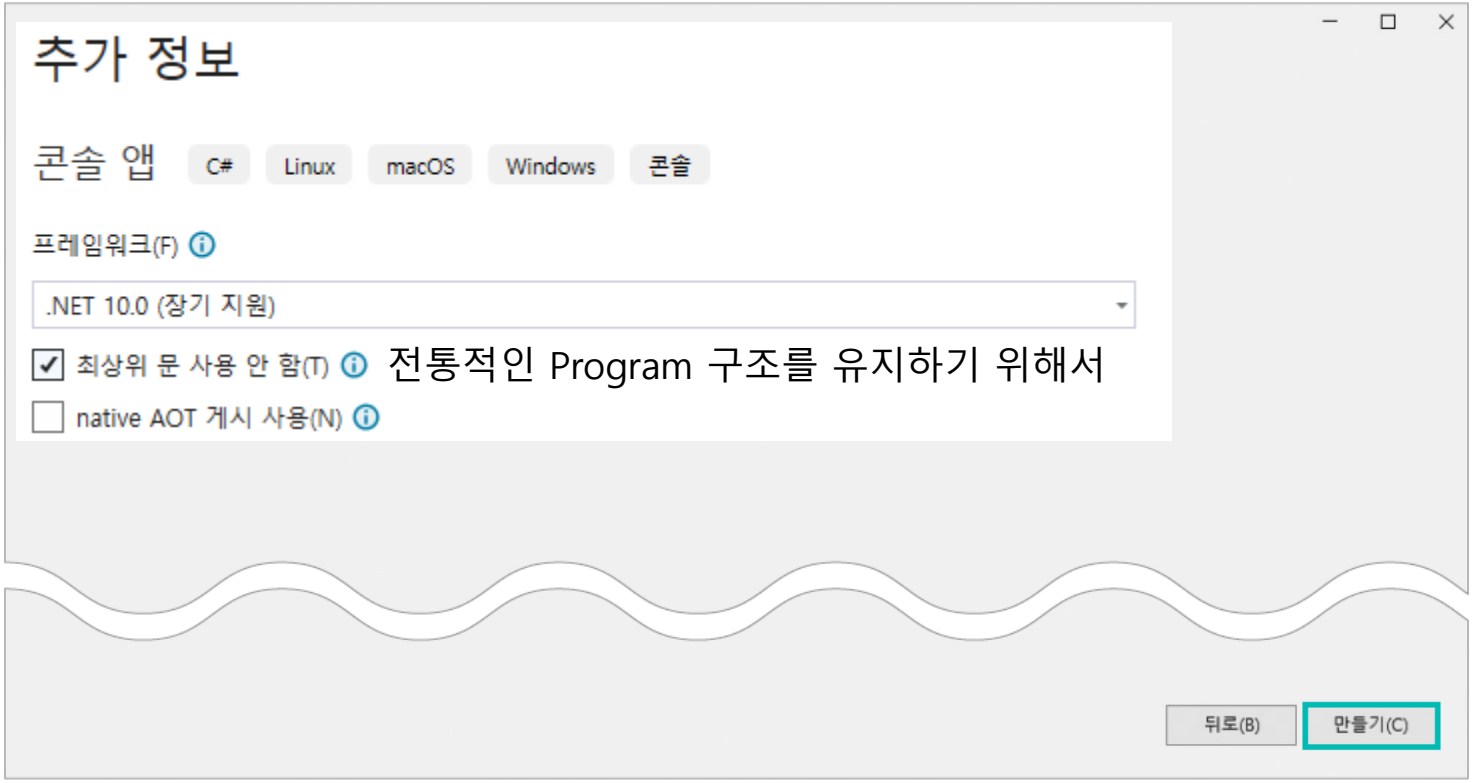
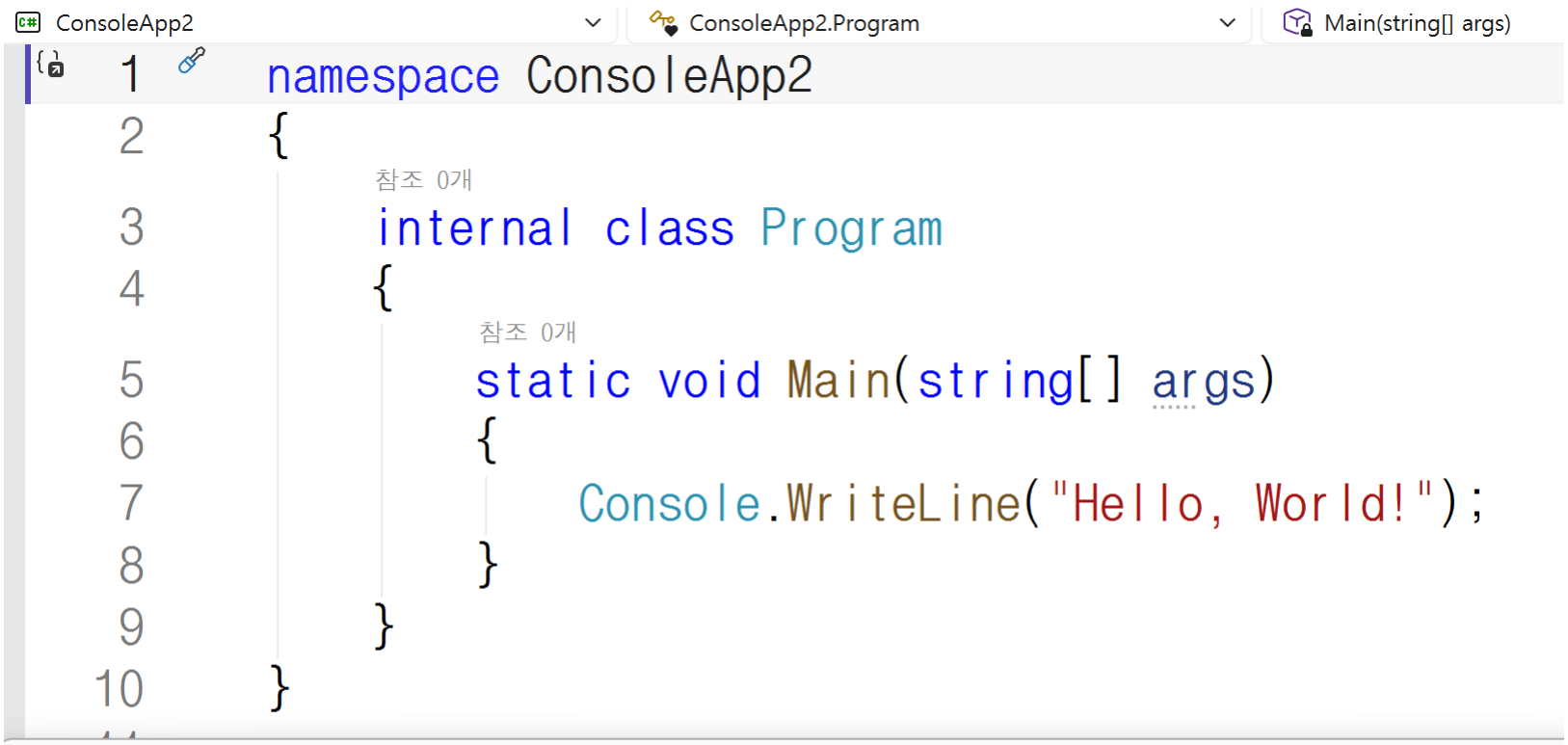
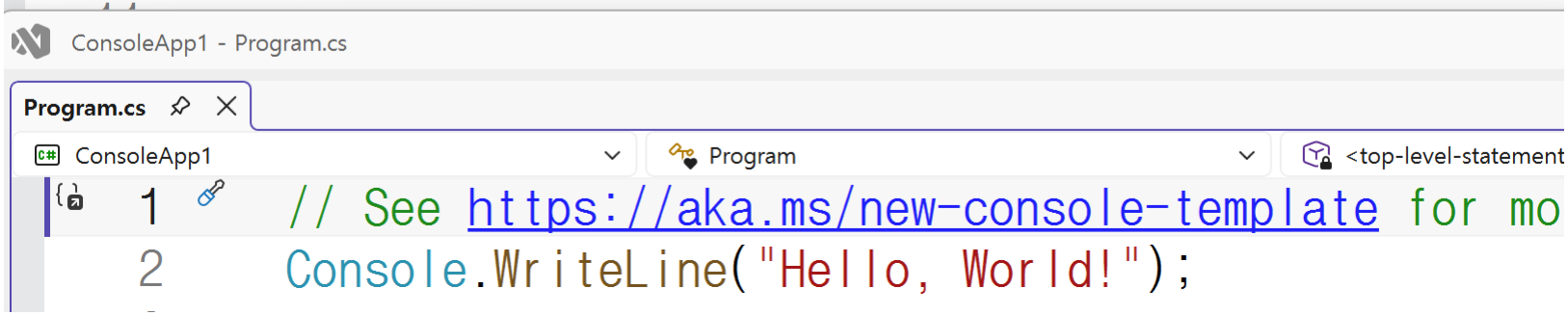


그림 1-24 [추가 정보] 대화상자

04 실습 환경 구축


ConsoleApp2
ConsoleApp2.Program
Main(string[] args)
namespace ConsoleApp2
{
참조 0개
internal class Program
{
참조 0개
static void Main(string[] args)
{
Console.WriteLine("Hello, World!");
}
}
}


ConsoleApp1 - Program.cs
Program.cs
ConsoleApp1
Program
<top-level-statement
// See https://aka.ms/new-console-template for mo
Console.WriteLine("Hello, World!");

■ 프로그램 생성

▪ 프로젝트 생성

- 프로젝트를 만들면 다음과 같은 화면이 표시되고, 오른쪽 [솔루션 탐색기] 패널을 보면 프로젝트의 구조를 확인할 수 있음

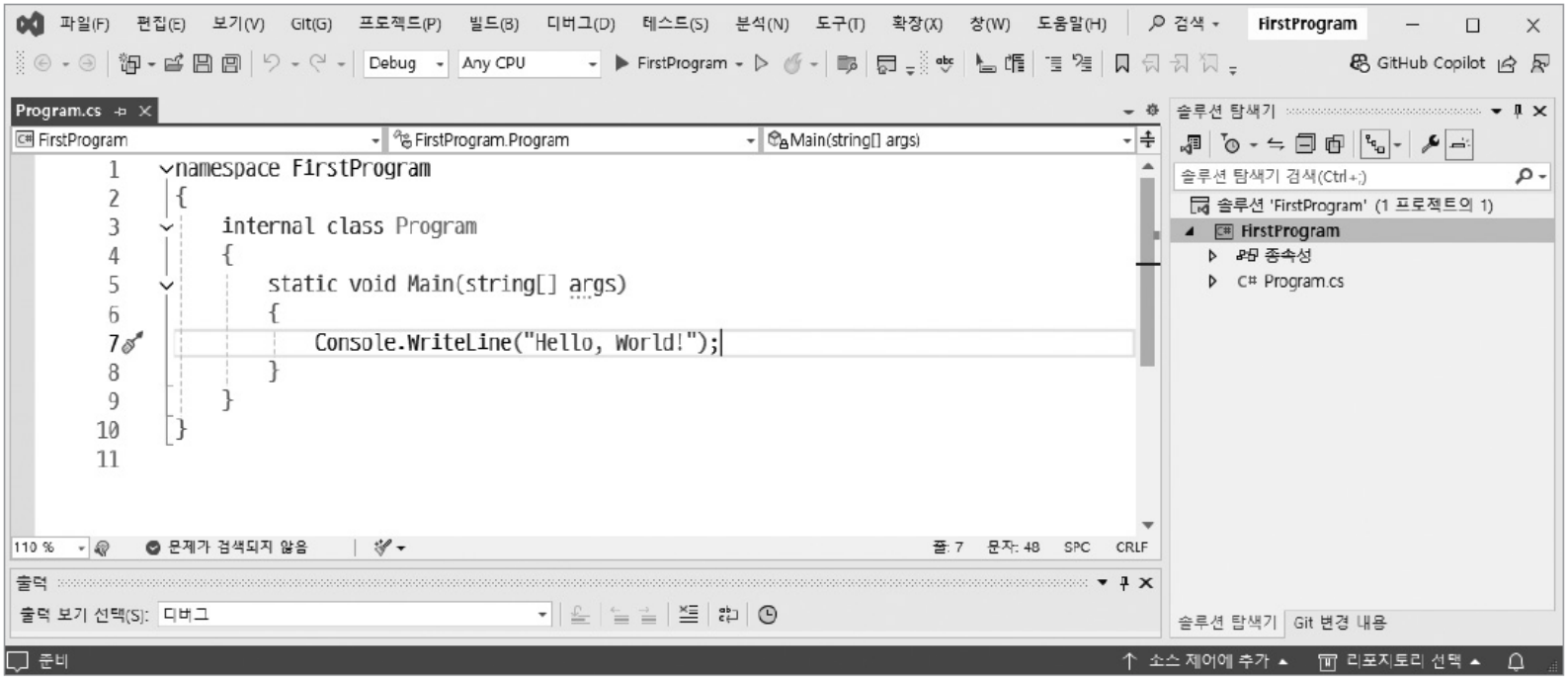


그림 1-25 생성된 프로젝트

■ 프로그램 생성

▪ 프로젝트 생성

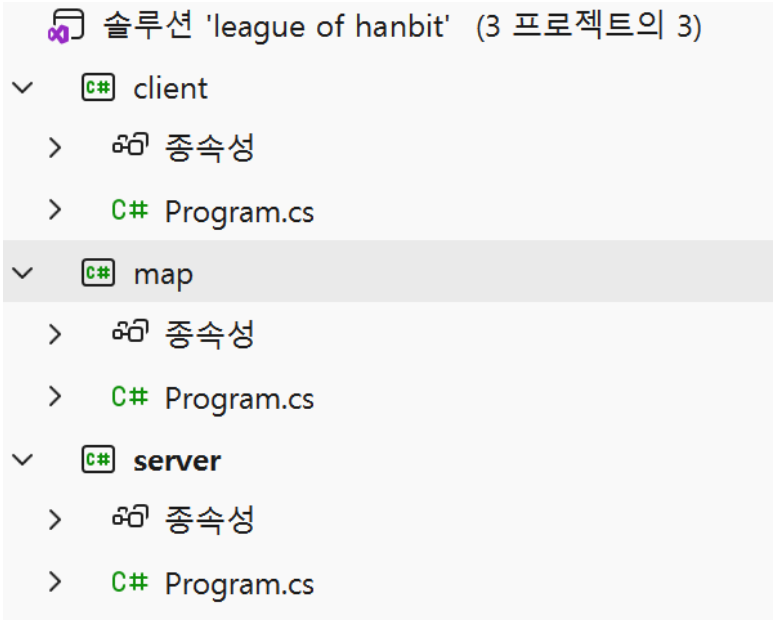
- 프로그램의 중심이 되는 Program.cs 파일을 열어보면 다음과 같이 작성되어 있음

```
namespace FirstProgram
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

여기서 잠깐!

■ 프로젝트와 솔루션

- 프로젝트(Project): 소프트웨어를 개발하는 데 필요한 파일, 소스 코드, 이미지 등을 포함하는 단위
- 솔루션(Solution): 이러한 프로젝트를 묶어서 관리하는 단위
- 예를 들어 '리그 오브 한빛'이라는 게임을 만든다면, '리그 오브 한빛 솔루션'이라는 거대한 솔루션 안에 '서버 프로젝트', '클라이언트 프로젝트', '맵 관리 프로그램 프로젝트' 등을 넣어 구성



■ 프로그램 생성

▪ 프로젝트 실행

- 프로젝트를 실행할 때는 상단의 ▶ 시작] 버튼 또는 [디버그]-[디버깅 시작].
- [F5] 키도 사용 가능

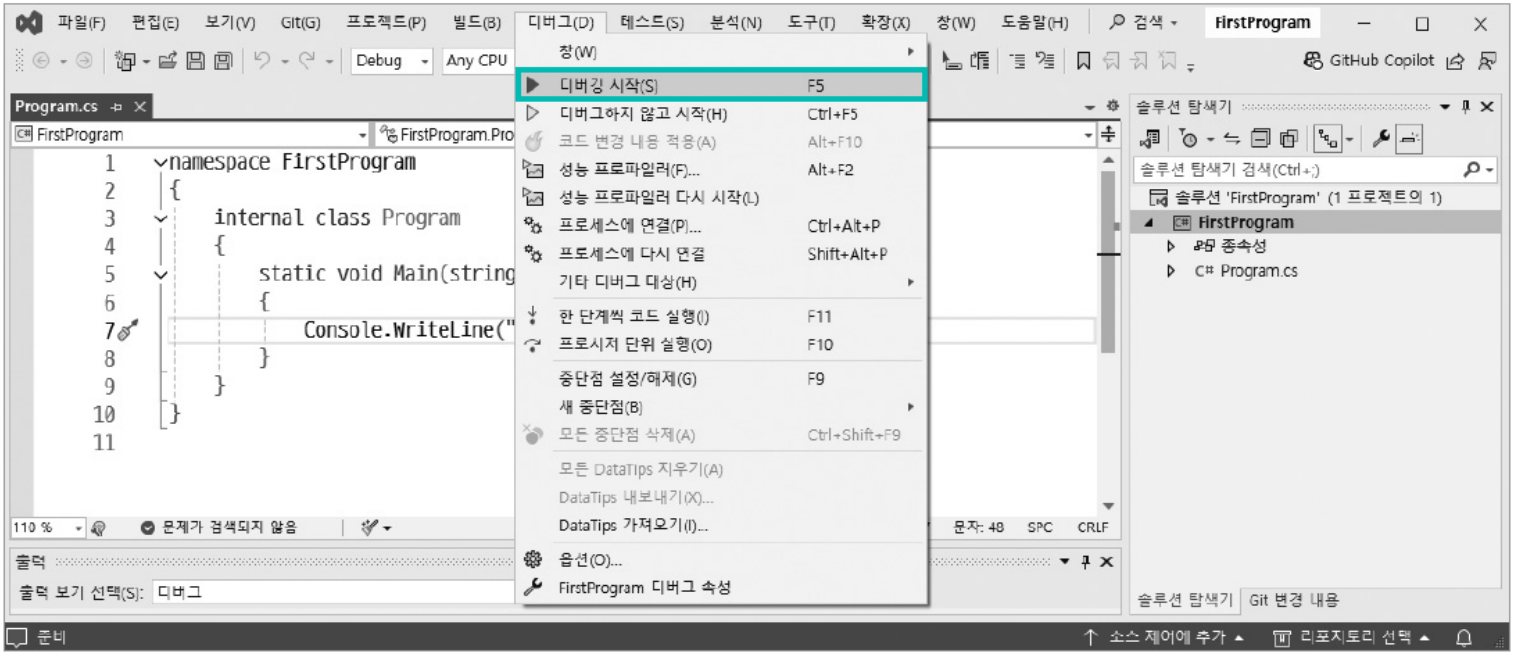


그림 1-26 프로젝트 실행

■ 프로그램 생성

▪ 프로젝트 실행

- 실행하면 콘솔창이 새로 열리고 다음과 같이 출력됨

```
Hello World!

C:\Users\arin\source\repos\FirstProgram\FirstProgram\bin\Debug\net8.0\FirstProgram.exe(프
로세스 21216개)이(가) 종료되었 습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로
콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

■ 프로그램 생성

▪ 오류 확인 방법

- 다음 코드를 실행

```
static void Main(string[] args)
{
    Console.WriteRine("Hello World");
}
```

- 오류가 있는 코드를 실행하면 다음과 같은 새로운 창이 나옴. “아니요”를 눌러줌

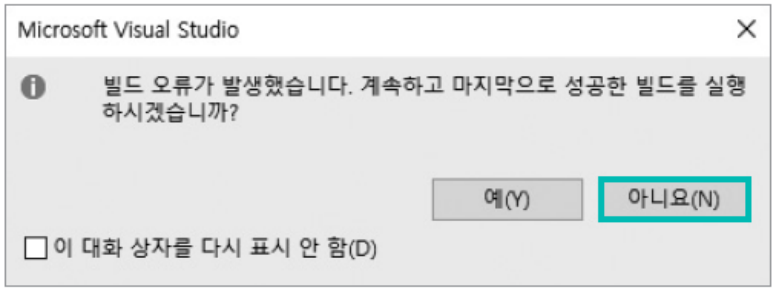


그림 1-27 오류 발생

■ 프로그램 생성

▪ 자동 완성 기능과 보조 기능

- 코드를 입력할 때 [Ctrl] + [Space] 단축기를 누르면 자동 완성 기능이 실행됨
- 그리고 현재 위치에서 사용할 수 있는 코드가 뜸

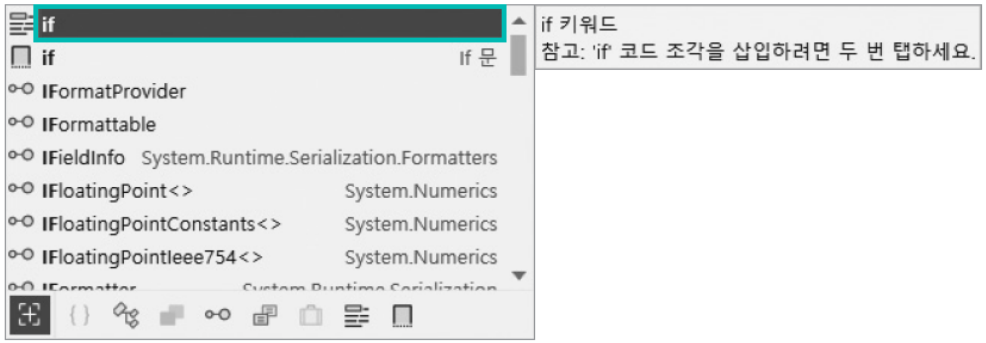


그림 1-28 자동 완성 기능 1

- 메서드를 사용할 때는 해당 메서드와 관련된 설명이 뜸

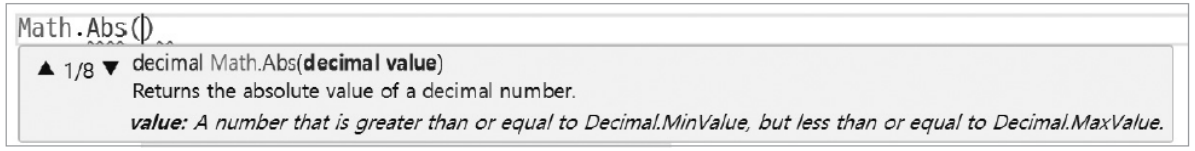


그림 1-29 자동 완성 기능 2

■ 프로그램 생성

▪ 프로그래밍을 잘 하는 습관

- 프로그래밍 언어도 하나의 언어이기 때문에 말을 많이 해보는 것이 중요
- 자동 완성 기능을 활용해보기. Console.WriteLine 을 입력하면 다음과 같이 자동 완성됨

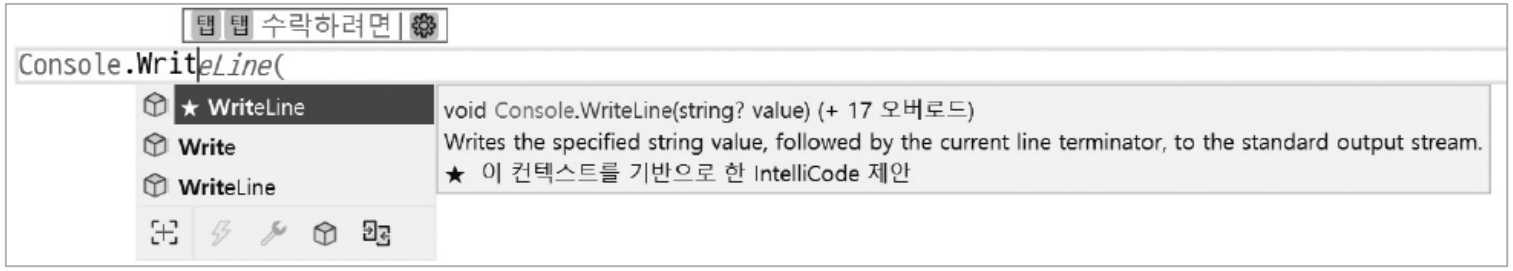


그림 1-30 write 관련 메서드 1

■ 프로그램 생성

- 프로그래밍을 잘 하는 습관
- 기본예제 1-1 Write( )와 WriteLine( ) 메서드 사용해보기 /1장/WriteAndWriteLine

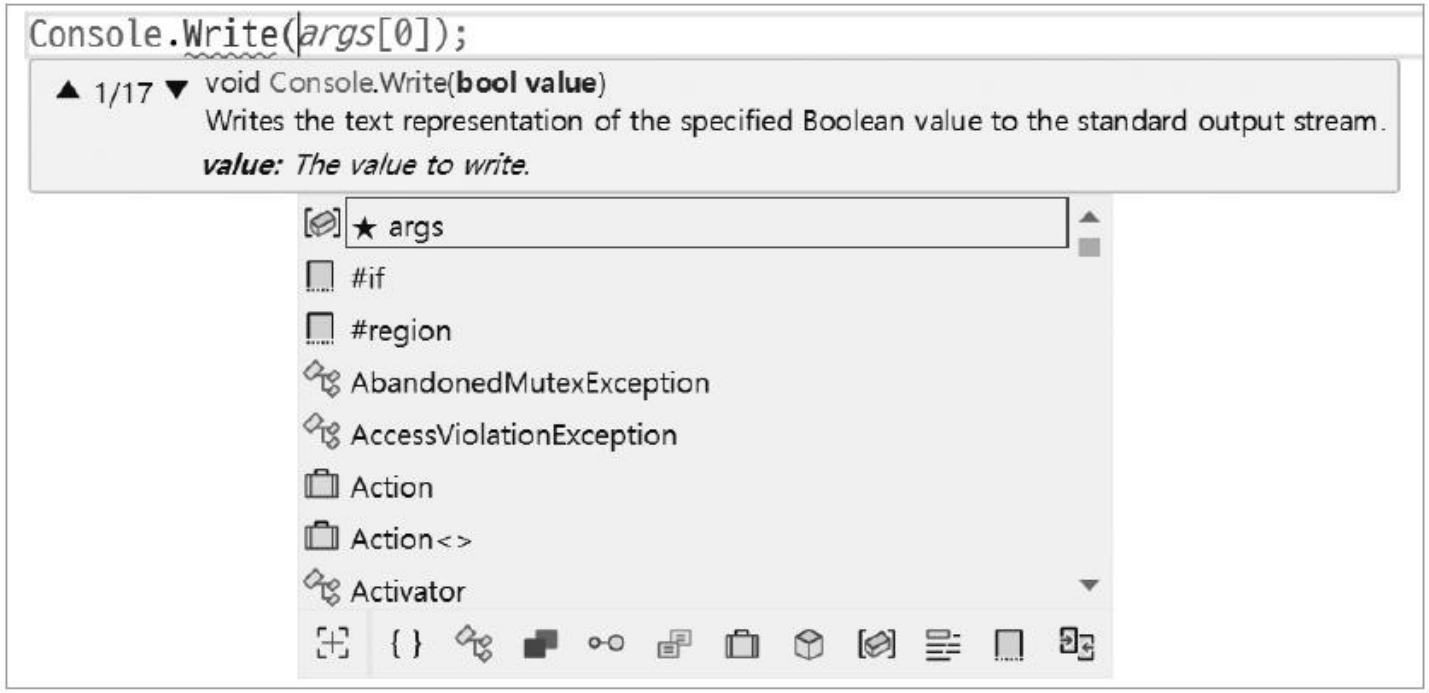


그림 1-31 write 관련 메서드 2

■ 프로그램 생성

- 프로그래밍을 잘 하는 습관
- 기본예제 1-1 Write( )와 WriteLine( ) 메서드 사용해보기 /1장/WriteAndWriteLine

코드 1-1 Write() 메서드 /1장/Write

```
01 static void Main(string[] args)
02 {
03     Console.Write("Write");
04     Console.Write("Write");
05     Console.Write("Write");
06     Console.Write("Write");
07     Console.Write("Write");
08 }
```

실행 결과

```
WriteWriteWriteWriteWrite
```

■ 프로그램 생성

- 프로그래밍을 잘 하는 습관
- 기본예제 1-1 Write( )와 WriteLine( ) 메서드 사용해보기 /1장/WriteAndWriteLine

코드 1-2 WriteLine() 메서드 /1장/WriteLine

```
01 static void Main(string[] args)
02 {
03     Console.WriteLine("WriteLine");
04     Console.WriteLine("WriteLine");
05     Console.WriteLine("WriteLine");
06     Console.WriteLine("WriteLine");
07     Console.WriteLine("WriteLine");
08 }
```

실행 결과

```
WriteLine
WriteLine
WriteLine
WriteLine
WriteLine
```

■ 프로그램 생성

- 프로그래밍을 잘 하는 습관
- **기본예제 1-1** Write( )와 WriteLine( ) 메서드 사용해보기 /1장/WriteAndWriteLine

코드 1-3 Write()와 WriteLine() 메서드의 차이 /1장/WriteAndWriteLine

```
01 static void Main(string[] args)
02 {
03     Console.Write("Write");
04     Console.WriteLine("WriteLine");
05
06     Console.WriteLine("WriteLine");
07     Console.WriteLine("WriteLine");
08
09     Console.Write("Write");
10     Console.Write("Write");
11 }
```

실행 결과

```
WriteWriteLine
WriteLine
WriteLine
WriteWrite
```

- Write() 메서드는 문자열을 출력하고 다음 줄로 넘어가지 않고, WriteLine() 메서드는 문자열을 출력하고 다음 줄로 넘어감