제02장

C 프로그래밍 첫걸음

Perfect C

C언어로 배우는 프로그래밍 기초

단원 목표

학습목표

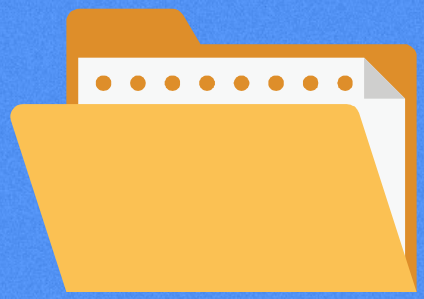
- **프로그램 구현과정과 통합개발환경을 설명할 수 있다.**
  - 프로그램 구상, 소스편집, 컴파일, 링크, 실행 과정
  - 통합개발환경의 필요 소프트웨어
- **비주얼 스튜디오로 C 프로그램을 개발할 수 있다.**
  - 솔루션과 프로젝트의 생성, 소스 생성
  - 빌드와 실행
- **함수 puts()와 printf()로 구성되는 C 프로그램을 구현할 수 있다.**
  - 함수의 요소와 이해
  - 솔루션에 여러 개의 프로젝트 생성
  - 오류의 종류와 원인 파악
  - 오류 발생에 따른 디버깅 과정

학습목차

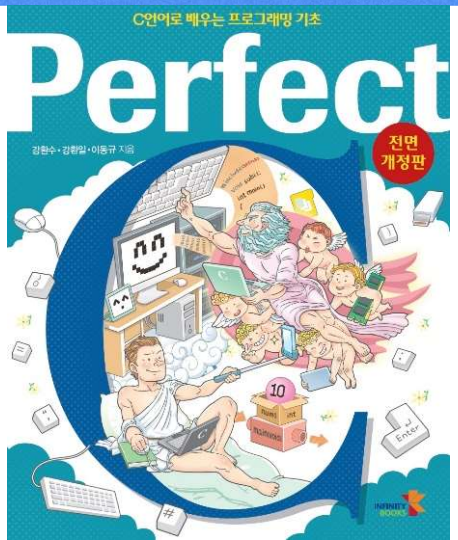
2.1 프로그램 구현 과정과 통합개발환경

2.2 비주얼 스튜디오 설치와 C 프로그램의 첫 개발

2.3 C 프로그램의 이해와 디버깅 과정



01. 프로그램 구현 과정과 통합개발환경



프로그램 구현 과정 5단계(1)

프로그램 구현 과정

프로그래밍 언어는 C로 선정 가정

- 프로그램을 구현하기 위해서는 프로그램구상, 소스편집, 컴파일, 링크, 실행의 5단계
- 간단한 프로그램은 하나의 소스파일로 구성, 프로그램이 커진다면 여러 개의 소스파일로 구성하는 것이 효율적

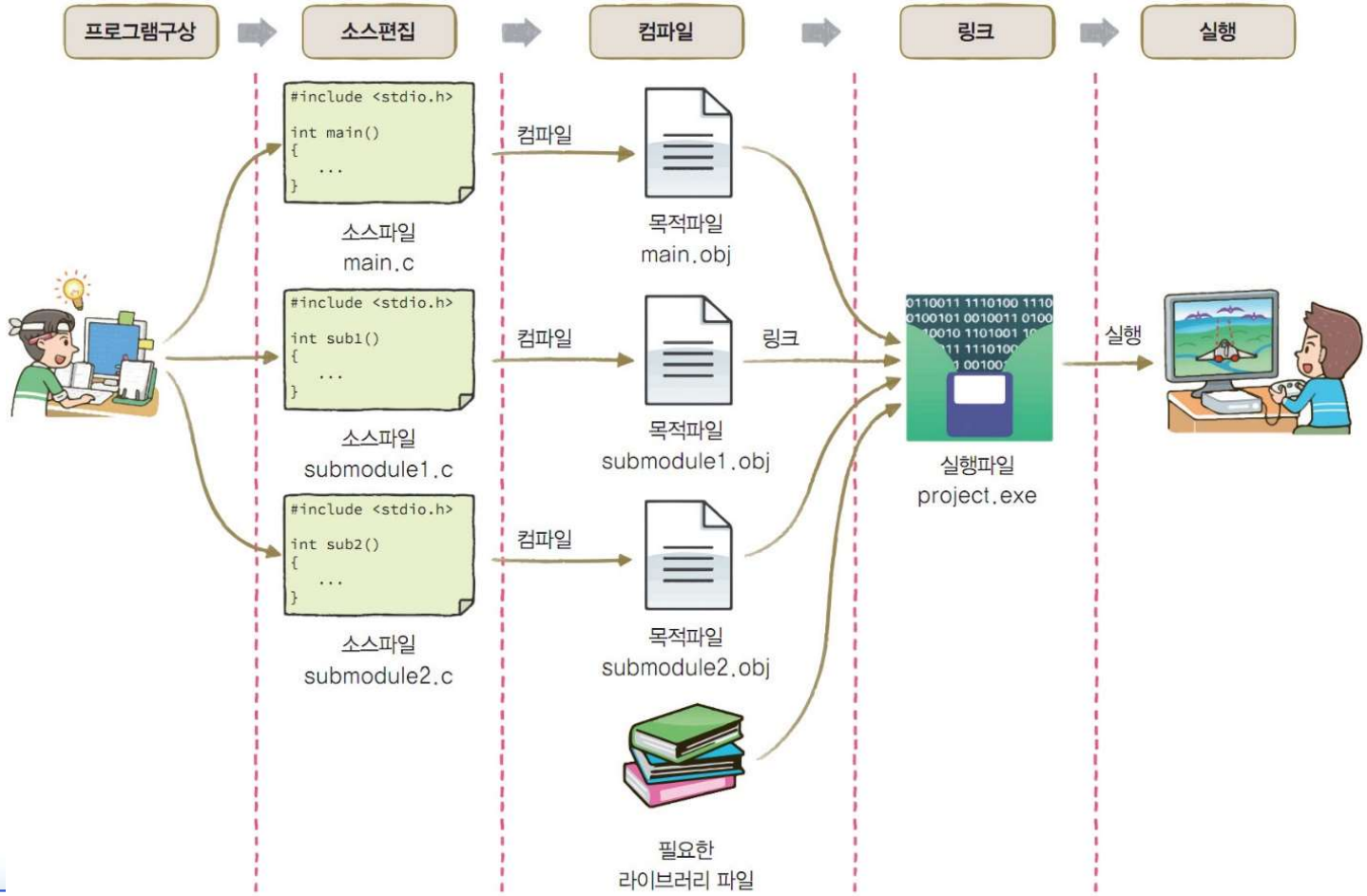


그림 2-1 C 프로그램 구현 과정

프로그램 구현 과정 5단계(2)

프로그램 구상과 소스편집

- 소스코드는 선정된 프로그래밍 언어인 C 프로그램 자체로 만든 일련의 명령문을 의미
- 소스파일(source file)
  - C와 같은 프로그래밍 언어로 원하는 일련의 명령어가 저장된 파일, 텍스트파일로 저장

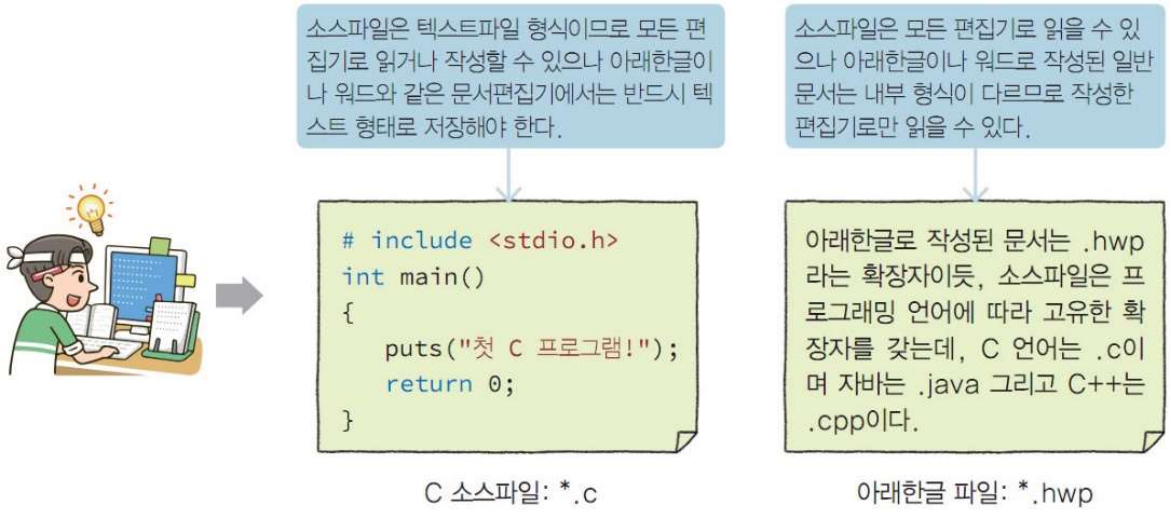


그림 2-2 소스 작성과 확장자

컴파일러

- 소스파일에서 기계어로 작성된 목적파일(object file)을 만들어내는 프로그램
- 컴파일러에 의해 처리되기 전의 프로그램을 소스코드(source code)라면 컴파일러에 의해 기계어로 번역된 프로그램은 목적코드(object code)

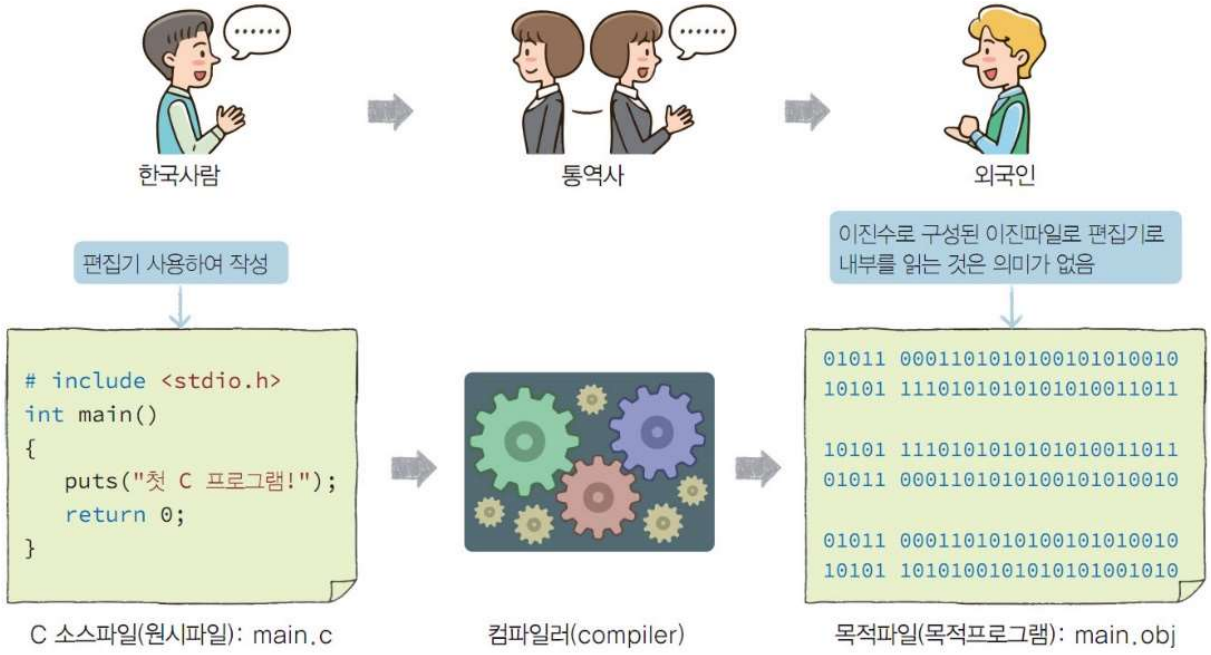


그림 2-3 컴파일의 이해

프로그램 구현 과정 5단계(3)

링크와 실행

- 링커(linker)
  - 하나 이상의 목적파일을 하나의 실행파일(execute file)로 만들어 주는 프로그램
  - 여러 개의 목적파일을 연결하고 참조하는 라이브러리를 포함시켜 하나의 실행파일을 생성
- 라이브러리(library)
  - 자주 사용하는 프로그램들은 프로그램을 작성할 때, 프로그래머마다 새로 작성할 필요 없이 개발환경에서 미리 만들어 컴파일해 저장해 놓는데, 이 모듈을 라이브러리(library)라 칭함

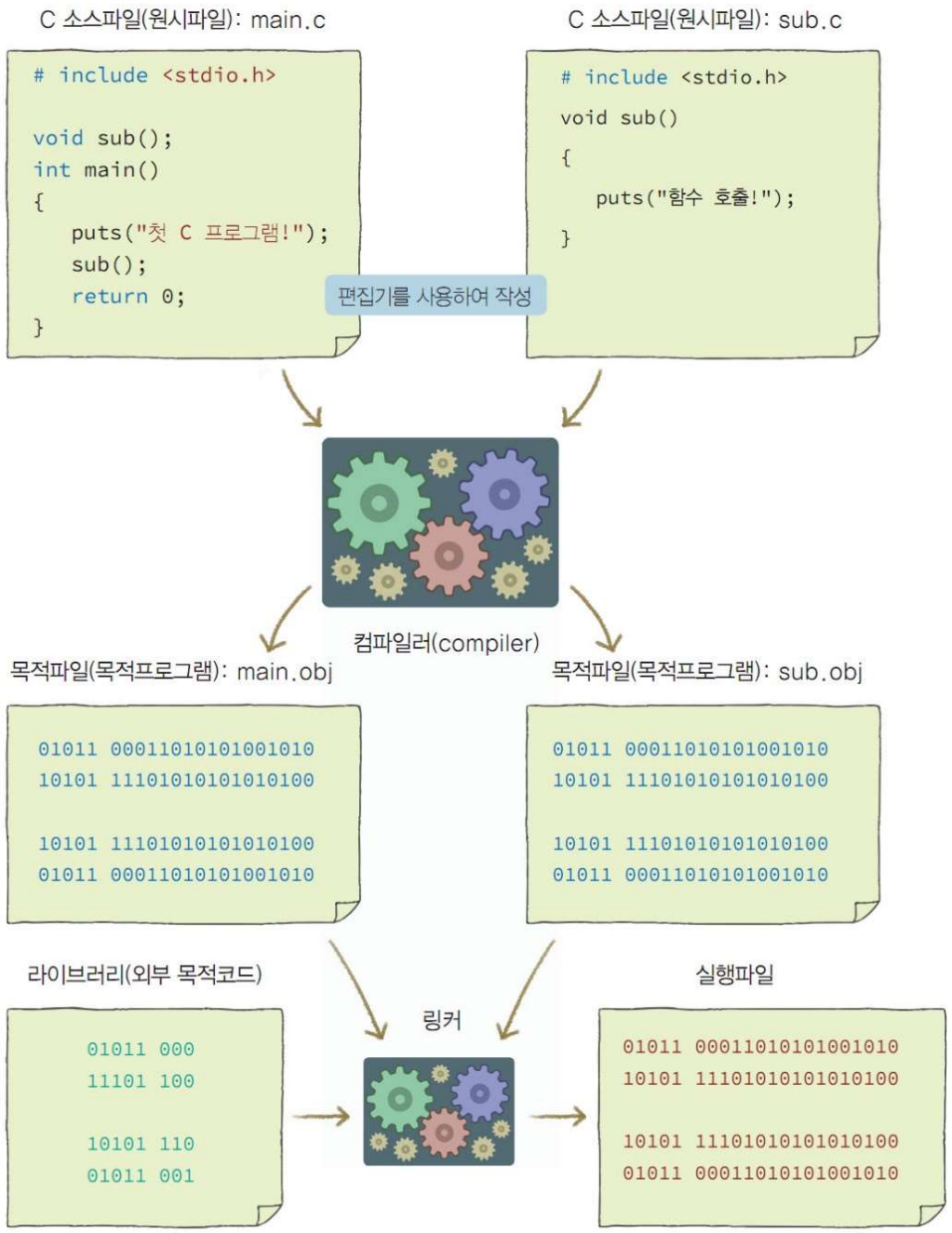


그림 2-4 링커의 이해

오류와 디버깅

- **오류 또는 에러(error)**
  - 프로그램 개발 과정에서 나타나는 문제
  - 발생시점에 따른 분류
    - 컴파일 오류
      - 오류 수정하기가 비교적 쉬움
    - 링크 오류
      - 컴파일 오류보다 상대적으로 적음
      - main() 함수 이름이나 라이브러리 함수 이름을 잘못 기술하여 발생
    - 실행 오류
      - 실행하면서 오류가 발생해 실행이 중지되는 경우
      - 문법적인 문제가 실행 오류까지 영향을 미치기도 함
  - 오류의 원인과 성격에 따른 분류
    - 문법 오류(syntax error)
      - 문법을 잘못 기술
    - 논리 오류(logic error)
      - 내부 알고리즘이 잘못되거나 원하는 결과가 나오지 않은 등의 오류

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

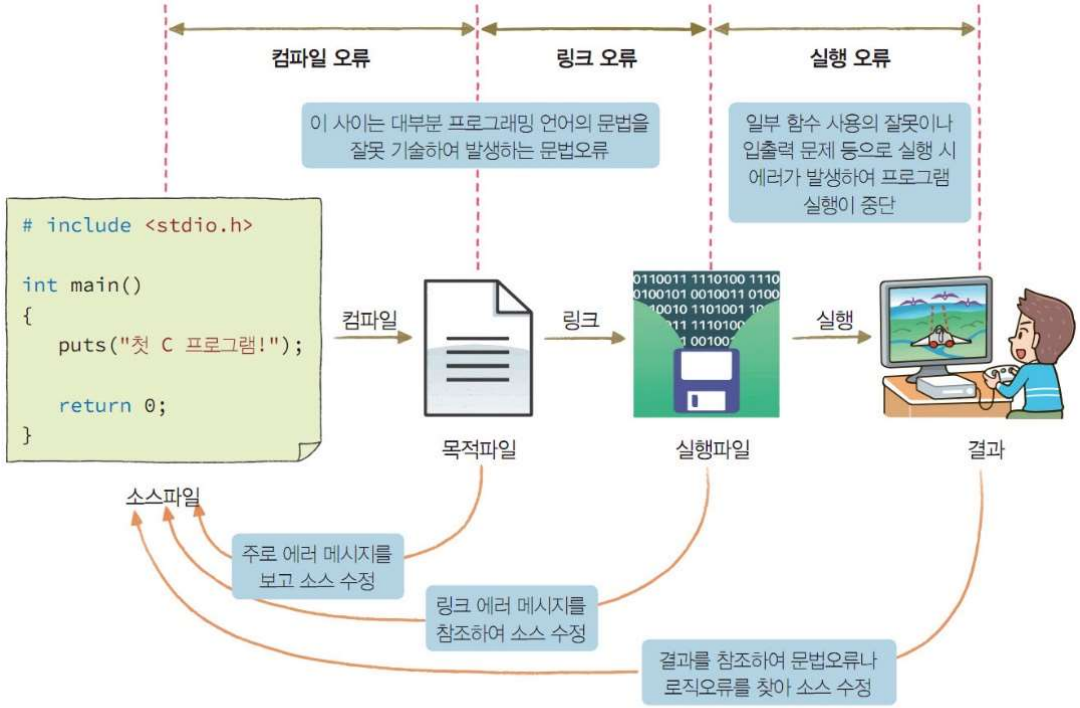


그림 2-5 디버깅의 순환과정

프로그램 구현 과정 순서도

- **컴파일, 링크, 실행 시 오류가 발생**
  - 대부분 소스 코드를 수정해서 다시 컴파일, 링크, 실행

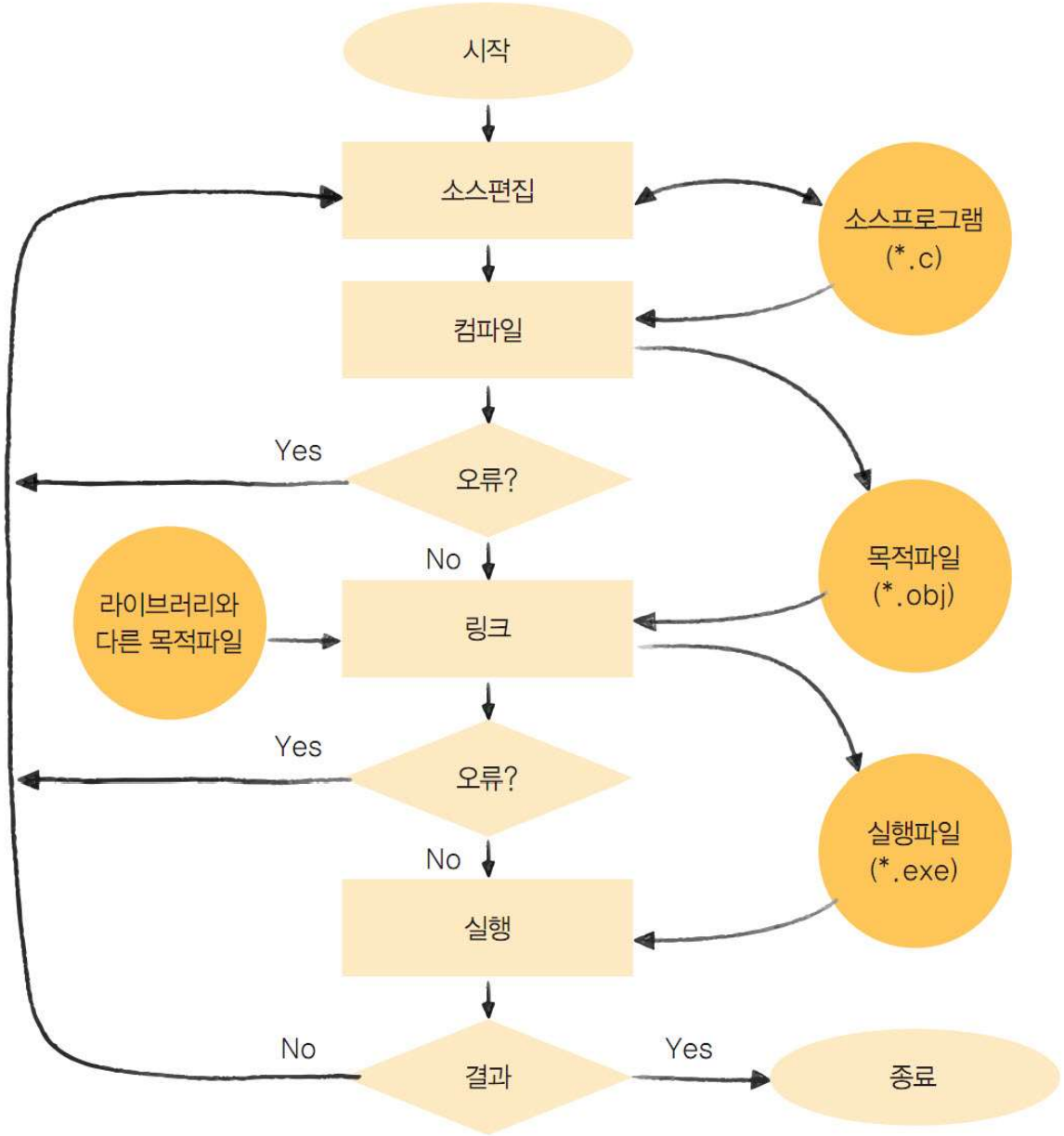


그림 2-9 프로그램 구현 과정 순서도

개발도구와 통합개발환경 IDE

- **패기지 해외여행**
  - 해외여행에 필요한 각종 예약 및 정보를 일일이 알아볼 필요가 없이
  - 패키지 여행만 따라다니면 해결
- **통합개발환경, IDE(Integrated Development Environment)**
  - 프로그램 개발에 필요한 편집기(editor), 컴파일러(compiler), 링커(linker), 디버거(debugger) 등을 통합하여 편리하고 효율적으로 제공하는 개발환경

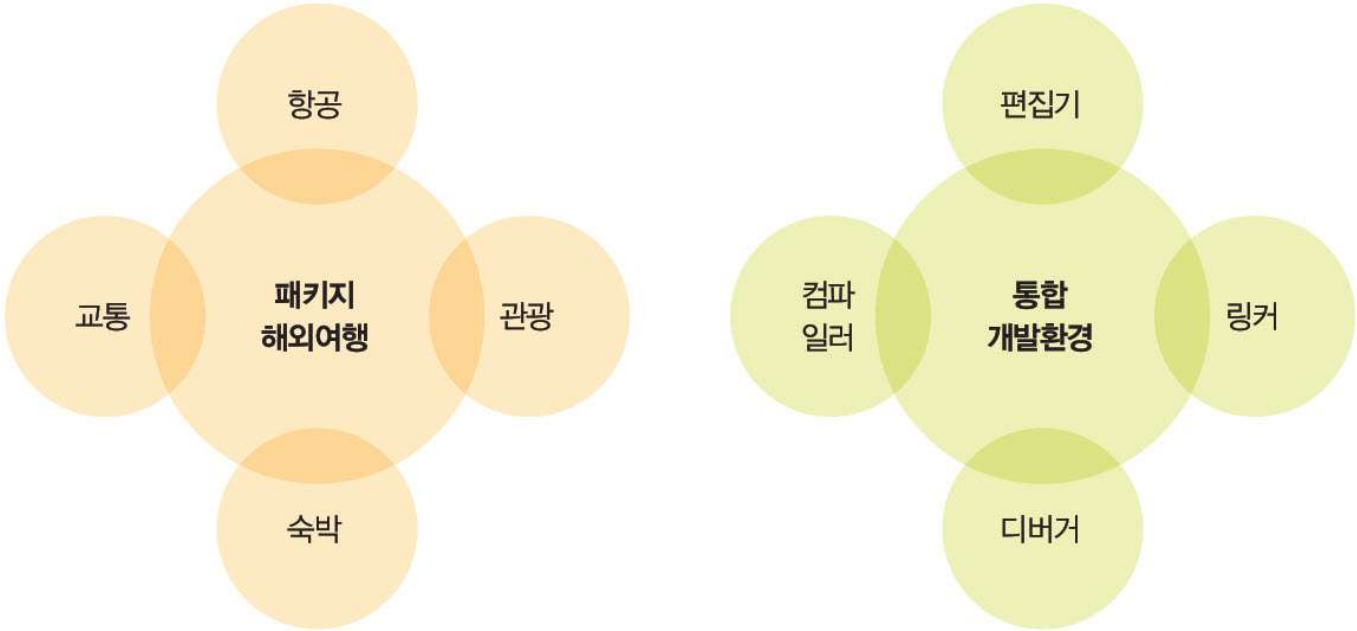


그림 2-10 통합개발환경의 이해

대표적인 통합개발환경(IDE)

- **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
- **이클립스 C/C++ 개발자용 IDE**
  - IBM이 주도하는 이클립스 컨소시엄이 개발
  - 모든 부분에 대해 개방형
    - PDE(Plug-in Development Environment) 환경을 지원하여 확장 가능한 통합개발환경
  - C/C++ 개발자용 IDE(Eclipse IDE for C/C++ Developers)
    - C/C++를 개발하기 위한 개발도구로 컴파일러는 따로 설치
    - C/C++ 컴파일러로는 주로 공개 모듈인 GNU의 GCC(GNU Compiler Collection)를 이용

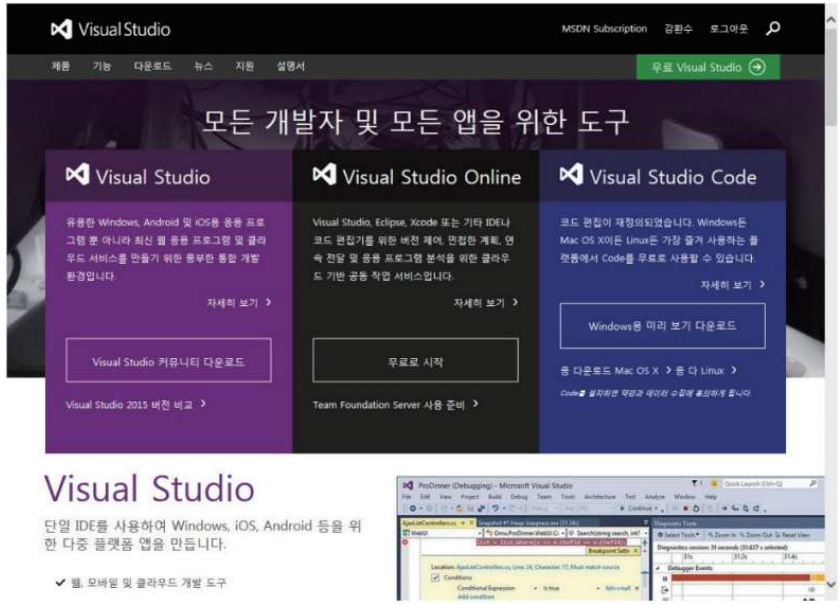


그림 2-12 비주얼 스튜디오의 홈페이지(www.visualstudio.com)

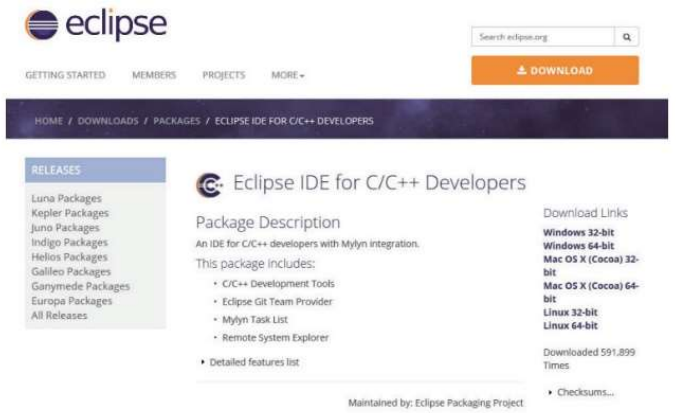
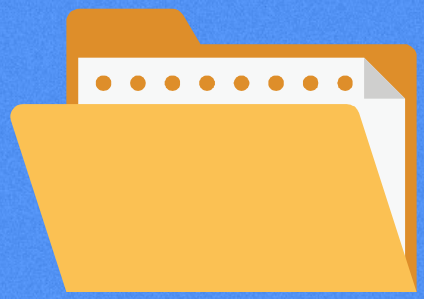
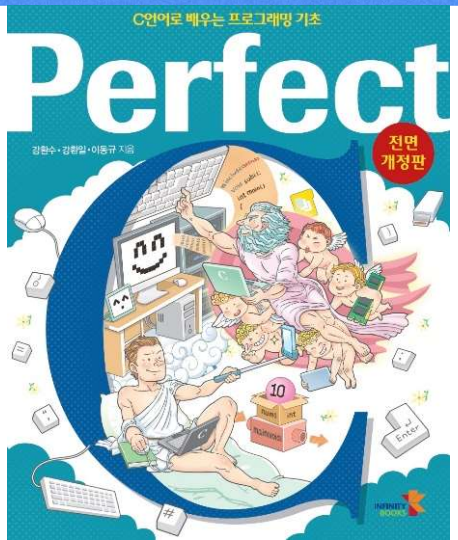


그림 2-14 C/C++ 개발자를 위한 이클립스 IDE 내려받기 페이지 (www.eclipse.org/downloads/packages/eclipse-ide-cc-developers)



02. 비주얼 스튜디오 설치와 C 프로그램의 첫 개발



비주얼 스튜디오 커뮤니티 설치와 이해

내려 받기와 설치, 실행

내려 받기

- 홈페이지(www.visualstudio.com)의 [Visual Studio Community 2015 다운로드]
- 또는 우측상단의 [무료 Visual Studio S]를 누르면 페이지가 표시

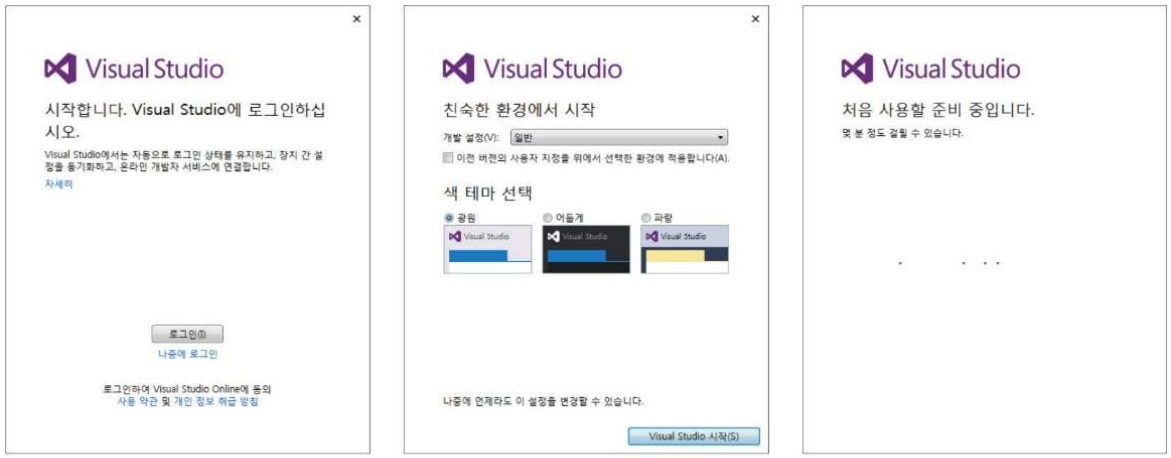


그림 2-20 비주얼 스튜디오 커뮤니티 2015 설치 후 진행과정

설치와 실행

- 비주얼 스튜디오 커뮤니티 버전의 실행이 시작된 후 표시되는 대화상자에서 [표준 설치] 설치 유형을 선택

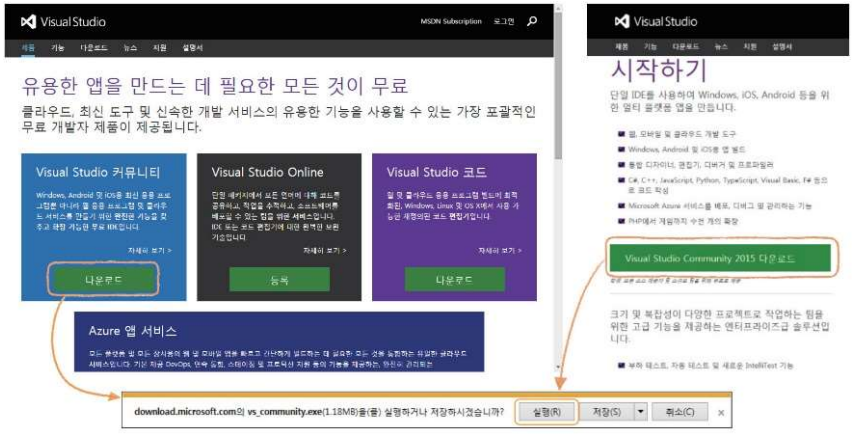


그림 2-18 무료 버전인 비주얼 스튜디오 커뮤니티 2015 내려 받기

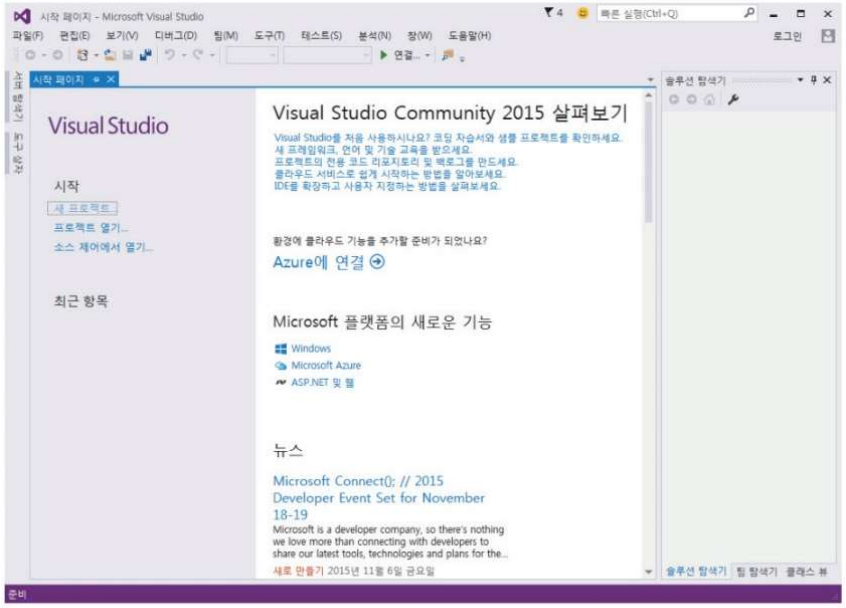


그림 2-21 비주얼 스튜디오 커뮤니티 2015 첫 화면

비주얼 스튜디오의 솔루션과 프로젝트 생성

가장 먼저 프로젝트 만들기

새 프로젝트 메뉴 선택

- 비주얼 스튜디오 첫 화면의 [시작 페이지] 하단에서 바로 [새 프로젝트...] 연결을 누르거나
- 주 화면에서 메뉴 [파일] -> [새로 만들기] -> [프로젝트]를 연이어 선택

그림 2-22 비주얼 스튜디오의 [새 프로젝트…] 연결

새 프로젝트 대화상자

- 왼쪽 '템플릿' 하단 'Visual C++'를 선택
- 누락된 기능이 있으면 종료 후 설치

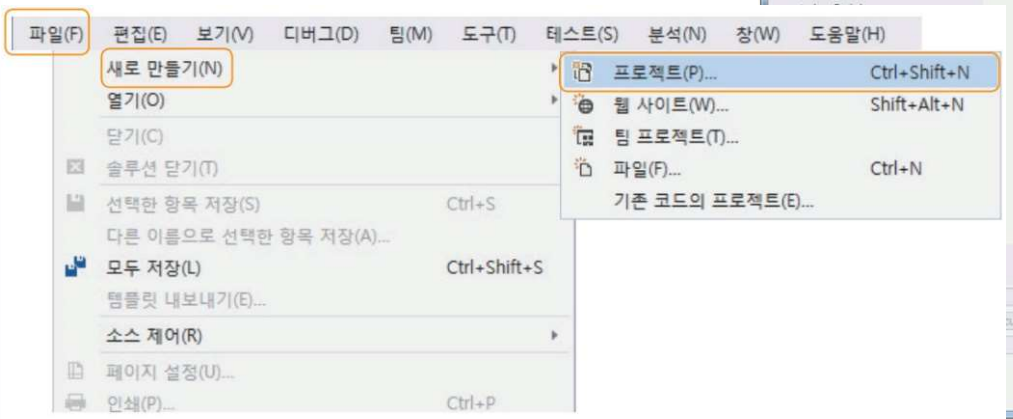


그림 2-23 메뉴 [파일] → [새로 만들기] → [프로젝트]를 선택

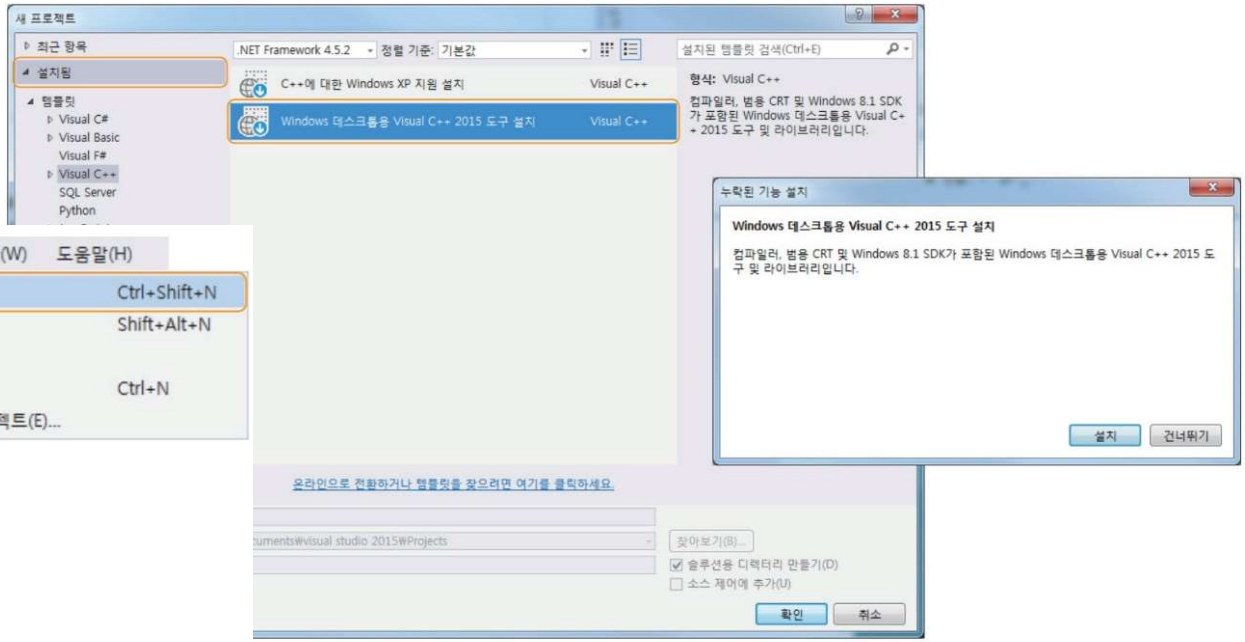


그림 2-24 미비된 Windows 데스크톱용 Visual C++ 2015 도구 설치

솔루션과 프로젝트를 생성하기 위한 여러 설정값

- **프로젝트 형식**
  - 선택된 템플릿 'Visual C++'에서 'Win32 콘솔 응용프로그램'으로 선택
  - '솔루션 이름'
    - 단원이름 'Ch02'을 지정
  - '이름'
    - 프로젝트 'First C Project'로 지정
  - '위치'
    - 솔루션과 프로젝트 관련 폴더와 여러 파일이 저장될 상위 폴더
    - '위치'에 지정되는 폴더는 없는 경우 자동으로 생성

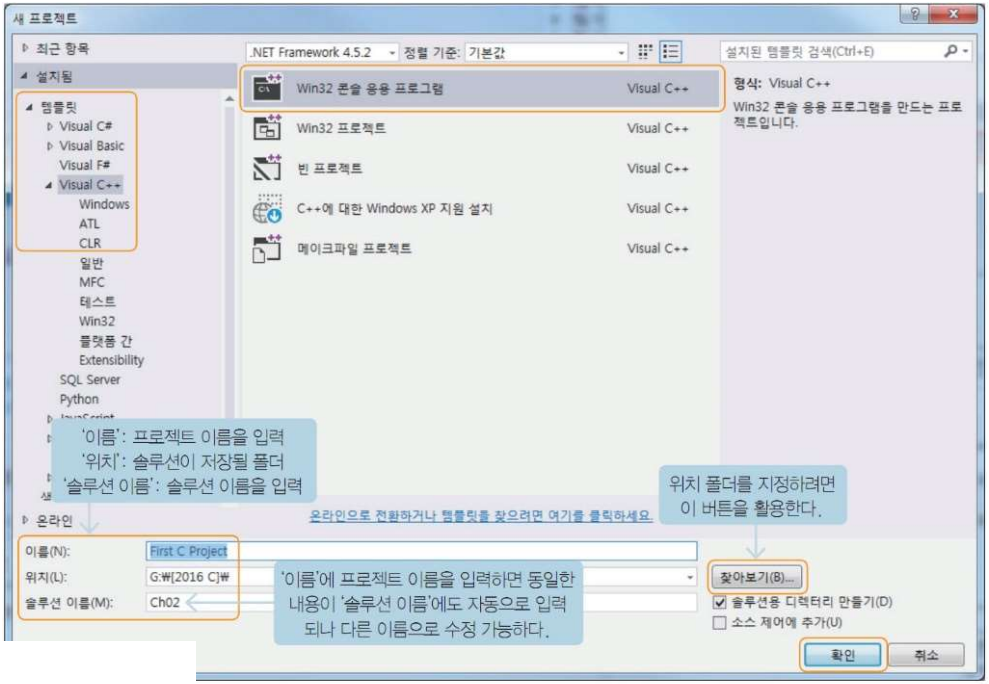


그림 2-26 그림 [새 프로젝트] 대화상자 주요 설정 내용 지정

표 2-1 [새 프로젝트]의 여러 설정 내용

주요 설정	설명	설정 내용
템플릿	개발하려는 환경	Visual C++
프로젝트 형식	다양한 프로젝트 형식 중에 하나 선택	Win32 콘솔 응용 프로그램
이름	만들려는 프로젝트 이름을 입력	First C Project
위치	솔루션과 프로젝트가 저장되는 폴더	G:\[2016 C]
솔루션 이름	만들려는 솔루션 이름을 입력	Ch02
솔루션을 디렉토리로 만들기	솔루션을 폴더로 지정하려면 체크	체크

응용 프로그램 설정

- **응용 프로그램의 종류와 옵션을 지정**
  - 대화상자 'Win32 응용 프로그램 마법사'의 첫 'Win32 응용 프로그램 마법사 시작' 화면
    - [마침]을 선택하지 말고 [다음]을 누르면
  - 프로그램의 종류
    - [콘솔 응용 프로그램]으로 지정
  - 추가 옵션
    - 초기에 기본적으로 해제되어 있는 [빈 프로젝트]를 선택
  - 다시 한번 확인한 후 [마침]을 누르면
    - 프로젝트와 솔루션이 생성되고 주 화면에 표시

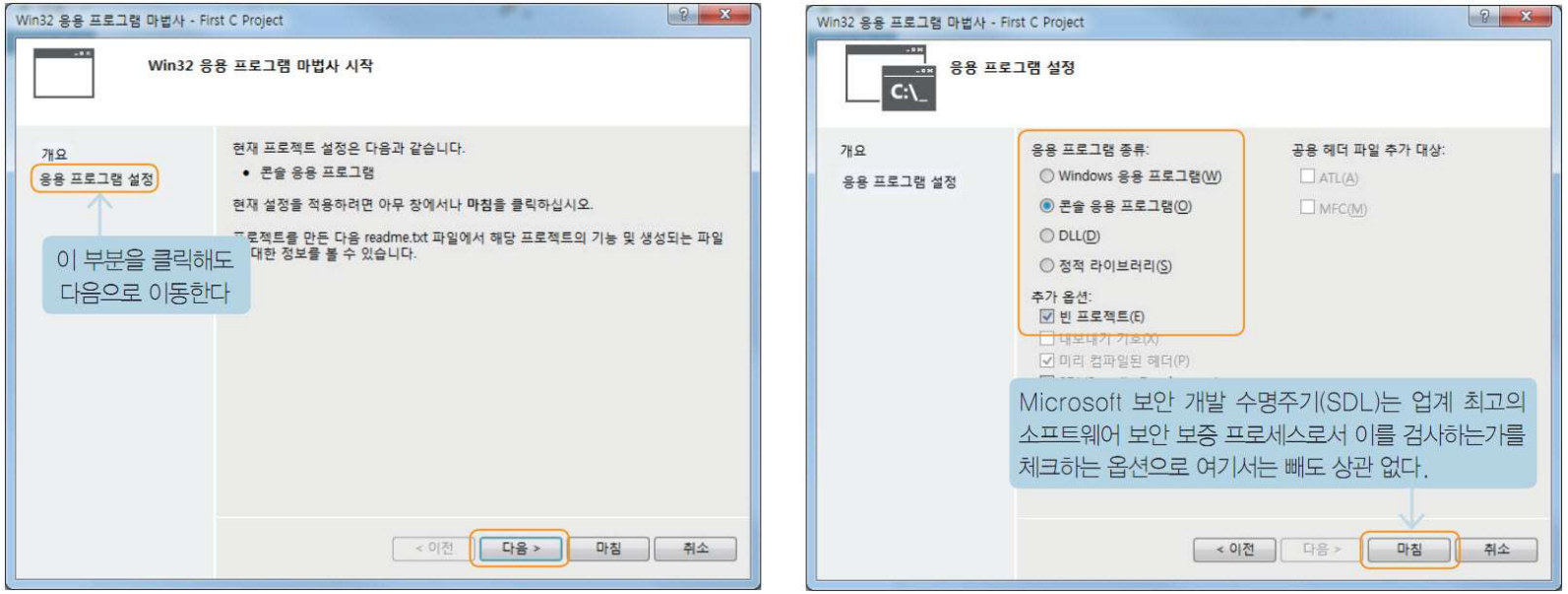


그림 2-27 대화상자 [Win32 응용 프로그램 마법사]의 두 화면

솔루션 탐색기

- **주 화면 오른쪽 '솔루션 탐색기'**
  - 생성된 솔루션과 프로젝트 표시
    - 전체 솔루션의 그래픽 뷰를 제공하여 응용 프로그램을 개발할 때 솔루션의 프로젝트와 파일을 쉽게 관리할 수 있도록 도움
  - 프로젝트 하단부
    - 관련 폴더인 리소스 파일, 소스 파일, 외부 종속성, 참조, 헤더 파일로 나뉨
  - 솔루션 탐색기 하단의 속성
    - 솔루션 탐색기에서 선택한 프로젝트 'First C Project'에 대한 속성정보가 표시

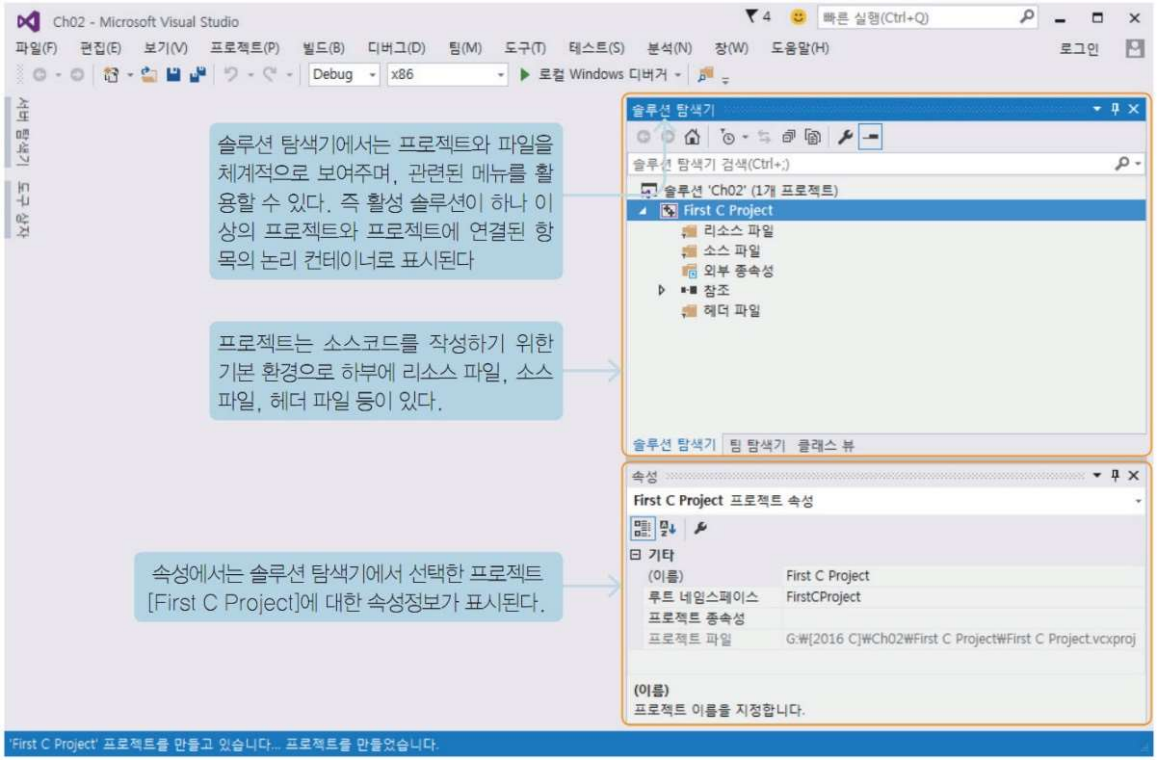


그림 2-28 솔루션과 프로젝트가 생성된 솔루션 탐색기

솔루션과 프로젝트 이해

여러 프로젝트가 모인 솔루션

단원의 예제

- 단원을 솔루션으로 만들고,
- 각각의 예제를 프로젝트로 생성하여 관리하면 매우 편리

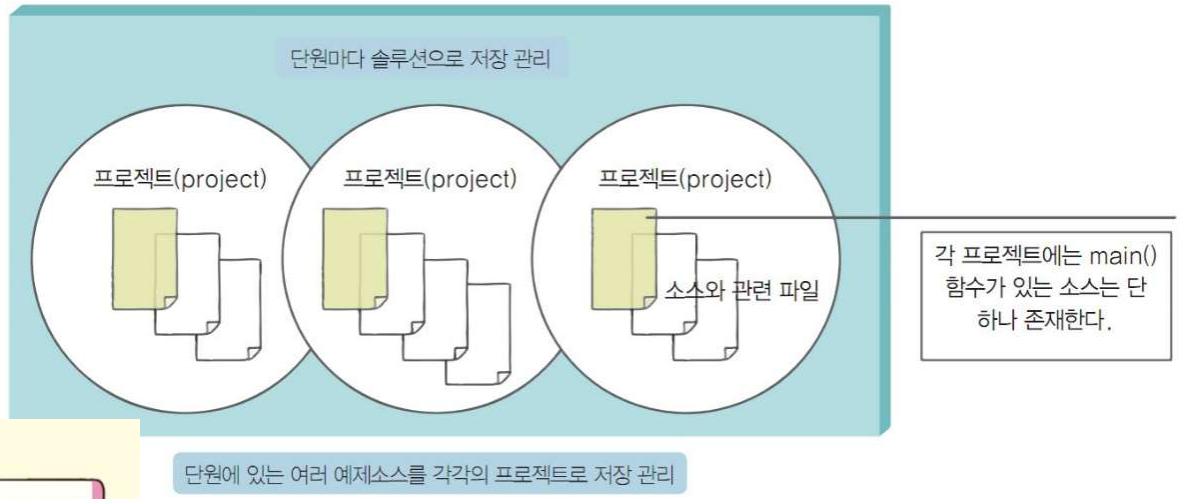


그림 2-29 단원을 솔루션으로, 예제를 프로젝트로 지정

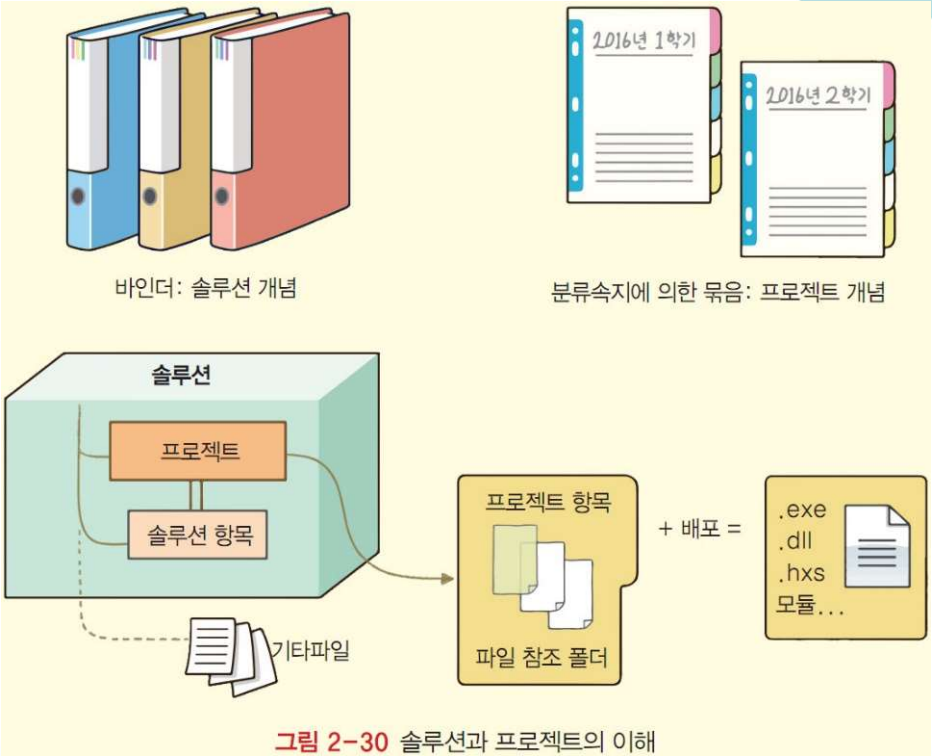


그림 2-30 솔루션과 프로젝트의 이해

'바인더'와 '분류속지'

- 여기서 '바인더'는 솔루션에 해당하고 '분류속지'는 프로젝트
- 즉 솔루션은 하나 이상의 프로젝트를 저장・관리하는 콘테이너 단위
- 프로젝트는 여러 소스와 관련 파일을 저장・관리하는 단위
  - 프로젝트 이름으로 하나의 실행파일이나 실행모듈을 생성

소스 파일 생성 편집

- **소스파일을 작성**
  - 메뉴 [프로젝트], [새 항목 추가]를 선택
  - '솔루션 탐색기'의 '소스파일' 폴더, 마우스 오른쪽을 클릭
    - 메뉴 [추가] -> [새 항목]을 선택
  - 표시된 대화상자 [새 항목 추가 – First C Project]
    - 각각 'Visual C++'와 'C++ 파일(cpp)'을 선택한 후
    - '이름'에 소스파일 이름 putstring.c를 입력
      - 파일이름에 반드시 확장자 .c를 입력
    - '위치'
      - '솔루션 폴더/프로젝트 폴더'인 'Ch01/First C Project'을 확인한 후 [추가]를 누름

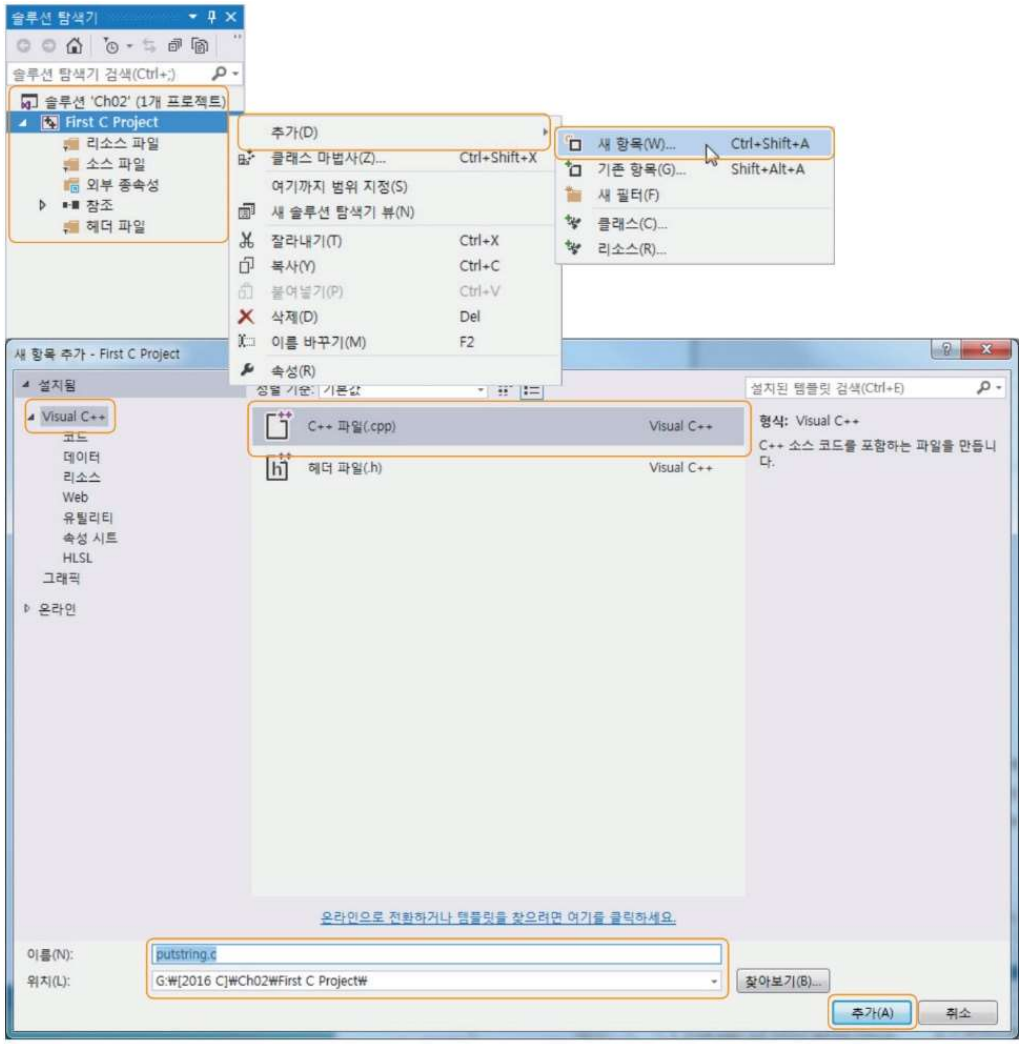


그림 2-31 소스파일 생성 방법

첫 예제 소스 간략한 이해와 작성

- **문자열 "첫 C 프로그램!"이 콘솔 창에 출력**
  - C 소스는 영문자의 대소문자를 구별
  - #, <, >, (,), ;, {, }와 같은 특별한 의미의 여러 문자들로 구성

실습예제 2-1 putstring.c

파일이름은 대소문자의 의미는 크게 없으며, 확장자 c를 붙이도록 한다.

```
01  #include <stdio.h>
02
03  int main()
04  {
05      puts("첫 C 프로그램!");
06
07      return 0;
08  }
```

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

설명

```
01  컴파일 되기 전에 일련의 작업을 지시하는 문장
02  빈 줄은 문법적인 의미는 없으나 시각적으로 필요
03  int: 함수의 반환 유형을 의미하며, 정수를 의미하는 int
03  main(): C 프로그램의 시작 부분을 알리는 약속된 단어
04  {: main() 함수의 몸체 구현 시작을 의미하며, 8줄의 몸체 구현 종료 문장 }와 대응
05  puts(): 매개변수로 입력된 문자열을 출력하는 문장
05  "첫 C 프로그램!": puts() 함수의 매개변수로, 콘솔에 출력할 문자열을 큰 따옴표로 묶음
05  문장을 종료할 때는 ;(세미콜론)을 입력
06  빈 줄은 문법적인 의미는 없으나 시각적으로 필요하면 사용
07  return: 함수의 결과값을 반환하는 문장으로 이어서 반환값을 기술
07  0: 반환하는 값으로 정수값 0을 지정하는데, 이 반환값이 정수이므로 3줄에서 int로 입력
08  }: 4줄의 함수 시작과 대응되는 함수 종료 의미
```

실행결과

```
첫 C 프로그램!
```

첫 프로그래밍에서의 주의점

- **함수 main()**
  - 대소문자로 구분하여 기술하고 중간에 공백이 들어갈 수 없으며
    - 소괄호 ( )와 중괄호 { }는 구분
    - 적당한 공백과 빈 줄은 소스의 이해력을 높이기 위해 필요
  - 소스 편집 시 입력되는 단어와 주의해야 할 문자
    - include, stdio.h, int, main, puts, return
    - # < > ( ) { } ; ""
- **컴파일러는 하나의 오타도 허용하지 않으므로**
  - 편집기에서 주의를 기울여, 행과 열을 맞추어 정확히 소스를 입력
  - 문장의 종료를 표시하는 세미콜론 ;을 콜론 :으로 잘못 입력하면 컴파일에 문제가 발생

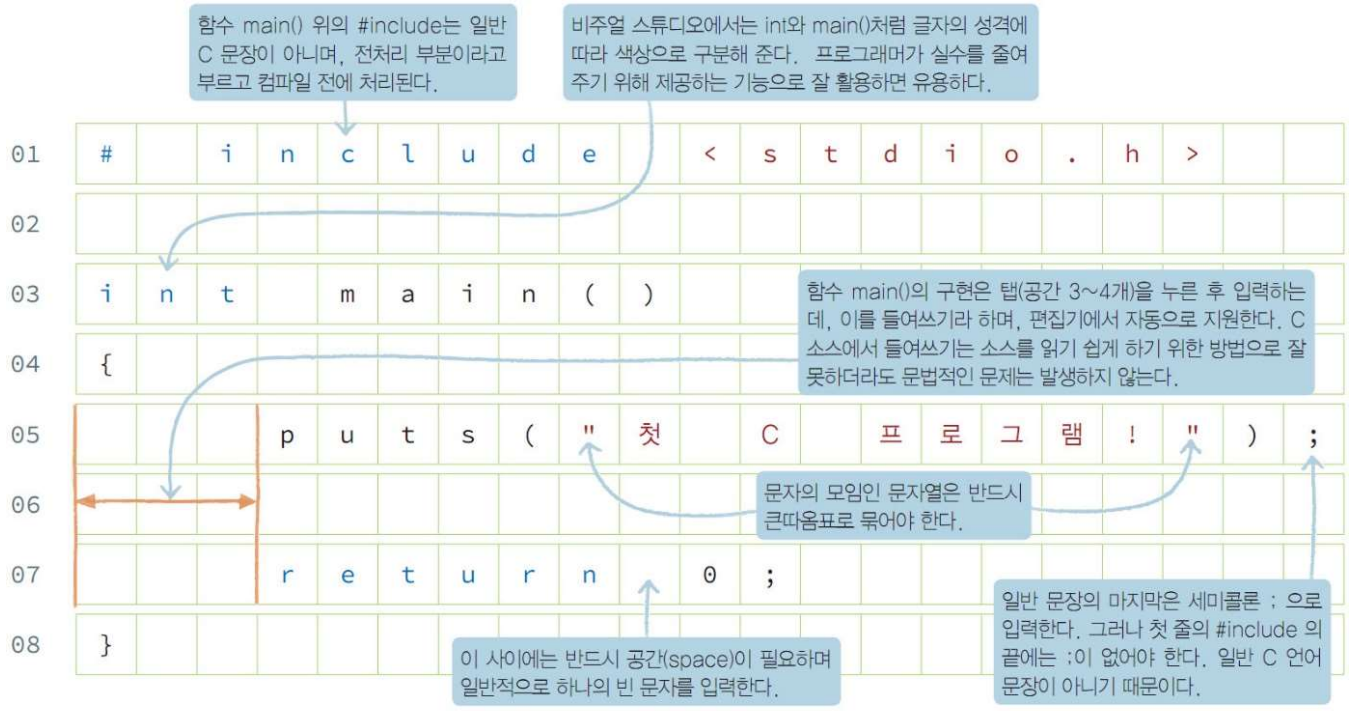


그림 2-34 소스가 기록된 원고지 모습과 주의점

프로젝트 실행

- **바로 실행 방법**
  - ❶ 작성된 소스에는 문제가 없어야 실행에 성공
  - ❷ [디버그] -> ❸ [디버깅하지 않고 시작]을 선택
  - ❹ 빌드를 묻는 대화상자가 먼저 표시, [예]를 눌러 진행
  - ❺ '출력'에 빌드 과정이 표시, 마지막 줄에 성공1, 실패 0과 같이 표시
- **빌드에 성공했다면 바로 실행결과 표시**
  - ❻ 검정색 바탕의 콘솔 화면이 표시
    - 프로젝트를 생성할 때 '콘솔 응용 프로그램'을 설정
    - 마지막 줄에 항상 '계속하려면 아무 키나 누르십시오...'라는 문장이 출력
      - 프로그램 내용과 관계없이 항상 콘솔 화면의 마지막에 표시되는 문구
      - '아무 키나 누르면 콘솔화면이 사라진다' 라는 안내문

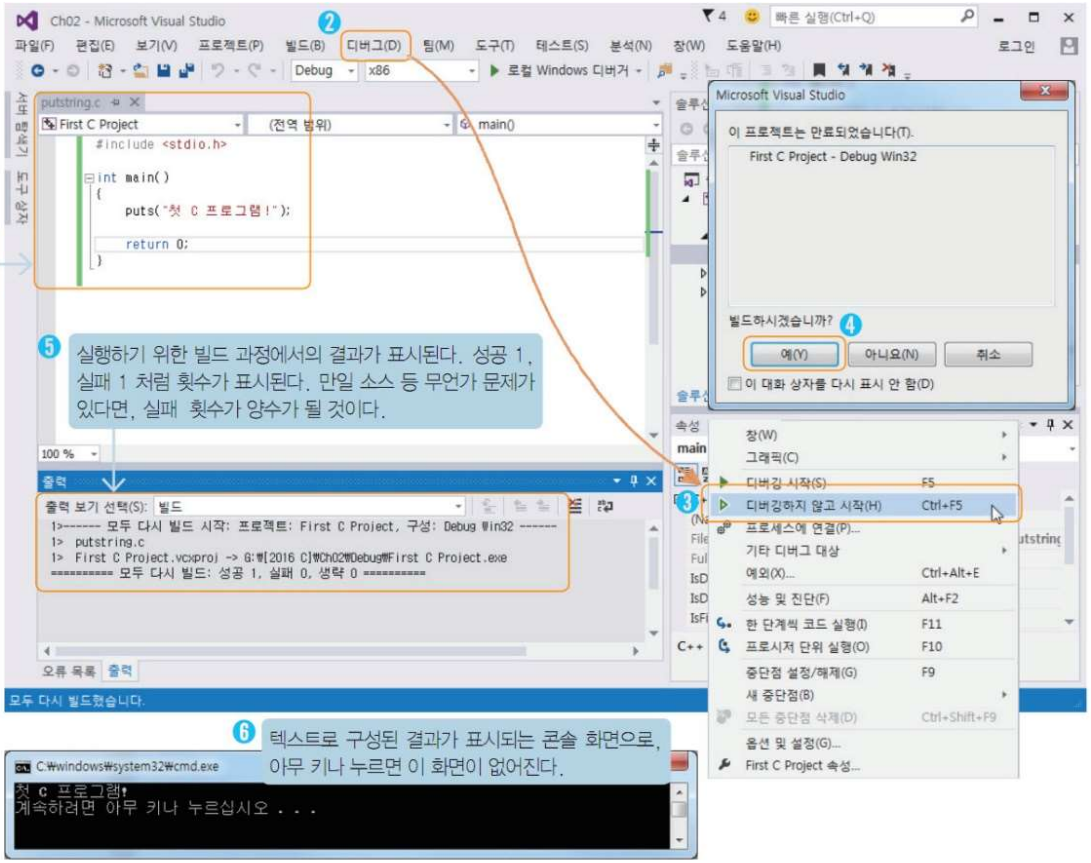


그림 2-37 프로젝트 실행을 위한 메뉴와 실행 과정, 그리고 실행 결과

컴파일과 링크

- **빌드(컴파일 + 링크) 과정을 직접 수행**
  - 메뉴 [빌드] -> [First C Project 빌드]를 선택
    - 화면 하단부의 출력 창에 빌드 과정과 그 결과가 표시
  - 메뉴 [빌드]에서 마지막 메뉴 [컴파일]을 선택하면 컴파일만 수행
  - 컴파일 후 메뉴 [빌드] -> [프로젝트만] -> [First c Project만 링크]를 선택
    - 링크만 구분하여 실행

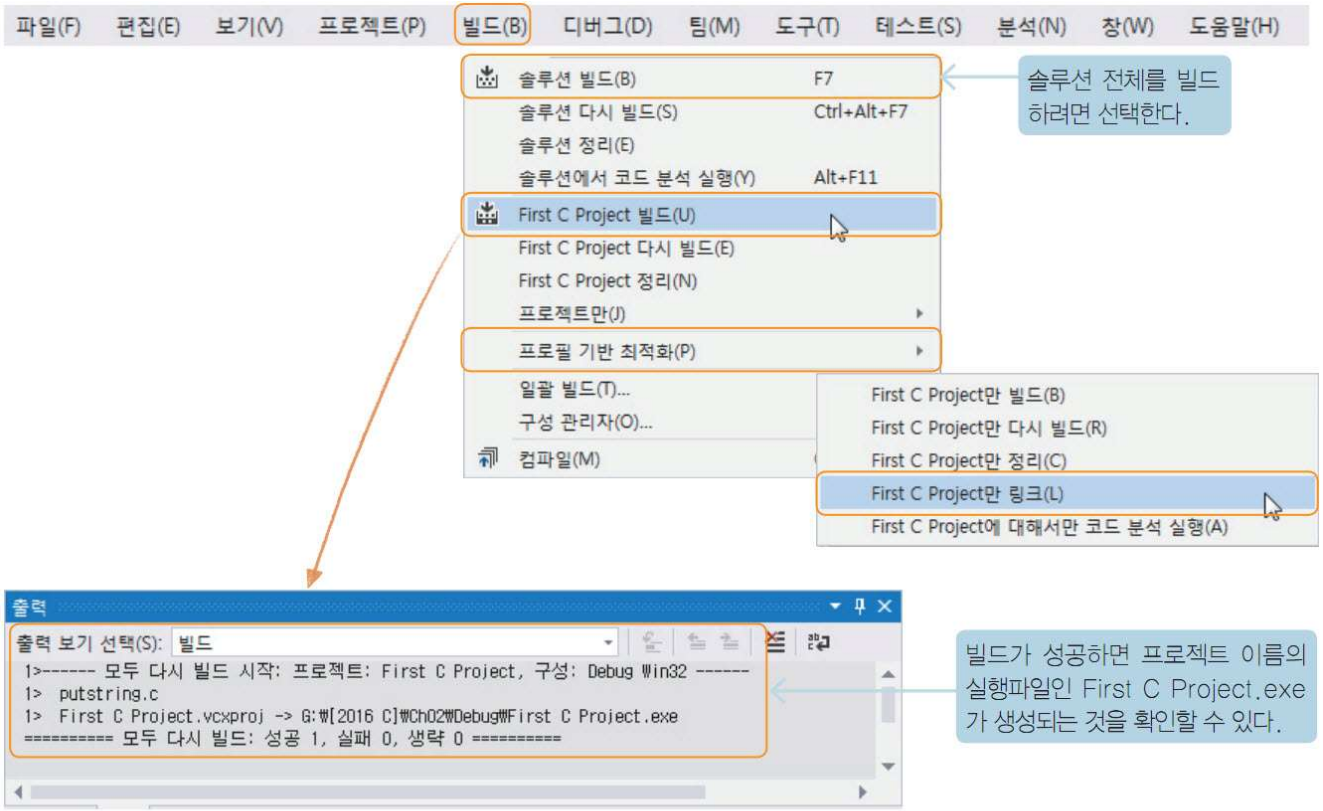


그림 2-38 빌드와 그 결과 화면 '출력'

비주얼 스튜디오 생성 파일

- **첫 솔루션과 프로젝트**
  - 솔루션 'Ch02' 하부
    - 프로젝트 'First C Project' 생성
  - 폴더 'Ch02/First C Project' 하부에 생성된 주요 파일
    - 프로젝트 파일 'First C Project.vcxproj'
    - 소스 파일 pustring.c
  - 솔루션 폴더 하부 'Ch02/Debug'
    - 프로젝트의 실행파일 'First C Project.exe'

표 2-2 비주얼 스튜디오 C 프로젝트 관련 파일

솔루션 폴더: Ch02	프로젝트 폴더: Ch02/First C Project		
파일명.확장명	**파일 이름**	**설명**	**위치**
Ch02.sln	솔루션	프로젝트, 프로젝트 항목 및 솔루션 항목을 솔루션으로 구성	Ch02
First C Project.vcxproj	프로젝트	비주얼 C++ 프로젝트 파일	Ch02/First C Project
putstring.c	소스	C 프로그램 소스 파일	
putstring.obj	목적	컴파일되었지만 링크되지 않은 개체 파일	Ch02/First C Project/Debug
First C Project.ilk	링크	링크 파일	Ch02/Debug
First C Project.exe	실행	실행 파일 또는 동적 연결 라이브러리 파일	
First C Project.pdb	디버그	프로그램 디버그를 위한 데이터베이스 파일	

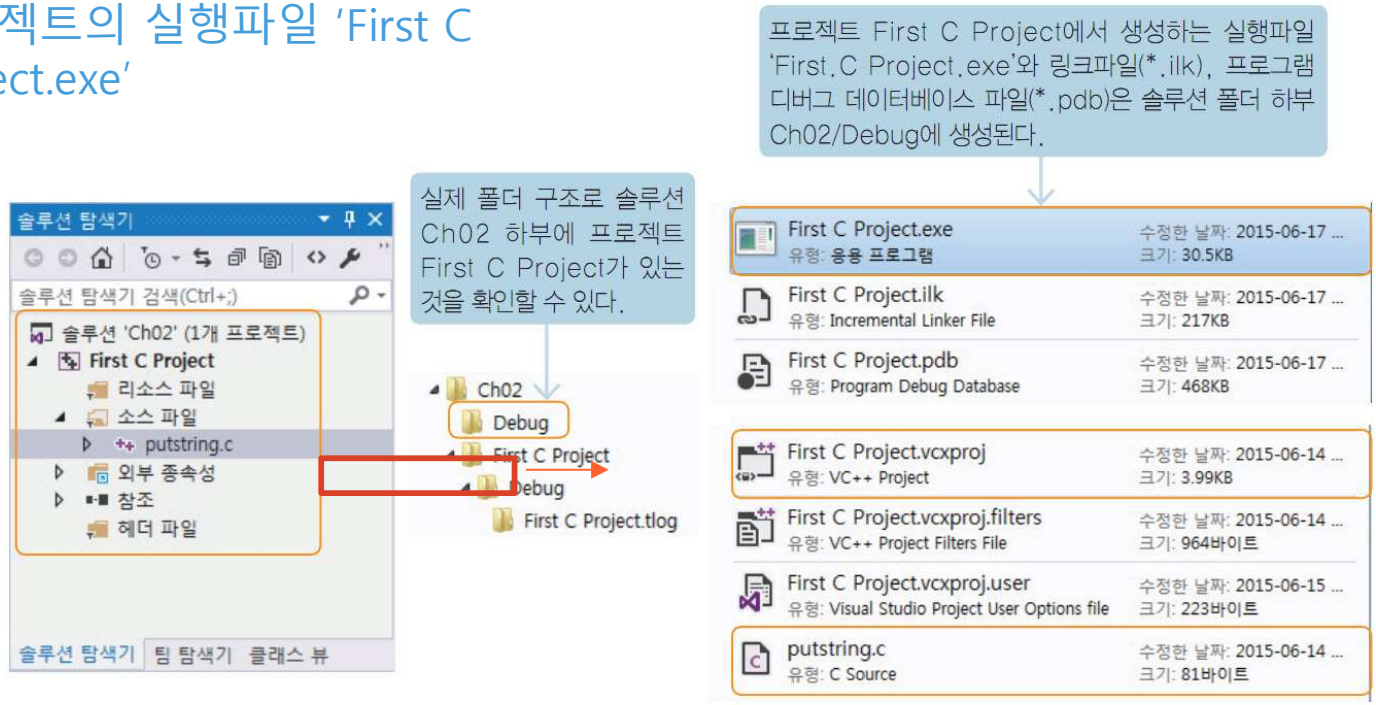


그림 2-39 솔루션 탐색기와 폴더 구조와 파일

솔루션 저장과 생성된 솔루션, 프로젝트 열기

- **비주얼 스튜디오 종료**
  - 프로젝트를 마치려면 메뉴 [파일] -> [모두 저장]을 누른 후
    - [파일] -> [끝내기]를 선택
- **이전에 생성한 솔루션을 열려면**
  - 메뉴 [파일] -> [최근에 사용한 프로젝트 및 솔루션]을 선택
  - 일반적으로 메뉴 [파일] -> [열기] -> [프로젝트/솔루션]을 선택
  - 솔루션 파일은 확장자가 sln이며, 프로젝트 파일은 vcxproj

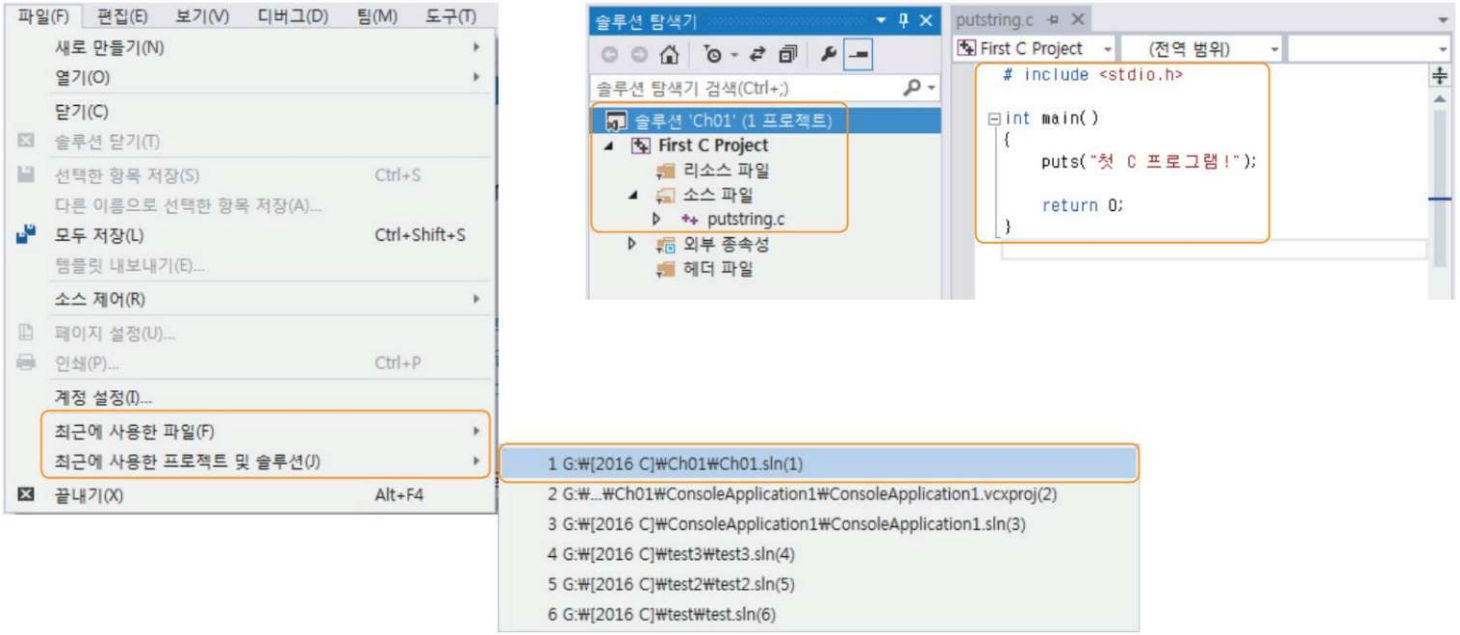
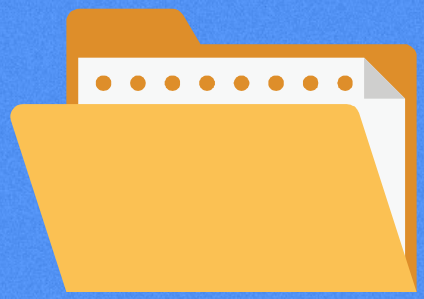
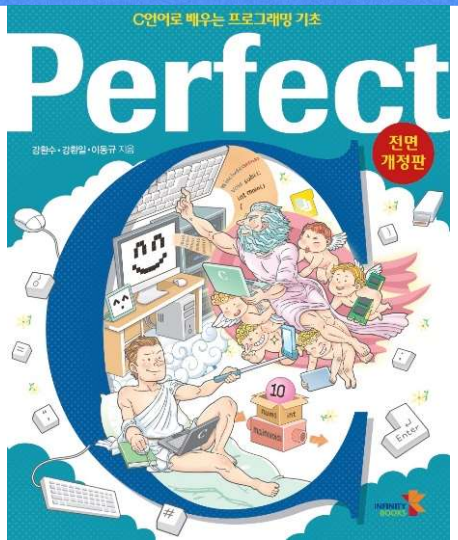


그림 2-40 최근에 사용한 프로젝트 및 솔루션 열기(open)

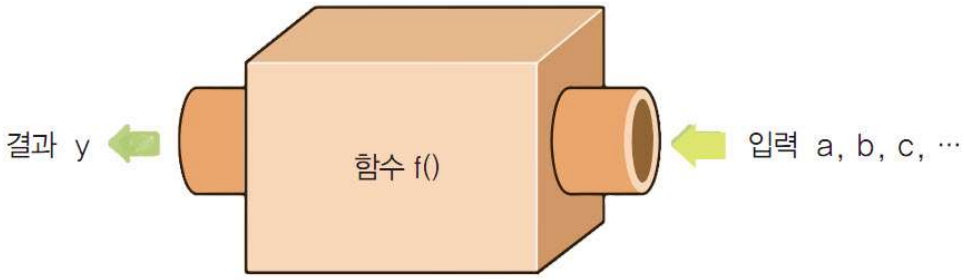


03. C 프로그램의 이해와 디버깅 과정



함수의 이해

- **함수 개요와 시작함수 main()**
  - C프로그램의 시작과 끝은 함수
  - C프로그램과 같은 절차지향 프로그램은 함수(function)로 구성
  - 함수 하나하나가 프로그램 단위
- **함수: 입력과 출력**
  - 함수는 'a, b, c...'와 같은 입력(input)을 받아
  - 'y'와 같은 결과(output) 값을 만들어 내는 기계장치와 유사
  - '입력'은 여러 개 사용될 수 있지만 결과값은 꼭 하나여야 한다는 점
- **사용자 정의 함수(user defined function)**
  - 프로그래머가 직접 만드는 함수
- **라이브러리 함수(library function)**
  - 시스템이 미리 만들어 놓은 함수



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

그림 2-41 함수는 입력과 출력 기능을 갖는 프로그램 단위

함수에서의 용어

- **함수 정의(function definition)**
  - 사용자 정의 함수를 만드는 과정
- **함수 호출(function call)**
  - 라이브러리 함수를 포함해서 만든 함수를 사용하는 것
- **매개변수(parameters)**
  - 함수를 정의할 때 나열된 여러 입력 변수
- **인자(argument)**
  - 함수 호출 과정에서 전달되는 여러 입력값

❶ 내가 가진 신용카드의 '한도액'을 알아야 하고
❷ '공인인증서'로 1차 인증을 한 후
❸ 신용카드번호를 입력하고, '비밀번호'를 입력한다.
❹ 휴대폰을 통해 전달된 '승인번호'를 입력하고
❺ 카드사의 '승인'을 기다려
❻ 최종 '결제'한다.

그림 2-42 신용카드와 사용절차

- **첫 프로그램에서의 main()**
  - 사용자가 직접 만드는 함수 정의 과정
- **puts(): 라이브러리 함수의 함수 호출**
  - 문장 puts("Hello World!")는 함수 호출 문장
  - 라이브러리 함수 puts()의 매개변수로 전달되는 인자
    - 문자열 "Hello World!"
    - 이 문자열이 표준출력으로 출력
  - 특별한 함수 main()을 제외하고는 프로그래머가 직접 만든 함수조차도 사용하기 위해서는
    - '함수 호출'이 필요
- **함수호출은 도서관에서의 도서 대출에 비유**
  - 필요한 자료나 서적이 있다면 '대출'하는 과정이 필요

시작함수 main()

- **main()함수의 정의 부분**
  - 함수 main() 정의의 첫 줄에 int와 void
    - 각각 함수가 자신의 작업을 모두 마친 후 반환하는 값의 유형
    - 함수로 값을 전달할 때 필요한 입력 형식
  - { … }
    - 중괄호 {와 }를 사용하여 함수의 기능을 구현
- **함수 main()이 실행되는 과정**
  - 프로그램이 실행되면
    - 운영체제는 프로그램에서 가장 먼저 main()함수를 찾고
      - 입력 형태의 인자로 main() 함수를 호출
    - 호출된 main()함수의 첫 줄을 시작으로 마지막 줄까지 실행하면
      - 프로그램은 종료
  - 만일 main() 함수 내부에서 puts()와 같이 라이브러리 함수를 호출
    - 라이브러리로 인자 "Hello World!"를 전달하여
    - puts()를 실행한 후 다시 main()으로 돌아와
    - 그 다음 줄인 return 0;을 실행

시작함수 main()

- **CRT 시작함수(C Runtime Startup function)**
  - 프로그램 실행 시 가장 먼저 호출되는 특별한 함수
  - 반환값
    - 함수 main()은 정상적인 작업을 마치면 정수 0을 반환

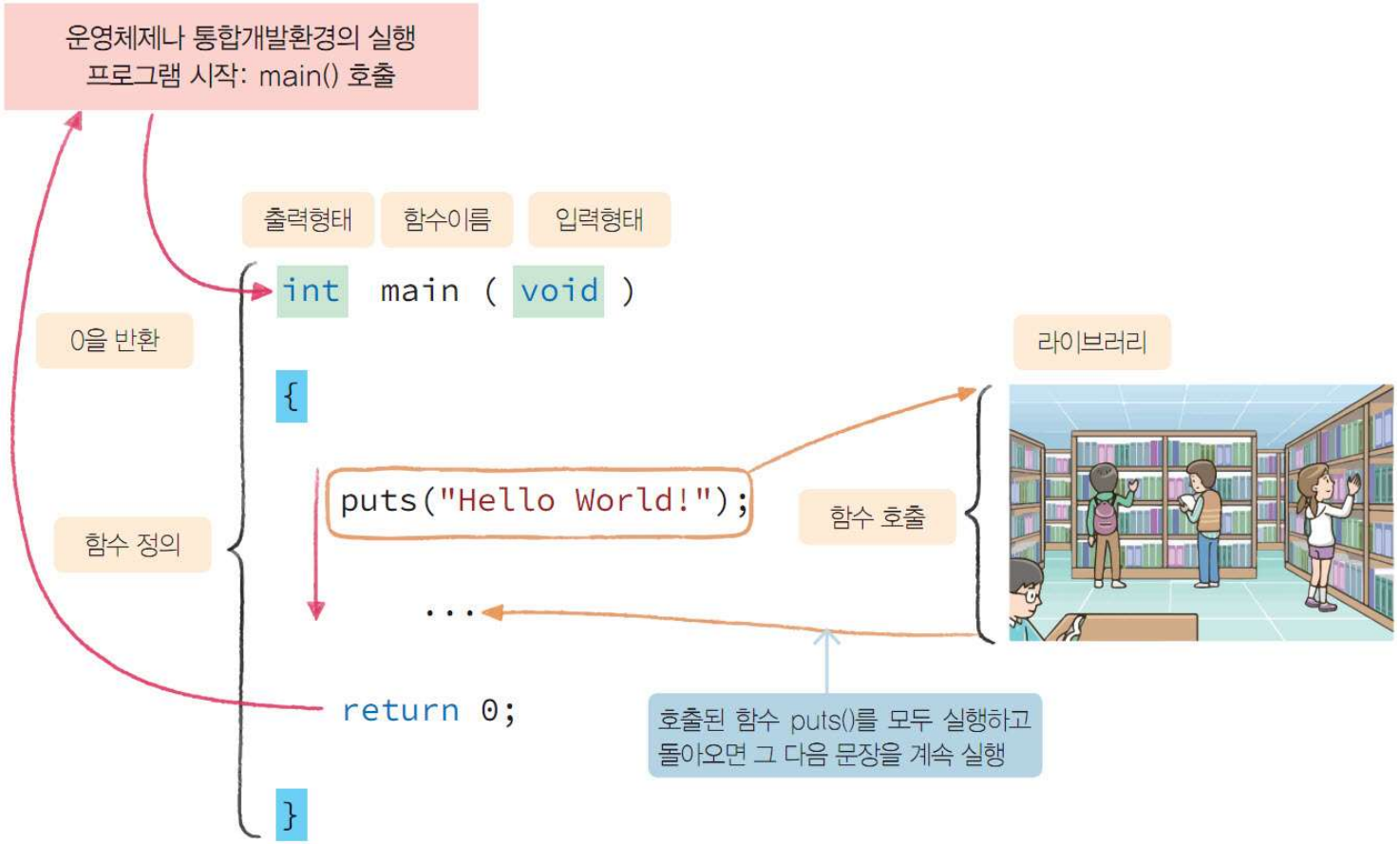


그림 2-43 CRT(C Runtime Startup function) 시작함수 main() 함수 정의와 라이브러리 함수 puts() 호출

두 번째 예제와 함수의 이해

- **문자열 "Hello World!"를 콘솔 창에 출력**
  - 어떤 프로그램 언어를 배우든지 가장 처음에 등장하는 유명한 예제
  - printf()라는 라이브러리 함수를 호출(call)
    - 함수 printf("문자열")는 인자인 문자열을 출력하는 기능을 수행

실습예제 2-2 putstring.c

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      printf("Hello World!\n");
06
07      return 0;
08  }
```

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

설명

```
01  #include: #과 include 사이는 빈 공간이 있어도 문제 없으나 일반적으로 #include처럼 붙여서
    기술하며, 이 첫 줄은 전처리기 지시자 include라 부름
01  <stdio.h>: stdio.h는 헤더 파일이라 부르며, 헤더 파일은 꺽쇠 모양의 문자 < >로 둘러쌈
03  int: int는 정수라는 integer에서 나온 자료유형이며, 함수 main()의 결과값의 유형
03  main(void): void는 함수 입력이 없다는 것이며, main 뒤의 괄호 ()는 반드시 필요
04  ~ 8 { … }: 집합 기호인 중괄호는 여러 문장을 하나로 묶는 블록이라 함
05  printf(): 함수 이름이 printf로 print formatted의 의미로 지정한 형식을 출력한다는 의미
05  "Hello World!\n": 문자열 Hello World!는 출력할 문자열이며, \n은 새로운 줄로 이동이
    라는 new line을 의미하는 문자를 표현하는데, \은 영문 키보드에서는 역슬래쉬 \임
07  return 0: 일반적으로 함수 main()은 0을 반환하면 프로그램이 정상적으로 종료됨을 의미
07  문장 종료인 ;(세미콜론)이 빠지면 컴파일 에러 발생
```

실행결과

```
Hello World!
계속하려면 아무 키나 누르십시오 . . .
```

함수의 머리와 몸체

- **C프로그램에서 main() 함수**
  - 자동차에 시동을 켜는 열쇠와 같은 역할
  - 반드시 정의되어야 함
- **함수 구현(정의)에서**
  - 함수 머리(function header)
    - int main(void)와 같이 함수에서 제일 중요한 결과값의 유형, 함수이름, 매개변수인 입력 변수 나열을 각각 표시
  - 함수 몸체(function body)
    - 함수 머리 이후 {...}의 구현 부분

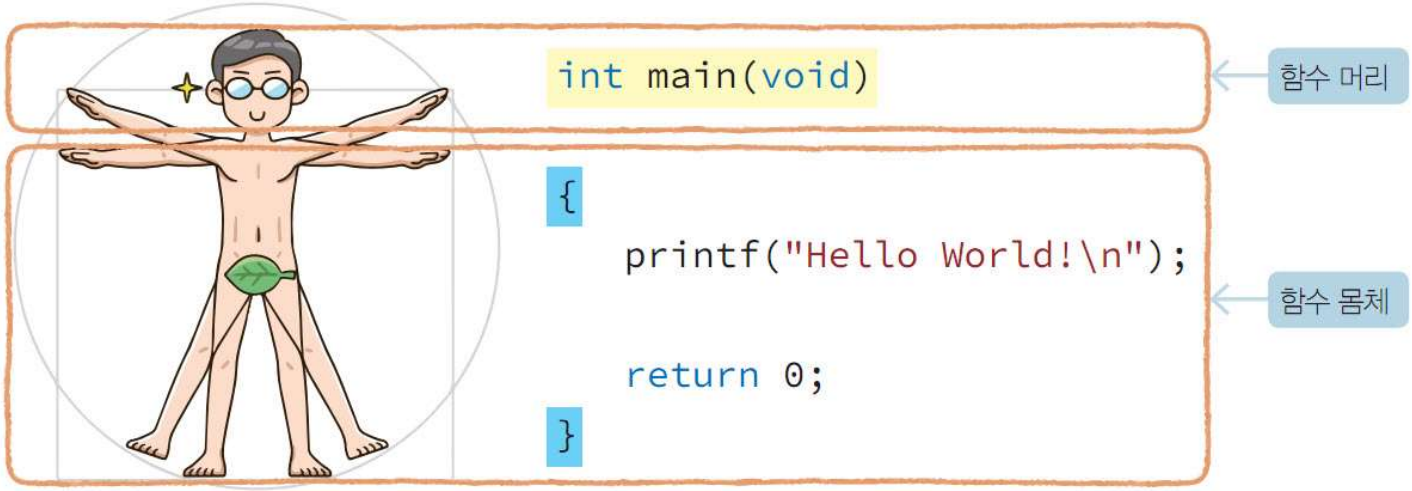


그림 2-44 사람과 같은 함수머리와 함수 몸체

순차실행과 들여쓰기

- **문장을 순차적으로 실행**
  - printf()를 처음으로 실행, 다음 return 0 문장을 실행하고 종료

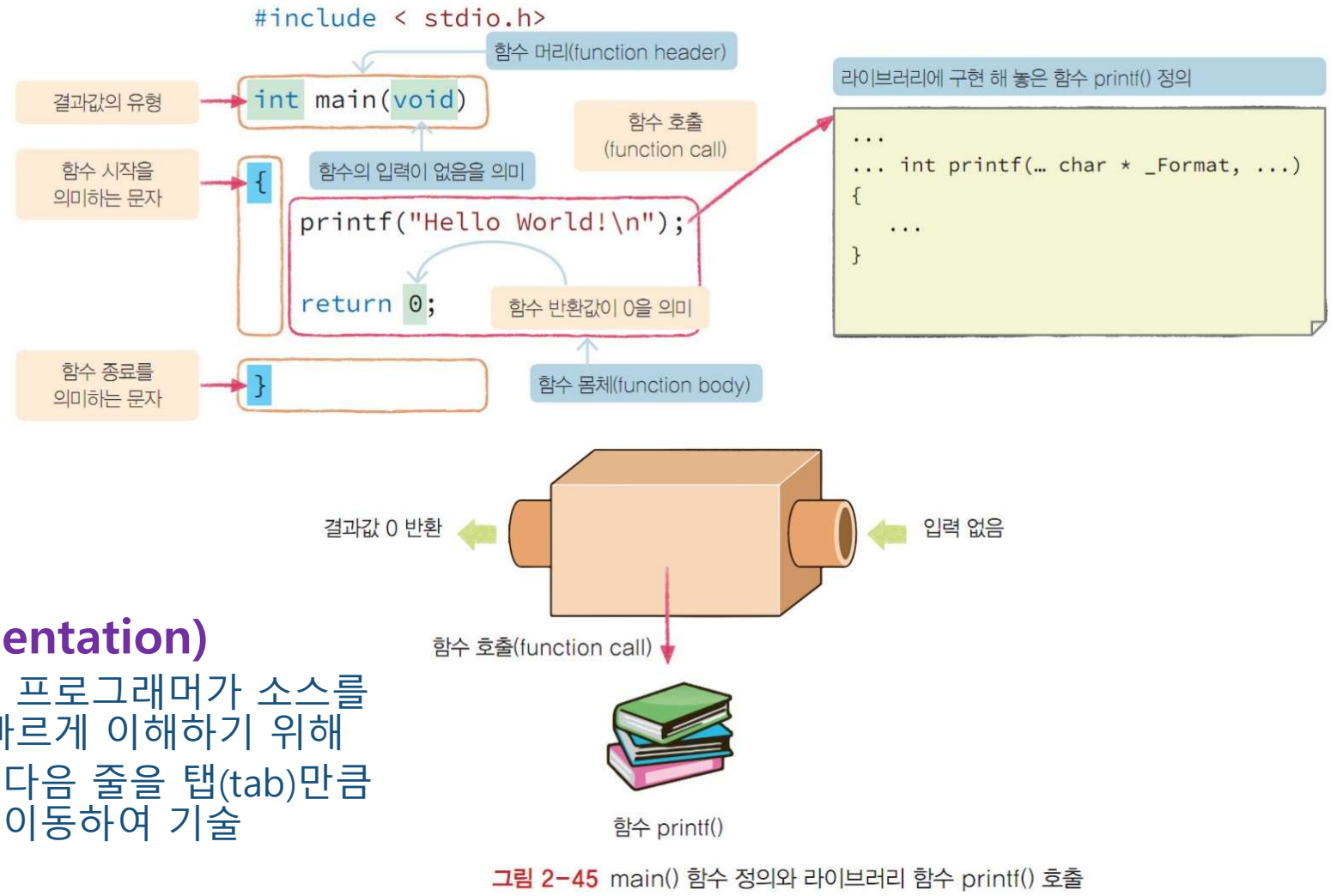


그림 2-45 main() 함수 정의와 라이브러리 함수 printf() 호출

- **들여쓰기(indentation)**
  - 함수 몸체는 프로그래머가 소스를 쉽게 읽고 빠르게 이해하기 위해
  - 블록 시작 { 다음 줄을 탭(tab)만큼 오른쪽으로 이동하여 기술

기존 솔루션에 프로젝트 추가

- **솔루션 'Ch02'에 새로운 프로젝트를 추가**
  - 프로젝트 이름: 'Second Project'
  - 소스 파일: printstring.c
- **메뉴 [파일] -> [추가] -> [새 프로젝트]를 선택**
- **❶ 솔루션 'Ch02'**
  - 오른쪽 마우스를 클릭해 [추가] -> [새 프로젝트]를 선택
  - 항목 '위치'
    - 솔루션 'Ch02'의 폴더임을 확인
  - ❷ 프로젝트 이름
    - 'Second Project'을 입력
  - [Second Project] 마법사
    - ❸ '콘솔 응용 프로그램' 라디오 버튼과 '빈 프로젝트' 체크박스를 선택

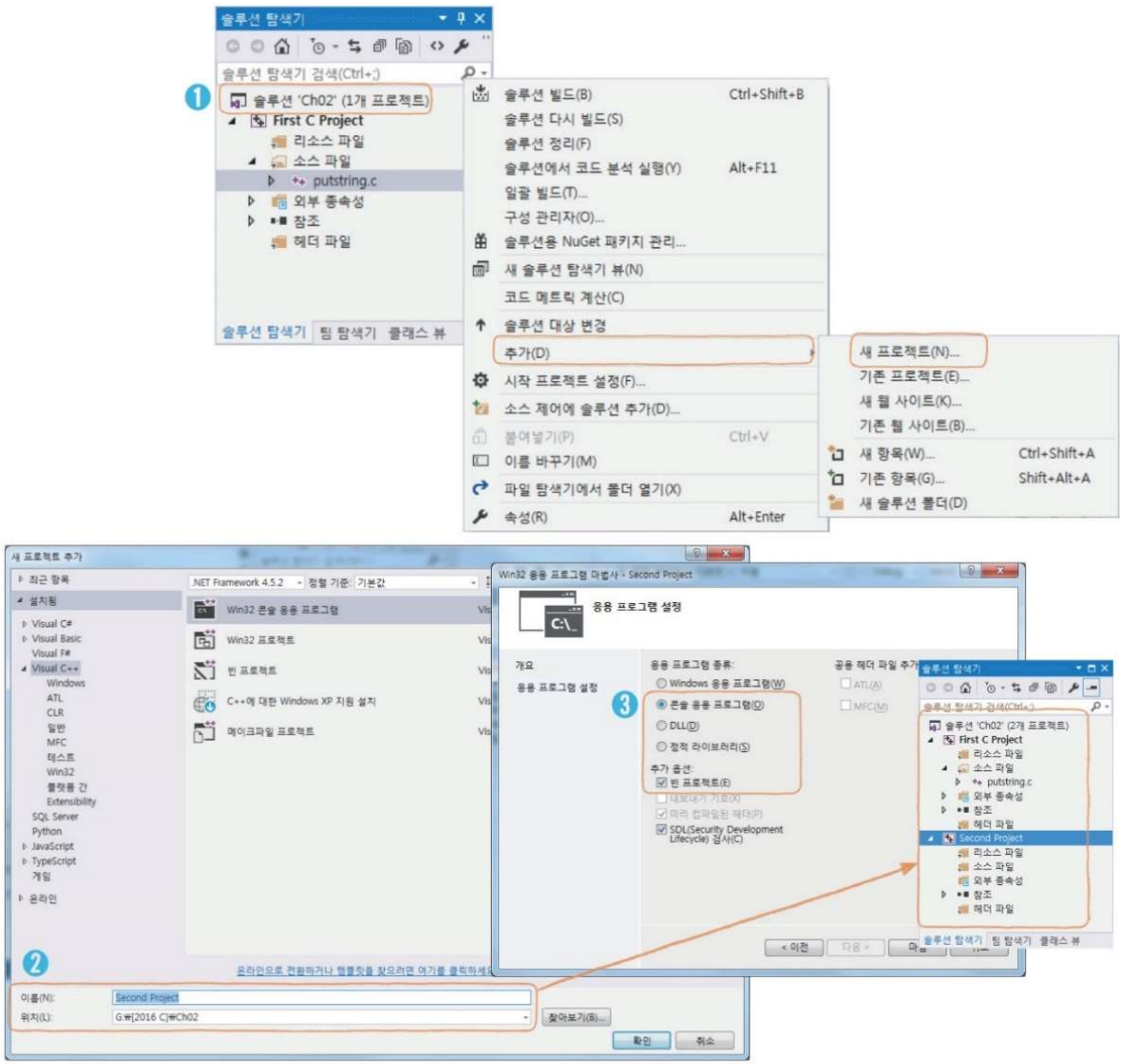


그림 2-46 [Second Project]을 위한 메뉴와 [새 프로젝트 추가] 대화상자와 설정

두 번째 소스 작성과 실행

- **프로젝트 'Second Project'**
  - 소스파일 printstring.c를 생성하여 소스를 편집
  - 메뉴 [빌드] -> [Second Project 빌드]를 선택

- **메뉴 [프로젝트] -> [시작 프로젝트로 설정]을 선택**
  - 먼저 'Second Project'를 클릭한 후
  - 시작 프로젝트로 설정을 하지 않으면?
    - 다른 설정된 시작 프로젝트가 실행되는 일이 발생

- **실행**
  - 메뉴 [디버그] -> [디버깅 하지 않고 실행] 선택
  - 단축키 ctrl + F5로 실행 결과를 확인

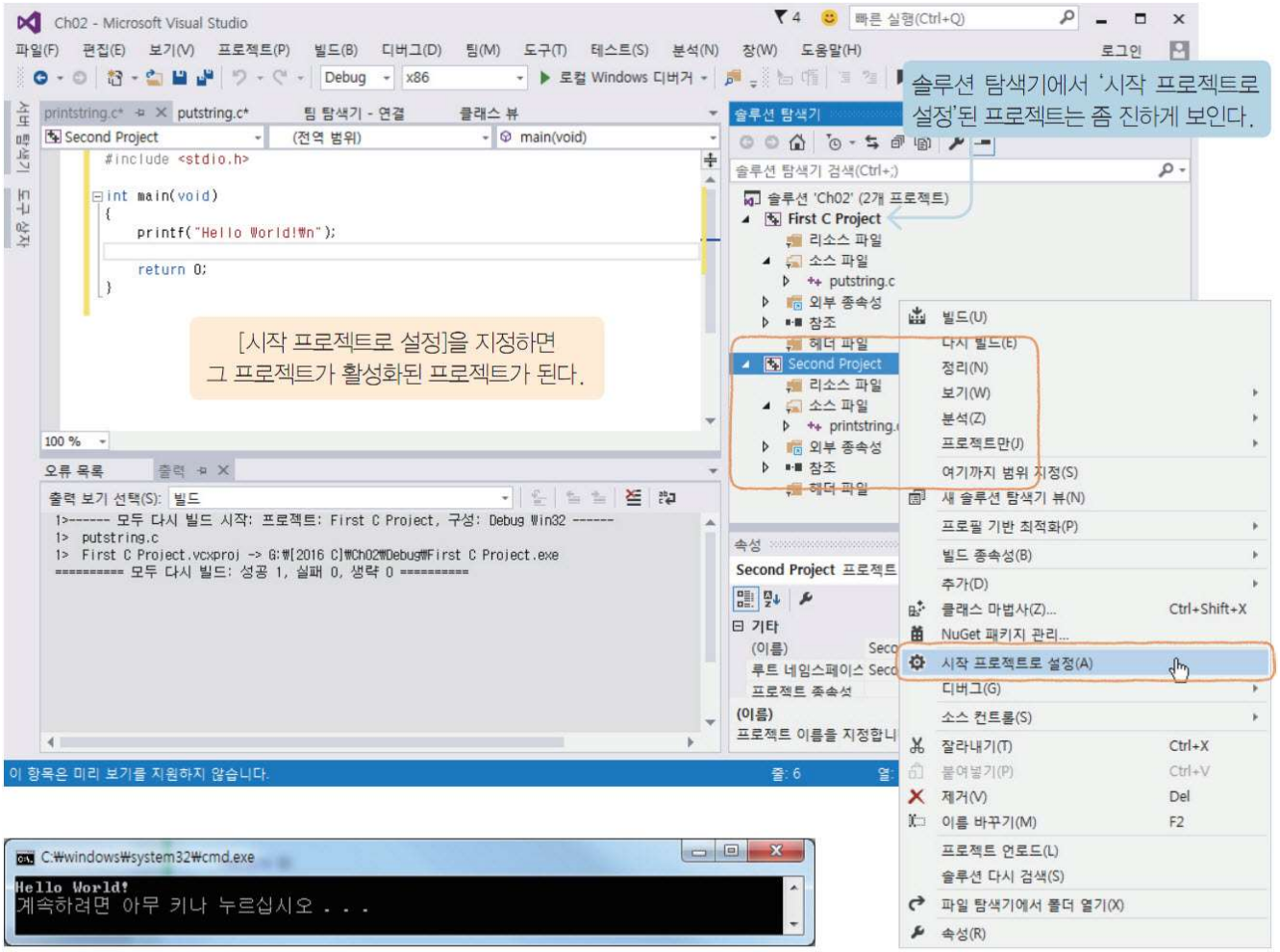


그림 2-47 [Second Project]를 시작 프로젝트로 설정 후 실행한 결과

여러 줄에 문자열을 출력

- **세 번째 실습예제 printmline.c**
  - 라이브러리 함수 puts()와 printf()를 호출하여 여러 줄에 문자열 정보를 출력
    - 함수 puts()는 문자열을 전용으로 출력하는 함수
    - 함수 printf("문자열")는 호출 시 전달되는 "문자열"과 같은 다양한 형태의 인자를 적절한 형식으로 출력하는 함수
  - ₩n에 주의하여 코딩
    - 문자열에 삽입된 새로운 줄을 의미
  - 솔루션
    - 기존 솔루션 [Ch02]
  - 프로젝트
    - Third Project
  - 소스파일
    - printmline.c

실습예제 2-3 printmline.c

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05     puts("세 번째 C 프로그램!");
06     puts("");
07     printf("-- 비주얼 스튜디오 C 프로그래밍 과정 --\n");
08     printf("\n");
09     puts("1. 솔루션과 프로젝트 만들기");
10     printf("2. 소스파일 편집\n");
11     printf("3. 실행\n");
12
13     return 0;
14  }
```

설명

```
05  ~ 11 { … }: 구동 함수 main()의 내부로 5줄에서 처음으로 시작해 11줄까지 순서대로 실행
05  puts(): 라이브러리 함수 puts() 호출하여 인자로 전달된 문자열 "세 번째 C 프로그램!"을 출력
    하고 다음 줄 첫 열로 이동하여 다음 출력을 준비
06  puts(""): 함수 호출 시 인자로 전달되는 “”은 소위 아무것도 없는 널(null) 문자로, 아무것도
    출력하지 않고 다음 줄로 이동하므로 빈 줄이 나타나는 효과
08  printf("\n"): 함수 호출 시 인자로 전달되는 “\n”은 ‘새 줄(new line)’을 의미하는 문자 표
    현으로, 현재 위치에 아무것도 출력하지 않고 \n에 의해 다음 줄로 이동하므로 빈 줄이 나타나는 효과
08  ~ 9 함수 printf()의 인자와 puts()의 인자는 모두 문자열이며, printf()의 인자는 문자열 마
    지막에 \n이 있으나 puts()의 인자는 \n이 없어야 현재 출력 위치에 문자열을 출력 한 후, 다음 줄
     첫 열로 이동하여 다음 출력을 준비
```

실행결과

```
세 번째 C 프로그램!

-- 비주얼 스튜디오 C 프로그래밍 과정 --

1. 솔루션과 프로젝트 만들기
2. 소스파일 편집
3. 실행
```

함수 puts()

- **#include <stdio.h>**
  - 라이브러리 함수 puts()와 printf()를 사용하려면
  - #include는 바로 뒤에 기술하는 헤더파일 stdio.h를 삽입하라는 명령어
- **함수 puts()**
  - 원하는 문자열을 괄호 (“원하는 문자열”) 사이에 기술
    - 인자를 현재 위치에 출력한 후 다음 줄 첫 열로 이동하여 출력을 기다리는 함수
  - 괄호 사이에 아무것도 없으면
    - 인자가 없으므로 오류가 발생
  - puts(“”)와 같이 공백 문자열을 입력
    - 현재 출력 위치에 공백 문자열을 출력한 후 다음 줄로 이동하는 효과

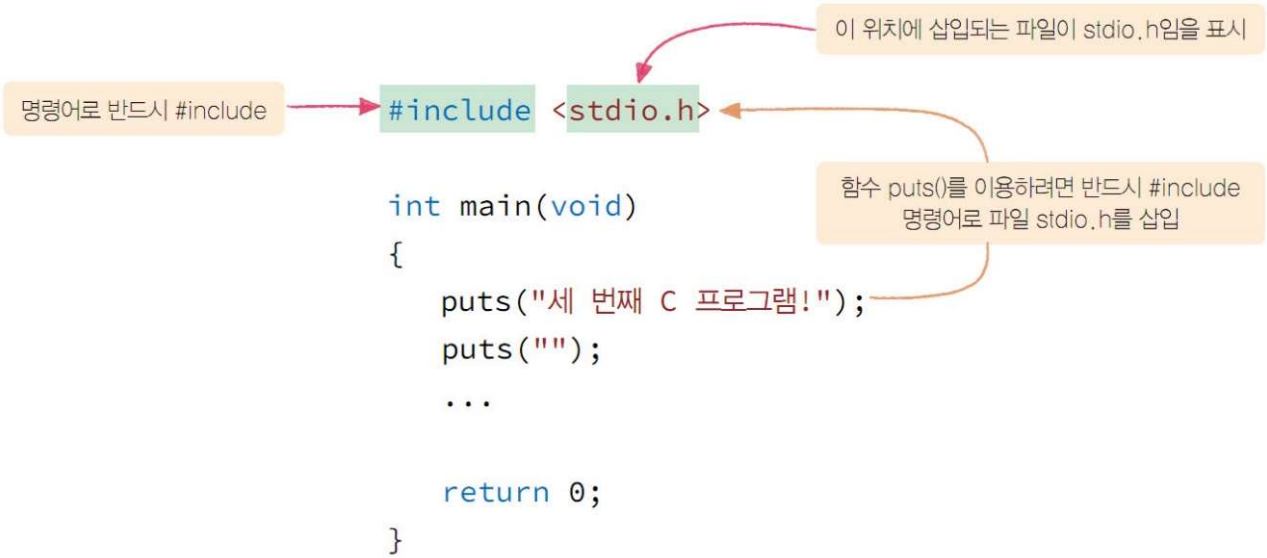


그림 2-48 #include와 헤더 파일

함수 printf()

- **함수 printf()**
  - 원하는 문자열을 괄호 (“원하는 문자열”) 사이에 기술
  - printf(“”)와 같이 공백 문자열을 인자로 전달
    - 현재 위치에 공백문자를 출력, 결과는 아무것도 출력되는 것이 없음
  - 함수 호출 printf(“₩n”)
    - 출력 위치를 새로운 줄 첫 열로 이동하게 하는 효과
- **주요 활용**
  - 인자인 문자열을 출력하고 다음 줄로 이동하여 출력 위치를 지정
    - 함수 puts(“문자열”) 또는 함수 printf(“문자열₩n”)로 호출
  - 아무것도 출력 없이 출력 위치를 다음 줄로 이동
    - 함수 puts(“”) 또는 함수 printf(“₩n”)로 호출

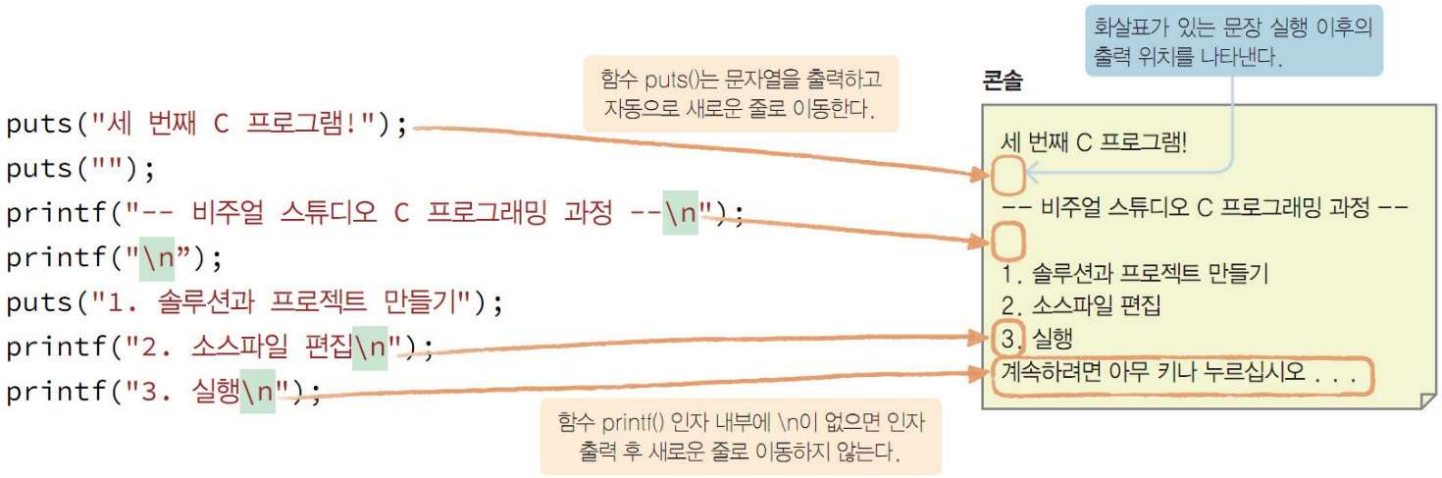


그림 2-49 함수 호출 puts()와 printf()의 차이

3 개의 프로젝트를 담은 솔루션 'Ch02'

- **솔루션 'Ch02'에는 총 3개의 프로젝트**
  - 예제 printmline.c는 솔루션 'Ch02'의 하부 프로젝트 'Third Project'에 저장
- **편집기의 왼쪽에 소스의 줄 번호 표시**
  - 메뉴 [도구] -> [옵션]을 선택
  - '옵션' 대화상자에서 [텍스트 편집기] -> [C/C++]를 선택
    - '설정'의 [줄 번호] 체크박스를 선택

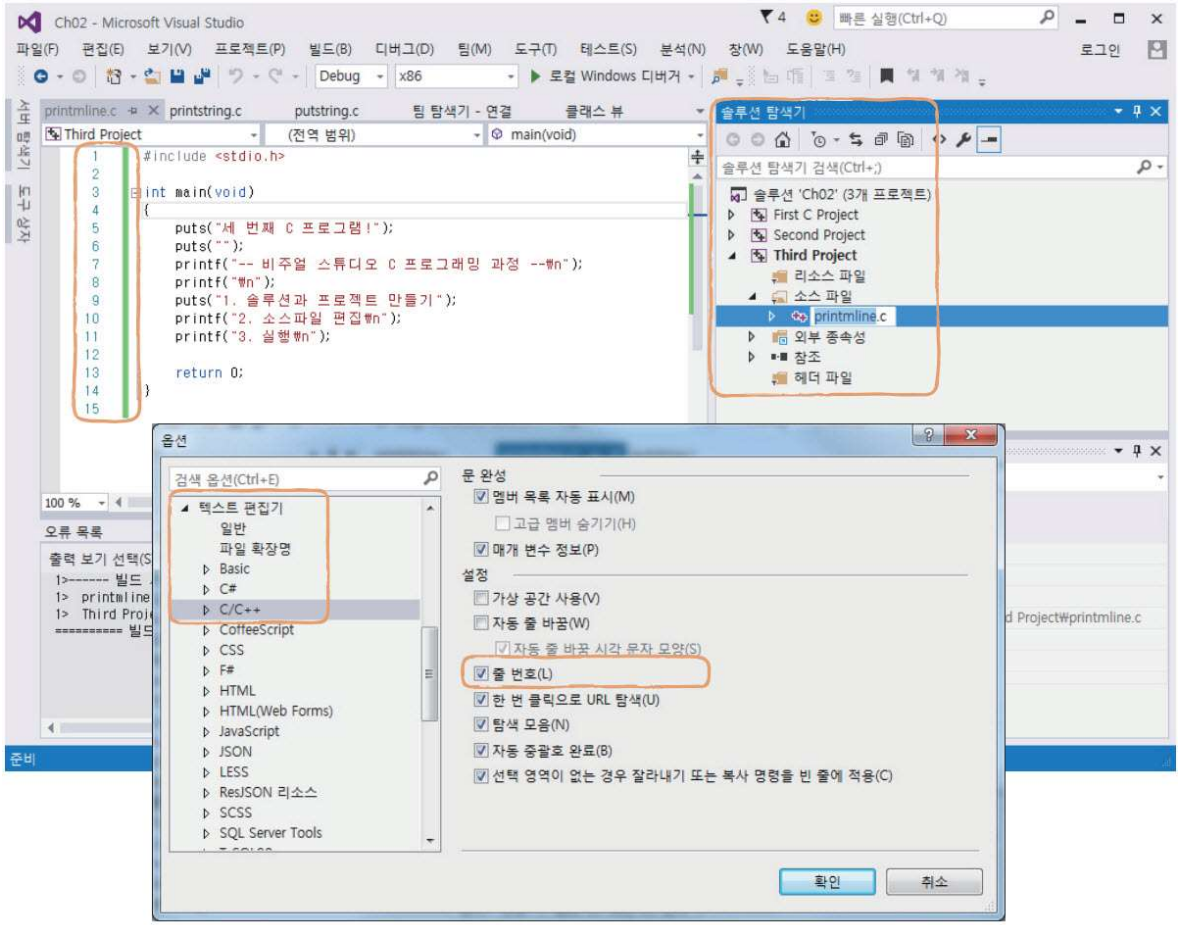


그림 2-50 솔루션 'Ch02'의 여러 프로젝트와 소스 파일의 줄 번호 보이기

디버깅 예제(1)

- **함수 printf()로 원하는 문자열(string)을 출력하는 프로그램을 작성**
  - 문법 오류가 발생하도록 의도적으로 소스에 오류를 심어 놓음
  - 솔루션: 기존 솔루션 'Ch02', 프로젝트: '4th Project', 소스파일: debugging.c
- **'저장'을 하면**
  - 오류가 의심되는 소스 부분에 붉은 색 밑줄이 생김
  - 이 부분에 마우스를 이동
    - 바로 "오류: ';'가 필요합니다."와 "오류: 닫는 따옴표가 없습니다."라는 정확한 오류 원인 표시
  - 만일 수정을 못하고 계속해서 컴파일이나 빌드를 수행
    - 컴파일 오류가 발생
    - 오류 목록 창에 '오류 내용'이 표시

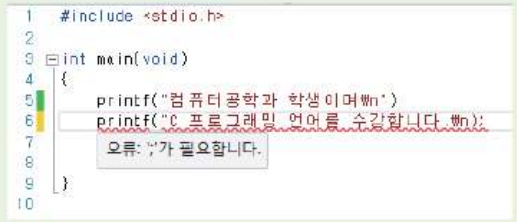
실습예제 2-4 debugging.c

```
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  }
```

설명

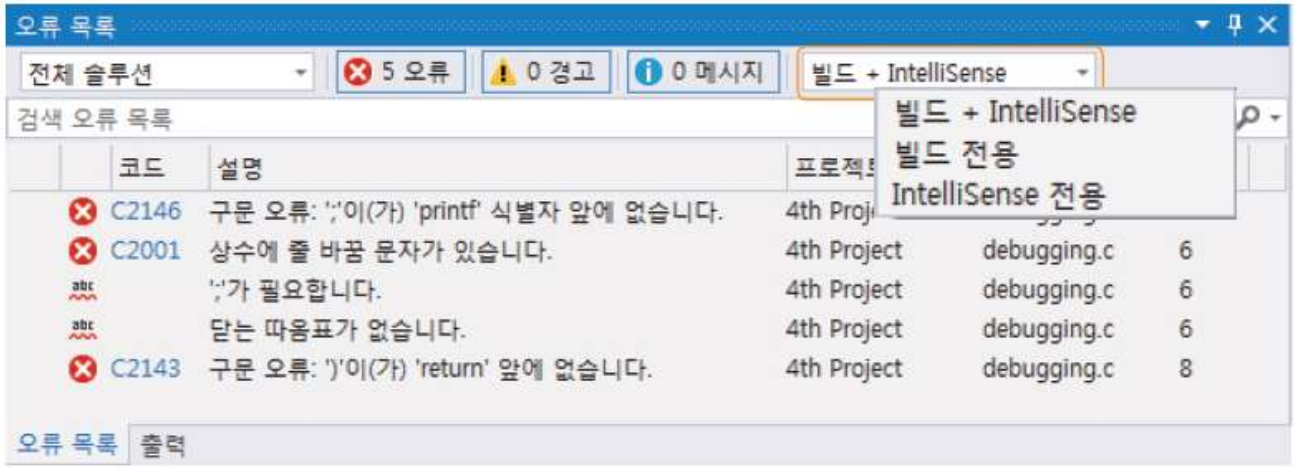
05 문장 종료를 나타내는 ;이 빠짐

06 문자열의 종료를 나타내는 "(닫는 큰 따옴표)가 빠짐



디버깅 예제(2)

- **'오류 목록' 창**
  - 일목요연한 오류 목록 표시
    - 오류 코드, 설명, 프로젝트, 파일, 줄 번호 등을 자세히 표시
    - 창에는 지능적인 오류 표시인 IntelliSense 오류 표시
- **오류목록 창의 하단 우측의 출력 선택**
  - ❶ 첫 줄에는 빌드를 시작한 프로젝트 이름
  - ❷ 두 번째 줄에는 컴파일한 소스인 debugging.c가 표시
  - ❸ 세 번째 줄부터 문제가 발생한 원인의 내용이 표시
  - ❹ 마지막으로 "= 빌드: 성공 0, 실패 1, 최신 0, 생략 0 ="와 같이 최종 빌드 결과가 표시



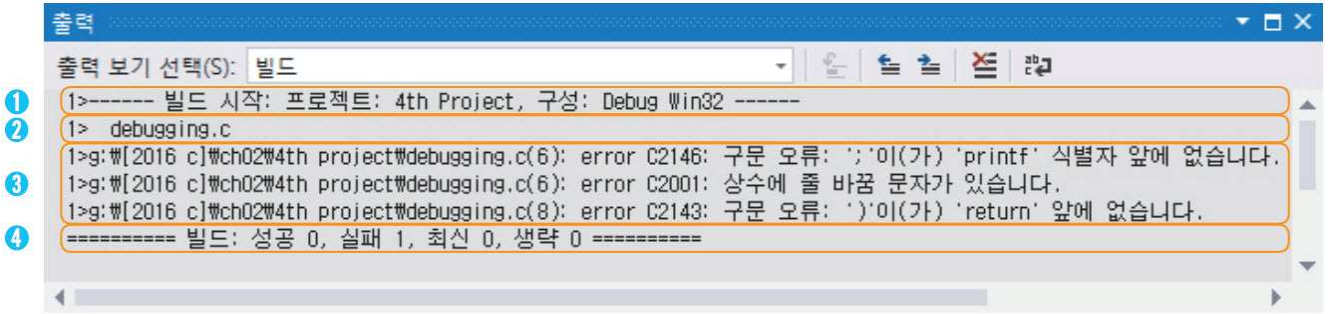


그림 2-52 출력 창

컴파일 오류 메시지의 이해와 소스 수정(1)

- **오류 내용 표시와 문장 종료 문자 ;이 빠진 오류 메시지 분석**
  - 4개의 요소가 3개의 콜론(:)으로 구분되어 표시
    - 오류가 발생한 파일이름(전체경로): g:₩[2016 c]₩ch01₩4th project₩debugging.c
    - 추정되는 오류발생 줄 번호: (6)
    - 오류 코드 번호: error C2146
    - 오류 원인 메시지: 구문 오류 : ';'이(가) 'printf' 식별자 앞에 없습니다.

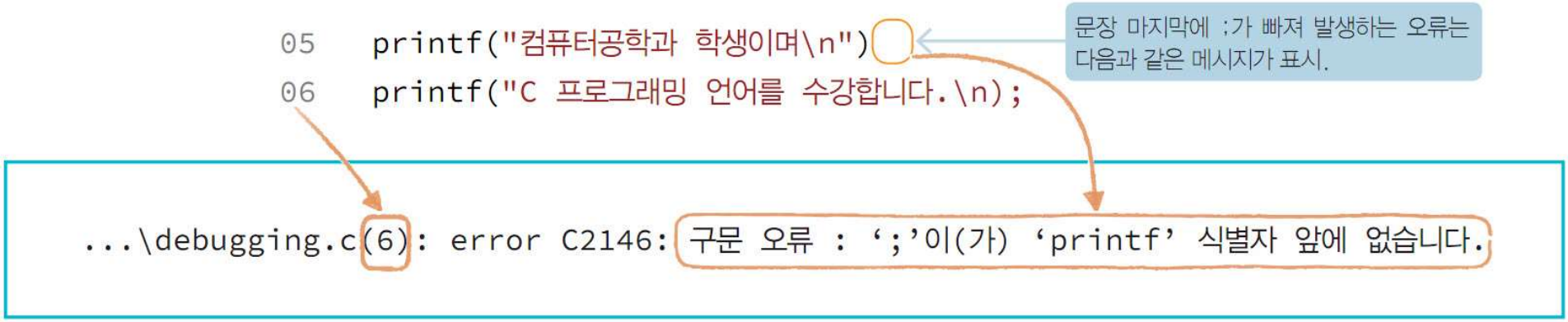


그림 2-53 문장 마지막에 ;이 빠진 오류 내용

  - 오류 발생 줄 번호는 ;이 빠진 5가 표시되지 않고
    - 그 다음 줄인 6이 표시
    - 그러나 에러 메시지는 6줄의 printf() 앞에 ;가 없다고 함
  - 출력 창의 오류 메시지 줄 위에서 마우스를 더블 클릭
    - 소스의 해당 줄로 이동
  - 다시 소스 5줄로 이동하여 마지막에 ;를 삽입하면, 이 컴파일 오류는 해결

컴파일 오류 메시지의 이해와 소스 수정(2)

- **문자열을 표시하는 "문자열" 중에서 뒤의 "가 빠진 오류 메시지 분석**
  - 오류 발생 줄 번호는 다음과 같이 6줄과 8줄로 2개가 표시
    - 오류 메시지도 쉽게 이해가 되지 않음

```
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

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

그림 2-54 문자열 닫힘 "이 없는 경우 발생하는 오류 내용

NOTE: 실제 오류와 IDE 결과 창의 오류 내용의 일치?

IDE는 소스에서 발생한 오류는 정확히 발견하지만, 줄 번호와 오류 원인은 생각보다 정확하지 않은 경우도 많다. 이와 같이 오류 발생 줄 번호는 정확하지 않을 수 있으니 줄 번호뿐 아니라 그 주위를 둘러 봐야 한다. 처음에는 이러한 오류를 발견하는 것이 쉽지 않을 수 있지만 자주 경험하면 오류의 위치와 원인을 쉽게 찾을 수 있을 것이다.

  - 이런 오류는 여러 번 경험해야 쉽게 그 원인을 찾아 수정 가능
  - 이 경우는 소스의 붉은 줄에 표시되는 오류 풍선의 메시지
    - '오류: 닫는 따옴표가 없습니다.'가 훨씬 효과적
  - 소스 debugging.c에서 발생한 오류 원인 2개를 수정
    - 5줄에 ;을, 6줄에 "을 삽입하면 컴파일 오류는 사라지고 결과가 출력

컴파일 오류 메시지의 이해와 소스 수정(3)

- **다음은 초보자에게 흔하게 발생하는 컴파일 오류의 예**
  - 오류 발생 부분 밑줄에 마우스를 이동하면 나타나는 오류 풍선
  - 결과 창의 오류 원인 메시지 등을 참고하여 수정
  - 오류 풍선
    - 대부분 정확한 오류 원인을 알려 줌
  - 오류 원인 메시지
    - 오류가 발생한 주위 코드에 따라서 여러 개의 오류 원인 메시지가 나오는 등 복잡한 경우가 많음
  - 컴파일 오류가 발생
    - 오류 풍선을 통해 먼저 오류 원인을 알아보고
    - 그 이후 오류 원인 메시지를 확인하여 문제를 해결하는 습관

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: '...\debugging.c(4)')이(가) 짝이 되기 전에 파일의 끝이 나타났습니다.

링크 오류 수정

자동차 생산에서 잘못된 부품을 조립하는 것과 유사

함수 print() 사용

- 대표적인 링크 오류는 라이브러리 함수인 printf()의 철자를 잘못 기술
- 빌드하면 경고 C4013(warning C4013)이 표시, 링크 오류도 발생
- 함수 print()의 호출은 컴파일 시간에는 경고 오류만 표시

```
05    print("컴퓨터공학과 학생이며\n");
```

라이브러리 함수 printf()로 수정하면 아무 문제없이 잘 실행된다.

```
...\debugging.c(5): warning C4013: 'print'이(가) 정의되지 않았습니다. extern은 int형을 반환하는 것으로 간주합니다.
debugging.obj : error LNK2019: _print 외부 기호(참조 위치: _main 함수)에서 확인하지 못했습니다.
...\Ch01\Debug\4th Project.exe : fatal error LNK1120: 1개의 확인할 수 없는 외부 참조입니다.
```

그림 2-58 링크 에러 메시지 "_print 외부 기호(참조 위치: _main 함수)에서 확인하지 못했습니다."

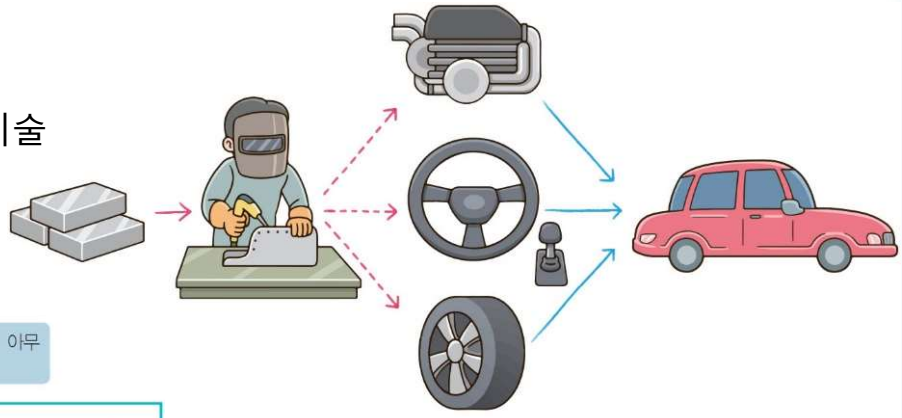


그림 2-56 자동차 생산에서의 링크 과정

함수 mein() 사용

- 구동 함수인 main()을 mein() 등으로 잘못 기술해도 링크 오류가 발생
- 컴파일 시간에는 오류가 발생하지 않으나, 빌드 시 2개의 링크 오류가 발생

```
03    int mein(void)
```

함수 정의 mein을 구동 함수 main()으로 수정하면 아무 문제 없이 잘 실행된다.

링크 오류는 줄 번호가 표시되지 않아, 오류 수정이 어려운 경우가 많다.

```
MSVCRTD.lib(crtexe.obj) : error LNK2019: _main 외부 기호(참조 위치: ___tmainCRTStartup 함수)에서 확인하지 못했습니다.
G:\[2016 C]\Ch01\Debug\4th Project.exe : fatal error LNK1120: 1개의 확인할 수 없는 외부 참조입니다.
```

그림 2-59 링킹 에러 메시지
"_main 외부 기호(참조 위치: ___tmainCRTStartup 함수)에서 확인하지 못했습니다."

실행시간 오류

- 컴파일과 링크가 성공해도 실행시간에 오류가 발생 가능
  - 일반적으로 메모리관리를 실수하거나
  - 0으로 나누는 식을 사용하는 등
  - 프로그램의 잘못으로 발생하는 경우가 대부분
  - 간혹 기계적 결함으로도 발생

논리 오류 수정

의도와 다른 결과가 나온다면 모두 논리 오류

출력 문자열의 오류

- 문자열에서 띄어 쓰기를 잘못한다거나 철자를 잘못 쓰는 것도 가장 흔한 논리 오류 중의 하나
- 논리 오류도 다른 문법 오류와 마찬가지로 소스 코딩을 잘못하여 발생하는 것이 대부분
- 문자열의 철자 오류와 같은 논리 오류는 문제를 찾기도 쉬우며 수정도 간단

```c
# include <stdio.h>

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

    return 0;
}
```

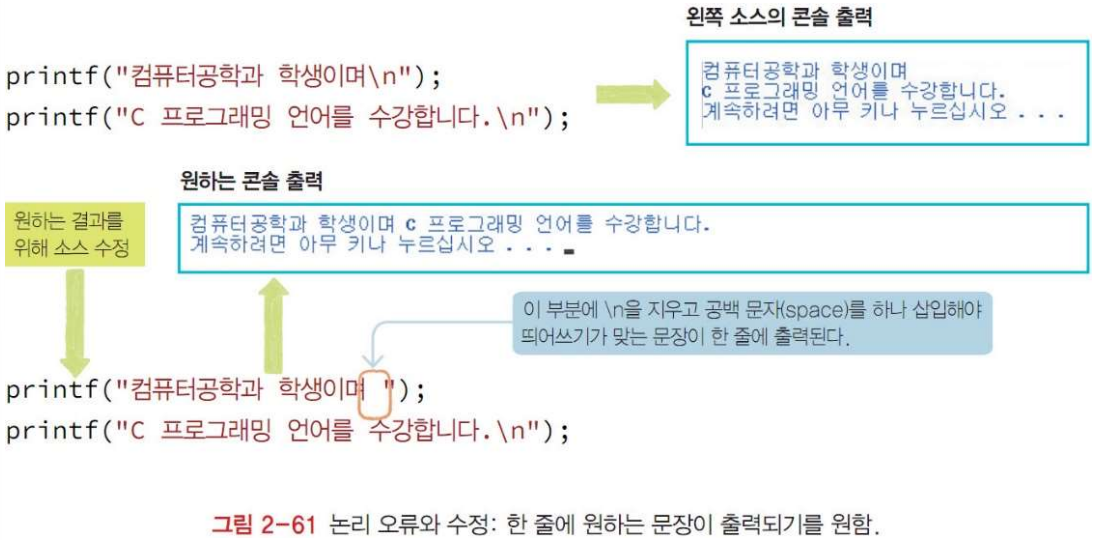


그림 2-61 논리 오류와 수정: 한 줄에 원하는 문장이 출력되기를 원함.

논리오류의 디버깅이 가장 어려움

- 복잡하고 큰 규모의 소프트웨어의 개발에서 다양한 문제로 발생하는 논리 오류
  - 찾기가 매우 어려운 경우가 많음
- 프로그램의 문제해결 절차인 알고리즘을 잘 만든 후
  - 이를 준수해서 소스를 코딩해야 논리 오류가 적은 프로그램을 완성

Thank you