

Chapter 01

C# 프로그래밍 첫걸음

01 플랫폼과 프로그래밍 언어

02 라이브러리와 프레임워크

03 C#으로 할 수 있는 일

04 실습 환경 구축

Section 01 플랫폼과 프로그래밍 언어

■ C#의 특징

- 형식 안정 Type-Safe적인 객체 지향 Object-Oriented 언어
- 기존 프로그래밍 언어의 생산성을 개선하여 성능이 굉장히 높음
- **현재 윈도우, 맥, 리눅스, 안드로이드, 아이폰 등의 다양한 운영체제나 플랫폼에서 동작**
- 윈도우에서 동작하는 닷넷 플랫폼과 대부분의 운영체제에서 동작하는 모노 플랫폼에서 작동하는 프로그램을 만들 수 있음

Section 01 플랫폼과 프로그래밍 언어

■ 플랫폼과 소프트웨어 플랫폼

- 플랫폼 Platform : 소프트웨어 응용 프로그램의 실행에 사용되는 하드웨어와 소프트웨어의 집합



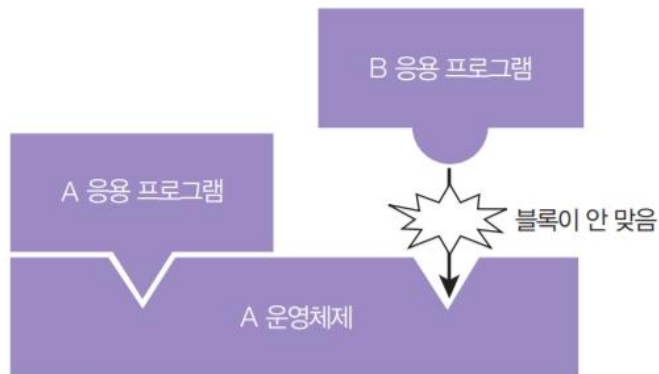
그림 1-1 플랫폼과 응용 프로그램

- 운영체제 = 플랫폼 (과거에는)

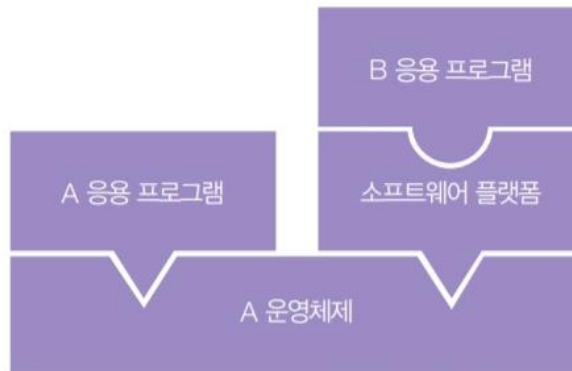
Section 01 플랫폼과 프로그래밍 언어

■ 소프트웨어 플랫폼

- [그림 1-2]와 같이 중간 레고 블록 역할을 함
- 예 : 자바 가상 머신, 액션스크립트, 닷넷 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

Section 01 플랫폼과 프로그래밍 언어

■ 닷넷 플랫폼

- 마이크로소프트 사가 만든 중간 레고 블록, 플랫폼의 기본적인 발전 형태
- 초기에는 윈도우에서만 동작했지만 현재는 마이크로소프트 사가 활용할 수 있는 모든 프로그래밍 언어(20개 이상)를 연결 가능
- C#은 모노 플랫폼이라는 소프트웨어 플랫폼 위에서도 동작(교재 그림 1-5)



그림 1-3 초기의 닷넷 플랫폼

■ C#의 기타 활용

- 게임 프레임워크(게임 엔진): 유니티
- 모바일 응용 프로그램 프레임워크(안드로이드와 아이폰): Xamarin

Section 02 라이브러리와 프레임워크

■ 라이브러리

- 코드를 쉽게 사용할 수 있게 미리 만들어준 코드
- 프로그램 소프트웨어를 만들 때 사용하는 클래스 또는 서브루틴의 집합
- 개발자가 사용해줘야만 하며, 스스로는 아무것도 하지 못함

사용자 코드

```
int Main() {  
    Console.WriteLine("출력");  
    Math.Abs(-273);  
    new Thread();  
}
```

라이브러리

```
void WriteLine() {...}  
int Abs() {...}  
public Thread() {...}
```

그림 1-6 라이브러리는 사용자 코드에게 호출되는 것

Section 02 라이브러리와 프레임워크

■ 프레임워크

- 제어 역전 Inversion of Control: IoC이 있는 대규모의 라이브러리
 - 제어역전 : 원래 개발자가 제어하던 코드를 프레임워크가 제어하는 것
- 프로그램의 초기화부터 종료까지의 흐름을 직접 관리
- 기본 틀 Framework을 모두 제공해주어 개발자는 개발에 집중 가능
- 현재는 “대규모의 라이브러리 = 프레임워크”로 많이 혼용
 - 닷넷 프레임워크는 닷넷 플랫폼과 클래스 라이브러리가 합쳐진 하나의 제품 이름

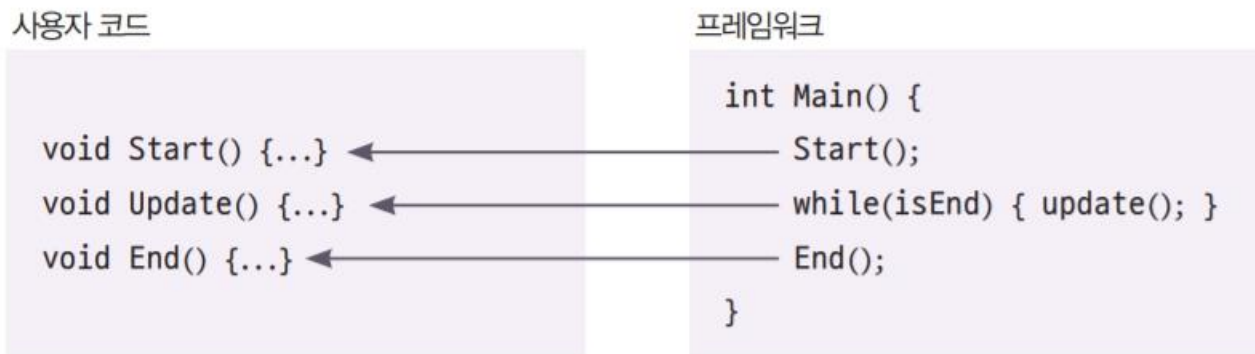


그림 1-7 프레임워크는 사용자 코드를 호출하는 것(제어 역전)

Section 03 C#으로 할 수 있는 일

■ GUI 개발

- 윈도우에서 작동하는 GUI(Graphical User Interface) 프로그램 개발을 위한 윈도우 폼과 WPF 제공
- 윈도우 폼(Windows Form)
 - C#의 가장 기초 프레임워크로, C++를 사용한 윈도우 개발을 C#으로 옮겨 놓은 형태
 - 개발자가 폼 디자이너 이용해 도구 상자에서 버튼이나 콤보 박스 등의 컨트롤을 윈도우에 배치할 때마다 폼 디자이너가 프로그램의 UI를 표시하면서 뒤로는 C# 코드를 자동으로 만들어 줌
- WPF(Windows Presentation Foundation)
 - 현대적인 개발 패턴 중에 MVC(Model-View-Controller) 패턴과 MVVM(Model-View-View Model) 패턴을 적용해 개발 생산성을 향상시킨 프레임워크
 - DirectX 등의 기능도 추가로 내장하여 3D 그래픽까지 자체적으로 처리 가능

Section 03 C#으로 할 수 있는 일

■ 웹 개발

- C#이 가장 대표적으로 사용되는 부분
- 마이크로소프트 사는 2가지 프레임워크 지원(ASP.NET 프레임워크와 ASP.NET MVC)

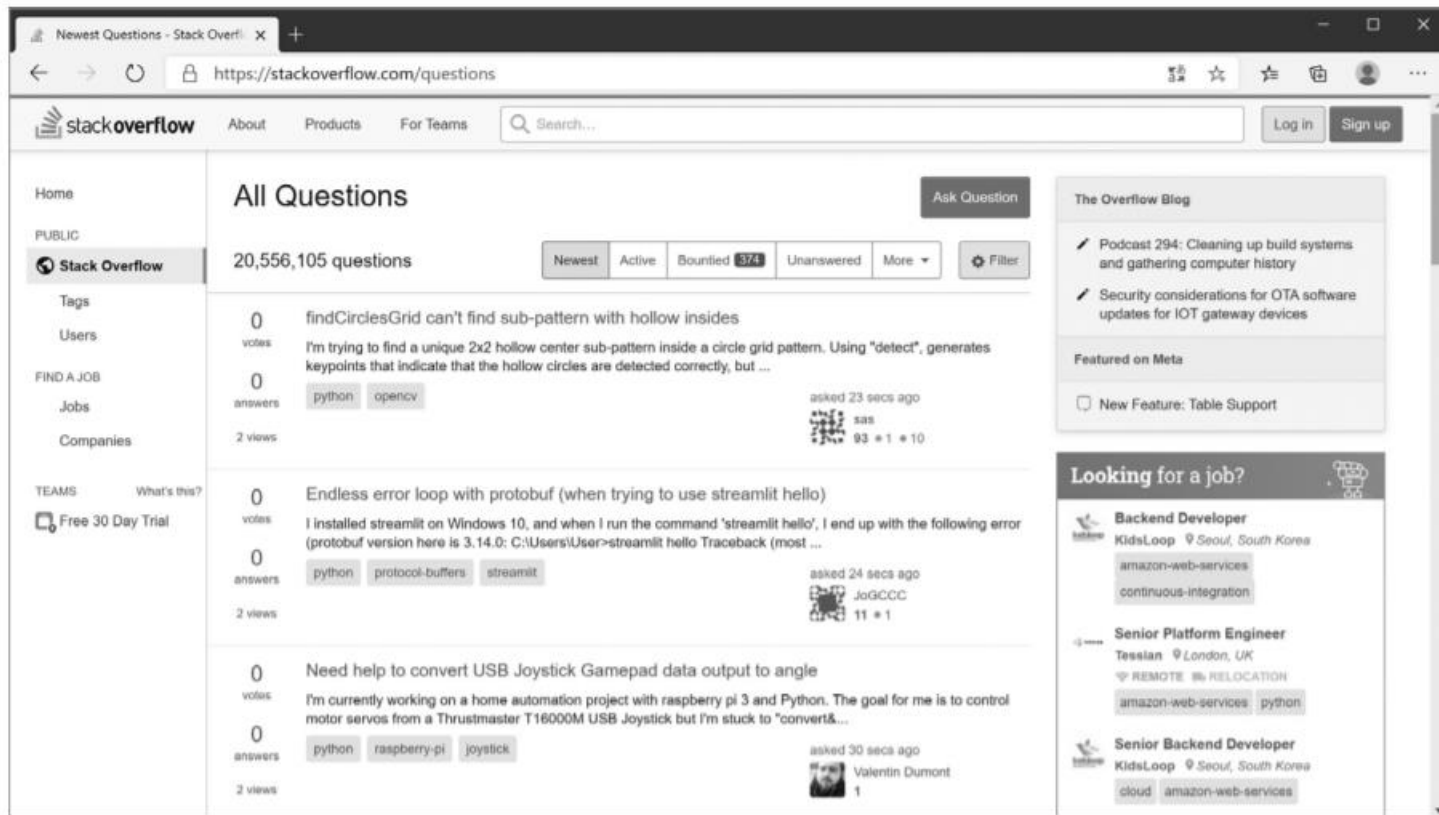


그림 1-10 ASP.NET MVC로 개발된 스택오버플로

Section 03 C#으로 할 수 있는 일

■ 게임 개발

- C#을 이용해 게임 클라이언트와 게임 서버 개발 가능
- 게임 클라이언트 개발
 - 유니티 엔진 개발로 C#으로 게임 클라이언트 개발 활성화
 - 유니티는 모노 플랫폼을 사용해 다양한 플랫폼에서 작동하는 게임 개발 가능

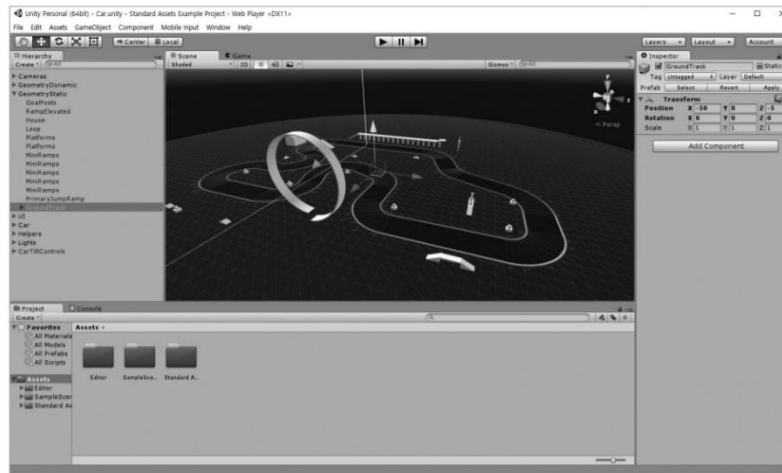


그림 1-11 유니티 엔진

- 게임 서버 개발
 - 간단한 모바일 게임 서버는 웹 서버처럼 개발
 - 대규모 MMORPG Massively Multiplayer Online Role-Playing Game 게임 서버도 개발

Section 03 C#으로 할 수 있는 일

■ IoT(사물 인터넷Internet of Things: IoT) 개발

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

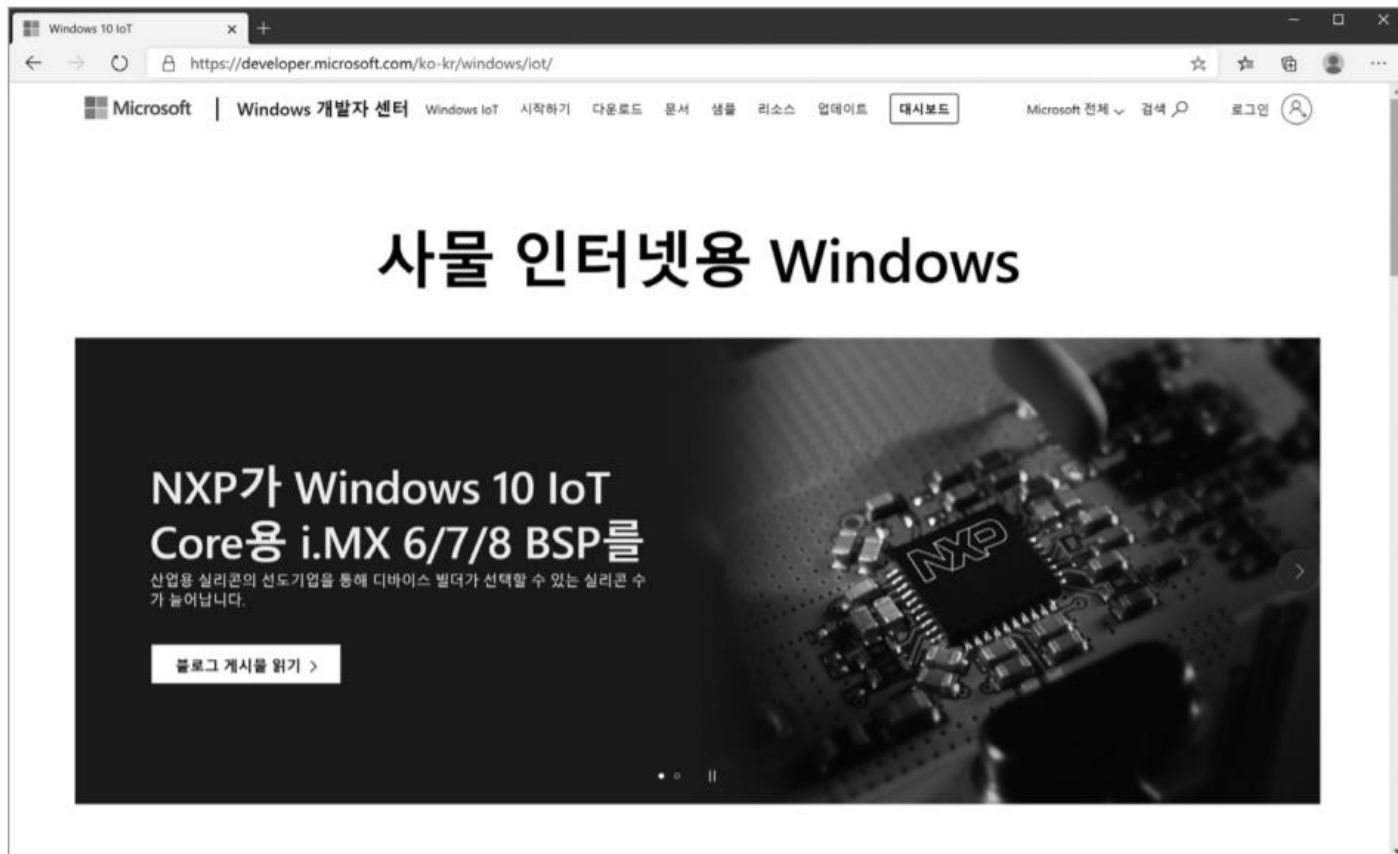


그림 1-12 윈도 10 IoT 코어

■ Visual Studio 2019 설치

- <https://visualstudio.microsoft.com/ko/vs/preview>에서 'Visual Studio 2019 Preview 커뮤니티 버전' 다운
- 반드시 커뮤니티 버전을 다운로드

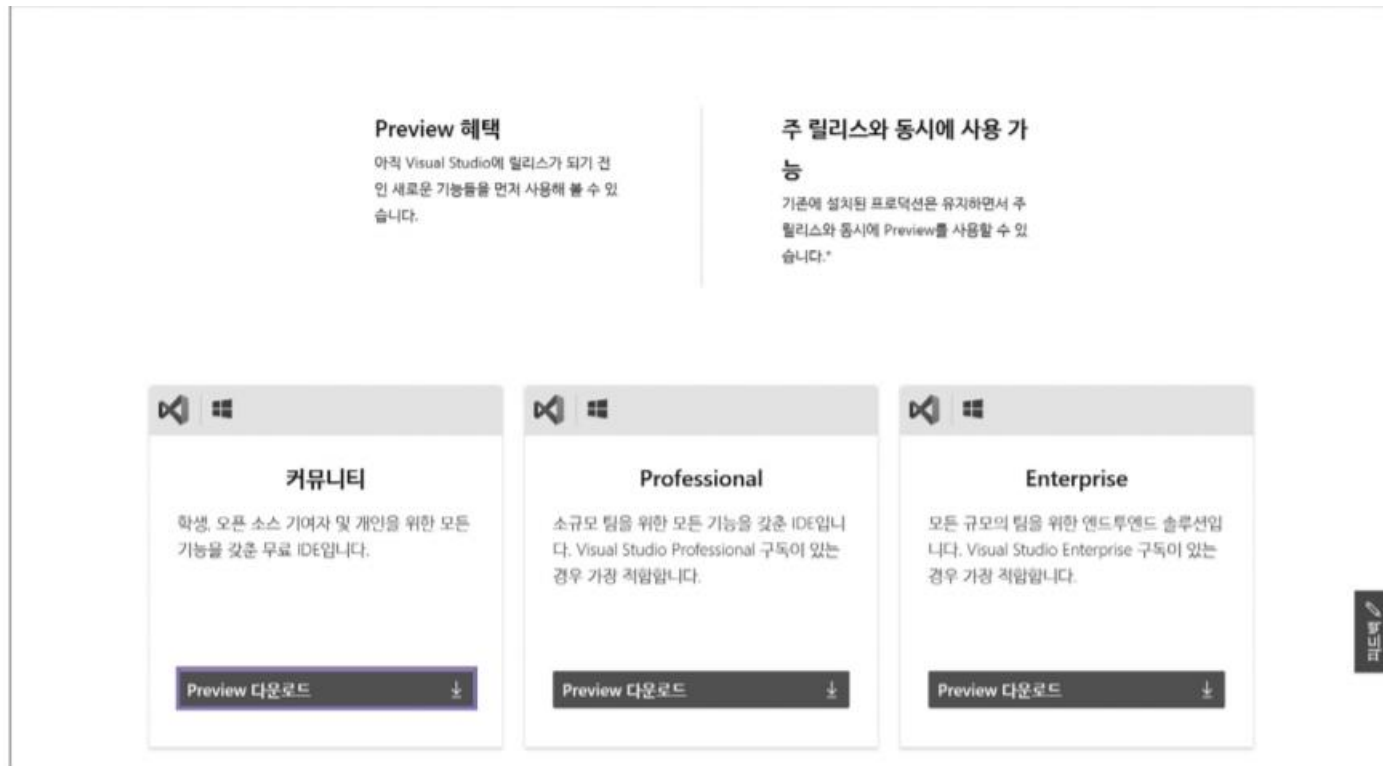


그림 1-13 Visual Studio 2019 Preview 페이지

Section 04 실습 환경 구축

■ Visual Studio 2019 설치

- 설치할 때 다음과 같은 팩 설치 화면이 나오면
[데스크톱 및 모바일]-[.NET 데스크톱 개발]에 체크하고 설치

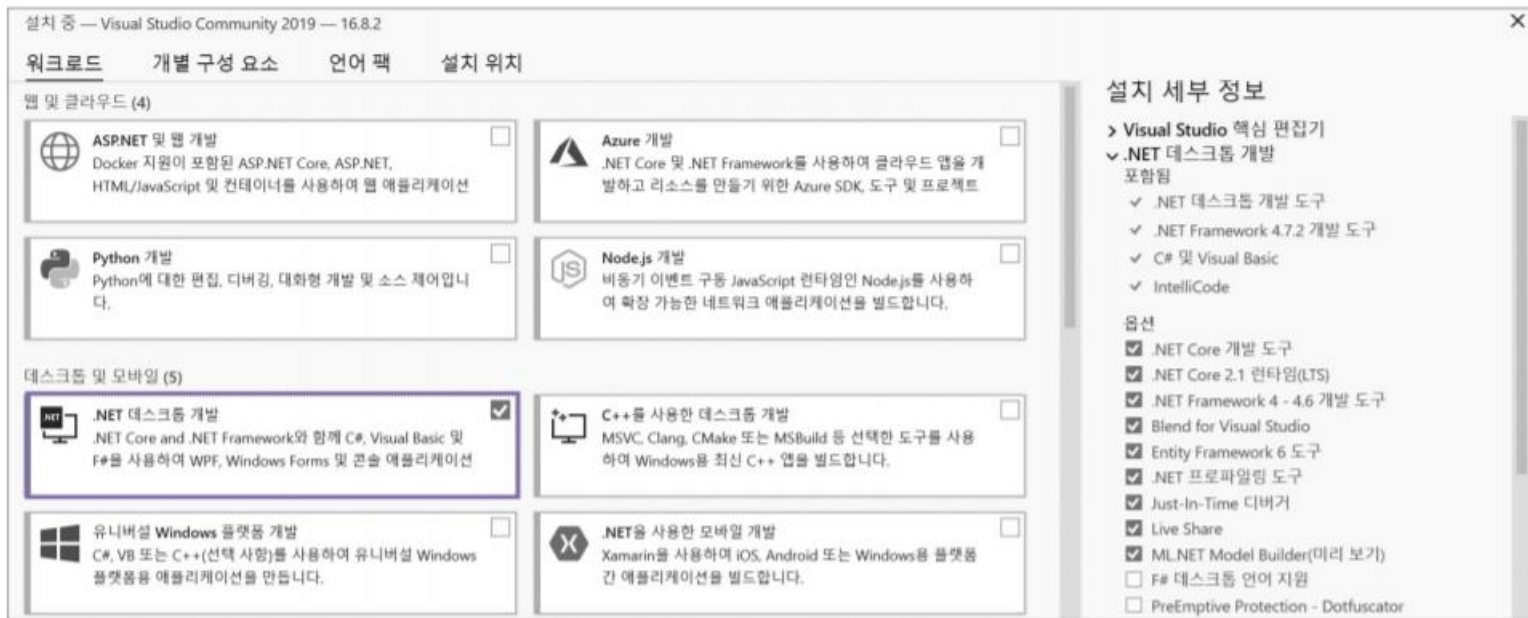


그림 1-14 설치 팩 선택 화면

■ .NET 5.0 SDK 설치

- 2020년 12월을 기준으로 닷넷은 5.0 버전이 최신으로 2022년 버전은 별도로 설치하지 않음
- Visual Studio 2019는 닷넷 최신 버전을 기본적으로는 지원하지 않음 → 따라서 별도로 설치해서 사용해야 함
 - <https://dotnet.microsoft.com/download/visual-studio-sdks>에 접속하고 'Visual Studio 2019 SDK'를 다운로드
 - 페이지에서 [.NET Core] 내부에 있는 [.NET 5.0]-[Visual Studio 2019 SDK]-[x64 SDK]를 클릭해서 다운로드 후 설치

Section 04 실습 환경 구축

■ .NET 5.0 SDK 설치

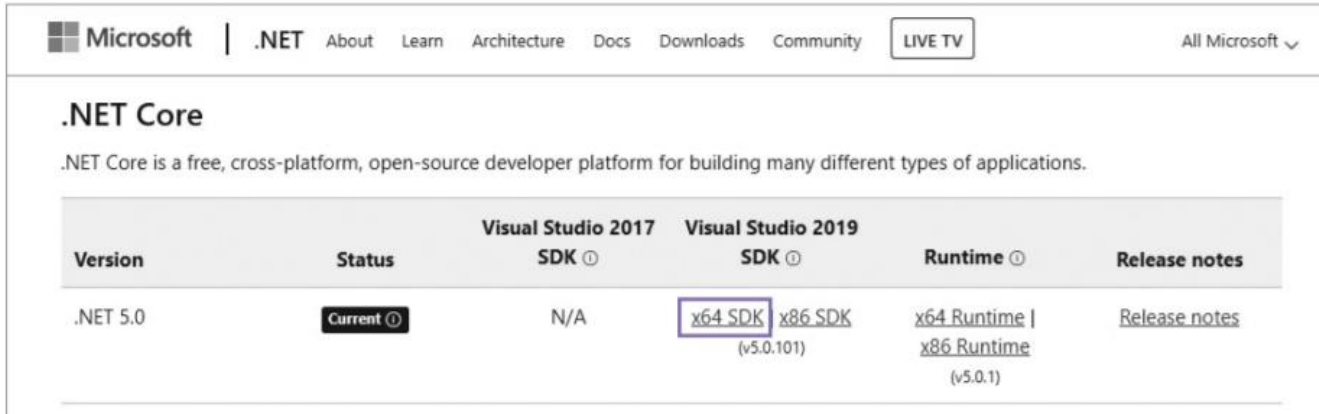


그림 1-15 Visual Studio .NET 다운로드 페이지



그림 1-16 Visual Studio .NET 설치 진행

■ 한국어 보조 기능 설치

- 2020년 12월을 기준으로 Visual Studio는 보조 기능을 영어로만 출력
- 처음 공부할 때는 한국어로 보조 기능을 살펴보는 것이 좋으므로 한국어 팩을 다운로드하고 적용
- <https://dotnet.microsoft.com/download/intellisense>에서 5.0.x 오른쪽에 있는 '한국어' 링크를 눌러서 한국어 팩을 다운로드

.NET Release	Localized IntelliSense Download for Windows
5.0.x	<ul style="list-style-type: none">• Deutsch• español• français• italiano• 日本語• 한국어• português (Brasil)• русский• 中文(简体)• 中文(繁體)

그림 1-17 NET 인텔리센스 지역화 팩 다운로드 페이지

■ 한국어 보조 기능 설치

- 다운로드한 파일의 압축을 해제하면 다음과 같은 폴더가 있음
- Microsoft.NETCore.App.RefWko
- Microsoft.WindowsDesktop.App.RefWko
- NETStandard.Library.RefWko
- 각각의 폴더를 다음 위치에 옮김
- C:\Program Files\dotnet\packs\Microsoft.NETCore.App.Ref\5.0.0\ref\net5.0
- C:\Program Files\dotnet\packs\Microsoft.WindowsDesktop.App.Ref\5.0.0\ref\net5.0
- C:\Program Files\dotnet\packs\NETStandard.Library.Ref\2.1.0\ref\netstandard2.1

Section 04 실습 환경 구축

■ 한국어 보조 기능 설치

- 폴더 내부에 있는 파일을 옮기는 것이 아니라 'ko'라는 이름의 폴더 자체를 옮겨서 넣는 것임을 주의



그림 1-18 폴더 이동

■ Visual Studio 2019 실행

- 마이크로소프트 계정이 없다면 먼저 회원 가입한 후 바로 로그인



그림 1-19 Visual Studio 2019 로그인 화면

■ Visual Studio 2019 실행

- 모든 과정을 제대로 수행했으면 비주얼 스튜디오 2019가 실행

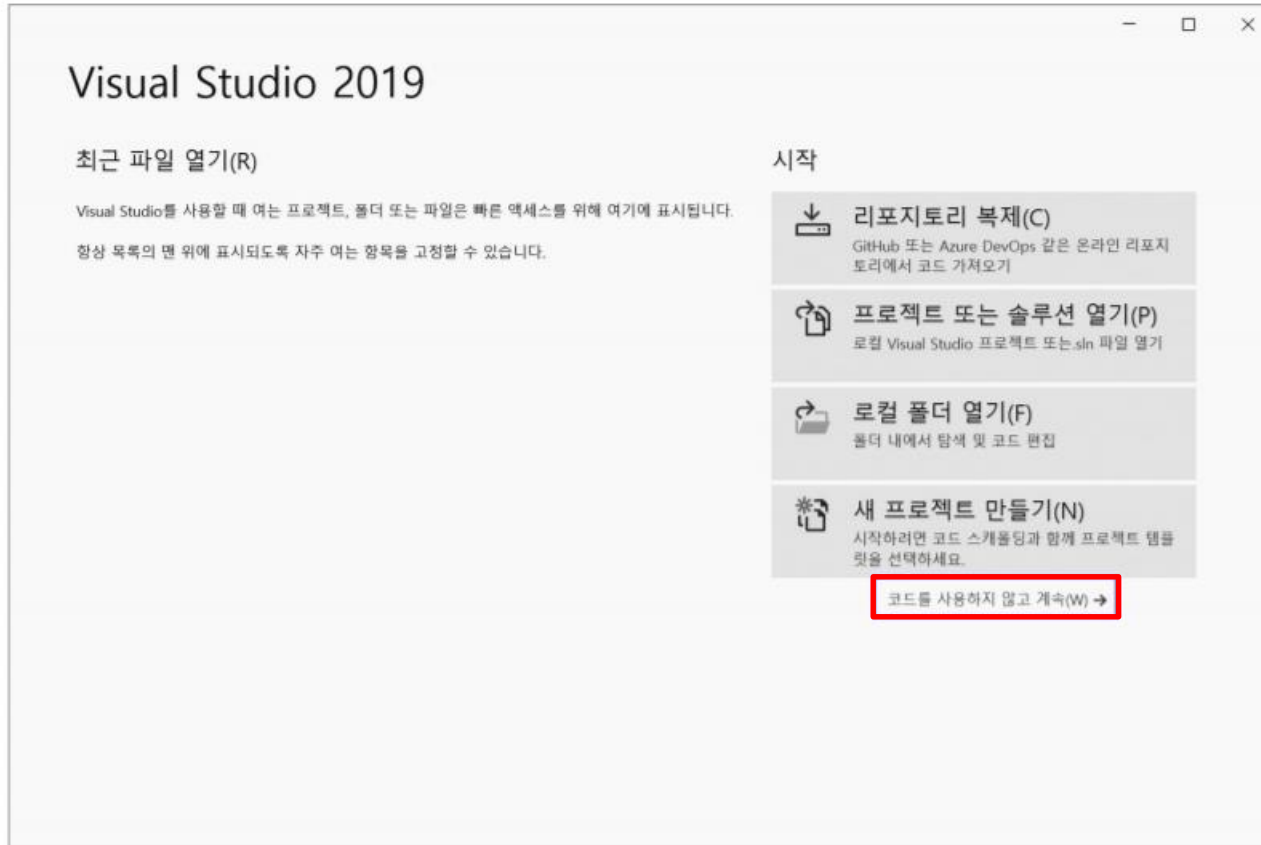


그림 1-20 Visual Studio 2019 첫 화면

Section 04 실습 환경 구축

■ Visual Studio 2019 실행

- 오른쪽 아래에 있는 '코드를 사용하지 않고 계속(W)'을 누르면, Visual Studio 기본 화면이 나옴

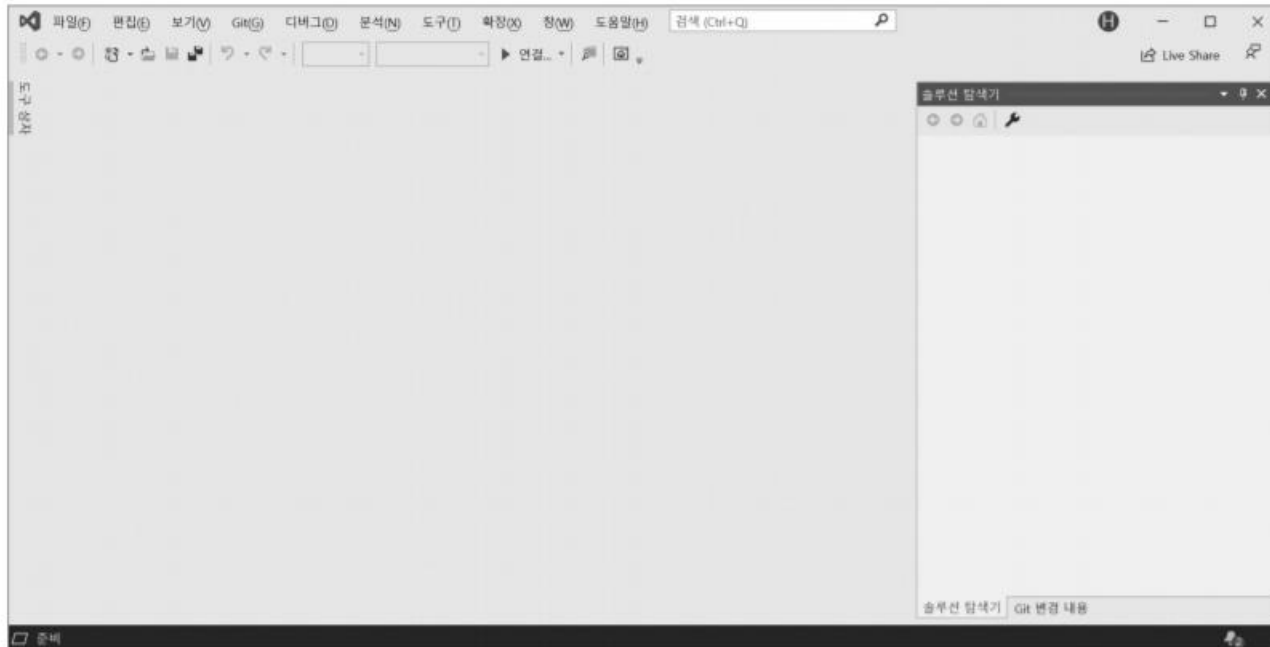


그림 1-21 Visual Studio 2019 기본 화면

Section 04 실습 환경 구축

■ 닷넷 5.0 프로젝트 활성화

- Visual Studio 2019의 [도구]-[옵션]으로 옵션 대화상자를 띄우고, [환경]-[미리 보기 기능]에서 '새 프로젝트 대화상자에 모든 .NET Core 템플릿 표시(다시 시작이(가) 필요함)'를 체크

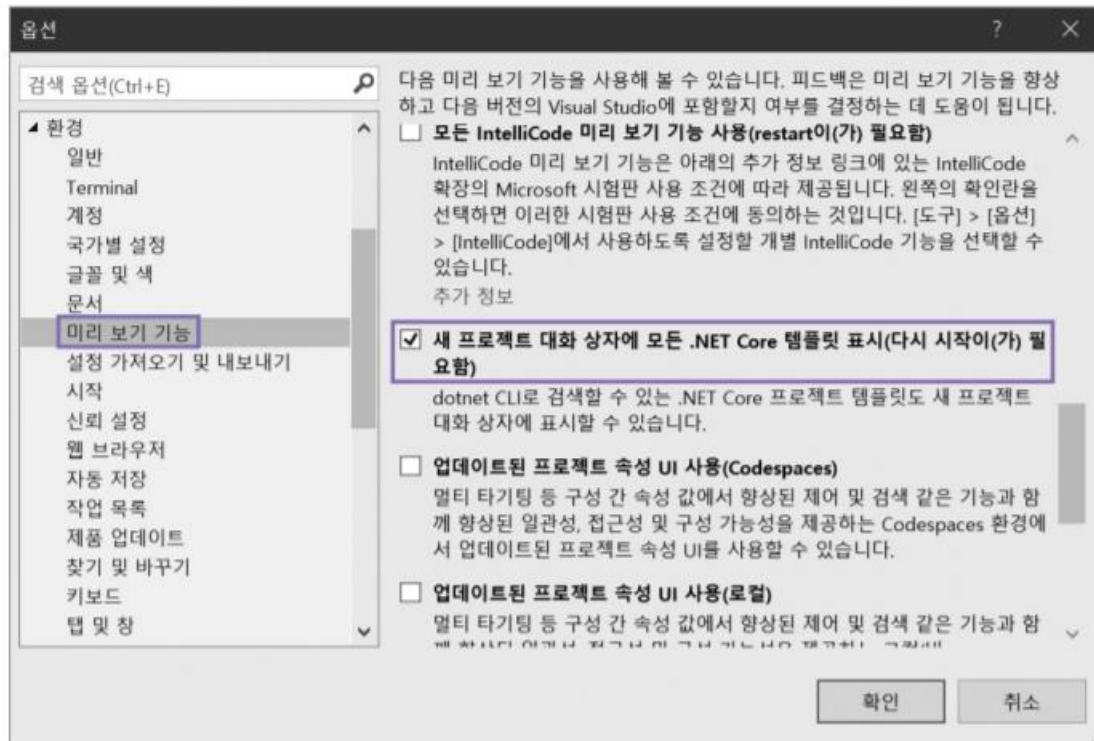


그림 1-22 미리 보기 기능 활성화

■ 프로그램 생성

■ 프로젝트 생성

- [파일]-[새로 만들기]-[프로젝트] 메뉴를 선택
- 새 프로젝트 만들기 대화상자가 나오면, Console Application(또는 콘솔 앱)을 선택(찾기 힘들다면 위의 드롭다운 메뉴에서 [C#]-[Windows]-[콘솔]을 선택)
- Console Application(또는 콘솔 앱)이 2개 이상 나오는 것을 볼 수 있음
- '.NET Framework' 등이 붙지 않은 단순한 Console Application(또는 콘솔 앱)을 선택하고 [다음]을 누름



그림 1-23 프로젝트 만들기 대화상자

Section 04 실습 환경 구축

■ 프로그램 생성

■ 프로젝트 생성

- 새 프로젝트 구성 대화상자가 나오면 프로젝트의 이름과 폴더 등을 지정
- 여기서는 프로젝트 이름만 'FirstProgram'으로 변경하고 이어서 [다음]을 누름

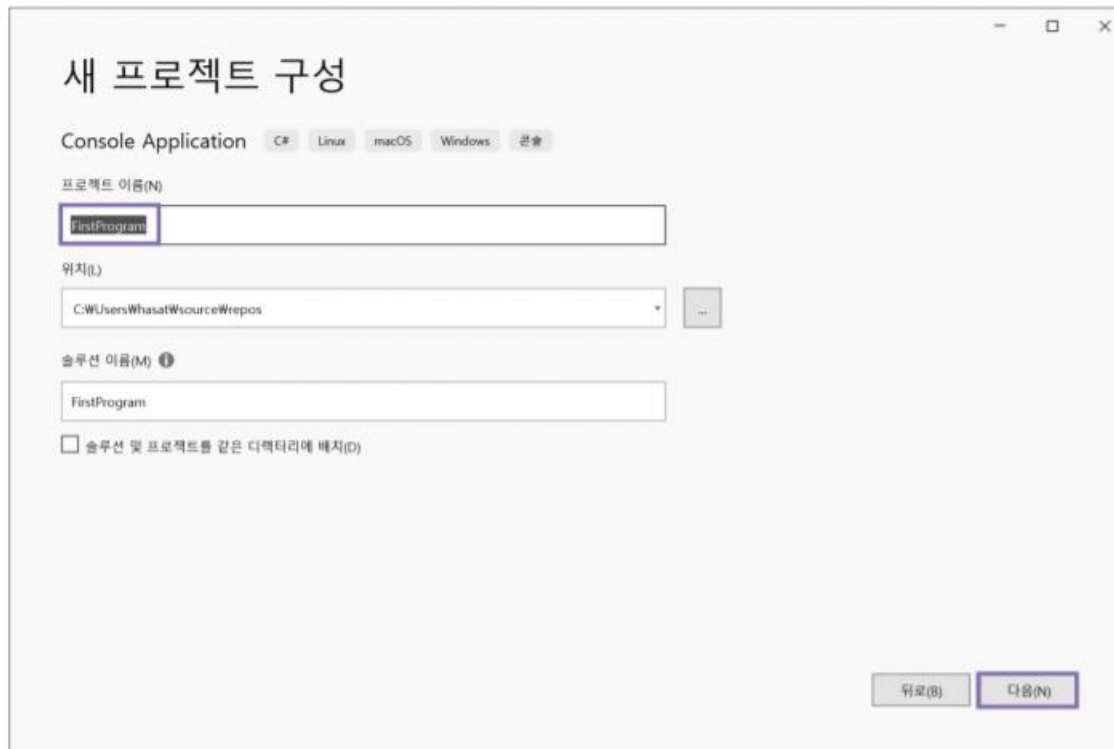


그림 1-24 새 프로젝트 구성 대화상자

Section 04 실습 환경 구축

■ 프로그램 생성

■ 프로젝트 생성

- 추가 정보 대화상자가 나오면, 대상 프레임워크를 선택
- .NET 6.0 설치되어 있으므로, 대상 프레임워크를 '**.NET 6.0**'으로 선택하고 [만들기] 버튼을 누르면 프로젝트가 생성됨

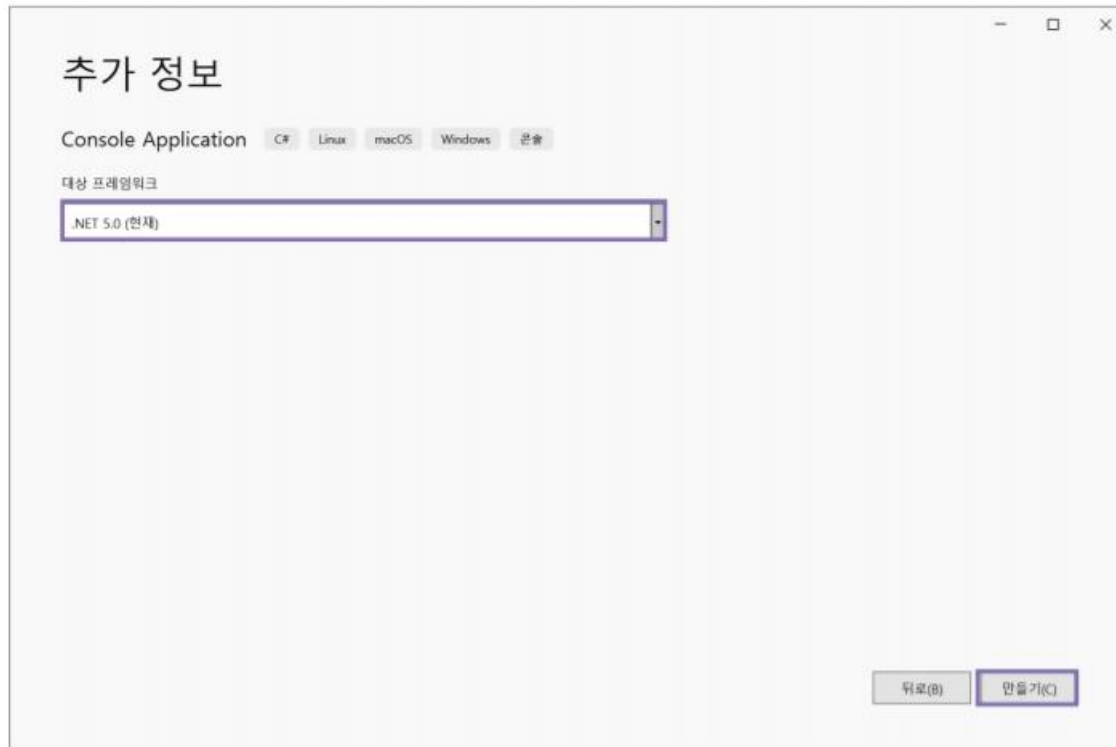


그림 1-25 생성된 프로젝트

Section 04 실습 환경 구축

■ 프로그램 생성

■ 프로젝트 생성

- 여기서 Program.cs가 프로그램의 중심이 되는 파일

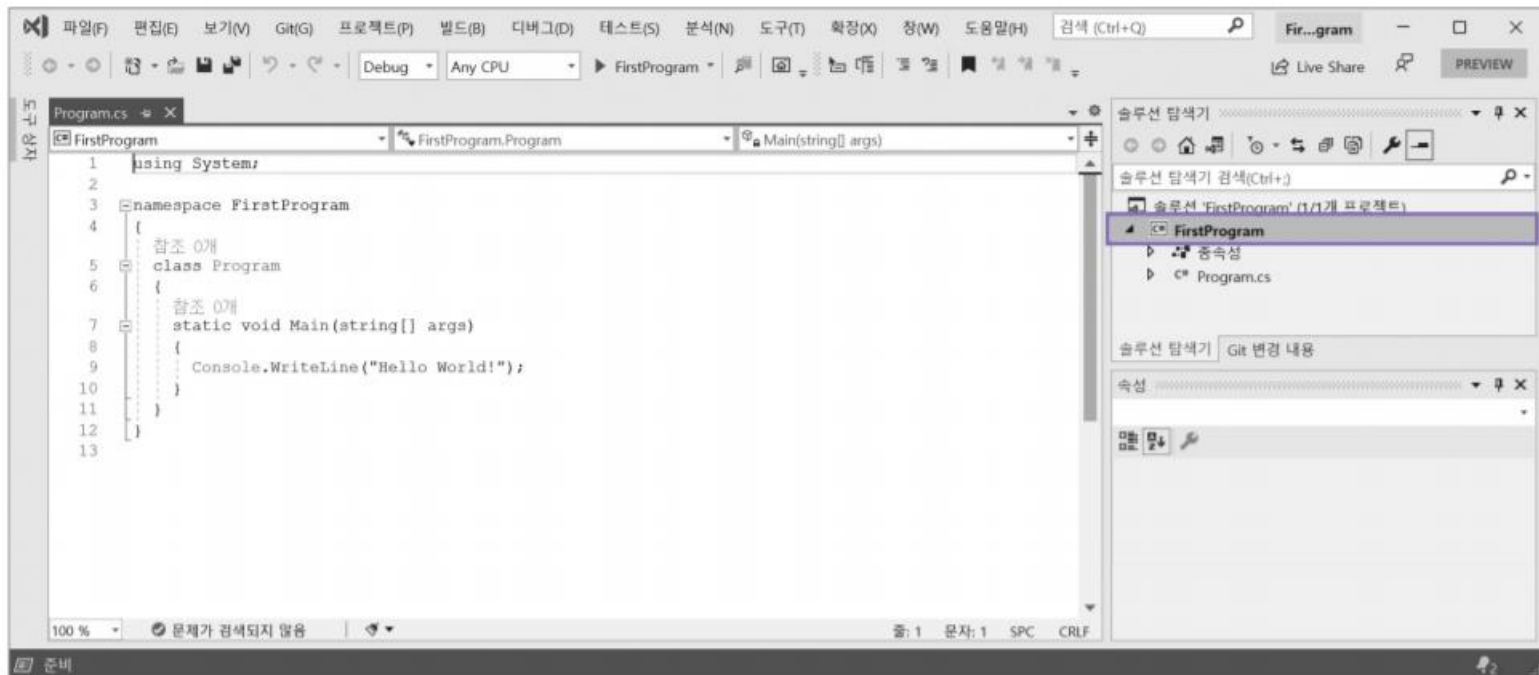


그림 1-26 솔루션 탐색기에서 생성된 프로젝트 확인

Section 04 실습 환경 구축

■ 프로그램 생성

■ 프로젝트 실행

- 프로젝트를 실행할 때는 상단에 있는 [시작] 버튼 또는 [디버그]-[디버깅 시작] 메뉴를 누름
- 또는 F5를 눌러 실행

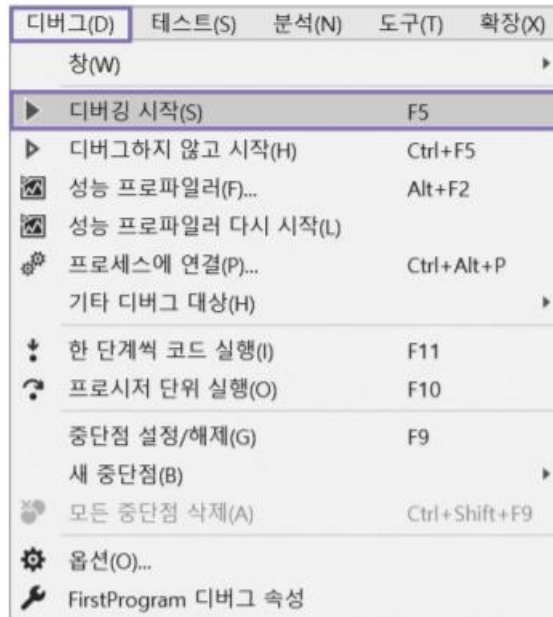


그림 1-27 프로젝트 실행

Section 04 실습 환경 구축

■ 프로그램 생성

■ 오류 확인 방법

- 오류가 있는 코드를 실행하면 다음과 같은 새로운 창이 나옴
- 오류가 발생했으므로 이전에 성공했던 코드로 실행하냐는 질문이므로 '아니요' 선택
- 이때 오류 창을 보면 해당 오류가 뜸
- 어떤 오류인지 설명이 나오며, 오류를 더블 클릭하면 해당 줄로 이동

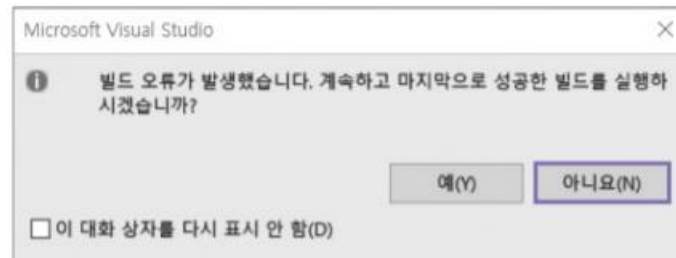


그림 1-28 오류 발생

Section 04 실습 환경 구축

■ 프로그램 생성

■ 자동 완성 기능과 보조 기능

- 코드를 입력할 때에 Ctrl + Space 단축키를 누르면 자동 완성 기능이 실행
- 자동 완성 기능이 실행되면 현재 위치에서 사용할 수 있는 코드가 뜸
- 메서드를 사용할 때는 해당 메서드와 관련된 설명이 뜸

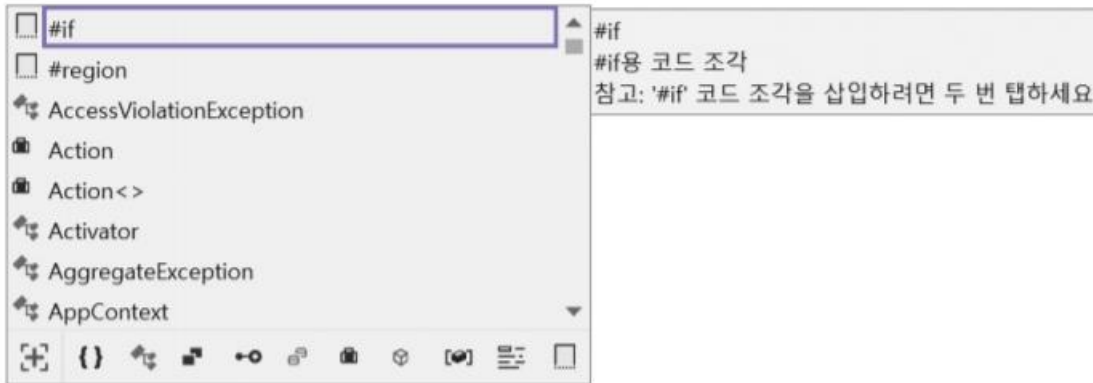


그림 1-29 자동 완성 기능 1

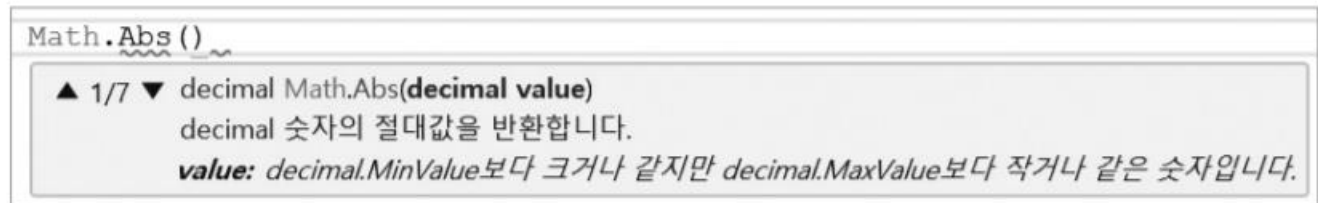


그림 1-30 자동 완성 기능 2

Section 04 실습 환경 구축

■ 프로그램 생성

- 프로그래밍을 잘 하는 습관
 - 자동 완성 기능을 활용을 자주 사용할 것
 - 직접 Write()와 WriteLine() 메서드의 차이를 비교해 볼 것

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
```