

# Chapter 01. C# 개요와 개발환경 구축

# 목차

1. 프로그래밍 언어의 개념과 종류
2. C# 소개
3. C# 프로그래밍 작성 방법
4. C# 개발환경 구축
5. 처음 작성하는 C# 프로그램

# 01

## 프로그래밍 언어의 개념과 종류

# 01. 프로그래밍 언어의 개념과 종류

## I. 프로그래밍 언어의 개념



그림 1-1 프로그래머, 프로그래밍 언어, 소프트웨어

- 프로그래밍 언어: 컴퓨터가 이해하는 언어
- 프로그래머: 프로그래밍 언어를 사용해서 소프트웨어나 앱을 만드는 직업인

# 01. 프로그래밍 언어의 개념과 종류

## I. 프로그래밍 언어의 개념



다양한 스포츠



다양한 프로그래밍 언어

그림 1-2 스포츠와 프로그래밍 언어

- 많이 사용되는 프로그래밍 언어: C, C++, 자바, HTML, PHP, 파이썬, C# 등
- C / C++: 엑셀, 한글, 웹브라우저 등 소프트웨어를 만드는 데 주로 사용
- 자바: 안드로이드 스마트폰의 앱을 구현하는 데 많이 활용

# 01. 프로그래밍 언어의 개념과 종류














---

## II. 프로그래밍 언어의 종류와 인기도

- 전 세계에 인간이 사용하는 언어의 종류는 2022년을 기준으로 7,151개 정도
- 컴퓨터 프로그래밍 언어도 700개가 넘는다고 알려져 있으며 각각의 문법과 표현력, 사용성을 지니고 있음

# 01. 프로그래밍 언어의 개념과 종류

## II. 프로그래밍 언어의 종류와 인기도

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%
6	6			Visual Basic	4.75%	-1.01%
7	7			JavaScript	2.17%	+0.09%
8	10	⬆		SQL	1.95%	+0.11%
9	8	⬇		PHP	1.61%	-0.30%
10	13	⬆		Go	1.24%	+0.26%
11	9	⬇		Assembly language	1.11%	-0.79%
12	15	⬆		MATLAB	1.08%	+0.28%
13	12	⬇		Delphi/Object Pascal	1.06%	-0.06%
14	23	⬆		Scratch	1.00%	+0.47%
15	17	⬆		Classic Visual Basic	0.98%	+0.38%
16	11	⬇		R	0.93%	-0.44%
17	30	⬆		Fortran	0.79%	+0.40%
18	16	⬇		Ruby	0.76%	+0.10%
19	26	⬆		Rust	0.73%	+0.22%
20	14	⬇		Swift	0.71%	-0.20%

02

C# 소개



## 02. C# 소개

### I. C#의 간단한 역사



그림 1-4 C# 로고 © 위키백과

- 1972년 벨 연구소의 데니스 리치(Dennis Ritchie)가 B 언어의 후속으로 개발한 언어가 C 언어
- 1983년, C 언어에 객체지향 개념이 추가된 C++가 개발
- 1991년 제임스 고슬링(James Gosling)에 의 해서 C++의 복잡성 등을 개선한 자바(Java) 언어가 개발

## 02. C# 소개

### I. C#의 간단한 역사

- 마이크로소프트에서 자바 언어의 확장판인 J++ 를 개발했으나, 곧 포기하고 2000년에 닷넷 프레임워크(.NET Framework)에서 사용하는 C# 언어를 발표했다.
- C#은 강력하고 배우기 쉬우며, 언어의 완성도가 높다는 좋은 평가를 받고 있다.
- C#은 비주얼 스튜디오(Visual Studio)로 불리는 통합 개발환경에서 개발한다.
- Visual Studio는 무료 버전과 유료 버전이 함께 제공되므로 별도의 비용 없이도 C# 응용 프로그램을 개발할 수 있다.

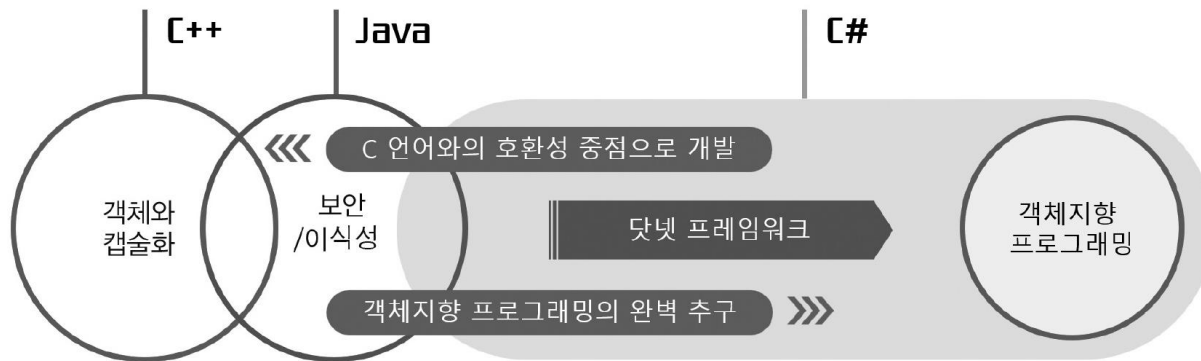


그림 1-1 C#의 탄생 유래

## 02. C# 소개

### II. C#과 Visual Studio



그림 1-5 Visual Studio 로고 © 위키백과

- C#의 버전은 통합 개발환경인 Visual Studio의 버전과 함께 발전했다.
- Visual Studio에서 처음 C#이 소개된 버전은 Visual Studio .NET 2002 버전이며 닷넷 프레임워크 1.0 버전 환경에서 구동되었다.
- 현재 Visual Studio는 유료 버전인 Enterprise, Professional 에디션과 무료 버전인 Community 에디션으로 나뉜다.

## 02. C# 소개

### II. C#과 Visual Studio

표 1-1 Visual Studio 버전에 따른 C# 버전

Visual Studio 버전(내부 버전)	닷넷 프레임워크 버전	C# 버전	기타
Visual Studio .NET 2002 (7.0)	.NET Framework 1.0	C# 1.0	.NET Framework 환경을 지원하기 시작
Visual Studio .NET 2003 (7.1)	.NET Framework 1.1	C# 1.1	
Visual Studio 2005 (8.0)	.NET Framework 2.0	C# 2.0	64bit OS 지원 무료 Express 버전 발표
Visual Studio 2008 (9.0)	.NET Framework 2.0~3.5	C# 3.0	Windows Vista 정식 지원
Visual Studio 2010 (10.0)	.NET Framework 2.0~4.0	C# 4.0	Windows 7 정식 지원
Visual Studio 2012 (11.0)	.NET Framework 2.0~4.5	C# 5.0	C++ 지원 강화
Visual Studio 2013 (12.0)	.NET Framework 2.0~4.5.1	C# 5.0	Windows 8.1 정식 지원
Visual Studio 2015 (14.0)	.NET Framework 2.0~4.6	C# 6.0	Windows 10 정식 지원
Visual Studio 2017 (15.0)	.NET Framework 3.5~4.7	C# 7.0	.NET Framework 3.5 미만은 지원 안 함
Visual Studio 2019 (16.0)	.NET Framework 3.5~4.8	C# 8.0	.NET 5 지원
Visual Studio 2022 (17.0)	.NET Framework 4.6.2~4.8	C# 9.0 C# 10.0 C# 11.0	64bit 전용 .NET 5 / 6 지원

# 1. C#의 탄생 배경과 특징

- C#의 특징
    - GUI와 게임 및 IoT 관련 프로그램 개발
    - C#의 확장자는 .cs
- 완벽을 추구하는 객체지향언어
  - 개발자가 사용하기 편리한 인터페이스 환경 제공
  - 자동 가비지 콜렉션으로 메모리에 대한 사용자 부담 감소
  - 타입과 문법의 엄격한 제한
  - 다양한 문법의 확장 용이
  - 닷넷의 모든 장점 보유 등

## 02. C# 소개

### III. C#의 특징

- C# 이전에 가장 많이 쓰인 객체지향 언어는 C++ 또는 자바였다.
- C#은 C++의 객체지향적인 장점을 그대로 가져오되 C++의 문제점을 보완해 만들어졌다.

#### ❶ 프로그래밍하기 위한 간결한 문법을 제공한다.

- 문법이 쉽기 때문에 프로그래밍 초보자가 배우기 적절한 언어다.
- 쉬운 문법만으로도 많은 기능을 제공하여 어려운 문법 때문에 프로그램이 복잡해지는 것을 최소화한다.
- C / C++의 강력하지만 문제점으로 꼽히는 포인터를 과감히 없앴다.

## 02. C# 소개

### III. C#의 특징

#### ② 엄격한 문법으로 프로그래머의 실수를 방지한다

- 형식 및 문법 체크가 기존보다 더욱 엄격하다.
- 개발자들이 학습에 불편함을 느끼기도 하지만, 대형 프로그램을 제작하는 경우에는 버그가 생길 확률이 적어져 더 유용하다고 평가를 받는다.

#### ③ 완전한 객체지향 언어이다

- 기존 C / C++에서 사용되던 구조적인 프로그래밍 기법을 그대로 지원한다.
- 현재 실무에서 가장 일반적으로 사용되는 객체지향 프로그래밍 기법을 100% 지원한다.

## 02. C# 소개

### III. C#의 특징

#### ④ 멀티 스레드 프로그래밍을 지원한다

- 하나의 프로세스(Process) 안에서 여러 개의 스레드(Thread)가 동시에 작동되도록 프로그래밍이 가능함으로 병렬 처리가 가능해져, 복잡한 대용량 작업을 빠른 시간에 처리할 수 있다.
- 자체 라이브러리에서 멀티 스레드 프로그래밍(Multi Thread Programming)을 지원하기 때문에 프로그래머는 비교적 쉽게 병렬 프로그래밍을 할 수 있다.

#### ⑤ 다양한 응용 프로그램을 작성할 수 있다

- 한글, 알집, 엑셀과 같이 일반 컴퓨터에서 작동하는 데스크톱 응용 프로그램, 웹 서버에서 작동하는 ASP.Net 프로그래밍, 데이터베이스 응용 프로그램, 유니티 환경의 게임 프로그래밍 등을 작성할 수 있다.



## 02. C# 소개

### III. C#의 특징

#### ⑥ CLR(Common Language Runtime, 공용 언어 런타임)환경에서 개발된다

- CLR과 통합 클래스 라이브러리 집합을 포함하는 닷넷 프레임워크 환경에서 실행
- CLR을 사용하면 코드를 실행하거나 개발하는 일이 쉽고 서로 다른 언어로 개발된 코드를 통합하거나 통신이 가능하도록 프로그램을 작성할 수 있다.
- CLR 사용의 장점
  - 실행 코드의 성능 향상
  - 다중 스레드 프로그래밍 지원
  - 타 언어로 개발된 구성요소 쉽게 호출해 사용
  - 예외 처리의 지원
  - 클래스 라이브러리에서 제공하는 기능 사용
  - 가비지 컬렉션
  - 객체지향 프로그래밍 기능
  - 안정성의 향상

## 02. C# 소개

### IV. 닷넷 프레임워크

- 닷넷 프레임워크 : 마이크로소프트에서 제공하는 Windows 프로그램 개발 및 실행 환경에 대한 이름
- 많은 Windows 응용 프로그램이 닷넷 프레임워크 환경에서 작동
- Windows 10, Windows Server 2016 이후 버전에서는 Windows 업데이트를 수행하면 자동으로 최신 닷넷 프레임워크가 설치

## 02. C# 소개

### IV. 닷넷 프레임워크

#### 여기서 잠깐

#### 닷넷 프레임워크, 닷넷 코어, 닷넷

- 닷넷 프레임워크(.NET Framework)
  - 2002년에 마이크로소프트에서 발표, Windows 전용 응용 프로그램 작성에 사용
  - 2022년 닷넷 프레임워크 4.8 버전을 마지막으로 새 버전 업그레이드 제공 안할 계획
  - 앞으로 나올 Windows 버전에는 닷넷 프레임워크가 기본으로 제공될 것이라 닷넷 프레임워크로 개발된 환경이 향후에도 계속 작동될 것으로 보임
- 닷넷 코어(.NET Core): 2014년 크로스 플랫폼(윈도우, 리눅스, 맥 등을 통칭)에서 작동하는 응용 프로그램을 작성하기 위한 용도로 출시
- 닷넷(.NET): 닷넷 프레임워크와 닷넷 코어를 통합한 이름
  - 2020년에 닷넷 5가, 2021년에 닷넷 6가 발표.
  - 닷넷 6부터는 모바일 환경까지 지원되는 진정한 크로스 플랫폼 환경을 지원

03

C# 프로그램 작성 방법

## 03. C# 프로그램 작성 방법

### I. C# 프로그램 작성 흐름 요약

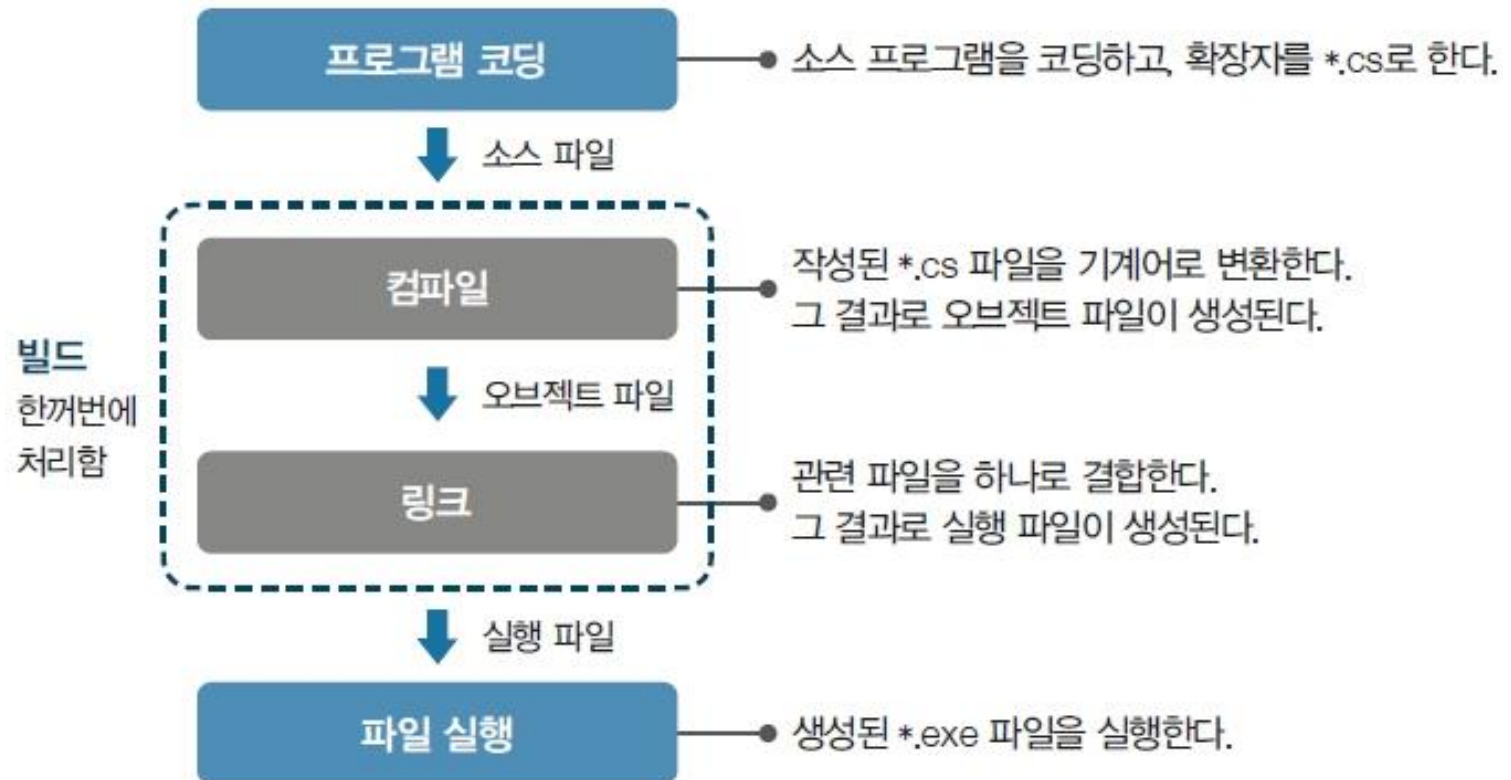


그림 1-6 C# 프로그램의 작성과 실행 순서

## 03. C# 프로그램 작성 방법

### I. C# 프로그램 작성 흐름 요약

#### 여기서 잠깐

#### 닷넷 프레임워크 환경에서의 컴파일과 실행

- C#은 닷넷 프레임워크 환경에서 작동하므로 그림의 마지막에 보이는 \*.exe 파일을 어셈블리 (Assembly)라 부른다.
- 어셈블리는 중간 단계의 언어로 IL(Intermediate Language)로도 부른다(자바의 바이트 코드와 비슷한 개념이다).
- \*.exe 파일을 실행하기 위해서는 닷넷 프레임워크 환경의 핵심인 CLR이 작동해서 내부적으로 다시 컴파일 과정 등을 거쳐 원시 코드(Native Code)로 변환해야 한다.
- 하지만 모두 내부적으로 처리되기 때문에 실제로는 그냥 \*.exe 파일이 바로 실행되는 것처럼 보인다.

## 03. C# 프로그램 작성 방법

### II. 프로그램 코딩

- 코딩(Coding) : 머릿속에 그려진 결과물을 실제로 만들기 위해 C# 컴파일러가 알아들을 수 있는 형식으로 문서를 작성한다는 의미

ex) 예를 들어, 100에서 50을 뺀 값을 계산하라는 프로그램을 작성한다고 했을 때 다음과 같이 코딩하면 컴퓨터는 알아듣지 못할 것이다.

어이 컴퓨터~

100에서 50을 뺀 결과가 뭐지?

모니터에 한번 출력해봐~~

## 03. C# 프로그램 작성 방법

### II. 프로그램 코딩

#### [소스 1-1] C# 프로그램 맛보기

```
1 static void Main(string[] args)
2 {
3     int result;
4     result = 100 - 50;
5     Console.WriteLine(result);
6 }
```

- C# 소스 코드 중 일부만 표현한 예
- 100에서 50을 뺀 결과를 모니터에 출력해주는 전형적인 C# 프로그램 형태



## 03. C# 프로그램 작성 방법

### II. 프로그램 코딩

#### SELF STUDY 1-1

[소스 1-1]을 수정하여 123과 456을 곱한 결과를 출력하는 C# 프로그램을 만들어보자.

## 03. C# 프로그램 작성 방법

### III. 컴파일과 링크

- 컴퓨터가 유일하게 이해하는 언어는 기계어로 작성된 코드이다.
- [소스 1-1]을 컴퓨터가 이해할 수 있도록 [그림 1-7]과 같은 '컴파일(Compile)' 과정을 거쳐야 한다.

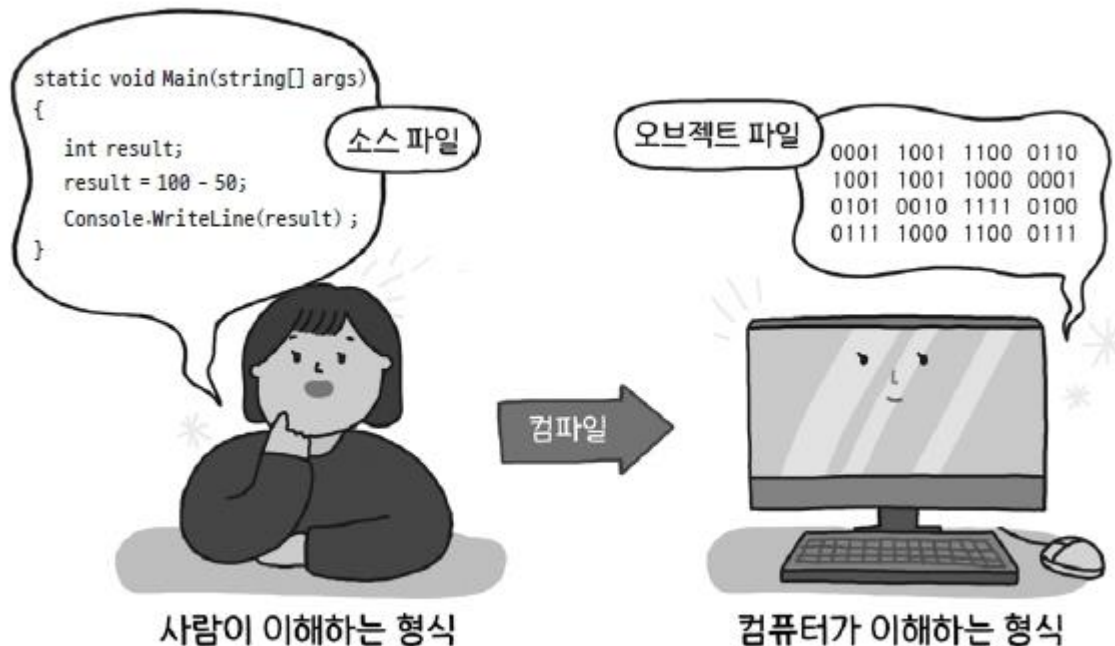


그림 1-7 컴파일의 개념도

## 03. C# 프로그램 작성 방법

### III. 컴파일과 링크

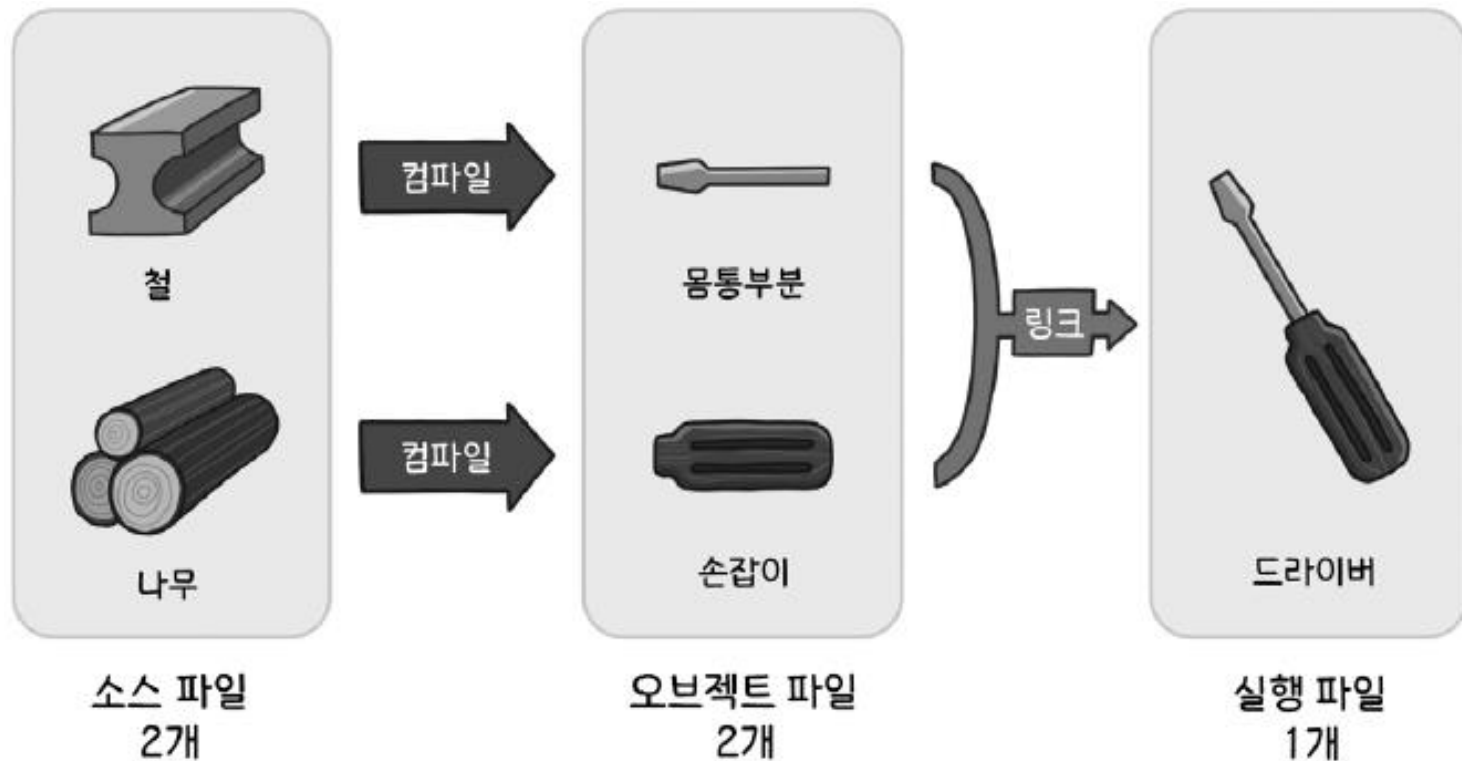


그림 1-8 컴파일 / 링크의 개념도

## 03. C# 프로그램 작성 방법

### III. 컴파일과 링크

- 컴파일 후 '링크(Link)'를 거쳐야 컴퓨터가 실행할 수 있는 '실행 파일(\*.exe)'이 생성
- 실행어리와 나무(소스 파일, \*.cs)를 가지고 작업(컴파일)을 거쳐 드라이버의 몸통과 손잡이(오브젝트 파일)를 만들 수 있음
- 몸통과 손잡이를 결합(링크)해야 비로소 사용할 수 있는(실행 가능한) 드라이버(실행 파일, \*.exe)가 완성되는 것

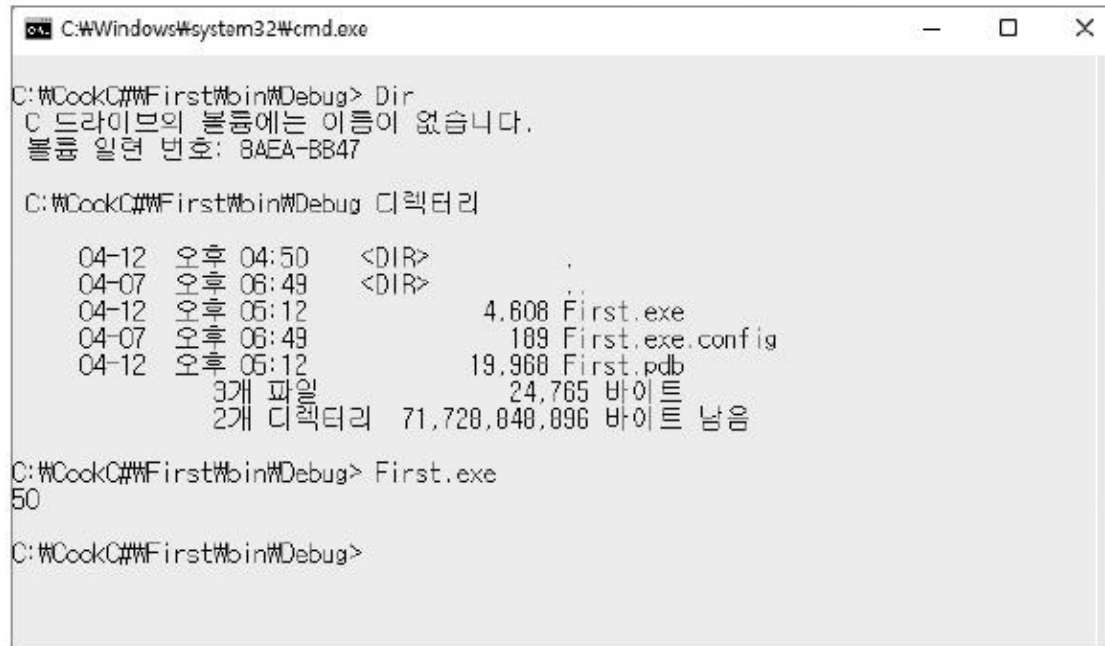
#### 여기서 잠깐

**빌드 = 컴파일 + 링크**

- 컴파일과 링크는 소스 파일이 하나뿐이더라도 반드시 수행해야 한다.
- 대부분의 컴파일러는 컴파일과 링크 과정을 별도로 처리하지 않기 때문에 컴파일과 링크를 합쳐서 '컴파일' 또는 '빌드'라고도 한다.
- 필요에 따라 컴파일과 링크를 별도로 수행할 수도 있다.

## 03. C# 프로그램 작성 방법

### IV. 프로그램 실행



```
C:\Windows\system32\cmd.exe
C:\CookC#\First\bin\Debug> Dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 8AEA-BB47

C:\CookC#\First\bin\Debug 디렉터리

   04-12 오후 04:50    <DIR>          .
   04-07 오후 06:49    <DIR>          ..
   04-12 오후 05:12             4,608 First.exe
   04-07 오후 06:49             189 First.exe.config
   04-12 오후 05:12            19,968 First.pdb
               3개 파일                24,765 바이트
               2개 디렉터리  71,728,848,896 바이트 남음

C:\CookC#\First\bin\Debug> First.exe
50

C:\CookC#\First\bin\Debug>
```

그림 1-9 컴파일 및 링크가 완료된 First.exe 파일

## 03. C# 프로그램 작성 방법

### IV. 프로그램 실행

#### 여기서 잠깐

#### 컴파일러 언어와 스크립트 언어

- 컴파일러(Compiler) 언어: 소스 코드를 실행 가능한 기계어로 일괄 번역한 후 번역이 완료된 파일(\*.exe, \*.class 등의 파일)이 실행되는 언어
  - 대표적인 컴파일러 언어: C, C++, 자바, C# 등
- 스크립트 언어(or 인터프리터 언어): 소스 코드를 한 줄씩 읽어가며 실행되는 언어
  - 한 줄 씩 처리하는 작업을 하는 프로그램을 인터프리터(Interpreter)라고 함.
  - 별도의 실행 파일이 생성되지 않음.
  - 대표적인 스크립트 언어: 파이썬, 자바스크립트(JavaScript), 펄(Perl)
  - 빠른 시간 안에 배울 수 있음.

04

C# 개발환경 구축

## 04. C# 개발환경 구축

### I. Visual Studio 버전 소개

- 마이크로소프트가 제공하는 Visual Studio 안에는 Visual C#이라고 불리는 C# 개발환경이 포함되어 있다.
- Visual Studio는 C++, C#, VB, F# 등의 다양한 언어를 포함하고 있는 개발환경이다.
- Visual Studio는 유료와 무료 버전으로 나뉜다



## 04. C# 개발환경 구축

### I. Visual Studio 버전 소개

표 1-2 Windows 버전에 따른 Visual Studio 버전

운영체제	설치 가능한 유료 버전	설치 가능한 무료 버전
Windows 7(sp1) Windows 8 Windows 8.1	Visual Studio 2005, 2008, 2010, 2012, 2013, 2015, 2017, 2019	Visual Studio Express 2010 Visual Studio Express 2012, 2013, 2015 for Windows Desktop Visual Studio Community 2013, 2015, 2017, 2019
Windows 10	Visual Studio 2012, 2013, 2015, 2017, 2019, 2022	Visual Studio Express 2012, 2013, 2015 for Windows Desktop Visual Studio Community 2013, 2015, 2017, 2019, 2022
Windows 11	Visual Studio 2013, 2015, 2017, 2019, 2022	Visual Studio Community 2013, 2015, 2017, 2019, 2022

## 04. C# 개발환경 구축

### II. Visual Studio Community 설치

#### [실습 1-1] Visual Studio Community 설치하기

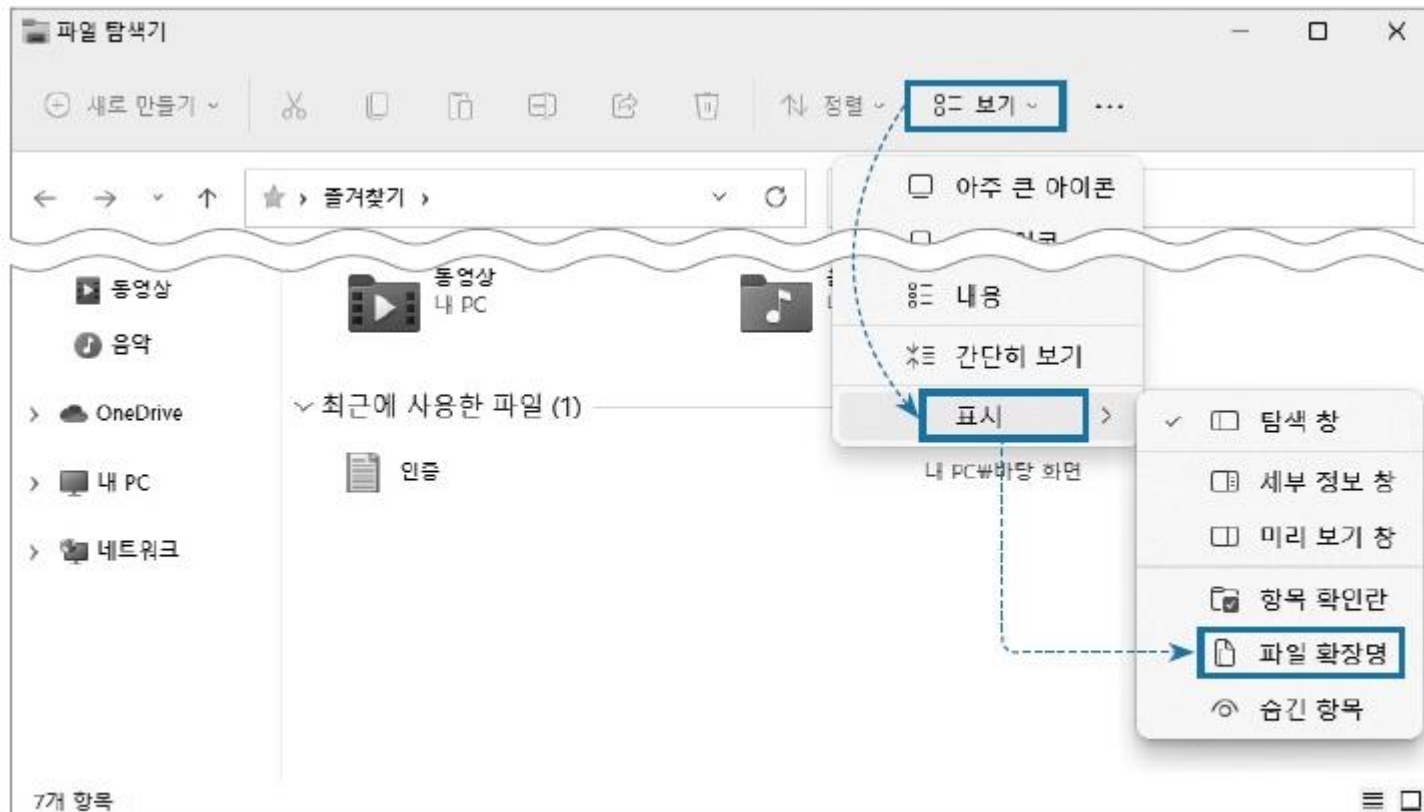


그림 1-10 Windows 11에서 확장명 보이기

## 04. C# 개발환경 구축

### II. Visual Studio Community 설치

#### [실습 1-1] Visual Studio Community 설치하기

- ① Windows 10은 파일 탐색기를 실행해 메뉴의 [보기] - [파일 확장명]을 체크하면 파일의 확장명을 볼 수 있다. Windows 11은 파일 탐색기의 메뉴에서 [보기] - [표시] - [파일 확장명]을 선택하면 된다.



그림 1-11 Visual Studio Community 다운로드

## 04. C# 개발환경 구축

### II. Visual Studio Community 설치

#### [실습 1-1] Visual Studio Community 설치하기

- ② <https://visualstudio.microsoft.com/>에서 Visual Studio Community 설치 파일을 다운로드한다.

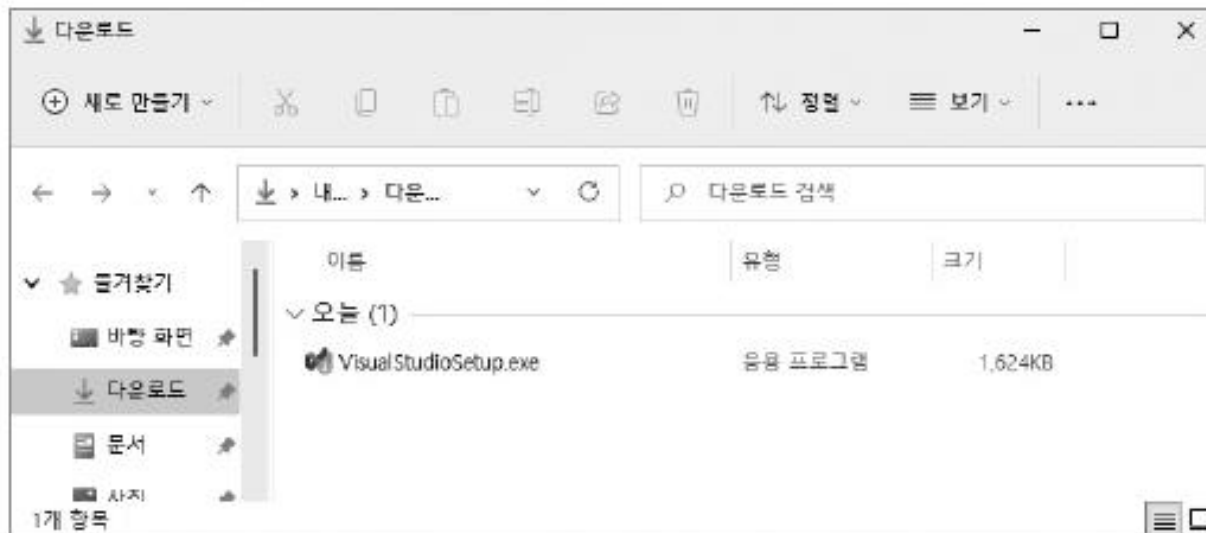


그림 1-12 설치 실행 파일

## 04. C# 개발환경 구축

### II. Visual Studio Community 설치

#### [실습 1-1] Visual Studio Community 설치하기

- ③ [그림 1-13]과 같은 [사용자 계정 컨트롤] 메시지 창이 나오면 <예>를 클릭한다.

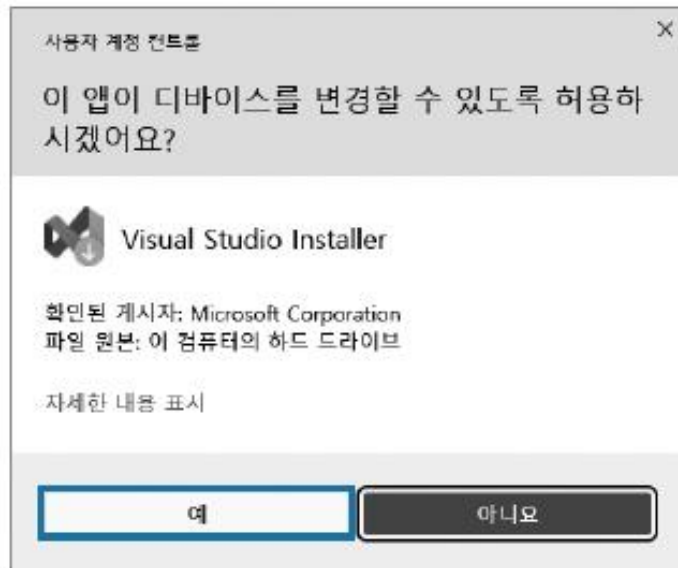


그림 1-13 Visual Studio Installer 실행 시 나오는 [사용자 계정 컨트롤] 창

## 04. C# 개발환경 구축

### II. Visual Studio Community 설치

#### [실습 1-1] Visual Studio Community 설치하기

- ④ 설치 초기화면에서 <계속>을 클릭한다.

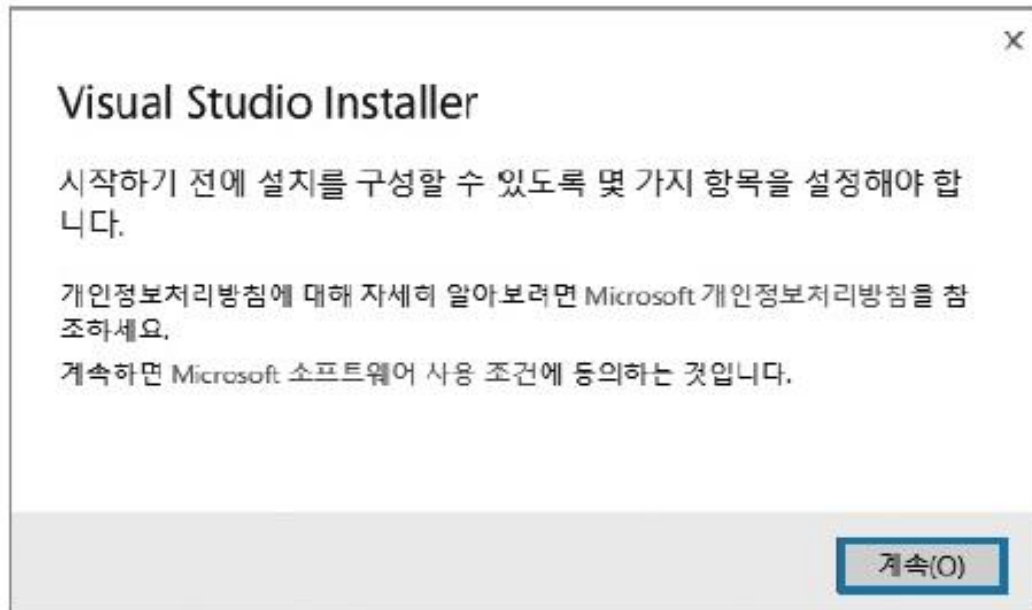


그림 1-14 설치 초기 화면

## 04. C# 개발환경 구축

### II. Visual Studio Community 설치

#### [실습 1-1] Visual Studio Community 설치하기

- ⑤ 필요한 파일을 다운로드 및 설치한다.

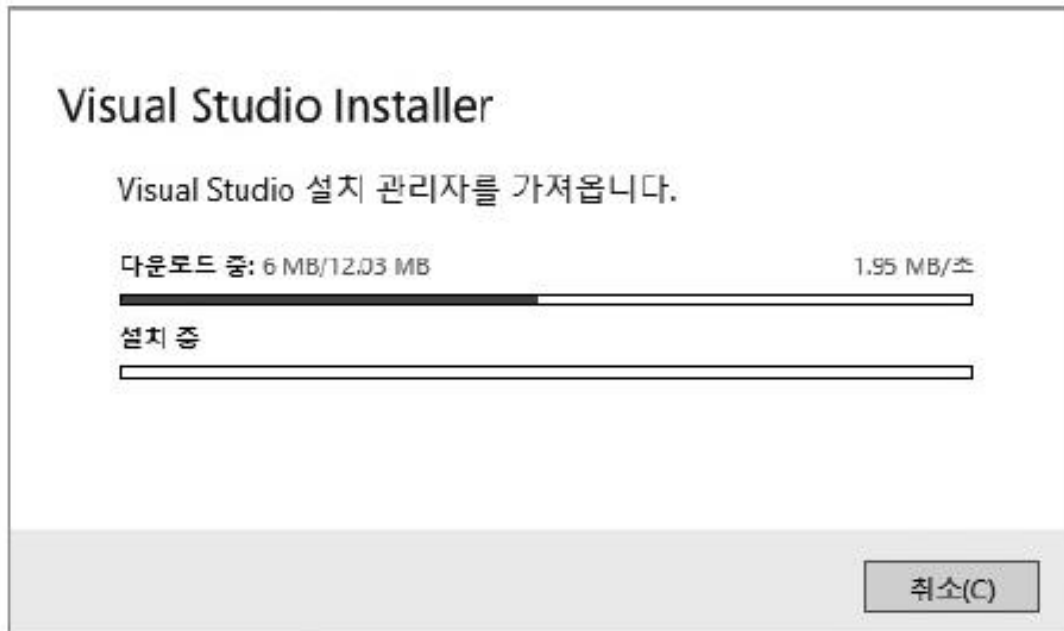


그림 1-15 초기 필요 파일 다운로드 및 설치

## 04. C# 개발환경 구축

### II. Visual Studio Community 설치

#### [실습 1-1] Visual Studio Community 설치하기

- ⑥ [워크로드] 창이 뜨면 [데스크톱 및 모바일] 아래의 [.NET 데스크톱 개발]만 체크하고 [설치]를 클릭한다.

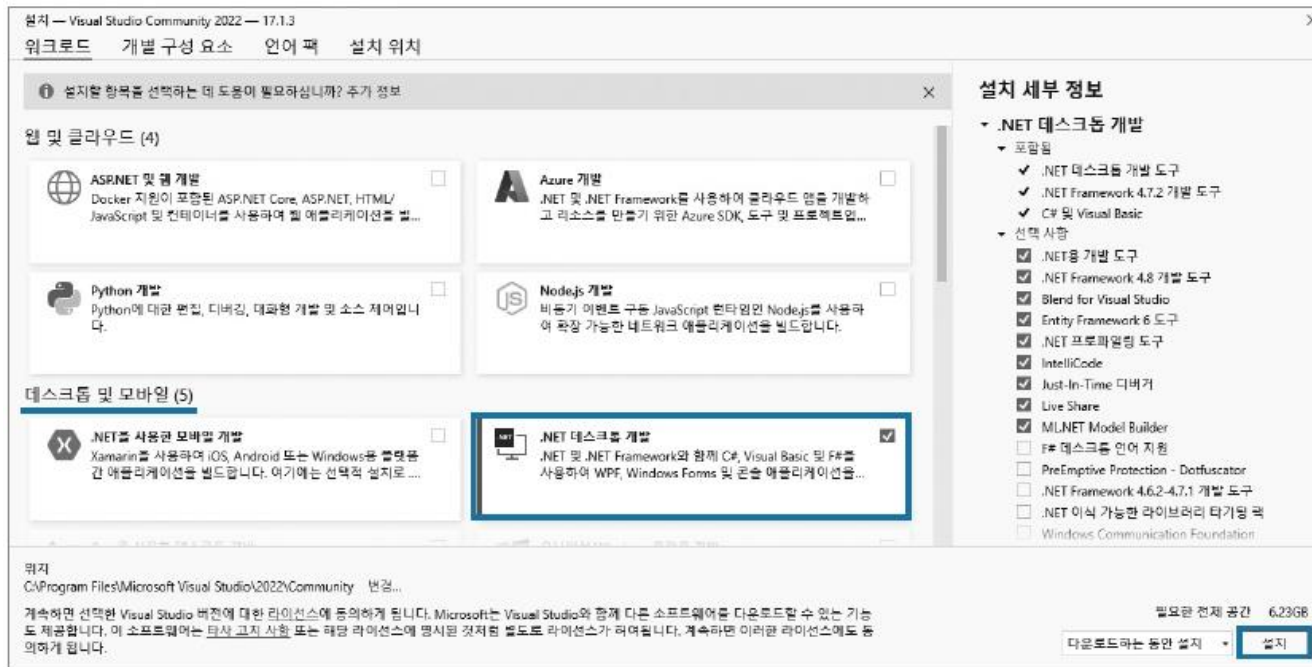


그림 1-16 설치할 기능 선택



## 04. C# 개발환경 구축

### II. Visual Studio Community 설치

#### [실습 1-1] Visual Studio Community 설치하기

- ⑦ 한동안 다운로드 및 설치가 진행된다. 설치 시간은 컴퓨터의 성능 및 네트워크 상황에 따라 달라진다.



그림 1-17 설치 진행

- **Step 04** | 도움말 뷰어 선택 : [개별 구성 요소] 탭에서 코드 도구 항목의 '☑️ 도움말 뷰어' 를 선택합니다.

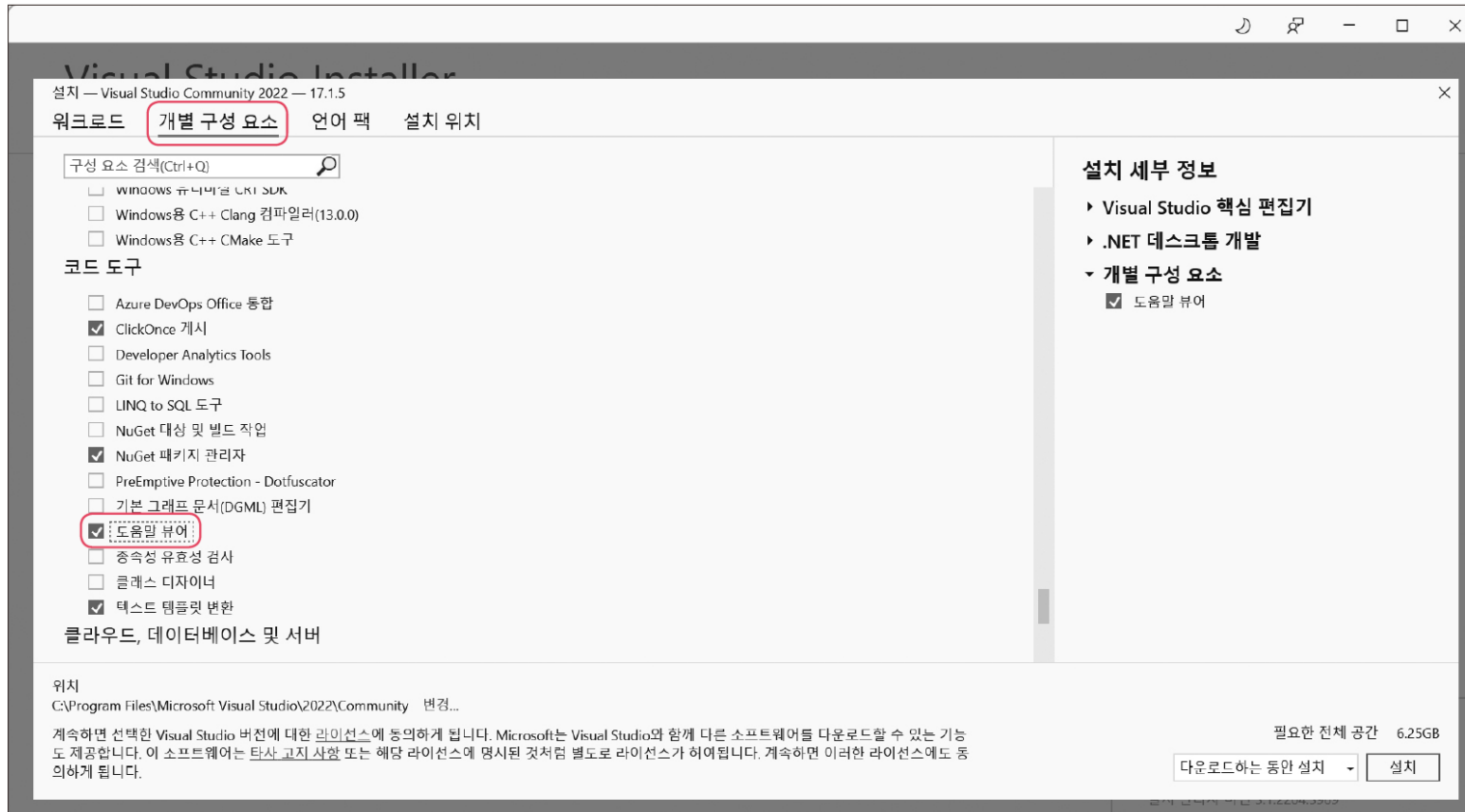


그림 1-5 도움말 뷰어 선택

- **Step 05** | 언어 팩 선택 : [언어 팩] 탭에서 '☑ 영어' 를 선택하고 <설치>를 누릅니다.

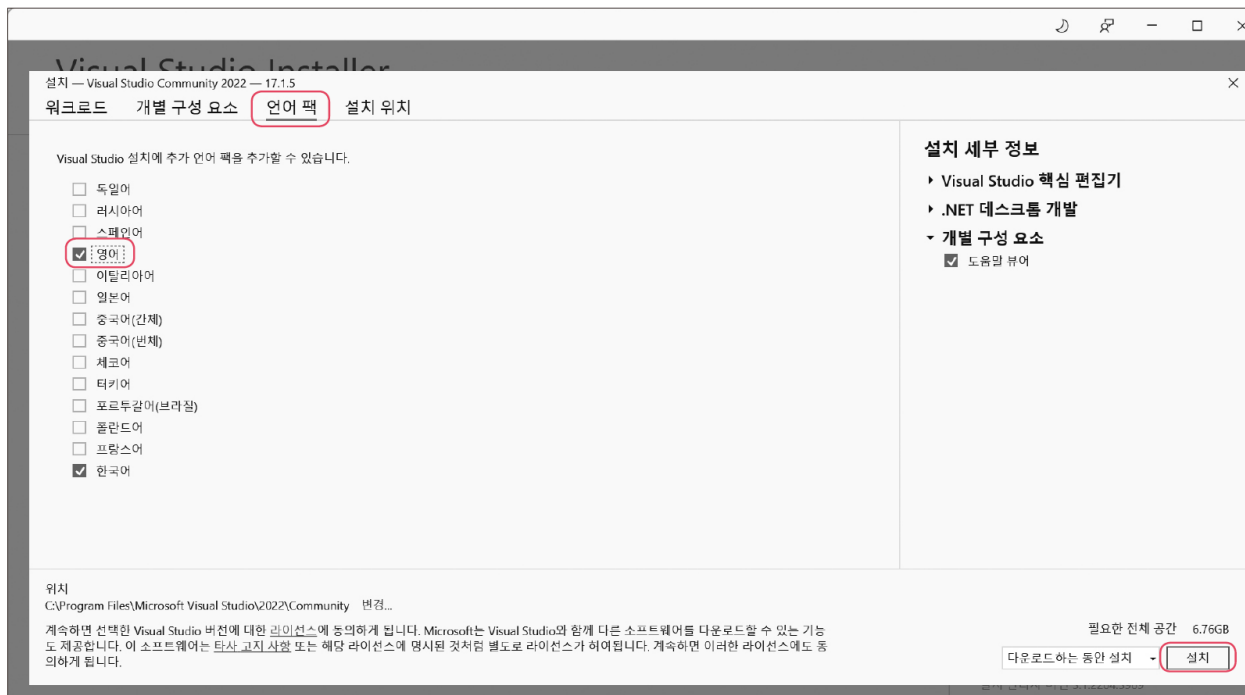


그림 1-6 언어 팩 선택

## 04. C# 개발환경 구축

### II. Visual Studio Community 설치

#### [실습 1-1] Visual Studio Community 설치하기

- ⑧ 설치가 완료되면 자동으로 Visual Studio가 실행된다. 설치가 완료된 [Visual Studio Installer] 창은 오른쪽 상단의 <X>를 클릭해서 닫는다.



그림 1-18 설치 완료

## 04. C# 개발환경 구축

### II. Visual Studio Community 설치

#### [실습 1-1] Visual Studio Community 설치하기

- ⑨ Visual Studio 로고가 잠깐 뜬 후, 로그인 창이 나오면 을 클릭한다. 그리고 마음에 드는 화면 환경을 선택해 <Visual Studio 시작>을 클릭한다.

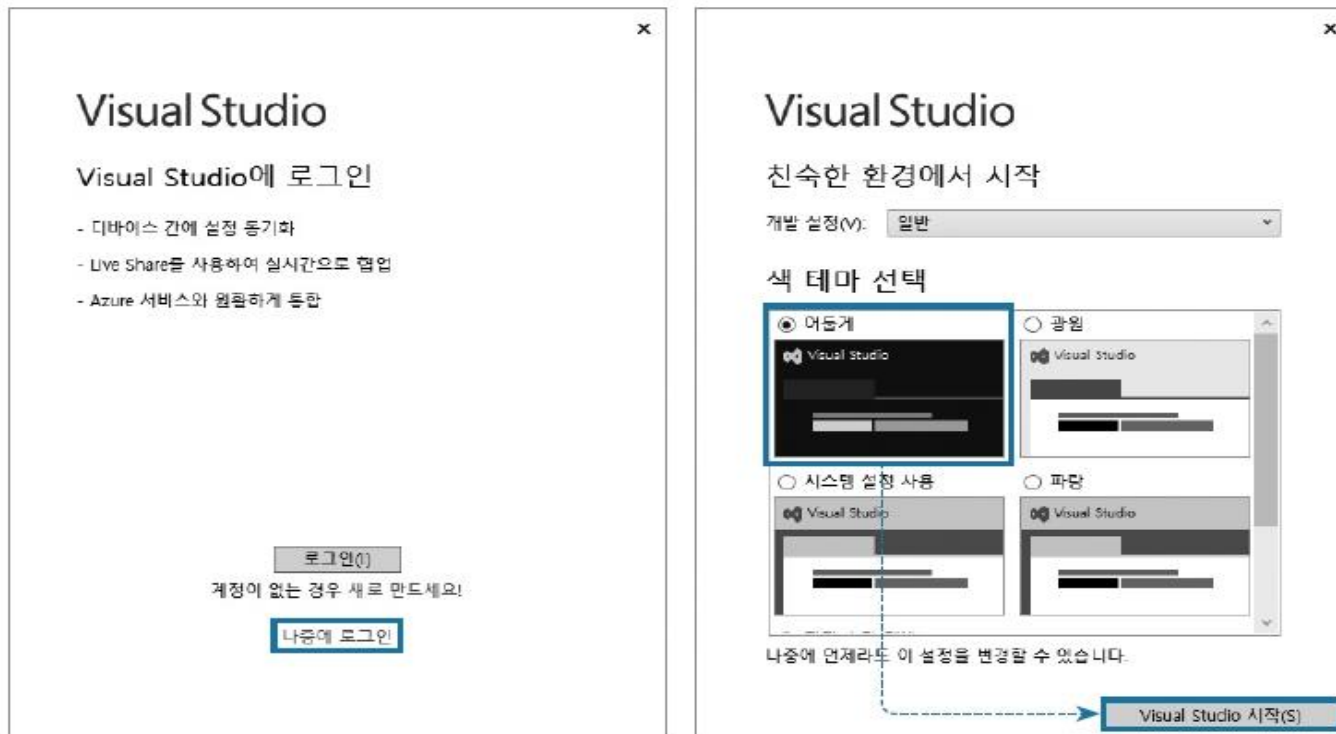


그림 1-19 Visual Studio 로그인 및 화면 환경 테마 선택

## 04. C# 개발환경 구축

### II. Visual Studio Community 설치

#### 여기서 잠깐

#### 마이크로소프트 계정으로 로그인

- Visual Studio Community는 마이크로소프트 계정으로 로그인해야 기간에 제한 없이 무료로 사용할 수 있다.
- 마이크로소프트 계정은 <https://account.microsoft.com/account>에서 무료로 만들 수 있다.
- 로그인을 하지 않으면 30일만 무료로 사용할 수 있으니 만료 전에만 로그인 하면 된다.
- 나중에 로그인이나 계정 등록을 하려면 Visual Studio의 [도움말] - [제품 등록] 메뉴를 선택하면 된다.

## 04. C# 개발환경 구축

### II. Visual Studio Community 설치

#### [실습 1-1] Visual Studio Community 설치하기

- ⑩ 처음 실행할 때는 잠시 시간이 걸린 후 Visual Studio [시작 페이지]가 나온다. 이상으로 설치를 완료했다. 오른쪽 위 <X>를 눌러 창을 닫는다



그림 1-20 Visual Studio 시작 페이지

## 2. 실습 환경 구축

### ■ 닷넷 플랫폼 설치

#### ■ 닷넷 SDK 설치

- SDK는 닷넷 앱과 라이브러리를 빌드하고 게시하는 용도로 사용
- 개발자에게 다른 프로그램에 추가하거나 커스텀 앱을 제작할 수 있는 기능을 제공하는 도구
- SDK를 설치하게 되면 3개(ASP.NET Core, 데스크톱, 닷넷 Core)의 런타임이 모두 포함됨

#### 예제 01-02

#### 닷넷 SDK 설치하기

- **Step 01** | 닷넷 SDK 설치 파일 다운로드 : 사이트에 접속하여 [.NET 6.0] 버전의 설치 파일 'x64 Runtime' 을 다운로드합니다.

▶ <https://dotnet.microsoft.com/en-us/download/visual-studio-sdks>



**.NET/.NET Core**

.NET is a free, cross-platform, open-source developer platform for building many different types of applications.

Version	Status	Visual Studio 2017 SDK ⓘ	Visual Studio 2019 SDK ⓘ	Runtime ⓘ	Release notes
.NET 7.0	Preview ⓘ	N/A	N/A	<a href="#">x64 Runtime</a>   <a href="#">x86 Runtime</a> (v7.0.0-preview.3)	<a href="#">Release notes</a>
.NET 6.0	LTS ⓘ	N/A	N/A	<a href="#">x64 Runtime</a>   <a href="#">x86 Runtime</a> (v6.0.4)	<a href="#">Release notes</a>
.NET 5.0	Current ⓘ	N/A	<a href="#">x64 SDK</a>   <a href="#">x86 SDK</a> (v5.0.407)	<a href="#">x64 Runtime</a>   <a href="#">x86 Runtime</a> (v5.0.16)	<a href="#">Release notes</a>

Feedback

그림 1-11 .NET 6.0 버전 선택

- **Step 02** | SDK 설치 파일 실행 : 다운로드한 'x64 Runtime' 파일을 더블클릭하면 설치 화면이 나타납니다. 화면에서 <설치>를 눌러 SDK 설치를 시작합니다.



그림 1-12 SDK 설치 시작

- **Step 03** | SDK 프로그램 설치 : SDK 프로그램 설치 과정에서 특별히 옵션을 선택하는 과정은 없으므로 잠시 기다립니다.

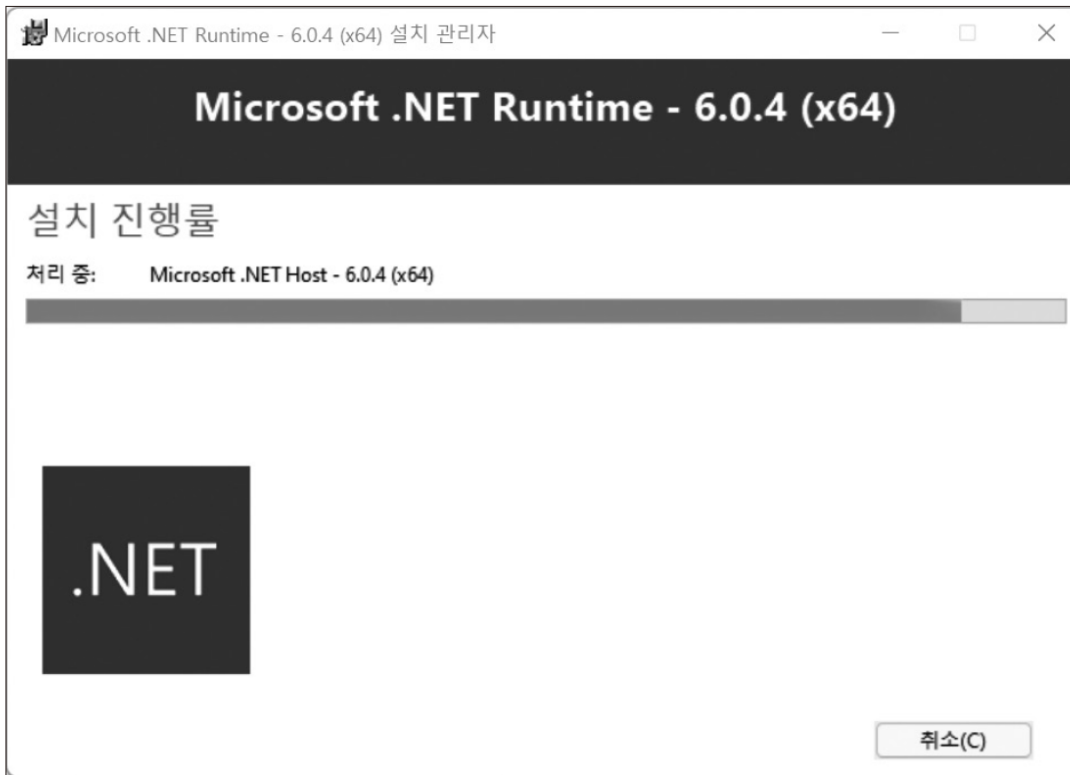


그림 1-13 SDK 프로그램 설치 중

- **Step 04** | SDK 프로그램 설치 완료 : SDK 프로그램 설치가 완료되면 <닫기>를 클릭합니다.



그림 1-14 SDK 설치 완료

05

처음 작성하는 C# 프로그램

## 05. 처음 작성하는 C# 프로그램

### I. C# 프로젝트 생성

#### [실습 1-2] Visual Studio에서 프로젝트 생성하기

- ① C:\CookC# 폴더를 미리 생성한다.

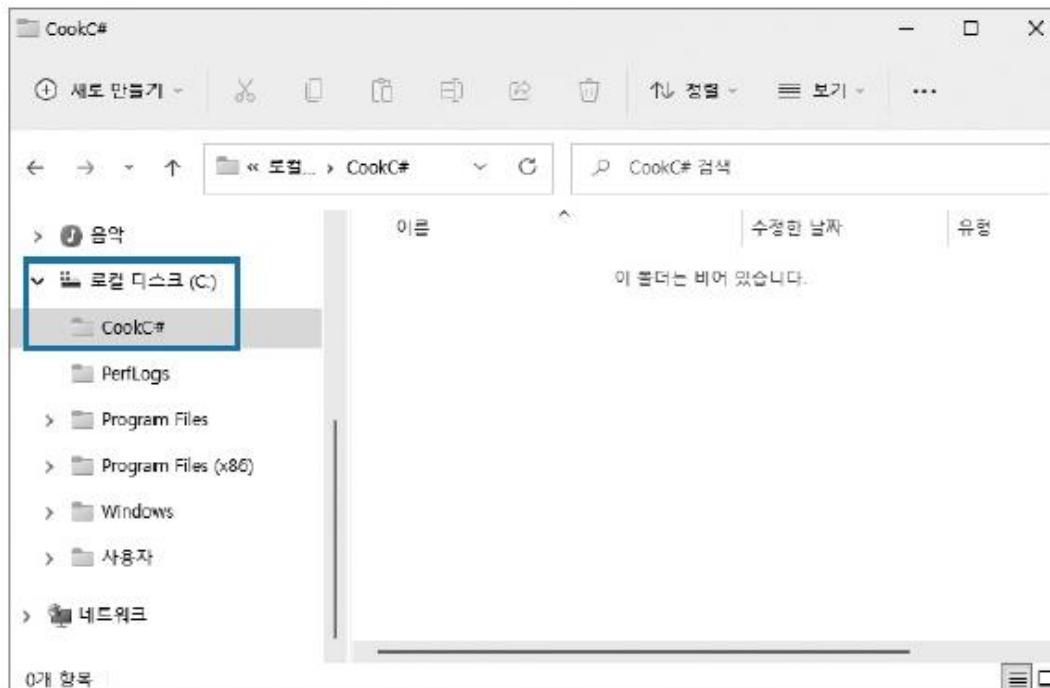


그림 1-21 폴더 생성

## 05. 처음 작성하는 C# 프로그램

### I. C# 프로젝트 생성

#### [실습 1-2] Visual Studio에서 프로젝트 생성하기

- ② Windows의 [시작] → [V] → [Visual Studio 2022]를 클릭해서 다시 실행한다

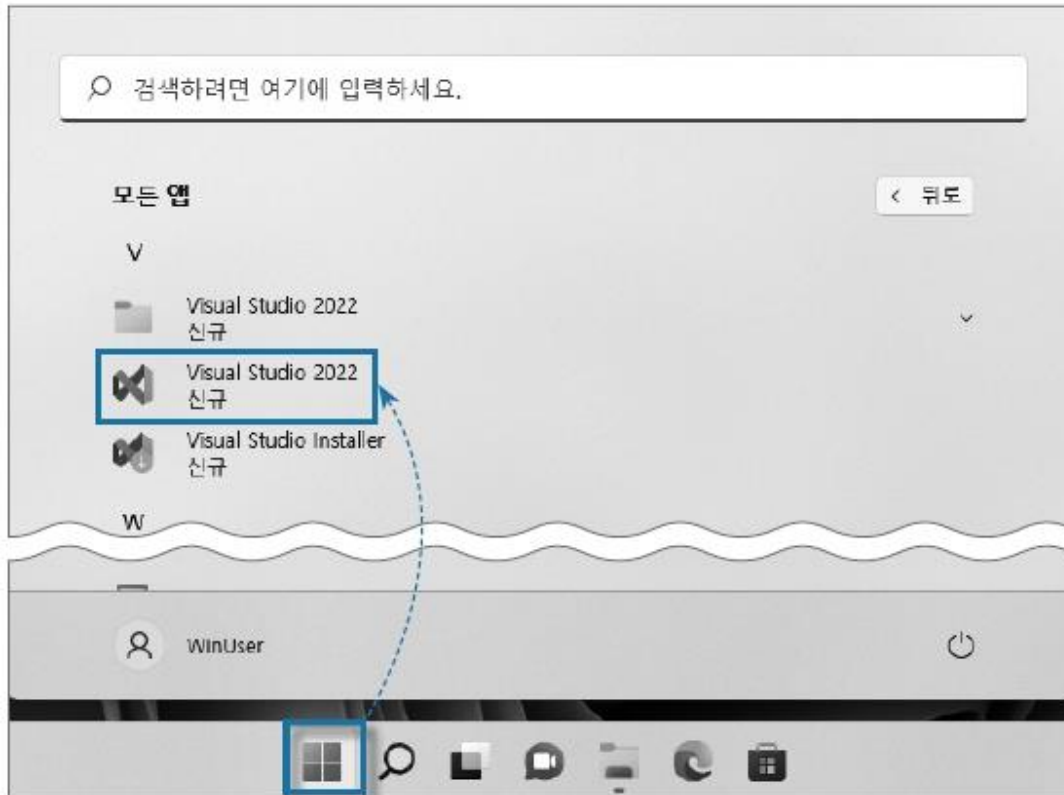


그림 1-22 Visual Studio 실행

## 05. 처음 작성하는 C# 프로그램

### I. C# 프로젝트 생성

#### 여기서 잠깐

#### 프로젝트 개념

- Visual C# 프로그래밍을 하려면 단순히 소스 파일만 필요한 것이 아니라 '프로젝트'라고 하는 것을 먼저 생성해야 한다.
- 프로젝트는 C#뿐 아니라 다른 고급 프로그래밍에서 도 중요한 개념이다.
- 프로젝트라는 개념은 여러 개의 C# 프로그램을 모아 놓은 묶음이라고 보면 된다.

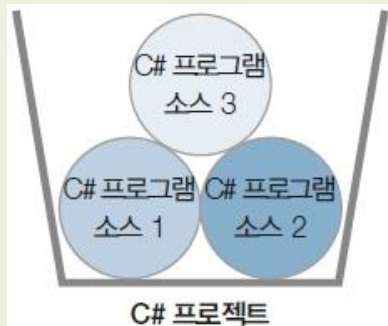


그림 1-23 프로젝트 개념도



## 05. 처음 작성하는 C# 프로그램

### I. C# 프로젝트 생성

[실습 1-2] Visual Studio에서 프로젝트 생성하기

- ③ Visual Studio 로고가 잠깐 보인다.



그림 1-24 Visual Studio 로고

## 05. 처음 작성하는 C# 프로그램

### I. C# 프로젝트 생성

#### [실습 1-2] Visual Studio에서 프로젝트 생성하기

- ④ 실행 창의 오른쪽에서 [새 프로젝트 만들기]를 클릭한다.



그림 1-25 새 프로젝트 만들기 1

## 05. 처음 작성하는 C# 프로그램

### I. C# 프로젝트 생성

#### [실습 1-2] Visual Studio에서 프로젝트 생성하기

- ⑤ [새 프로젝트 만들기] 창에서 언어를 [C#]으로 선택한다. 창 아래쪽으로 스크롤해서 프로젝트 템플릿을 [콘솔 앱(.NET Framework)]로 선택 후, <다음>을 클릭한다.



그림 1-26 새 프로젝트 만들기 2

## 05. 처음 작성하는 C# 프로그램

### I. C# 프로젝트 생성

#### [실습 1-2] Visual Studio에서 프로젝트 생성하기

- ⑥ [새 프로젝트 구성] 창이 뜨면 '프로젝트 이름'에 **First**라고 입력한 후, '위치'에서 <...>을 클릭해 앞에서 **만든 C:\WCookC#** 폴더를 선택 → <솔루션 및 프로젝트를 같은 디렉터리에 배치>를 체크 → '프레임 워크'는 4.8 버전을 선택하고 <만들기>를 클릭



그림 1-27 새 프로젝트 만들기 3

## 05. 처음 작성하는 C# 프로그램

### I. C# 프로젝트 생성

#### [실습 1-2] Visual Studio에서 프로젝트 생성하기

⑦ 최종적으로 프로젝트가 완성된다.

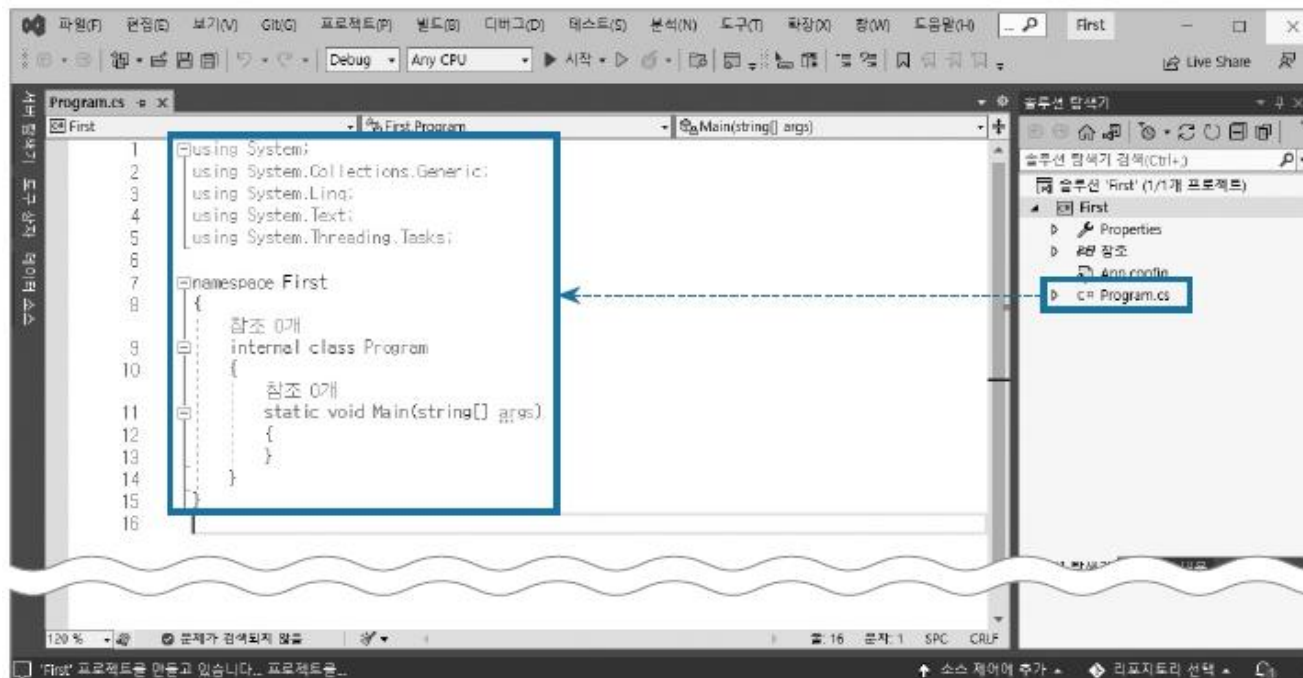


그림 1-28 생성된 프로젝트 화면

## 05. 처음 작성하는 C# 프로그램

### II. C# 프로젝트 코딩

#### [실습 1-3] C# 소스 코드 작성하기

- ① Program.cs 파일의 소스 코드를 변경해보자.

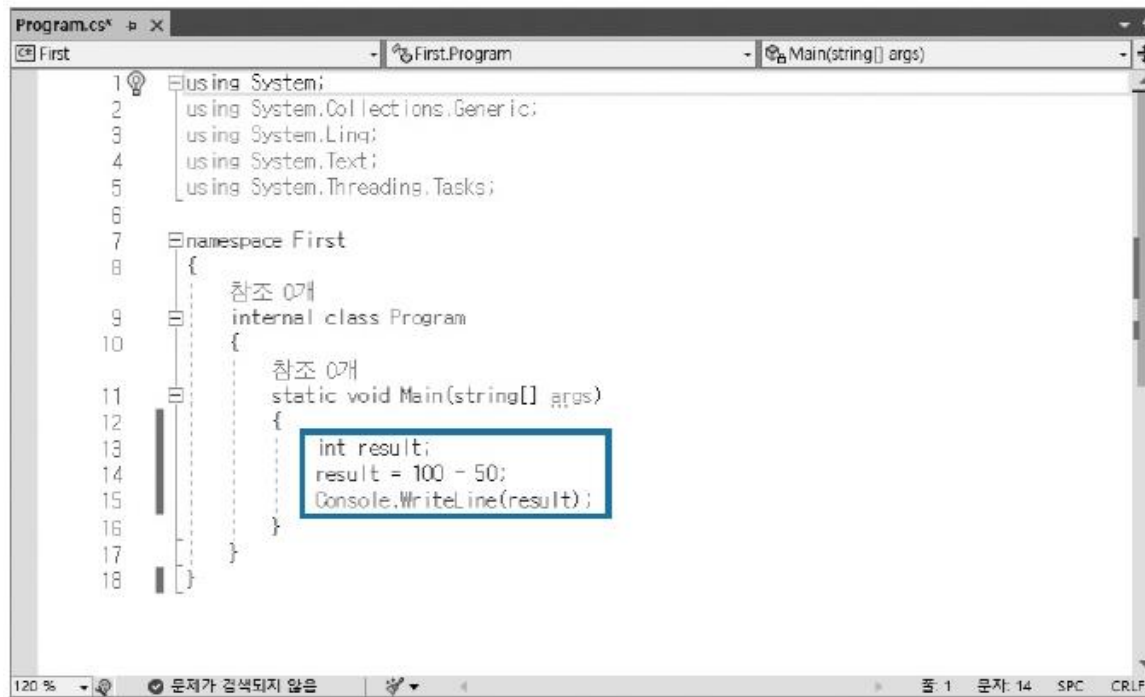


그림 1-29 C# 소스 코드 입력하기

# 05. 처음 작성하는 C# 프로그램

## II. C# 프로젝트 코딩

### [소스 1-2] 처음으로 만드는 C# 프로그램

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace First
8  {
9      internal class Program
10     {
11         static void Main(string[] args)
12         {
13             int result;
14             result = 100 - 50;
15             Console.WriteLine(result);
16         }
17     }
18 }
```

## 05. 처음 작성하는 C# 프로그램

### II. C# 프로젝트 코딩

#### [실습 1-3] C# 소스 코드 작성하기

- ② 입력을 완료했으면 [파일] - [Program.cs 저장] 메뉴를 선택하거나 저장 아이콘을 클릭한다.
  - 1~5행의 **using**은 네임스페이스의 클래스를 사용하겠다는 의미이므로 전체 키워드를 쓰지 않아도 된다.
  - 7~18행에서는 네임스페이스를 선언한다. 기본적으로 프로젝트의 이름과 동일하다.
  - 9~17행은 클래스를 선언한다. 파일명(Program.cs)과 이름이 동일하다.
  - 11~16행은 실제로 작동하는 메인 코드 부분이다.
  - 13행은 정수형 변수 `result`를 선언했다.



## 05. 처음 작성하는 C# 프로그램

### II. C# 프로젝트 코딩

#### [실습 1-3] C# 소스 코드 작성하기

- 14행에서는 100에서 50을 뺀 결과를 result 변수에 대입했다.
- 15행의 `Console.WriteLine( )`은 괄호 안의 값을 모니터에 출력하라는 기능을 제공하는 메서드이다.
- 13~15행을 제외하고는 모두 자동 완성된 부분으로 지금은 실제로 작성한 13~15행만 자세히 보면 된다.

## 05. 처음 작성하는 C# 프로그램

### III. 빌드(컴파일 및 링크)

- **빌드(Build)**: 컴파일과 링크는 한꺼번에 수행하는 과정

#### [실습 1-4] 소스 코드 빌드하기

- ① 메뉴의 [빌드]-[솔루션 빌드]를 선택한다. 단축키 Ctrl + Shift + B 또는 F6 키를 누른다.

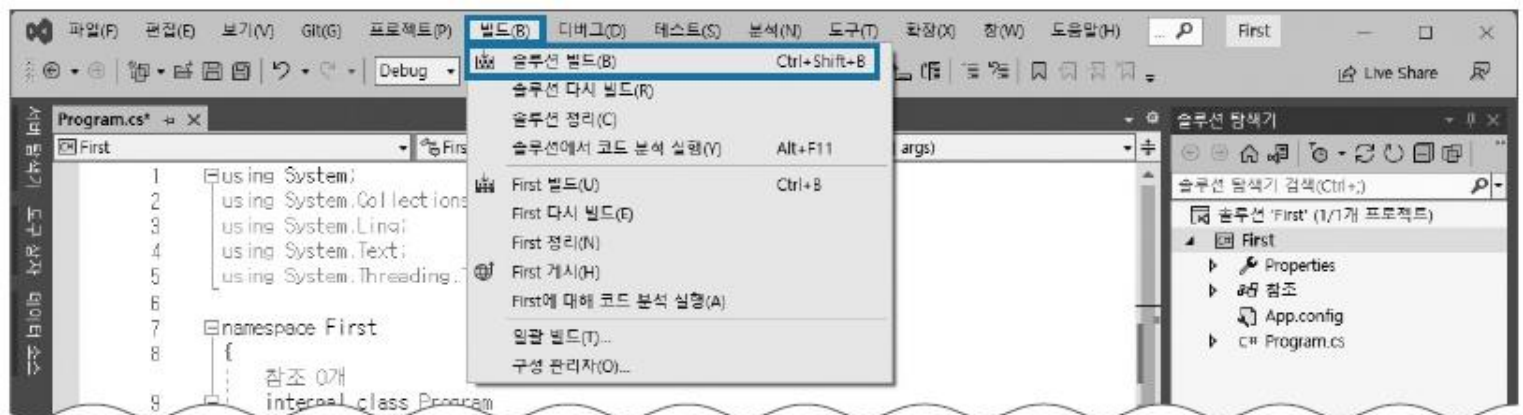


그림 1-30 빌드 실행

## 05. 처음 작성하는 C# 프로그램

### III. 빌드(컴파일 및 링크)

#### [실습 1-4] 소스 코드 빌드하기

- ② 소스에 아무런 오류가 없다면 Visual Studio 왼쪽 아래에 **빌드에 성공했습니다** 메시지가 나온다.

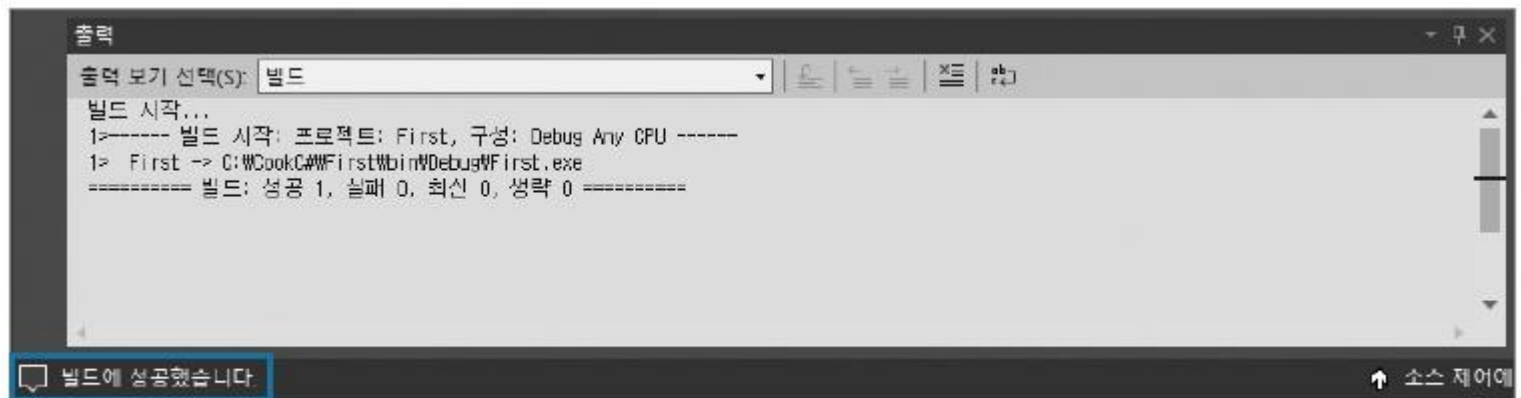


그림 1-31 빌드 결과 – 이상 없음

## 05. 처음 작성하는 C# 프로그램

### III. 빌드(컴파일 및 링크)

#### [실습 1-4] 소스 코드 빌드하기

- ③ [소스 1-2]의 13행에서 제일 뒤에 붙은 세미콜론(;)을 지우고, 메뉴의 [빌드] - [솔루션 빌드]를 선택해서 다시 빌드하면 오류 목록 화면이 뜰 텐데 대부분의 경우 코드가 틀렸을 때 나타난다.

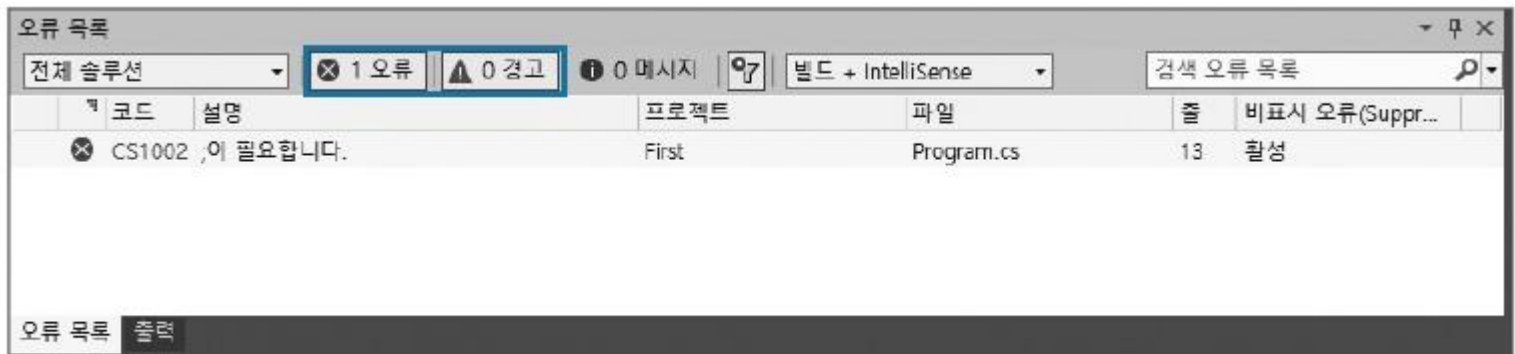


그림 1-32 빌드 결과 - 오류 발생

## 05. 처음 작성하는 C# 프로그램

### III. 빌드(컴파일 및 링크)

#### [실습 1-4] 소스 코드 빌드하기

- ④ 소스 코드에서 틀린 곳을 찾아보자. Visual C#은 틀린 부분을 직접 알려준다. 오류 목록에서 오류가 난 부분을 마우스로 더블 클릭하면 해당 위치로 바로 이동할 수 있다. 오류 메시지 설명에서 **;이 필요합니다**라고 알려주고 있다.

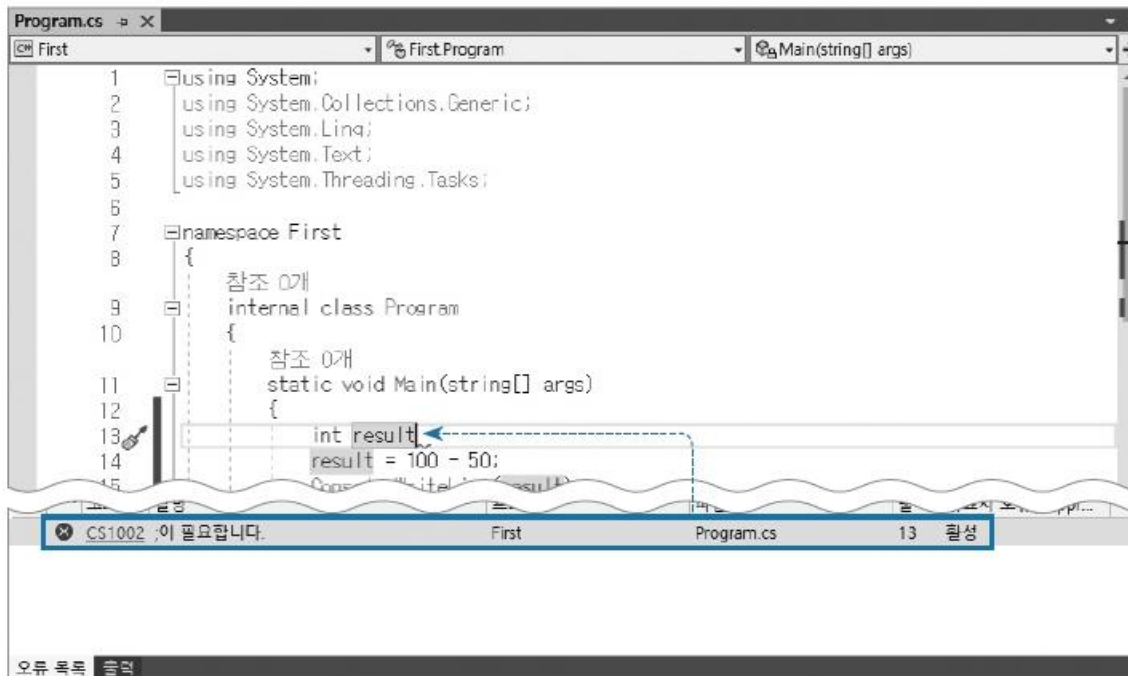


그림 1-33 소스 코드의 틀린 부분 찾기

## 05. 처음 작성하는 C# 프로그램

### III. 빌드(컴파일 및 링크)

#### [실습 1-4] 소스 코드 빌드하기

- ⑤ 오류가 발생한 부분과 원인을 찾았으면 오류를 수정해보자. 13행의 제일 뒤쪽에 세미콜론을 작성한 후 파일을 저장한다. 메뉴의 [빌드] - [솔루션 빌드]를 눌러서 다시 빌드하면 **빌드에 성공했습니다** 메시지가 나올 것이다.

#### 여기서 잠깐

#### C# 소스 코드의 줄바꿈

C# 언어는 줄을 바꾼다고 문장이 끝나는 것이 아니라 세미콜론(;)을 만나야 문장이 끝난다. 아래 네 문장은 표현 방식은 다르지만 모두 같은 의미이다.

① `Console.WriteLine(result);`

② `Console.WriteLine  
(result);`

③ `Console.WriteLine(  
result);`



④ `Console.WriteLine(result  
)  
;`

## 05. 처음 작성하는 C# 프로그램

### IV. 파일 실행

- .exe 파일이 생성되었는지 확인, 실행하는 방법
  - 명령 프롬프트에서 \*.exe 파일이 생성된 폴더로 이동 후 직접 명령을 입력해 실행
  - Visual Studio에서 직접 실행하는 방법

#### [실습 1-5] 명령 프롬프트에서 실행하기

- ① 명령 프롬프트 창에서 직접 실행해보자. + 키를 눌러 [실행] 창이 떴을 때 'CMD'를 입력하고 Enter 를 누르면 명령 프롬프트 창이 나온다. 다음 명령을 차례대로 입력하여 실행 파일이 들어 있는 폴더로 이동한다.

```
CD \  
CD CookC#  
CD First  
CD bin  
CD Debug  
DIR
```

## 05. 처음 작성하는 C# 프로그램

### IV. 파일 실행

#### 여기서 잠깐

#### 기본 도스 명령어

- 도스(DOS): 명령 프롬프트 환경의 운영체제

표 1-3 Windows 도스 명령어

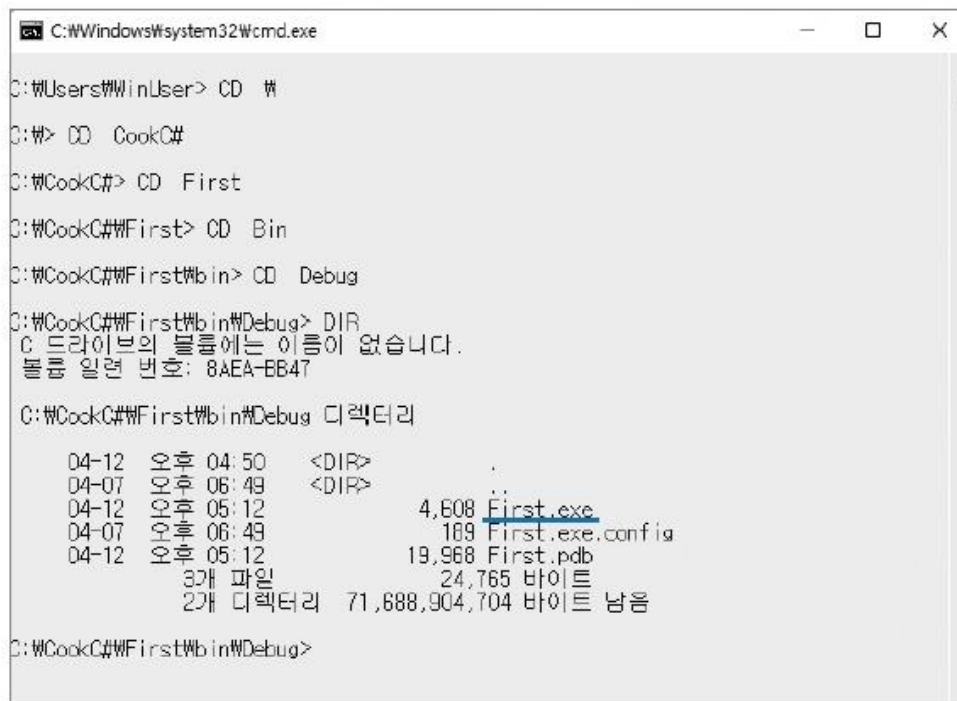
명령어(대·소문자 구분 없음)	설명
DIR	현재 폴더의 파일 목록을 보여줌
CD [폴더 이름]	폴더를 이동함
COPY [원본] [사본]	원본 파일을 사본 파일로 복사함
TYPE [파일명]	텍스트 파일의 내용을 화면에 출력함
DEL [파일명]	파일을 삭제함
MD [폴더명]	새로운 폴더를 생성함
RD [폴더명]	폴더를 삭제함(단, 폴더는 비어 있어야 함)
CLS	화면을 깨끗이 지움



## 05. 처음 작성하는 C# 프로그램

### IV. 파일 실행

#### [실습 1-5] 명령 프롬프트에서 실행하기



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

C:\Users\WinUser> CD \
C:\> CD CookC#
C:\CookC#> CD First
C:\CookC#\First> CD Bin
C:\CookC#\First\Bin> CD Debug
C:\CookC#\First\Bin\Debug> DIR
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 8AEA-BB47

C:\CookC#\First\Bin\Debug 디렉터리

04-12 오후 04:50 <DIR> .
04-07 오후 06:49 <DIR> ..
04-12 오후 05:12          4,608 First.exe
04-07 오후 06:49          189 First.exe.config
04-12 오후 05:12       19,968 First.pdb
                24,765 바이트
                3개 파일
                2개 디렉터리 71,688,904 바이트 남음

C:\CookC#\First\Bin\Debug>
```

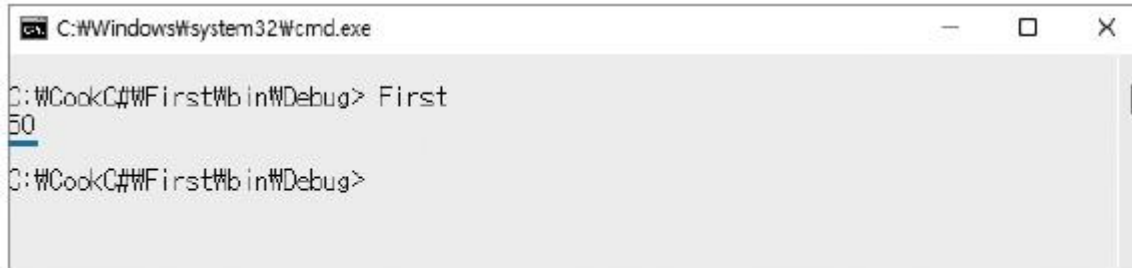
그림 1-34 명령 프롬프트에서 컴파일 결과 확인하기 1

## 05. 처음 작성하는 C# 프로그램

### IV. 파일 실행

#### [실습 1-5] 명령 프롬프트에서 실행하기

- 그 결과, 실행 파일인 **First.exe**가 확인된다. 실행 파일 이름인 First를 입력해서 실행해보면 우리가 코딩한 결과를 확인할 수 있다. 확장명이 .exe인 파일은 확장명을 생략하고 실행해도 된다.



```
C:\Windows\system32\cmd.exe
C:\CookC#\First\bin\Debug> First
50
C:\CookC#\First\bin\Debug>
```

그림 1-35 명령 프롬프트에서 컴파일 결과 확인하기 2

## 05. 처음 작성하는 C# 프로그램

### IV. 파일 실행

#### [실습 1-5] 명령 프롬프트에서 실행하기

- ② 소스 코드를 수정해서 100-50이 아니라 100-99의 결과가 나오도록 코딩해보자.  
[소스 1-2]를 그대로 사용하되 14행만 **100-50**을 **100-99**로 변경한 후 [모두 저장] 아이콘을 클릭해서 변경한 내용을 저장한다.

#### [소스 1-3] 소스 코드 수정하기

1~8행 생략. [소스 1-2]와 동일

```
9     internal class Program
10    {
11        static void Main(string[] args)
12        {
13            int result ;
14            result = 100 - 99 ;
15            Console.WriteLine(result) ;
16        }
17    }
18 }
```

## 05. 처음 작성하는 C# 프로그램

### IV. 파일 실행

#### [실습 1-5] 명령 프롬프트에서 실행하기

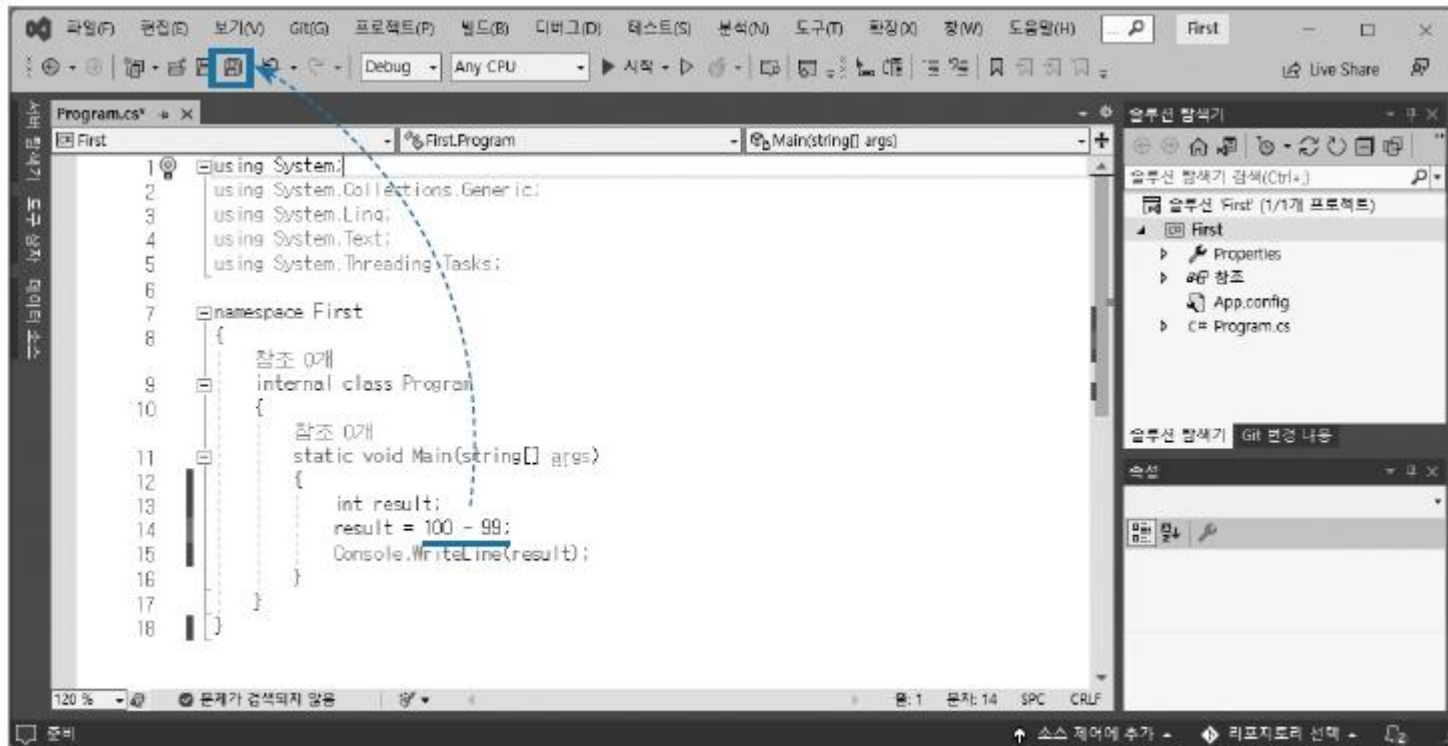


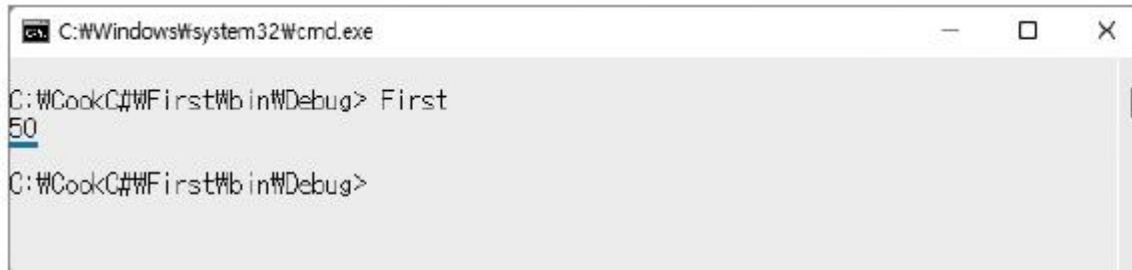
그림 1-36 소스 코드 수정 후 저장

## 05. 처음 작성하는 C# 프로그램

### IV. 파일 실행

#### [실습 1-5] 명령 프롬프트에서 실행하기

- ③ 새로운 결과를 보기 위해 다시 명령 프롬프트 창에서 실행해보자.



```
C:\Windows\system32\cmd.exe
C:\CookC#\First\bin\Debug> First
50
C:\CookC#\First\bin\Debug>
```

그림 1-37 결과 화면 보기 1

- 왜 변경된 내용에 맞춰 결과가 나오지 않았을까?
  - 그 이유는 방금 수정한 소스 코드를 빌드하지 않았기 때문이다. 지금 나오는 결과는 방금 수정한 소스 코드를 실행한 결과가 아니라, 예전에 실행된 소스 코드의 결과이다. 그러므로 소스 코드에서 한 글자라도 변경했다면 반드시 다시 빌드해야 한다.

## 05. 처음 작성하는 C# 프로그램

### IV. 파일 실행

#### 여기서 잠깐

#### 실행 파일의 배포

- 만약 누군가가 100에서 99를 뺀 결과가 출력되는 프로그램을 만들어 달라고 요청했다면, 지금 완료된 First.exe 파일만 복사해서 주면 된다. 그러면 의뢰인은 자신의 PC에서 First.exe 파일을 실행해 원하는 결과를 얻을 수 있다.
- 현재 판매되는 대부분의 소프트웨어(한글, 엑셀, 게임 등)는 판매자 쪽 프로그래머가 직접 프로그램을 작성하고 빌드한 뒤, 구매자에게는 확장자가 .exe인 실행 파일만 주는 것이다.
- 실행 파일 외에 필요한 파일이 더 있을 수는 있지만, 원칙적으로는 실행 파일(.exe)만 판매하며 원본 소스 코드는 공개하지 않는다.
- 요즘은 원본 소스 코드를 공개하는 '오픈 소스(Open Source)' 프로그램도 많이 나오고 있는 추세다

## 05. 처음 작성하는 C# 프로그램

### IV. 파일 실행

#### [실습 1-5] 명령 프롬프트에서 실행하기

- ④ Visual Studio 메뉴에서 [빌드] - [솔루션 빌드]를 선택해 다시 빌드한다. Visual Studio 왼쪽 아래에 **빌드에 성공했습니다** 메시지가 나왔다면 다시 명령 프롬프트에서 실행해본다.



```
C:\Windows\system32\cmd.exe
C:\Cook\#\First\bin\Debug> First
50
C:\Cook\#\First\bin\Debug> First
1
C:\Cook\#\First\bin\Debug>
```

그림 1-38 결과 화면 보기 2

- 이렇게 명령 프롬프트를 사용하는 방식보다 Visual Studio의 환경에서 바로 실행하는 방법을 사용하면 좋다. 별도로 명령 프롬프트를 실행하거나 폴더를 이동하는 과정을 거치지 않아도 돼 한결 수월하기 때문이다.

## 05. 처음 작성하는 C# 프로그램

### IV. 파일 실행

#### [실습 1-6] Visual Studio 환경에서 실행하기

- ① 빌드한 후에 명령 프롬프트에서 결과를 바로 보여주는 단축키인 Ctrl + F5 키를 눌러보자.
- ② 명령 프롬프트창이 나타나며 실행된 결과가 보일 것이다.



그림 1-39 실행 결과



## 05. 처음 작성하는 C# 프로그램

### IV. 파일 실행

#### [실습 1-6] Visual Studio 환경에서 실행하기

- ③ 프로그램 작성 및 실행이 모두 끝났으면 메뉴의 [파일] - [솔루션 닫기]를 선택해 현재 작업 중인 프로젝트를 종료한다.

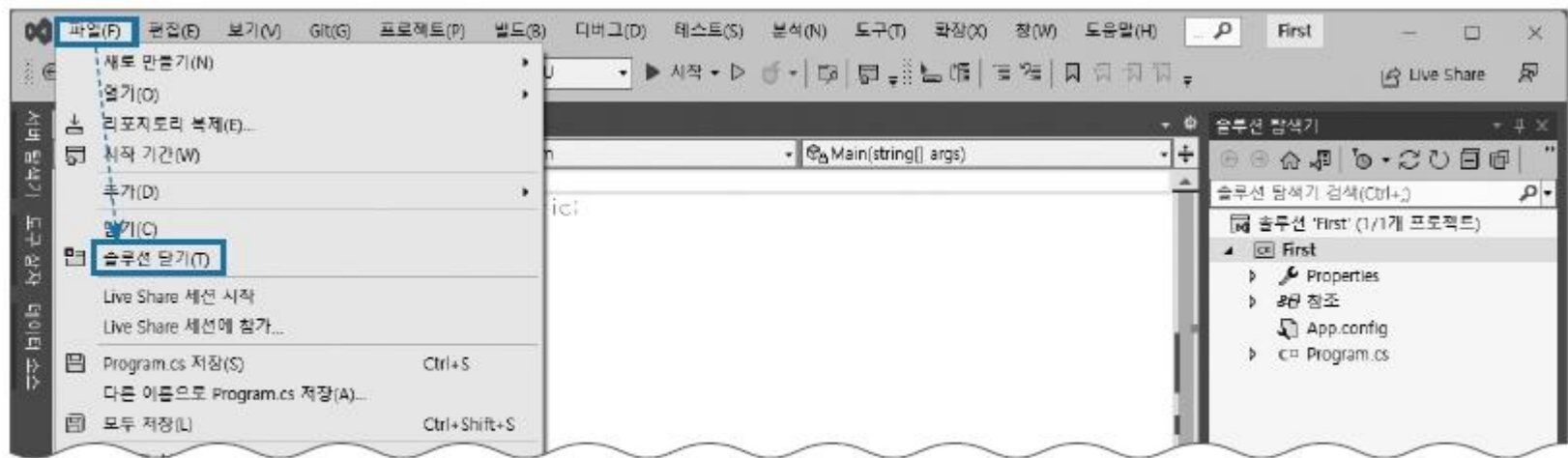


그림 1-40 프로젝트(솔루션) 닫기

- ④ [원하는 작업을 선택하세요] 창이 나오면 화면 오른쪽 상단의 <X>를 눌러 창을 닫는다. 그리고 다시 메뉴의 [파일] - [끝내기]를 선택해 Visual Studio를 종료한다.

## 05. 처음 작성하는 C# 프로그램

### V. 기존의 C# 프로젝트 실행

#### [실습 1-7] 기존의 C# 프로젝트 다시 열기

- ① Visual Studio를 실행한다.
- ② 시작 화면의 [프로젝트 또는 솔루션 열기]를 선택한 후 전에 작업했던 **C:\WCookC#\WFirstW** 폴더의 First.sln을 선택한다.

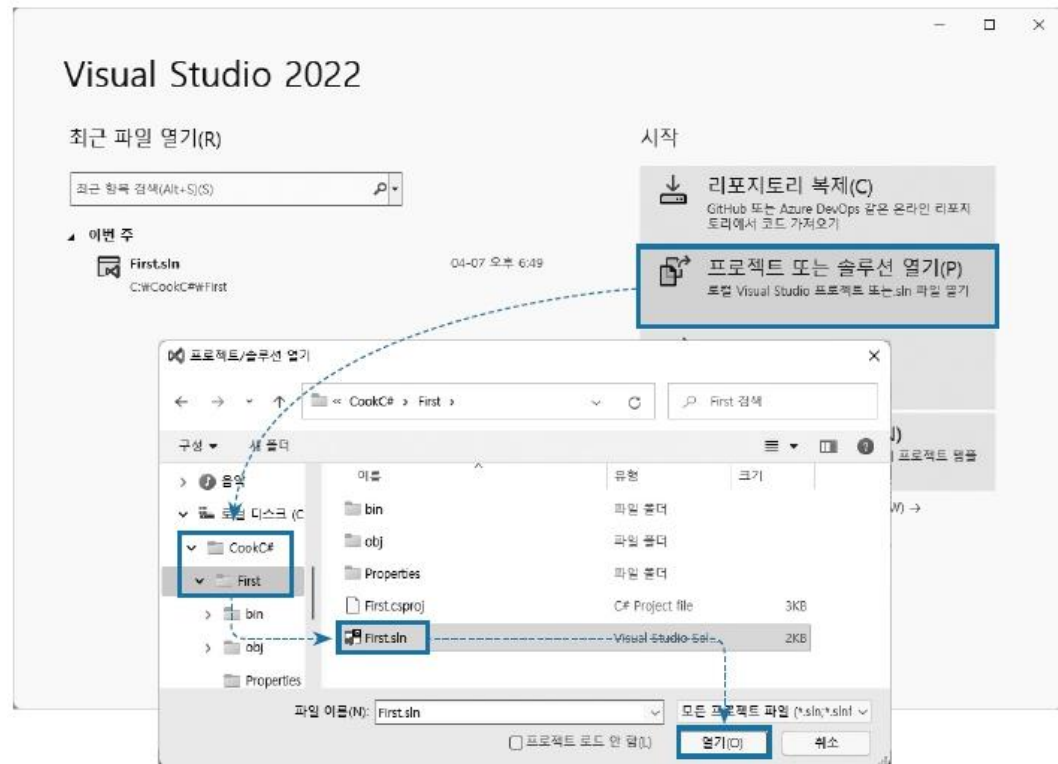


그림 1-41 기존 프로젝트 열기

## 05. 처음 작성하는 C# 프로그램

### V. 기존의 C# 프로젝트 실행

#### [실습 1-7] 기존의 C# 프로젝트 다시 열기

- ③ 기존에 작성했던 Program.cs 파일이 보일 것이다. 소스 코드가 보이지 않는다면 오른쪽 [솔루션 탐색기] 탭을 클릭한 후 [First]를 확장하고, [Program.cs]를 더블클릭한다.

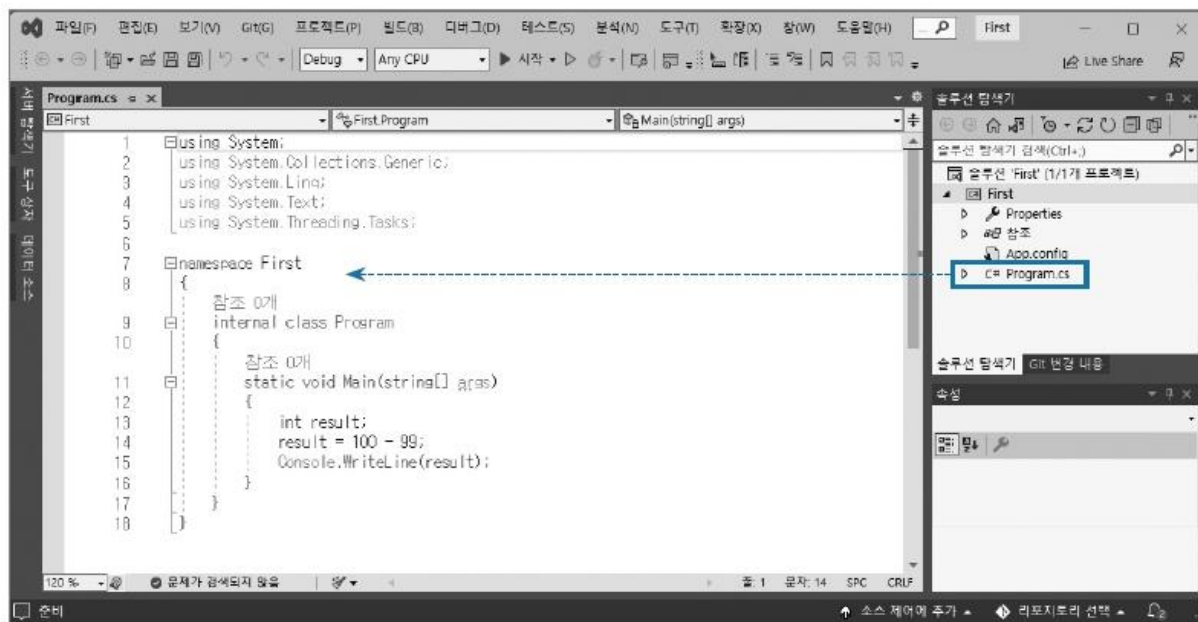


그림 1-42 기존 프로젝트 열기 완료

## 05. 처음 작성하는 C# 프로그램

### V. 기존의 C# 프로젝트 실행

#### [실습 1-7] 기존의 C# 프로젝트 다시 열기

- ④ 실행 화면 메뉴의 [파일] - [끝내기]를 선택해서 Visual Studio를 종료한다.

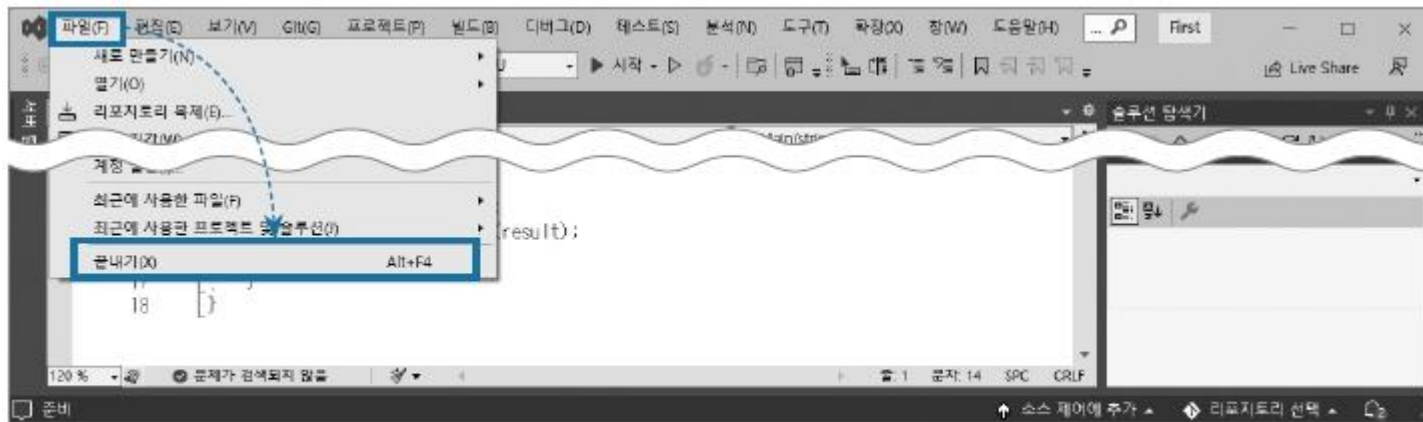


그림 1-43 Visual Studio 종료

## 05. 처음 작성하는 C# 프로그램

### V. 기존의 C# 프로젝트 실행

#### SELF STUDY 1-2

[소스 1-3]을 수정하여  $1000 + 2000$ 의 결과가 나오도록 프로그램을 만들어보자.

**힌트** 프로젝트 이름 : Test , 소스 이름 : Program.cs

# Thank You !