제 2 장 C프로그래밍 첫걸음

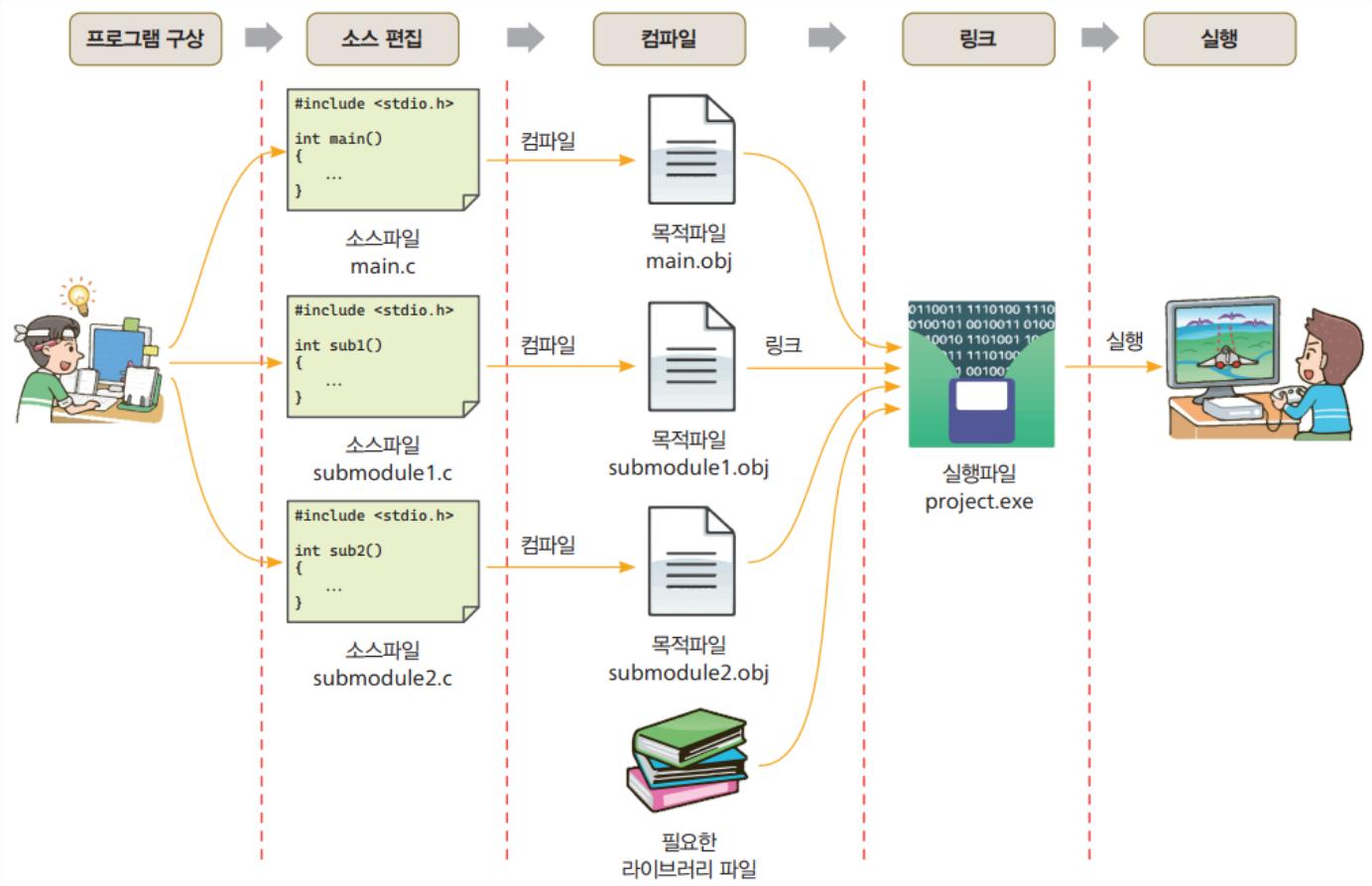
01 프로그램 구현 과정과 통합개발환경
02 비주얼 스튜디오 설치와 c 프로그램의 첫 개발
03 c 프로그램 실행 과정의 이해
04 오류와 디버깅

학습목표

- 프로그램 구현 과정과 통합개발환경을 설명할 수 있다.
  - 프로그램 구상, 소스 편집, 컴파일, 링크, 실행 과정
  - 통합개발환경의 필요 소프트웨어
- 비주얼 스튜디오로 C 프로그램을 개발할 수 있다.
  - 솔루션과 프로젝트의 생성, 소스 생성
  - 빌드와 실행
- 함수 printf()와 puts()로 구성되는 C 프로그램을 구현할 수 있다.
  - 솔루션에 여러 개의 프로젝트 생성
  - 오류의 종류와 원인 파악
  - 오류 발생에 따른 디버깅 과정

프로그램 구현 과정

- **SW 개발 5단계**
  - 요구 분석, 설계, 구현, 검증, 유지보수
  - 프로그램 구현 절차



소스 편집

- **소스파일(source file) 또는 소스코드(source code)**
  - 프로그래밍 언어로 일련의 명령어가 저장된 파일
  - 텍스트 파일
- **소스 파일 확장자**
  - C 언어 .c
  - 자바 .java, C++ .cpp

소스파일은 텍스트파일 형식이므로 모든 편집기로 읽거나 작성할 수 있으나 아래한글이나 워드와 같은 문서편집기에서는 반드시 텍스트 형태로 저장해야 한다.

```c
# include <stdio.h>

int main()
{
   printf("hello, world\n");

   return 0;
}
```

C 소스파일: *.c

소스파일은 모든 편집기로 읽을 수 있으나 아래한글이나 워드로 작성된 일반 문서는 내부 형식이 다르므로 작성한 편집기로만 읽을 수 있다.

아래한글로 작성된 문서는 .hwp라는 확장자이듯, 소스파일은 프로그래밍 언어에 따라 고유한 확장자를 갖는데, C 언어는 .c이며 자바는 .java 그리고 C++는 .cpp이다.

아래한글 파일: *.hwp

컴파일

- **컴파일러(compiler)**
  - 소스파일에서 기계어로 작성된 목적파일(object file)을 만들어내는 프로그램
    - **소스파일이 main.c, submoudule1.c, submoudule2.c**
    - **3개의 목적파일, main.obj, submoudule1.obj, submoudule2.obj가 생성**

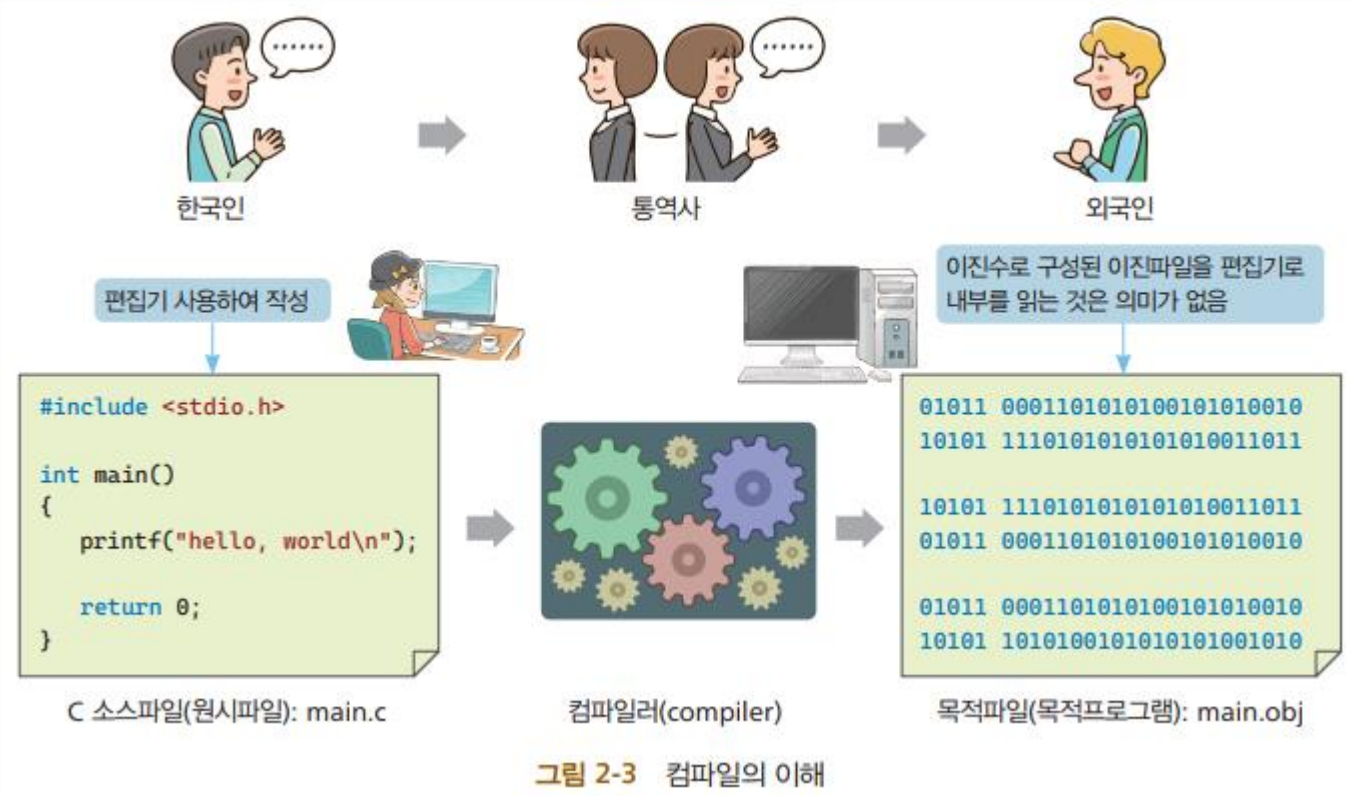


그림 2-3 컴파일의 이해

링크와 실행

- **링크(link) 또는 링킹(linking)**
  - 링커(linker)가 수행하는 과정
    - **여러 개의 목적 파일을 연결하여 하나의 실행 파일(execute file)을 생성해 주는 과정**
      - **참조하는 여러 라이브러리를 포함시킴**
    - **링크의 결과인 실행 파일의 확장자는 .exe 또는 .dll, .com 등**
- **비주얼 스튜디오. 빌드(build)를 제공**
  - 컴파일과 링크 과정을 하나로 합친 메뉴
    - **성공하면 파일 확장자가 .exe인 하나의 실행 파일이 생성**

- **라이브러리란?**
  - 자주 사용하는 프로그램을 이미 만들어 놓고 사용하는 모듈
    - **개발자마다 새로 작성할 필요 없이 개발환경에서 미리 만들어 컴파일해 저장해 놓은 모듈**
    - **리포트를 만들 때 자신의 지식과 보관 자료가 부족하면 도서관의 자료를 찾듯**

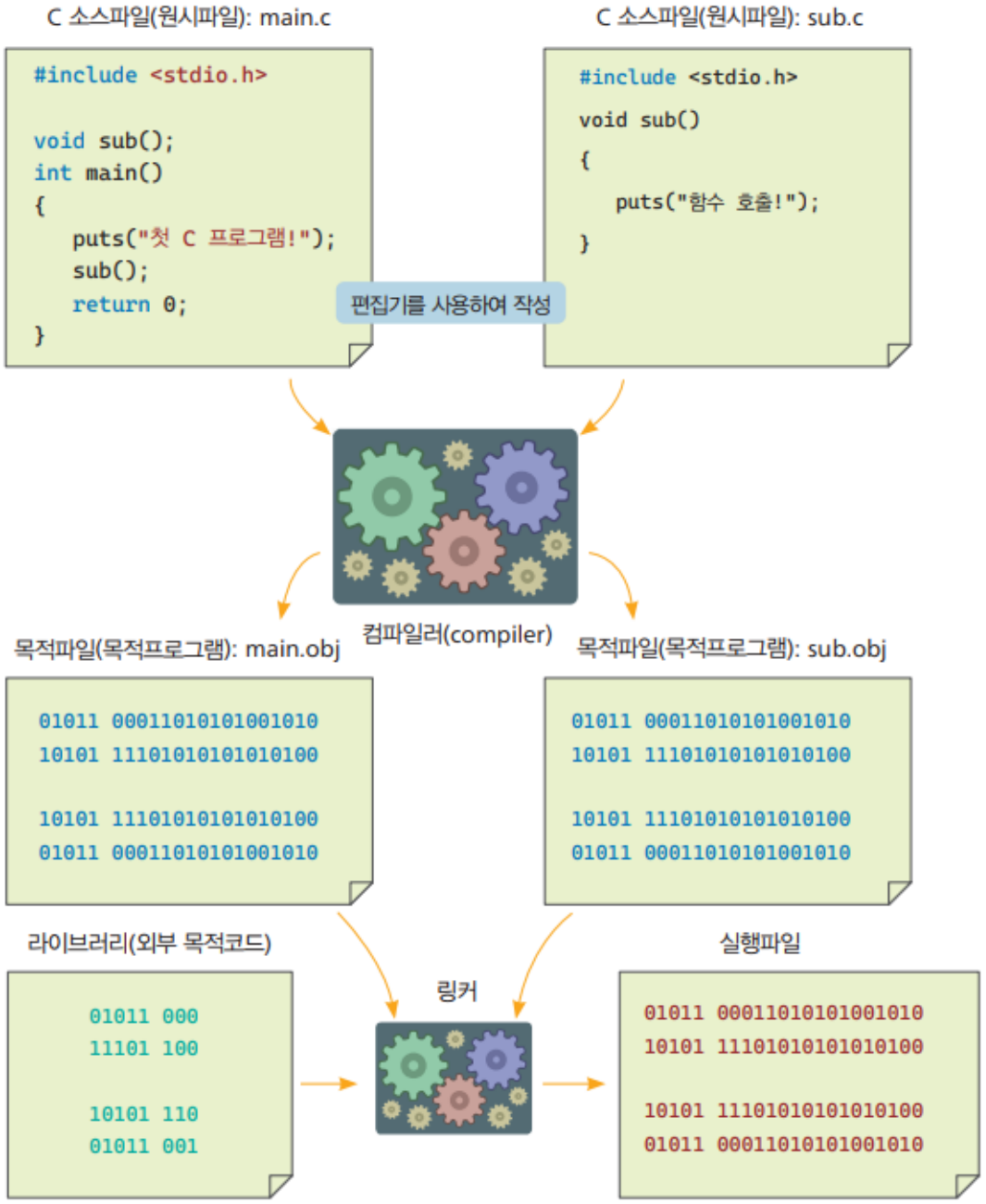

C 소스파일(원시파일): main.c
#include <stdio.h>
void sub();
int main()
{
puts("첫 C 프로그램!");
sub();
return 0;
}
C 소스파일(원시파일): sub.c
#include <stdio.h>
void sub()
{
puts("함수 호출!");
}
편집기를 사용하여 작성
컴파일러(compiler)
목적파일(목적프로그램): main.obj
01011 00011010101001010
10101 11101010101010100
10101 11101010101010100
01011 00011010101001010
목적파일(목적프로그램): sub.obj
01011 00011010101001010
10101 11101010101010100
10101 11101010101010100
01011 00011010101001010
라이브러리(외부 목적코드)
01011 000
11101 100
10101 110
01011 001
링커
실행파일
01011 00011010101001010
10101 11101010101010100
10101 11101010101010100
01011 00011010101001010

그림 2-4 링커의 이해

오류

- **오류 또는 에러(error)**
  - 프로그램 개발 과정에서 나타나는 모든 문제
- **발생 시점에 따른 구분**
  - 컴파일(시간) 오류
    - **개발환경에서 오류 내용과 오류 발생 위치를 어느 정도 알려주므로**
      - **오류를 찾아 수정하기가 비교적 용이**
  - 링크(시간) 오류
    - **컴파일 오류보다 상대적으로 희소**
    - **main() 함수 이름이나 라이브러리 함수 이름을 잘못 기술하여 발생**
  - 실행(시간) 오류
    - **실행하면서 오류가 발생해 실행이 중지되는 경우**

- **오류의 원인과 성격에 따른 구분**
  - 구문 오류(syntax er ror, 또는 문법 오류)
    - **프로그래밍 언어 문법(syntax)을 잘못 기술**
    - **문법이 잘못된 소스로 발생하는 오류**
  - 논리 오류(logical error)
    - **내부 알고리즘이 잘못되거나 원하는 결과가 나오지 않는 등의 오류**
      - **원의 면적을 구하는데 '2 × 3.14(원주율) × 반지름'으로 잘못 계산**
      - **해당 년도의 평균평점을 계산했는데 잘못된 결과가 나오는 등**

디버깅

- **디버깅(debugging)**
  - 다양한 오류를 찾아 소스를 수정하여 다시 컴파일, 링크, 실행하는 과정
  - 디버거(debugger)
    - **디버깅을 도와주는 프로그램**
    - **벌레라는 단어의 버그(bug)란 바로 오류**

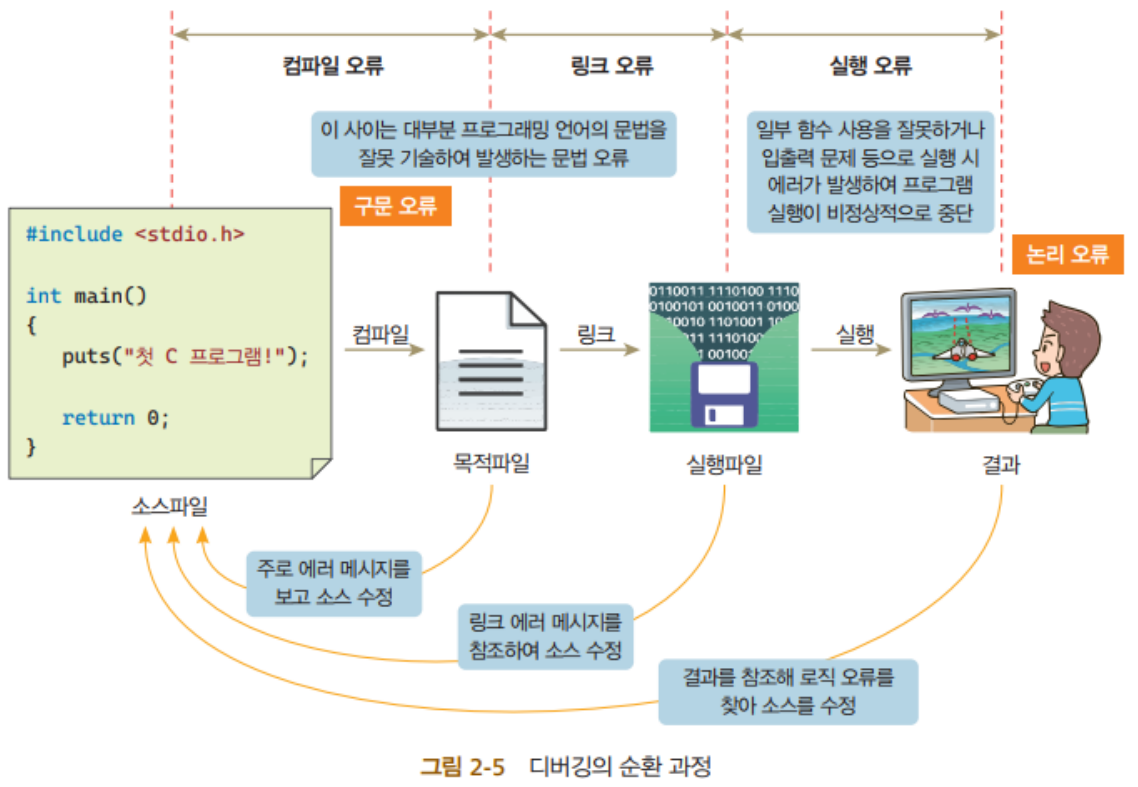


그림 2-5 디버깅의 순환 과정

프로그램 구현 과정

- **컴파일과 링크, 실행 시 오류가 발생**
  - 대부분 소스코드를 수정해서 다시 컴파일, 링크, 실행

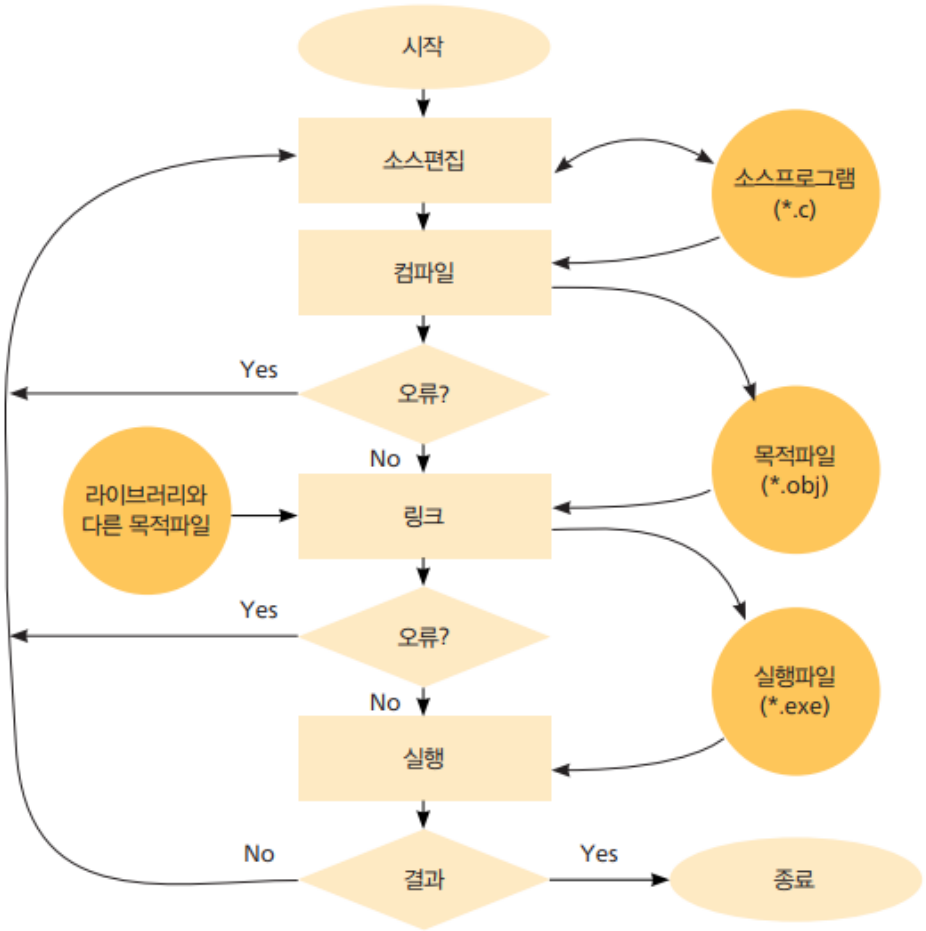


그림 2-9 프로그램 구현 과정 순서도

기사 디버깅의 유래

"아, 이 나방(bug)이 문제였구나!"

컴퓨터에서 일어나는 오작동이 벌레(bug) 때문이라는 것을 최초로 발견한 사람은 미 군함계산식 프로젝트를 맡아 하버드대에서 마크II컴퓨터를 담당하던 그레이스 호퍼 중위였다. 1947년 9월 9일, 호퍼 중위는 컴퓨터가 갑자기 멈추게 되자 컴퓨터 패널의 릴레이를 살펴보게 되었는데 바로 여기에 나방이 끼어 있었던 것이다.

마크II 컴퓨터는 그녀의 상관인 해군 예비역 중령 하워드 에이킨이 만든 것이었는데, 그 당시 최고 성능의 최신식 디지털 컴퓨터였다. 에이킨은 하버드대 물리학과 출신으로 치밀한 성격의 소유자였는데 IBM과 더불어 마크컴퓨터 시리즈를 개발한 천재이자 0세대 컴퓨터계의 대표적인 인물이기도 하다.

호퍼 중위는 릴레이 사이에서 벌레가 발견된 전후 상황을 컴퓨터 로그기록 노트에 자세하게 적었다.

그림 2-6 벌레를 발견한 그레이스 호퍼

"09:00 마크II 작동 시작, 10:00 작동을 멈추다. 패널 릴레이를 교환하다. 11:00 코사인 테이프 시작. 15:25 멀티 덧셈기를 테스트하다. 15:47 70번 패널 릴레이에서 벌레(나방)가 끼인 것을 확인하다. 이것은 벌레가 발견된 실제 케이스이다."

그림 2-7 벌레 발견 정황을 자세하게 기록한 로그 노트

그녀는 곧바로 그 나방을 테이프로 노트에 붙여 놓았다. 지금과 달리 당시의 컴퓨터는 기계식이었고 자전거 체인과 같은 릴레이로 된 방식이 대부분이었다. 이러한 작동부분이 몇 군데나 되었기 때문에 당시에는 나방이나 각종 벌레, 심지어 쥐가 기계를 고장내는 일이 다반사였다. 호퍼 중위가 나방을 발견했을 때 '버그(bug)'라는 단어가 구체적인 의미를 나타나게 되었다. 이후 호퍼의 보고서에 등장한 '버그'는 2차 대전 중 사용되는 레이더의 오작동을 설명하는 데 사용되는 등 컴퓨터나 정밀기기에 가장 많이 쓰이게 되는 단어가 된다.

통합개발환경

- **IDE(Integrated Development Environment)**
  - 프로그램 개발에 필요한 편집기(editor), 컴파일러(compiler), 링커(linker), 디버거(debugger) 등을 통합하여 편리하고 효율적으로 제공하는 개발환경

그림 2-10 통합개발환경의 이해

마이크로소프트(MS) 사의 비주얼 스튜디오

- **여러 프로그래밍 언어와 환경을 지원하는 통합개발환경**
  - 프로그램 언어 C/C++ 뿐만 아니라
    - **C#, JavaScript, Python, Visual Basic 등 여러 프로그램 언어를 이용**
- **응용 프로그램 및 앱을 개발할 수 있는 다중 플랫폼 개발 도구**
  - 비주얼 스튜디오 프로페셔널(professional)
  - 비주얼 스튜디오 엔터프라이즈(enterprise)
  - 무료버전: 비주얼 스튜디오 커뮤니티(community)

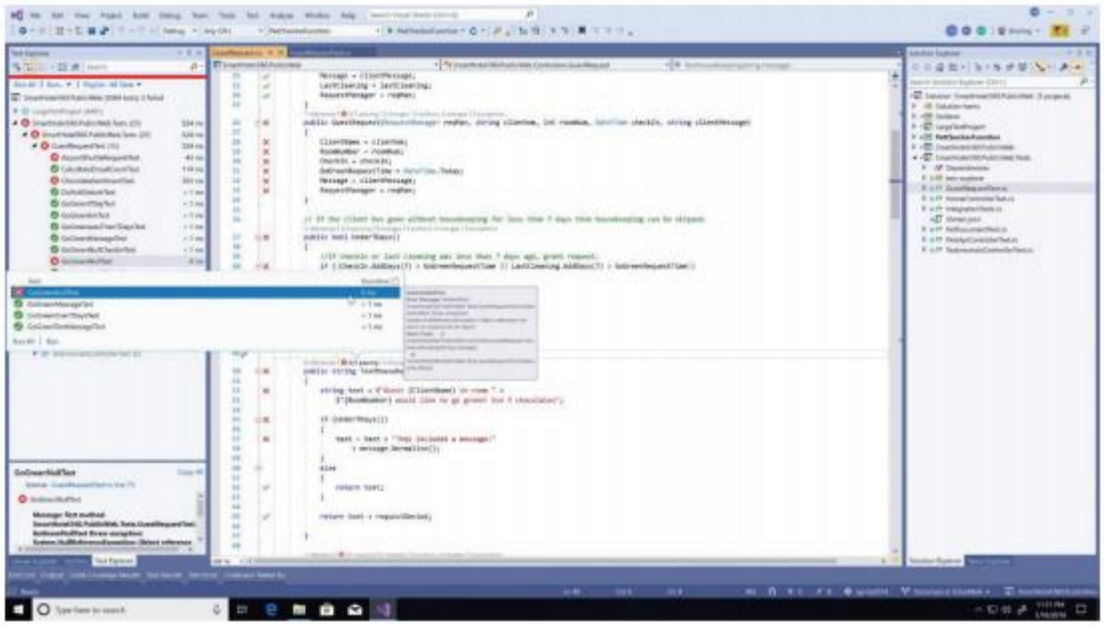


그림 2-11 대표적 IDE인 마이크로소프트사의 비주얼 스튜디오 IDE 화면

비주얼 스튜디오 커뮤니티 설치와 이해

- **비주얼 스튜디오 커뮤니티 내려 받기**
  - 무료인 '비주얼 스튜디오 커뮤니티'
  - 비주얼 스튜디오 홈페이지(visualstudio.microsoft.com/ko)에 접속
    - **'Visual Studio 다운로드' 하부의 'Community 2019'를 누르면**
      - **하단에 표시된 저장 표시 화면에서 [저장(S)]을 누르면 저장할 수 있으며, [실행(R)]을 눌러 설치를 시작**

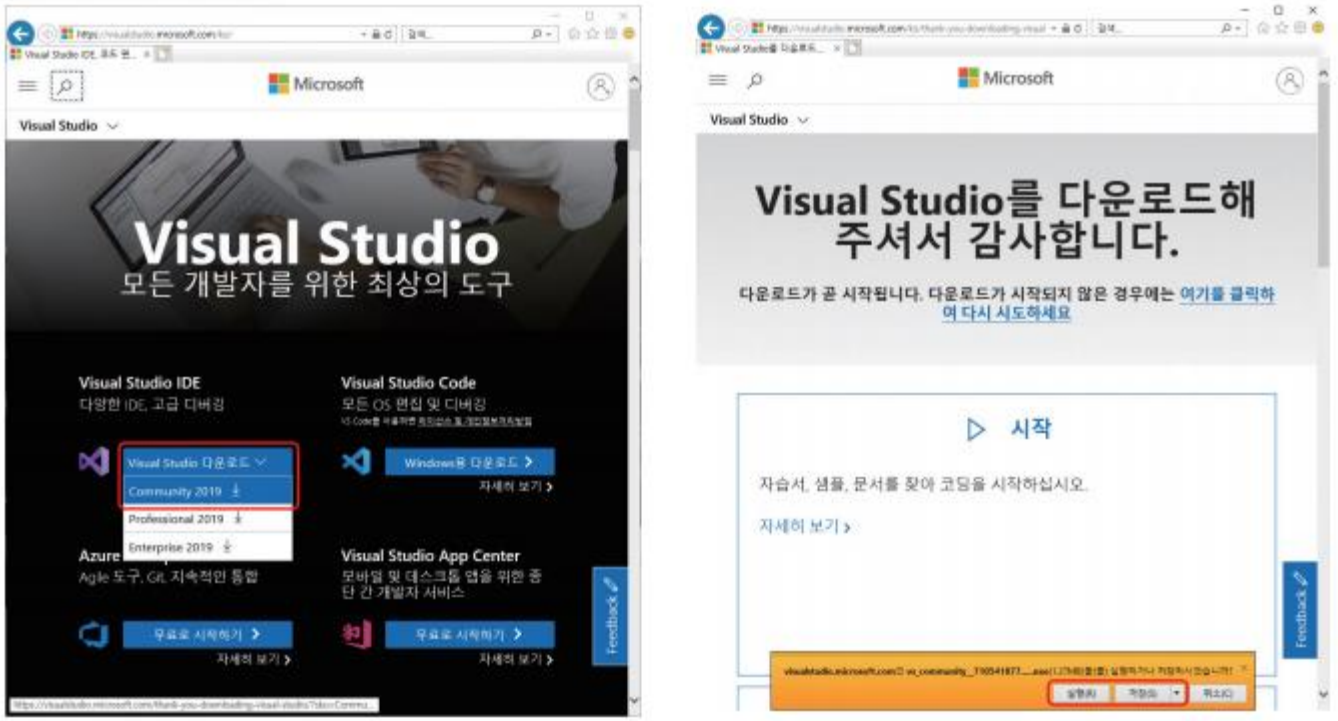
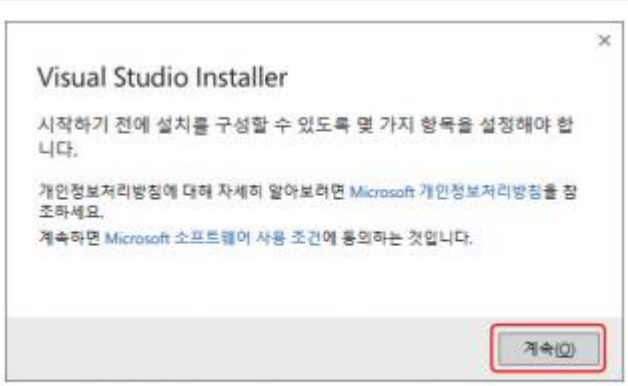


그림 2-22 비주얼 스튜디오의 홈페이지(visualstudio.microsoft.com/ko)

비주얼 스튜디오 설치와 실행



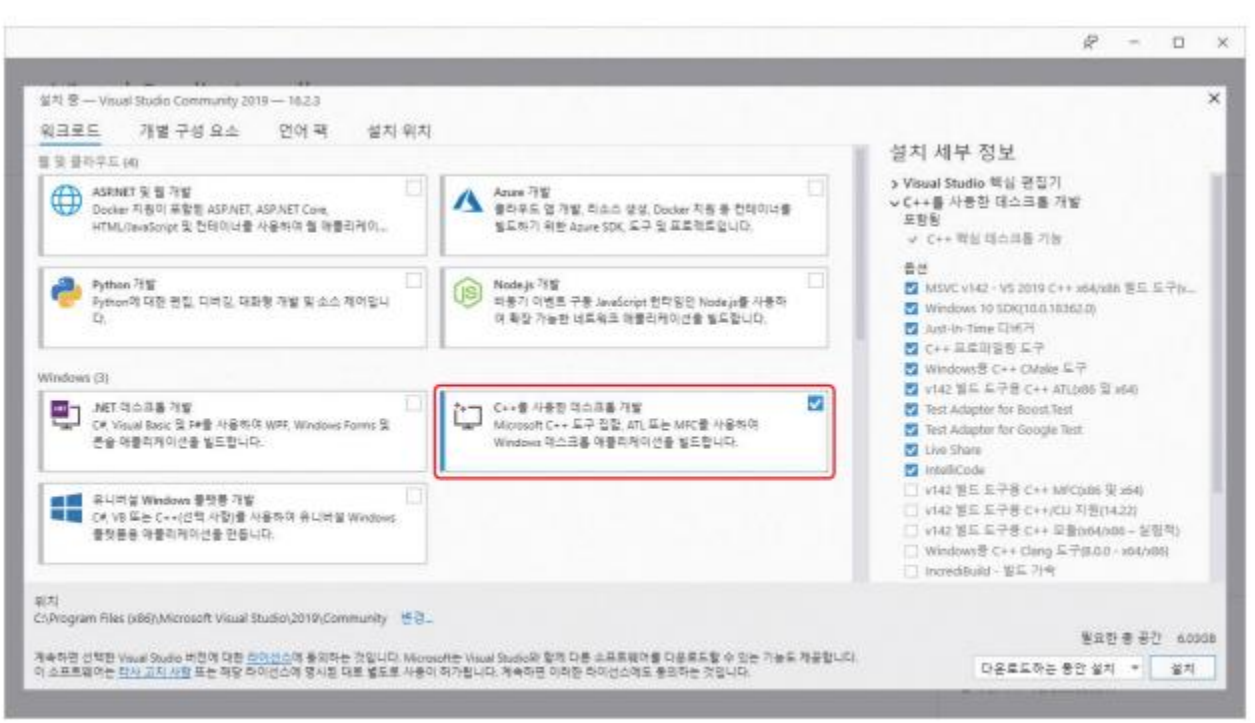


그림 2-24 비주얼 스튜디오 커뮤니티 2019 설치 과정에서 [C++를 사용한 데스크톱 개발]을 선택

설치

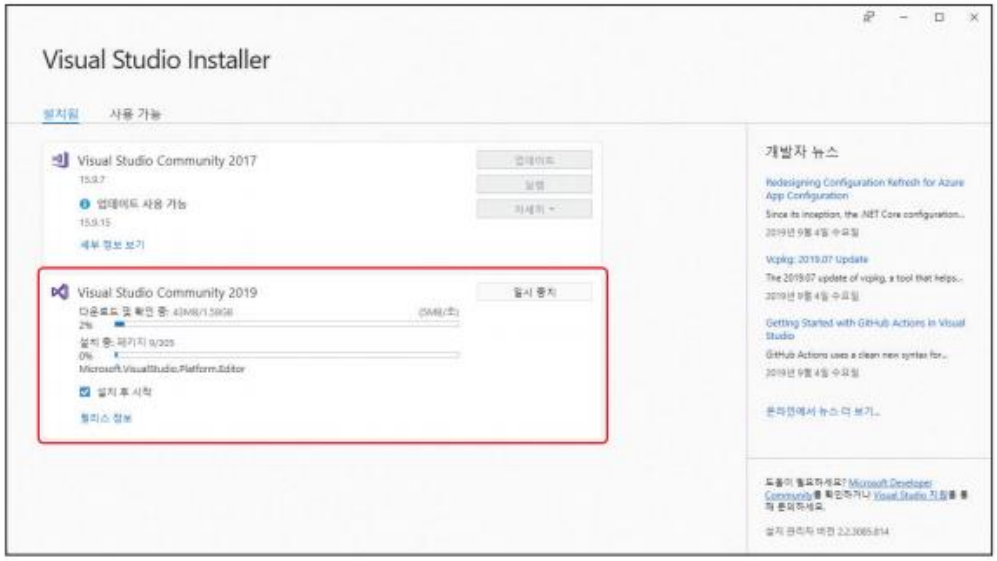


그림 2-25 비주얼 스튜디오 커뮤니티 2019 설치 과정

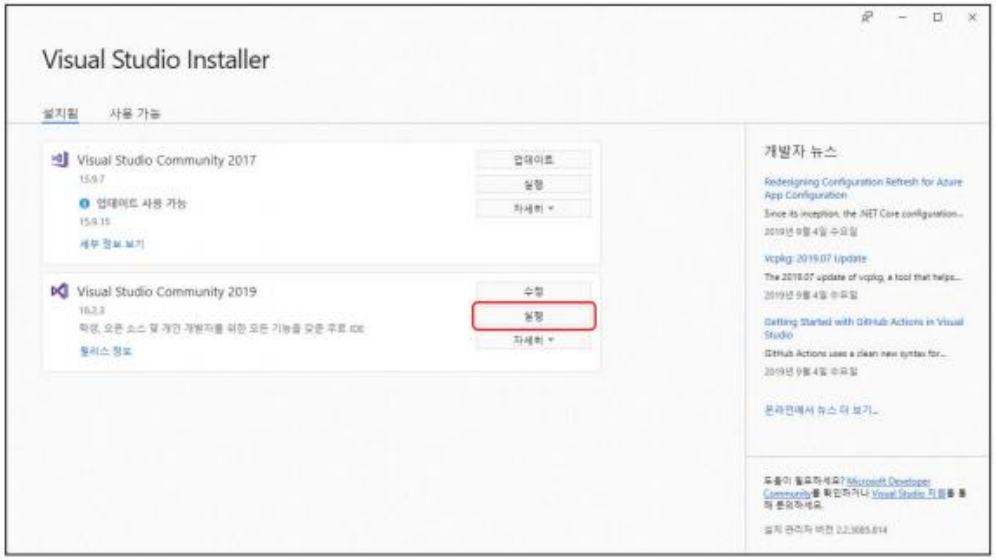
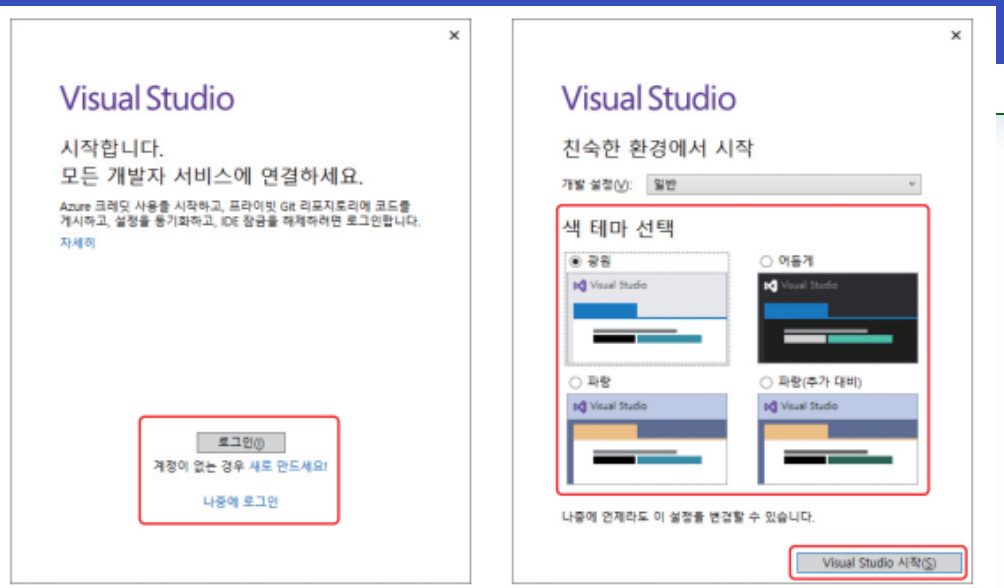
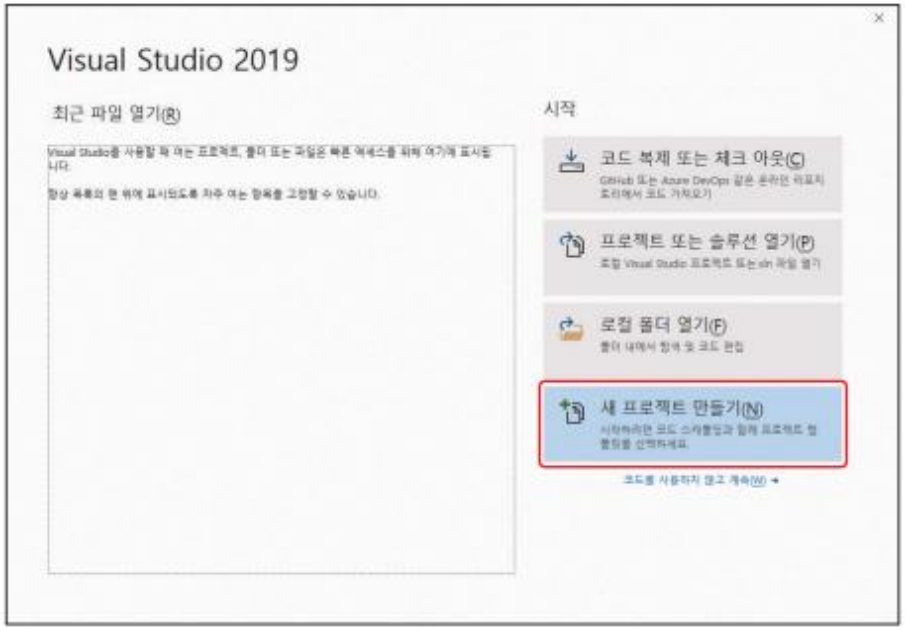


그림 2-26 Visual Studio Installer에서 Visual Studio Community 2019 실행

실행


Visual Studio
시작합니다.
모든 개발자 서비스에 연결하세요.
Azure 크레딧 사용을 시작하고, 프라이빗 Git 리포지토리에 코드를 게시하고, 설정을 동기화하고, IDE 잠금을 해제하려면 로그인합니다.
자세히
로그인(I)
계정이 없는 경우 새로 만드세요!
나중에 로그인
Visual Studio
친숙한 환경에서 시작
색 테마 선택
Visual Studio 시작(S)


Visual Studio 2019
최근 파일 열기(R)
시작
코드 복제 또는 체크 아웃(C)
프로젝트 또는 솔루션 열기(P)
로컬 폴더 열기(F)
새 프로젝트 만들기(N)

첫 프로젝트 생성

- **로고 화면 이후 표시된 위 화면**
  - [새 프로젝트 만들기(N)]
  - [빈 프로젝트]를 선택

그림 2-29 새 프로젝트 만들기의 빈 프로젝트 선택

솔루션과 프로젝트 생성

- **임의로 지정된 프로젝트 이름과 위치, 그리고 솔루션 이름 지정**

주요 설정	설명	설정 내용
이름	만들려는 프로젝트 이름을 입력하며, 최종 실행파일 이름 P01-HelloWorld.exe이 됨	P01-HelloWorld
위치	솔루션과 프로젝트가 저장되는 폴더	D:\C code
솔루션 이름	만들려는 솔루션(여러 프로젝트의 모임) 이름을 입력	ch02

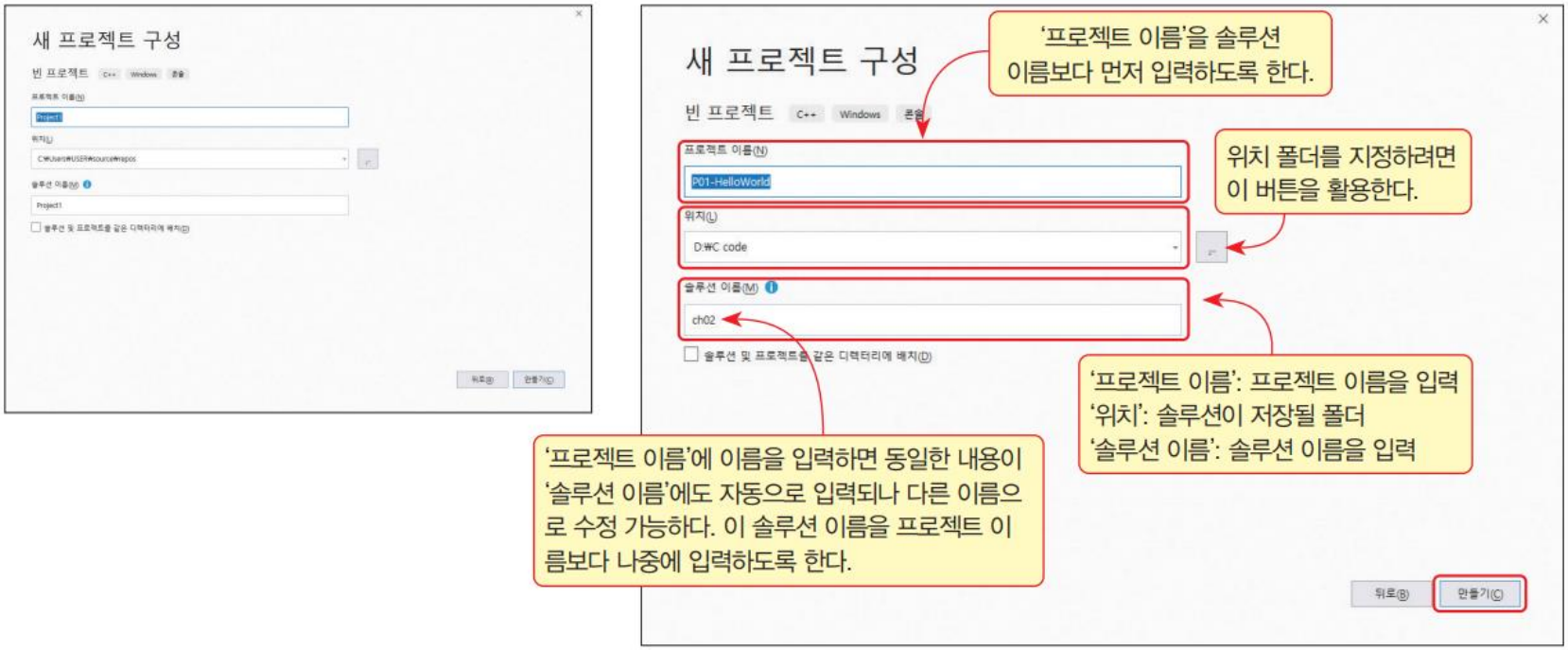
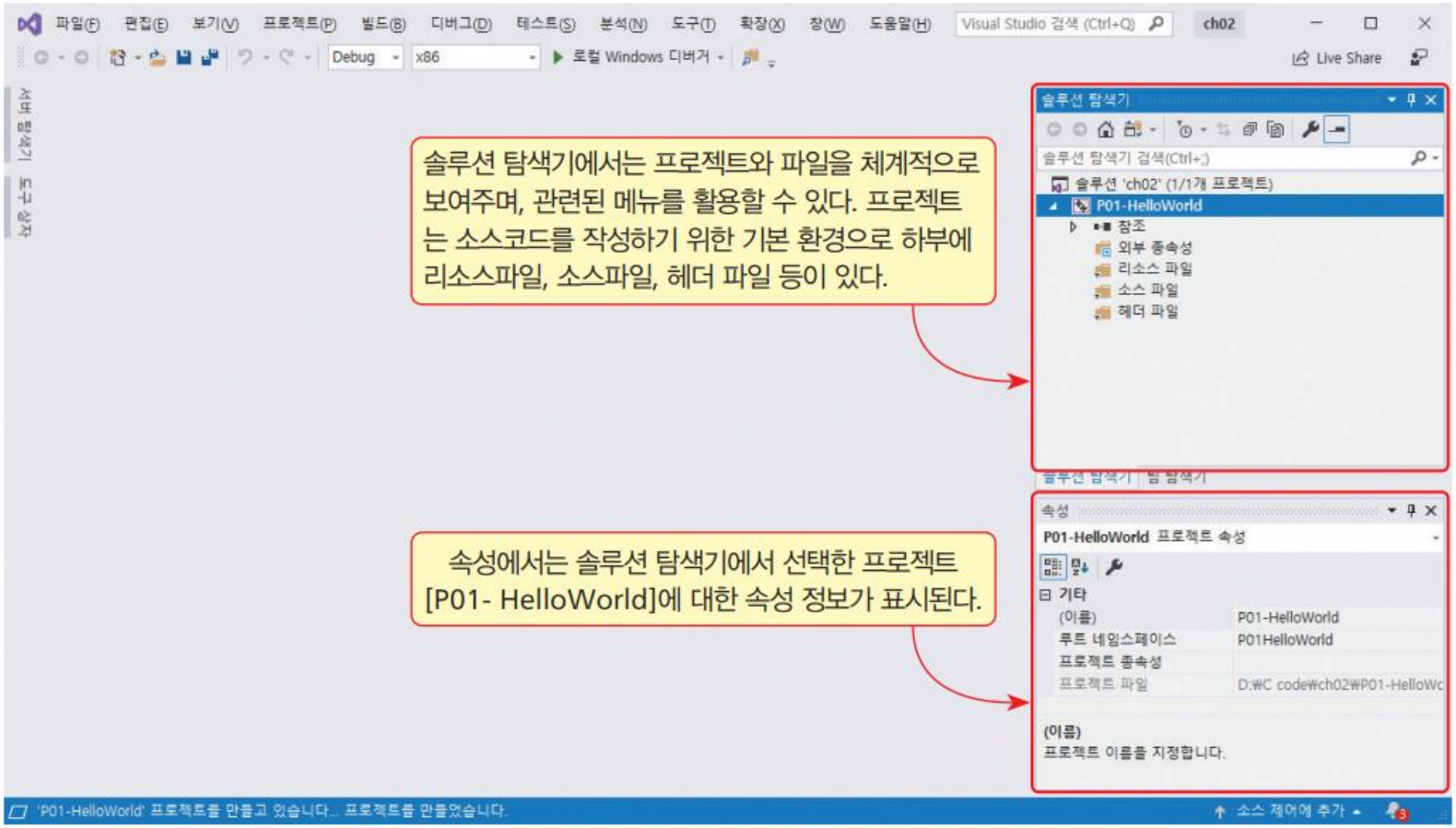


그림 2-30 새 프로젝트 구성에서 프로젝트 이름, 위치, 솔루션 이름 지정

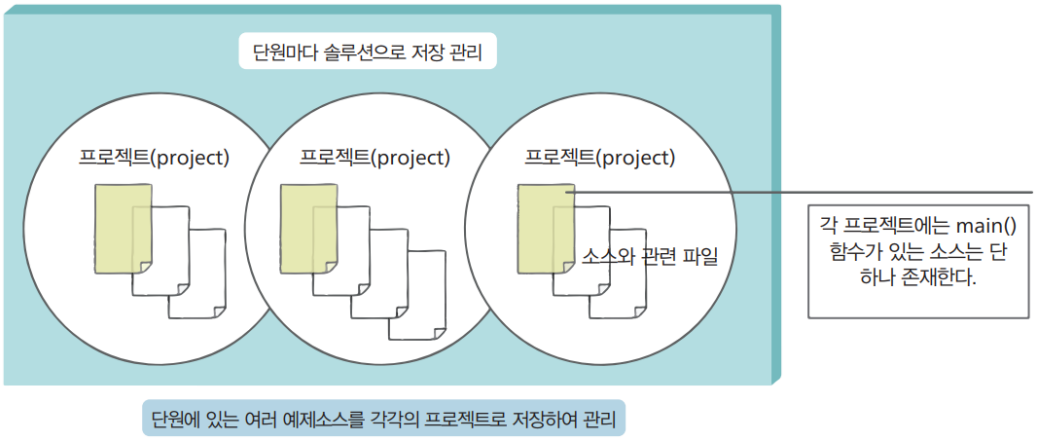
솔루션 탐색기

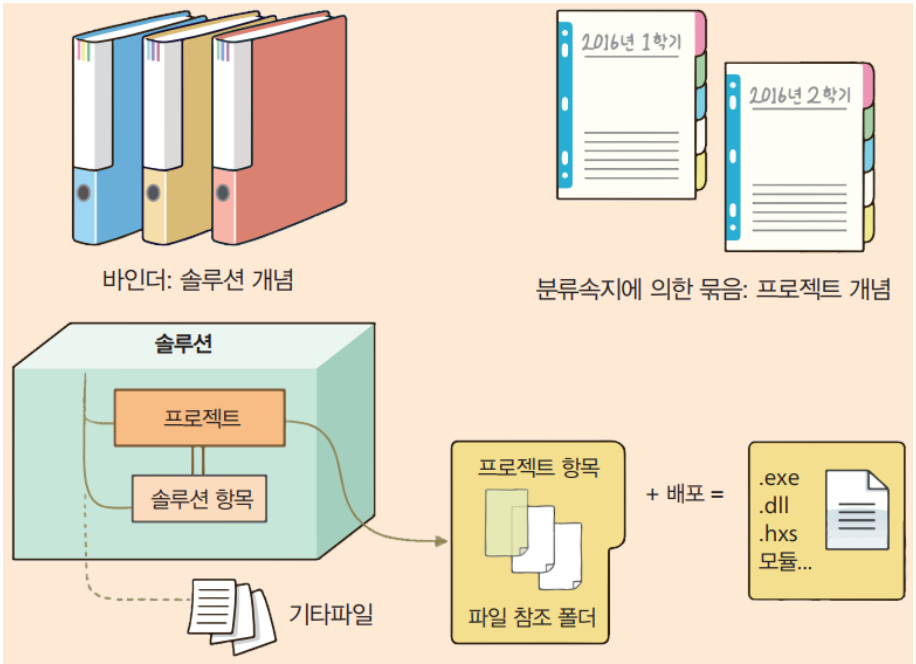
- **주 화면 오른쪽 '솔루션 탐색기'**
  - 생성된 솔루션과 프로젝트
    - **솔루션의 프로젝트와 파일을 쉽게 관리**



솔루션과 프로젝트

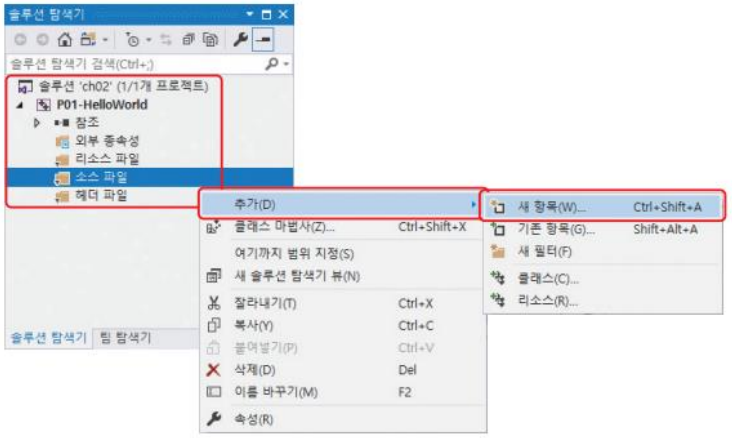
- **솔루션: 단원**
  - 프로젝트: 각각의 예제
- **'바인더'**
  - 솔루션에 해당
  - 하나 이상의 프로젝트를 저장・관리하는 콘테이너 단위
- **'분류속지'**
  - 프로젝트
  - 여러 소스와 관련 파일을 저장・관리하는 단위
  - 프로젝트 이름으로
    - **하나의 실행파일이나 실행 모듈을 생성**

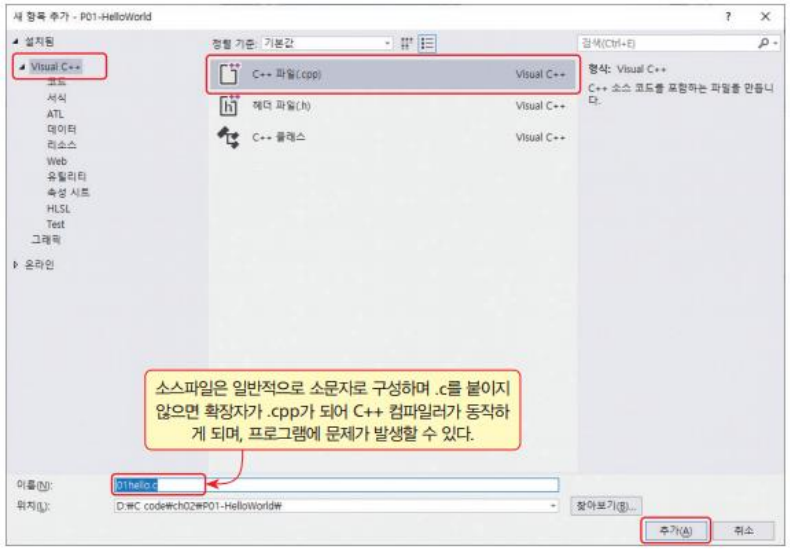




프로젝트에서 소스 작성

- **소스파일 생성 편집**
  - 메뉴 [프로젝트/새 항목 추가]
    - **또는 '솔루션 탐색기'의 '소스파일' 폴더에서 메뉴 [추가] / [새 항목]을 선택**
  - 대화상자 [새 항목 추가 P01-HelloWrold]
    - **각각 'Visual C++'와 'C++ 파일(.cpp)'을 선택**
      - **'이름': 소스파일 이름 01hello.c를 입력**
      - **'위치': '솔루션 폴더/프로젝트 폴더'인 'ch02/P01-HelloWrold'를 확인**
      - **'이름': 소스파일 이름 01hello.c 입력**





NOTE: **파일 이름에 반드시 확장자 .c를 입력하도록 하자.**

입력하지 않으면 자동으로 확장자가 .cpp가 되어 C++ 컴파일러가 동작하기 때문에 C 소스코드 컴파일 시 문제가 발생한다.

텅 빈 소스파일 01hello.c, 생성된 화면 모습

- 솔루션 탐색기의 '소스파일' 폴더 하단 부
  - 파일 01hello.c가 표시
- 왼쪽에는 소스를 편집할 수 있는 창
- 오른쪽은 솔루션 탐색기
  - 각 소스파일을 선택
    - 하단 속성 탭에서 선택된 파일의 자세한 정보 표시

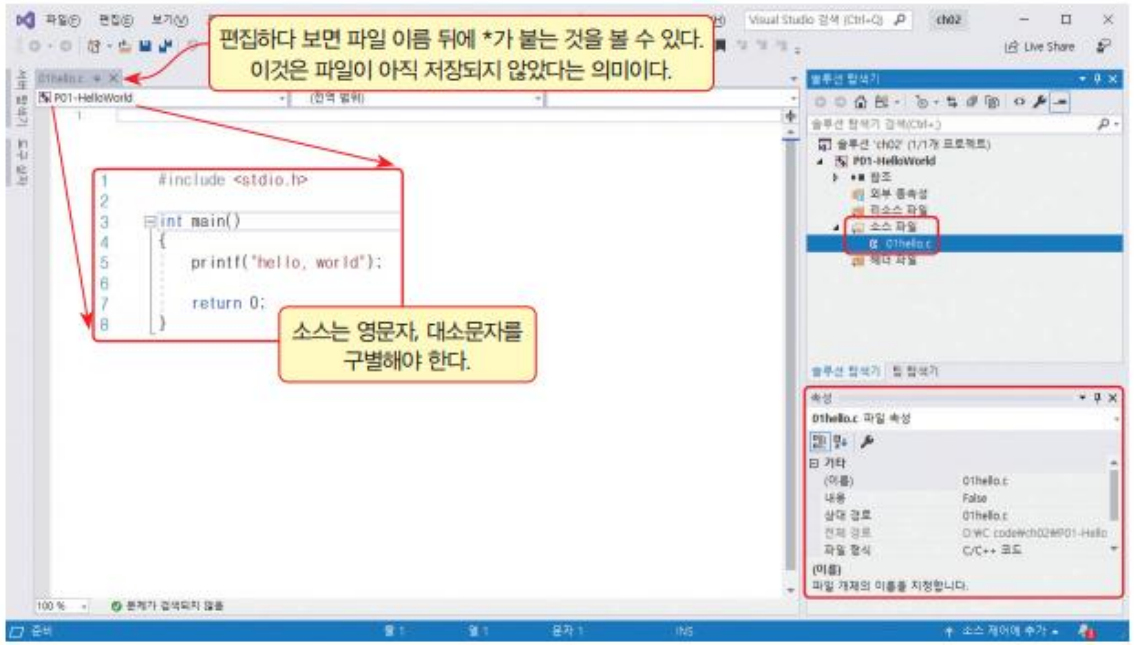


그림 2-35 소스파일 생성과 편집

솔루션과 프로젝트 그리고 프로그램 소스

- **하나의 솔루션**
  - 하나 이상의 프로젝트를 저장
  - 하나의 프로젝트는 여러 개의 프로그램 소스
    - **하나의 main() 함수 만을 구성**

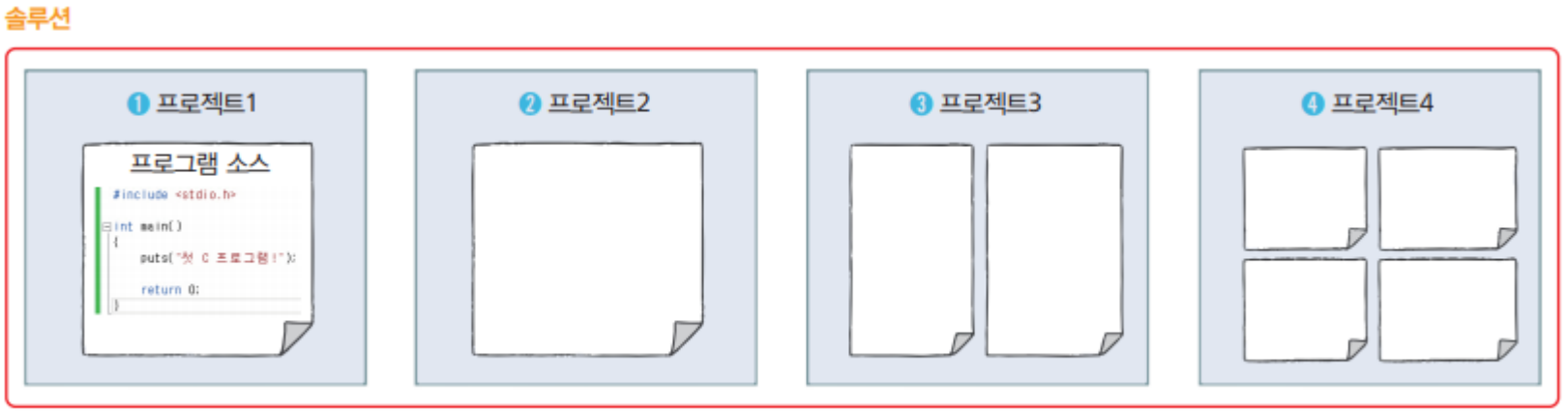


그림 2-36 솔루션과 프로젝트 그리고 프로그램 소스

첫 예제

- **문자열 "hello, world"가 콘솔 창에 출력되는 프로그램**
  - printf()의 괄호 사이에 기술된 "hello, world"가 출력
    - **대소문자를 구별**
      - **include, stdio.h, int, main, printf, return**
    - **특별한 의미의 여러 문자들로 구성**
      - **#, <, >, (,), ;, {, }**

실습예제 2-1 | P01-HelloWorld | 01hello.c | 콘솔에 hello, world 출력 | 난이도: ★

```
01  #include <stdio.h>
02
03  int main()
04  {
05      printf("hello, world");
06
07      return 0;
08  }
```

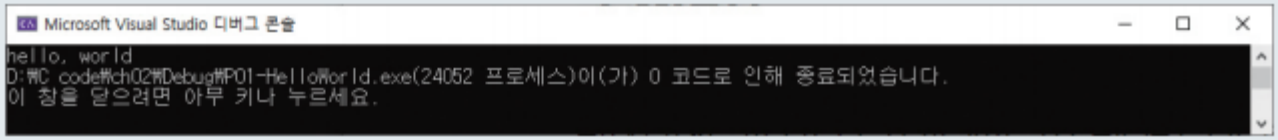
printf() 함수가 콘솔에 출력할 문자열을 큰 따옴표로 묶어야 한다.

문장을 종료할 때는 ; (세미콜론)을 입력해야 한다.

5 줄에서 시작해서 6 줄을 지나, 7 줄 return 0로 0을 반환하고 프로그램이 종료된다.

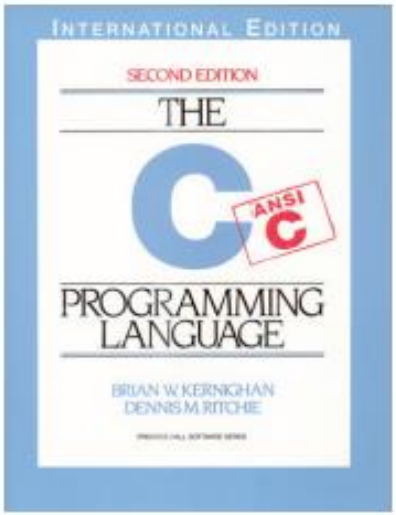
결과

```
hello, world
```



NOTE: **C 언어와 'hello, world' 출력 프로그램의 유래**

1978년 프로그래밍 언어 C를 개발한 데니스 리치(Dennis Ritchie)는 동료 브라이언 커닝핸(Brian Kernighan)과 함께 너무나 잘 알려진 "The C Programming Language" 라는 책을 출판한다. 흔히 이 책을 K&R C라고 부르는데, K&R C는 이후 1988년 미국국립표준협회(ANSI: American National Standards Institute)의 ANSI C 버전으로 2판이 다시 출간됐다. 바로 이 서적의 첫 번째 예제(아래 ❶이 첫 버전의 책의 내용이며, ❷는 2판의 책의 내용임)가 콘솔 화면에 "hello, world"를 출력하는 프로그램이었고, 이것이 유명해지면서 모든 프로그래밍 언어의 첫 번째 예제로 "hello, world"를 출력하는 프로그램을 사용하게 되었다.



❶

program text somewhere, compile it successfully, load it, run it, and find out where your output went. With these mechanical details mastered, everything else is comparatively easy.

In C, the program to print "hello, world" is

```
main()
{
    printf("hello, world\n");
}
```

❷

text somewhere, compile it successfully, load it, run it, and find out where your output went. With these mechanical details mastered, everything else is comparatively easy.

In C, the program to print "hello, world" is

```
#include <stdio.h>

main()
{
    printf("hello, world\n");
}
```

그림 2-37 "The C Programming Language" 2판 표지와 hello world 프로그램의 변화

소스 작성의 주의점과 들여쓰기의 이해

- **세미콜론 ;**
  - 문장의 종료를 표시
  - 콜론 :으로 잘못 입력하면 컴파일에 문제가 발생
- **들여쓰기(indentation)**
  - IDE의 편집기에서 자동으로 맞추어 주며
  - 소스의 가독성(readability)을 위해 반드시 필요

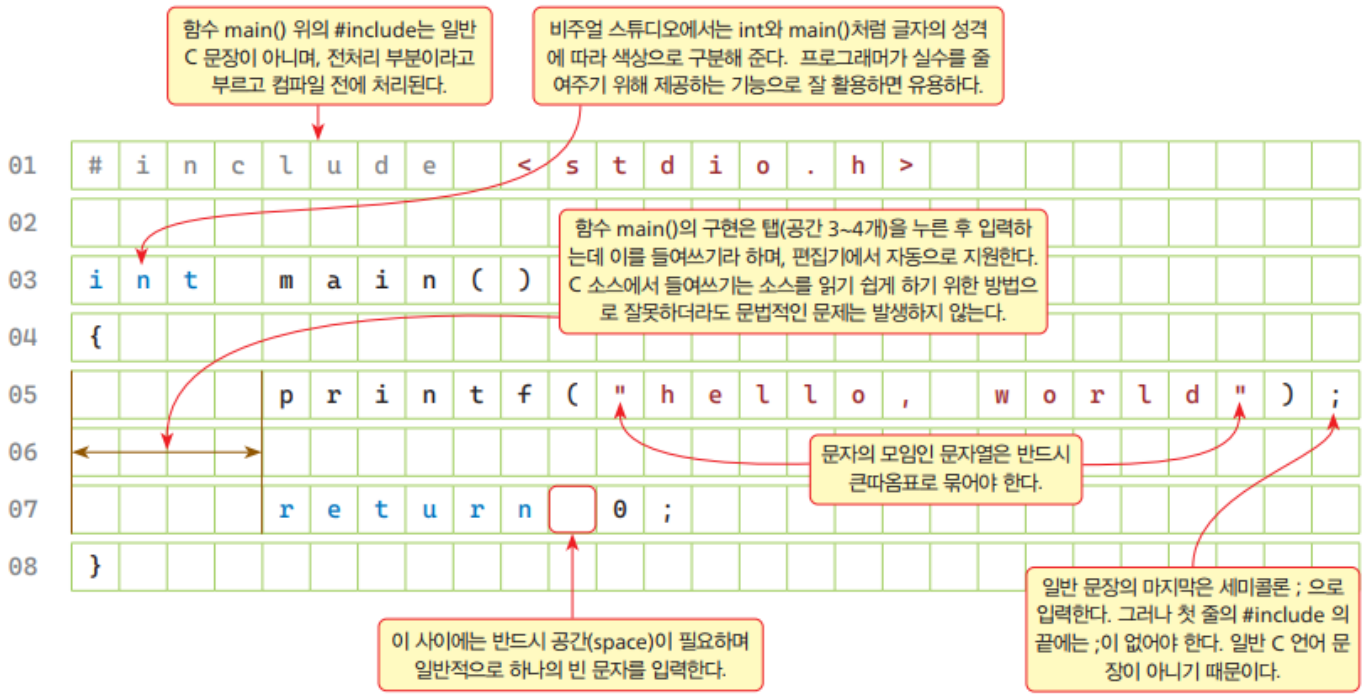


그림 2-38 소스가 기록된 원고지 모습과 주의점

프로젝트 실행 ctrl + F5

- ❶ 작성된 소스에는 문제가 없어야 실행에 성공
  - ❷ [디버그] > ❸ [디버깅하지 않고 시작]을 선택
  - ❹ '출력'에 빌드 과정이 표시
    - **마지막 줄에 성공 1, 실패 0와 같이 표시**
  - ❺ 검정색 바탕의 콘솔 화면이 표시
- **콘솔 프로그램**
  - 텍스트 기반의 프로그램
  - 마지막 줄에 항상 '이 창을 닫으려면 아무 키나 누르세요' 라는 문구가 표시
    - **프로그램 내용과 관계없이 항상 콘솔 화면의 마지막에 표시되는 문구**

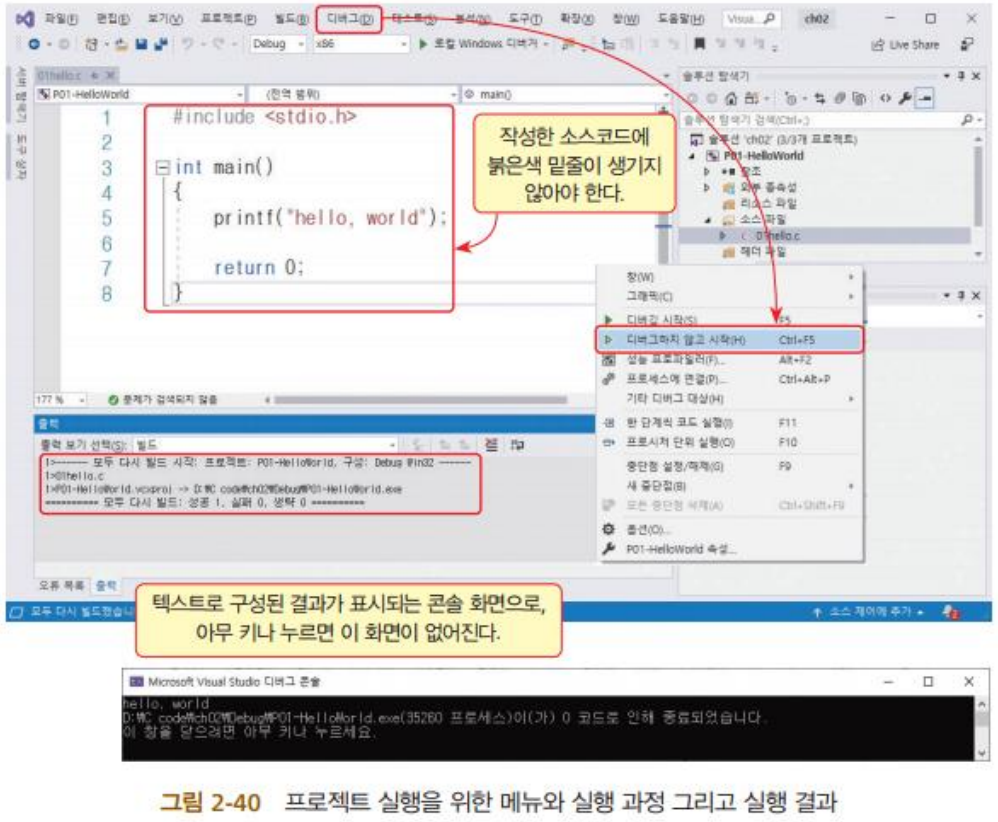


그림 2-40 프로젝트 실행을 위한 메뉴와 실행 과정 그리고 실행 결과

컴파일과 링크

- **빌드 과정을 직접 수행**
  - 비주얼 스튜디오
    - **컴파일과 링크 과정을 거쳐 실행 파일을 생성하는 과정을 빌드**
  - 메뉴 [빌드] / [P01-HelloWorld 빌드]를 선택
    - **화면 하단부의 출력 창에 빌드 과정과 그 결과가 표시**
- **컴파일과 링크를 구분해 실행**
  - 컴파일만 수행
    - **메뉴 [빌드]의 마지막 메뉴 [컴파일(M) ctrl + F7]을 선택**
  - 링크만 구분하여 실행
    - **컴파일 후 메뉴 [빌드] / [프로젝트만] / [P01-HelloWorld만 링크]를 선택**

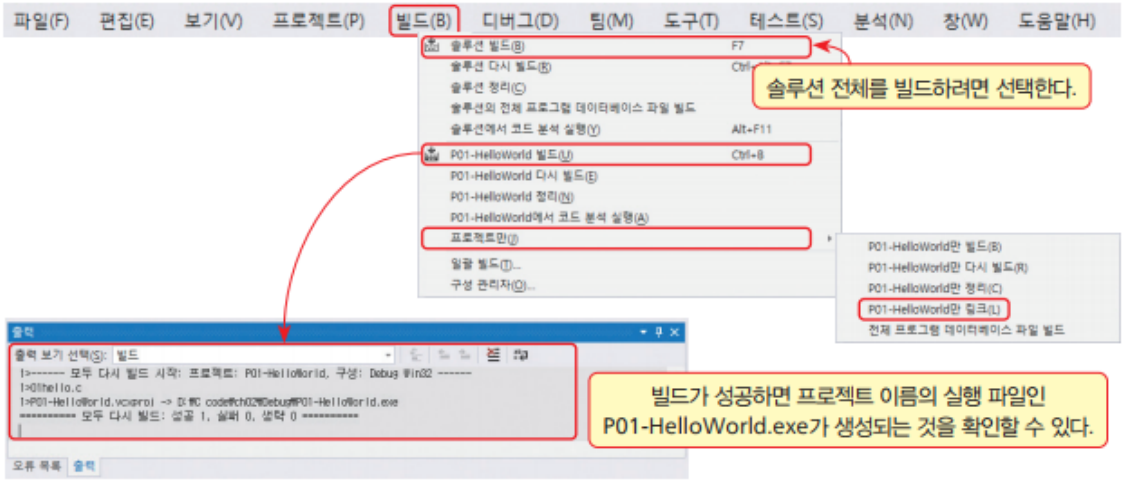


그림 2-41 빌드와 그 결과 화면 '출력'

비주얼 스튜디오 생성 파일

표 2-2 비주얼 스튜디오 C 프로젝트 관련 파일

솔루션 폴더: ch02	프로젝트 폴더: ch02/P01-HelloWorld		
파일명.확장명	파일 이름	설명	위치
ch02.sln	솔루션	프로젝트, 프로젝트 항목 및 솔루션 항목을 솔루션으로 구성	ch02
P01-HelloWorld.vcxproj	프로젝트	비주얼 C++ 프로젝트 파일	ch02/P01-HelloWorld
01hello.c	소스	C 프로그램 소스파일	
01hello.obj	목적	컴파일되었지만 링크되지 않은 목적 파일	ch02/P01-HelloWorld/Debug
P01-HelloWorld.ilk	링크	목적파일과 사용되는 라이브러리를 합친 링크 파일	ch02/Debug
P01-HelloWorld.exe	실행	실행 가능한 실행 파일	
P01-HelloWorld.pdb	디버그	프로그램 디버깅을 위한 데이터베이스 파일	

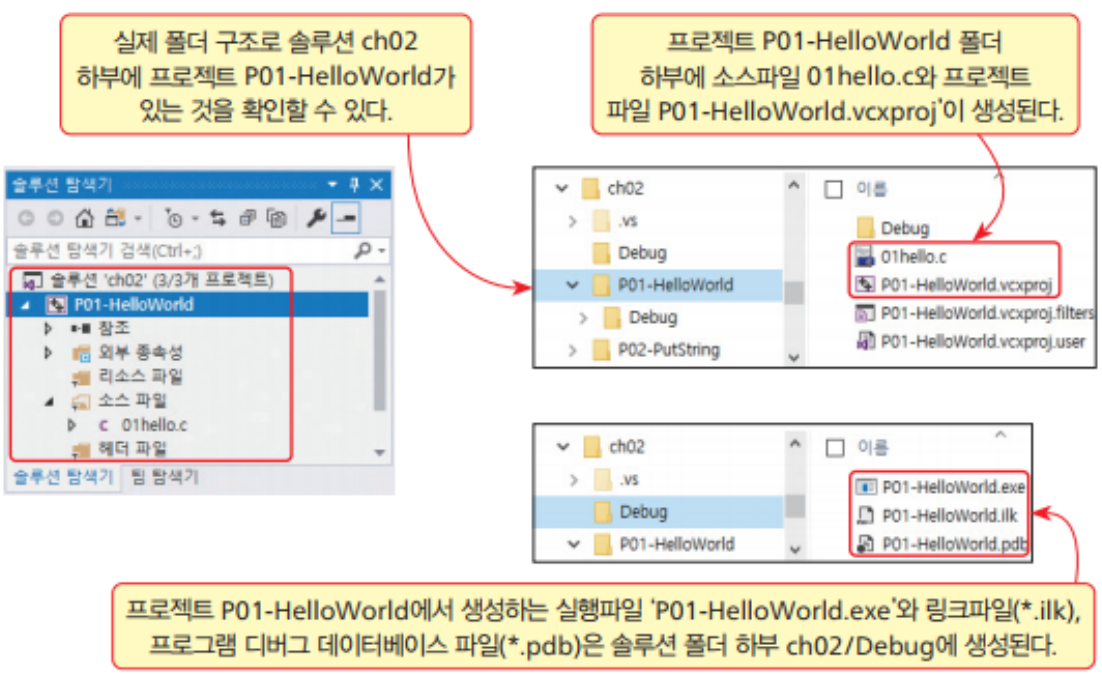


그림 2-42 솔루션 탐색기와 폴더 구조와 파일

솔루션 저장과 생성된 솔루션, 프로젝트 열기

- **프로젝트를 마치려면**
  - 메뉴 [파일] / [모두 저장] 을 누른 후 [파일] / [끝내기]를 선택
  - '모두 저장'은 아이콘 메뉴도 가능
- **다시 열기**
  - [최근 파일 열기] 목록에 표시되는 기존의 파일을 열 수 있으며
  - [시작] 표시되는 [프로젝트 또는 솔루션 열기]나 [새 프로젝트 만 들기]를 선택

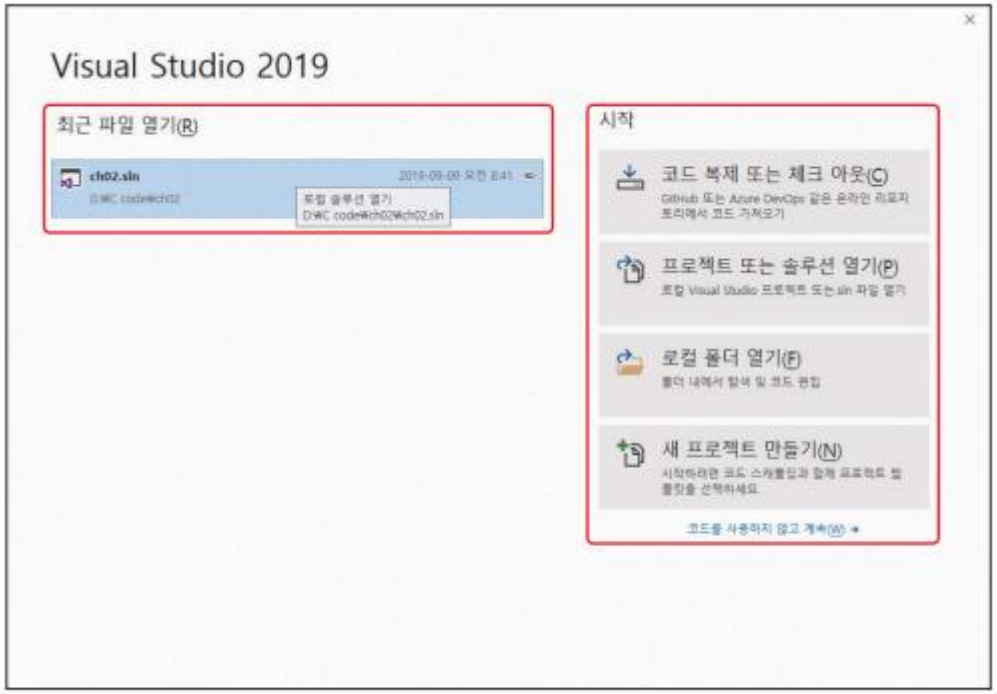


그림 3-43 비주얼 스튜디오 초기 화면

솔루션 ch01에 두 번째 프로젝트 추가

- 프로젝트 이름: 'P02-PutString'
- 소스파일: 02put.c로 편집

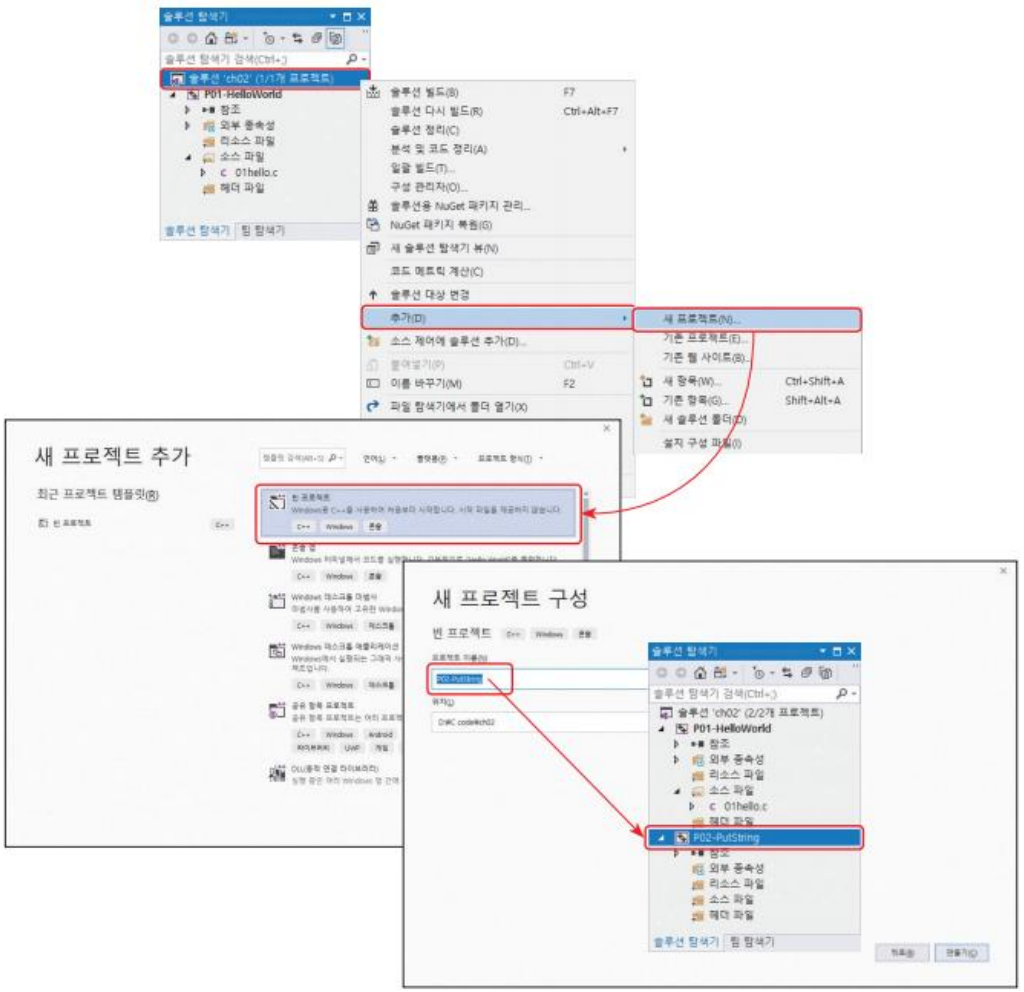


그림 2-45 새 프로젝트 P02-PutString 추가

프로젝트 P02-PutString

- **소스파일 02put.c**
  - 문자열 "hello, world"와 "C 언어, 재미있다!"를 콘솔의 두 줄에 출력하는 프로그램
  - puts(''문자열")
    - **문자열을 출력(put string)하는 기능**
  - ₩n
    - **다음 줄(new line)로 이동**

실습예제 2-2 | P02-PutString | 02put.c | 문자열 두 개를 콘솔의 두 줄에 출력 | 난이도: ★

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      printf("hello, world\n");
06      puts("C 언어, 재미있다!");
07
08      return 0;
09  }
```

void는 함수 입력이 없다는 것이며, main 뒤의 괄호 ()는 반드시 필요

문자열 Hello World!는 출력할 문자열이며, \n은 새로운 줄로 이동이라는 new line을 의미하는 문자를 표현하는데, \은 키보드에서는 ₩와 같음

결과

```
hello, world
C 언어, 재미있다!
```

여러 프로젝트 중 시작 프로젝트로 설정한 후 실행

- **메뉴 [프로젝트] / [시작 프로젝트로 설정]을 선택**
  - 또는 솔루션 탐색기에서
    - **프로젝트 P01-PutString의 우측 버튼 메뉴, [시작 프로젝트로 설정]**

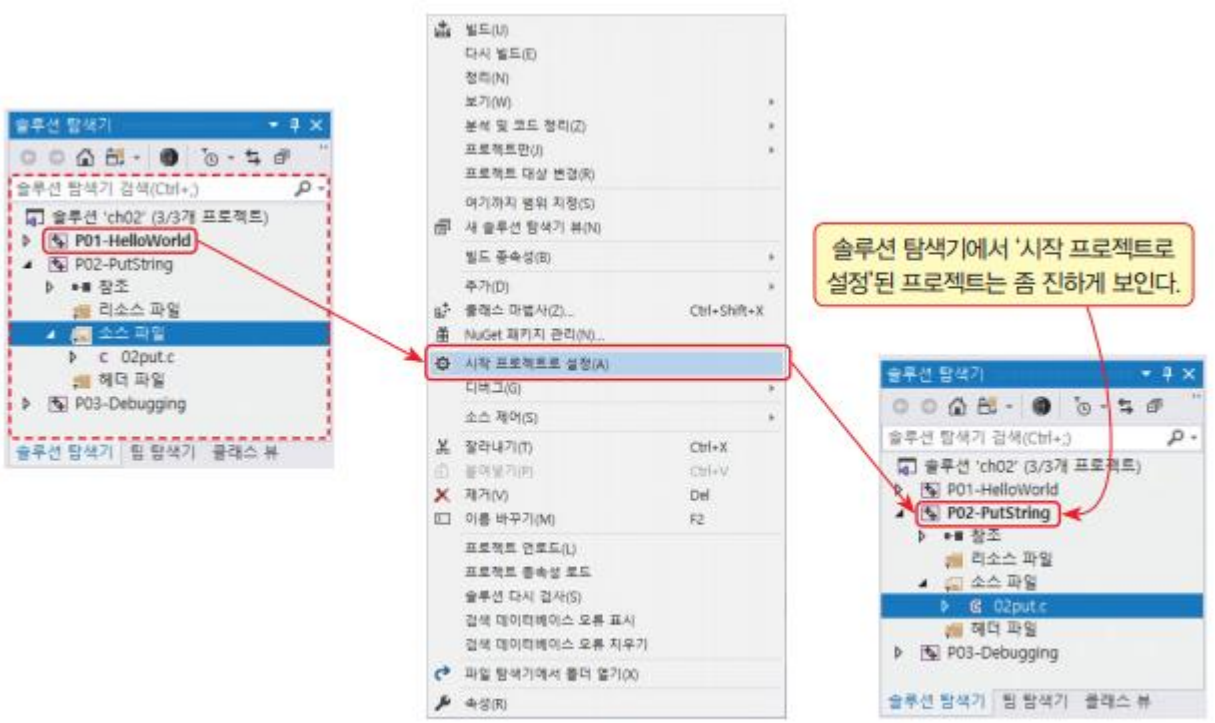


그림 2-46 실행할 프로젝트를 [시작 프로젝트로 설정]으로 활성화

헤더 파일 개요

- **함수 puts()와 printf()를 사용**
  - 첫 줄에 #include <stdio.h> 삽입

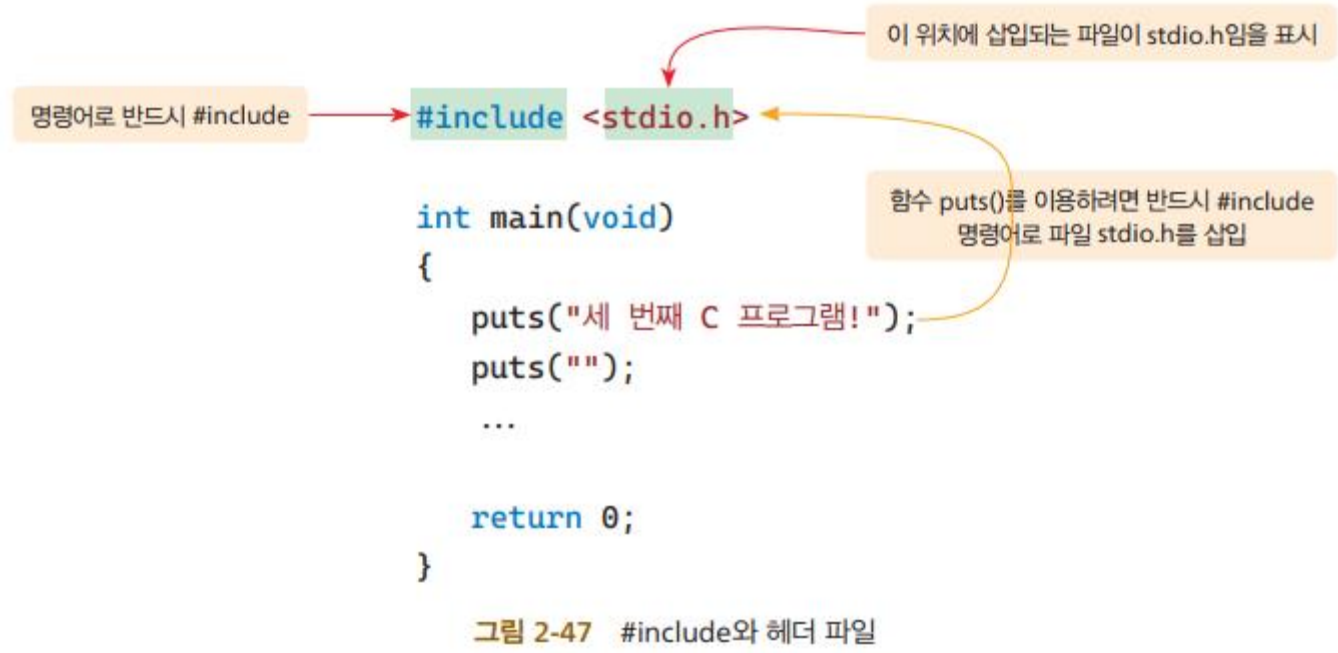
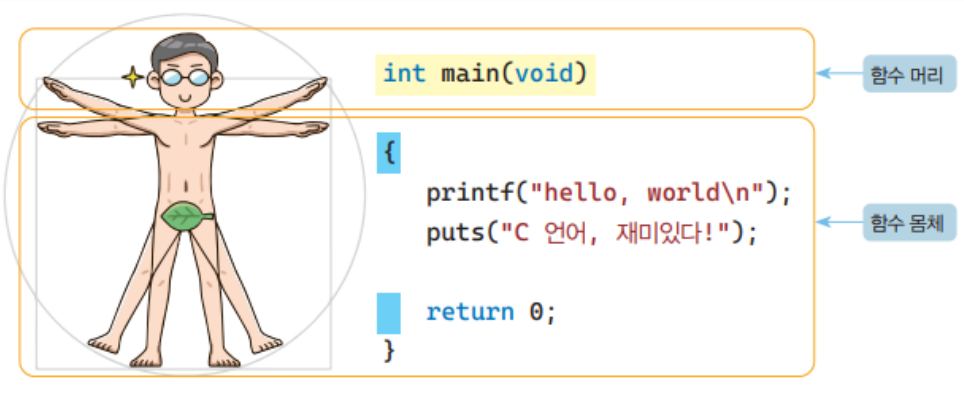
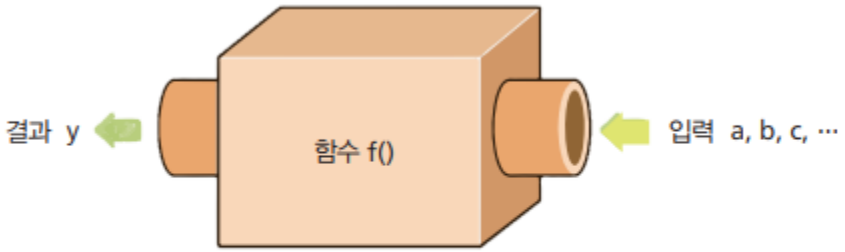


그림 2-47 #include와 헤더 파일

함수

- **C프로그램에서 main() 함수**
  - 자동차에 시동을 켜는 열쇠와 같은 역할
  - 프로그램 실행 시 가장 먼저 호출되는 특별한 함수
    - **C Runtime Startup function**
- **함수의 형태**
  - 함수 머리 (function header), 함수 몸체(function body)
  - '사용자 정의 함수(user defined function)'
    - **프로그래머가 직접 만드는 함수**
  - '라이브러리 함수(library function)'
    - **시스템이 미리 만들어 놓은 함수**





$$y = f(a, b, c, \cdots)$$

프로젝트 'P03-Debugging'

- **구문 오류가 발생하도록 의도적으로 소스에 오류가 존재**
  - 솔루션: 기존 솔루션 'ch02'
  - 프로젝트: 'P03-Debugging'
  - 소스파일: 03debug.c

실습예제 2-3 | P03-Debugging | 03debug.c | 2개의 구문 오류를 발견하고 수정 | 난이도: ★

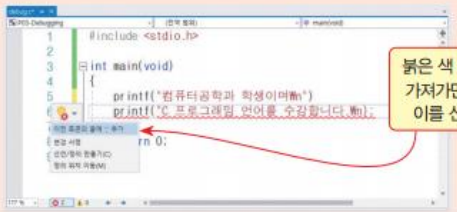
```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      printf("컴퓨터공학과 학생이며\n")
06      printf("C 프로그래밍 언어를 수강합니다.\n);
07
08      return 0;
09  }
```

문장 종료를 나타내는 ;이 빠짐

문자열의 종료를 나타내는 "(닫는 큰 따옴표)가 빠짐

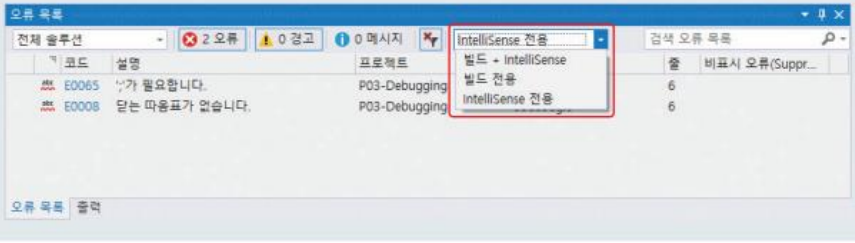
닫는 따옴표가 없습니다.
온라인 검색

오류 수정



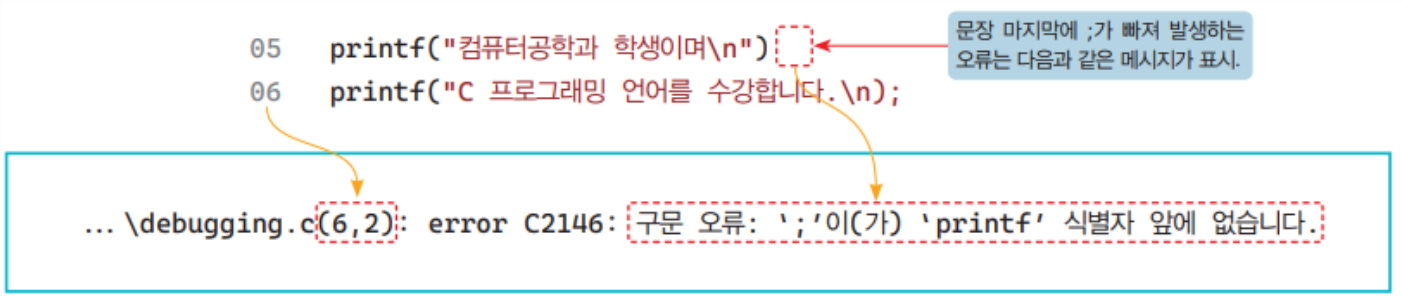
붉은 색 밑줄이 쳐진 줄 번호에 마우스를 가져가면 수정할 수 있는 항목이 있으며 이를 선택하면 오류가 자동 수정된다.

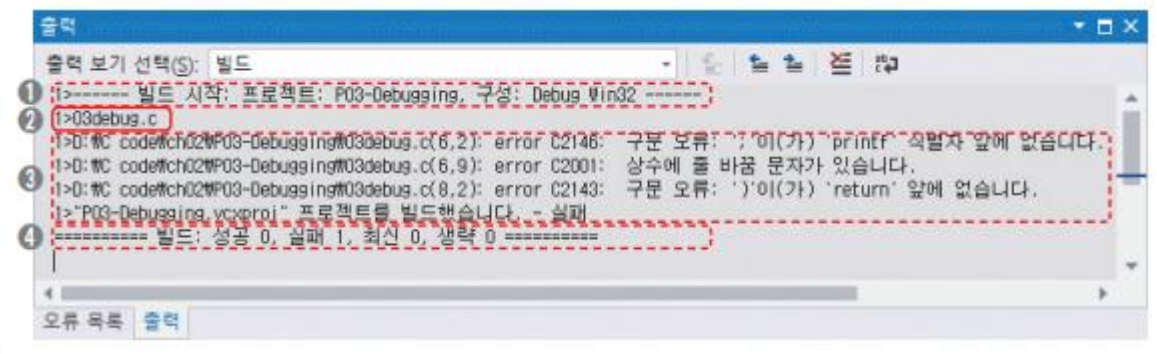
결과



구문 오류 메시지의 이해와 소스 수정

- **오류 내용**
  - 오류발생 파일의 경로를 포함한 전체이름, 추정되는 오류발생 줄과 열 번호, 오류 코드 번호, 오류 원인 메시지 등
    - **4개의 요소가 3개의 콜론(:)으로 구분되어 표시**
      - **오류가 발생한 파일이름(전체경로): D:₩C Code₩ch02₩P03-Debugging₩03debug.c**
      - **추정되는 오류발생 줄 번호와 열(column) 번호: (6,2)**
      - **오류 코드 번호: error C2146**
      - **오류 원인 메시지: 구문 오류: ';'이(가) 'printf' 식별자 앞에 없습니다.**





다양한 구문 오류 수정

```
06    printf("C 프로그래밍 언어를 수강합니다.\n");
07
08    return 0
```

문자열의 마지막을 알리는 "이 빠져 생기는 컴파일 오류

문자열의 마지막을 알리는 "이 빠져,);도 문자열로 인식하여 아직 printf()가 종료되지 않은 것으로 인식

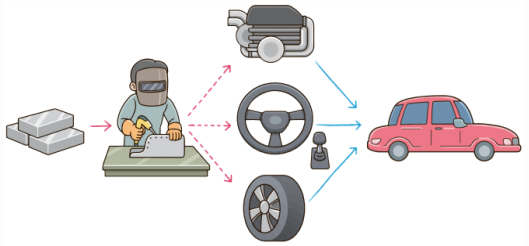
```
... \03debug.c(6,9): error C2001: 상수에 줄 바꿈 문자가 있습니다.
... \03debug.c(8,2): error C2143: 구문 오류 : ')'이(가) 'return' 앞에 없습니다.
```

표 2-3 다양한 컴파일 오류와 오류 풍선, 오류 원인 메시지

오류	바른 입력	오류 풍선	오류 원인 메시지
#incude	#include	인식할 수 없는 전처리 지시문입니다.	'incude' 전처리기 명령이 잘못되었습니다.
stdi.h	stdio.h	파일 소스를 열 수 없습니다. "stdi.h"	포함 파일을 열 수 없습니다. 'stdi.h': No such file or directory
inte	int	식별자 "inte"가 정의되어 있지 않습니다.	error C2061: 구문 오류 : 식별자 'main' error C2059: 구문 오류 : ';' error C2059: 구문 오류 : '형식'
retun	return	식별자 "retun"이 정의되어 있지 않습니다.	error C2065: 'retun' : 선언되지 않은 식별자입니다. error C2143: 구문 오류 : ';'이(가) '상수' 앞에 없습니다.
{ 빠짐	{	{가 필요합니다.	error C2059: 구문 오류 : ';' error C2059: 구문 오류 : '문자열' error C2143: 구문 오류 : ')'이(가) '문자열' 앞에 없습니다. error C2143: 구문 오류 : '{'이(가) '문자열' 앞에 없습니다.
} 빠짐	}	표시되지 않음	왼쪽 중괄호 '{'(위치: '…\03debug.c(4)')이(가) 짝이 되기 전에 파일의 끝이 나타났습니다.

링크 오류 수정

- **부품을 조립하는 링크 과정에서 발생하는 오류가 '링크 오류'**
- **대표적인 링크 오류**
  - 라이브러리 함수인 printf()의 철자를 잘못 기술하는 경우
    - **마지막 f를 빼고 print()로 기술**



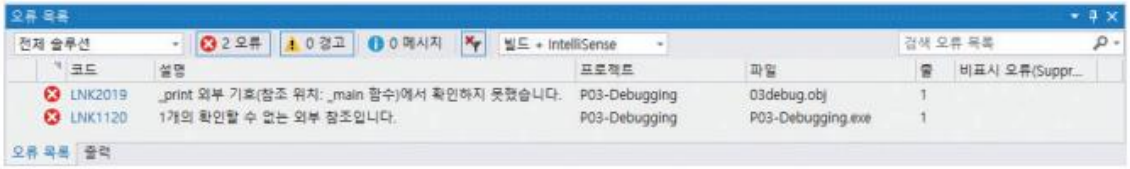


그림 2-55 함수 이름이 잘못된 컴파일 오류의 경고

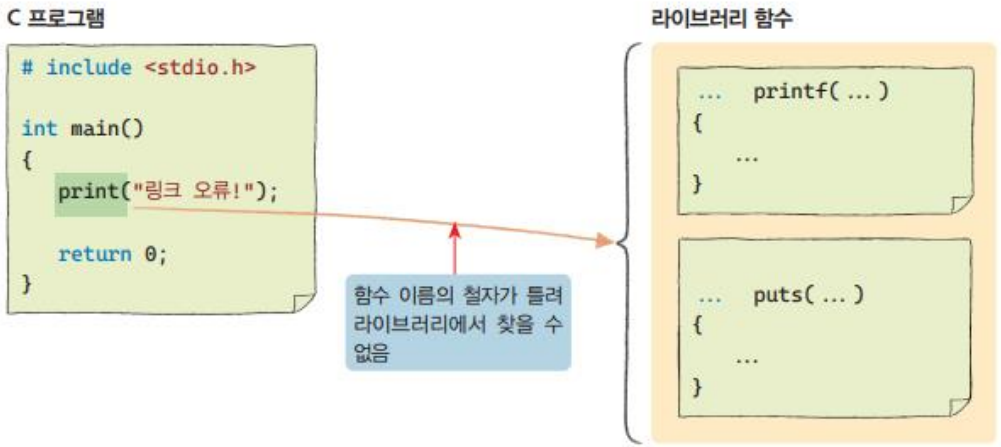


그림 2-56 링크 오류 결과

논리 오류 수정

- **논리 오류 (logical error) 예**
  - 교과목의 성적을 산출하는 프로그램
    - **잘못된 성적 결과**
- **복잡하고 큰 규모의 소프트웨어의 개발에서**
  - 다양한 문제로 발생하는 논리 오류는 찾기가 매우 어려움
    - **결국 가능한 한 프로그램의 문제해결 절차인 알고리즘을 잘 만든 후**
    - **이를 준수해서 소스를 코딩해야 논리 오류가 적은 프로그램을 완성**

```c
# include <stdio.h>

int main()
{
   printf("무지 개같은 내 친구!");

   return 0;
}
```

그림 2-57 문자열 출력의 논리 오류 예

감사합니다.