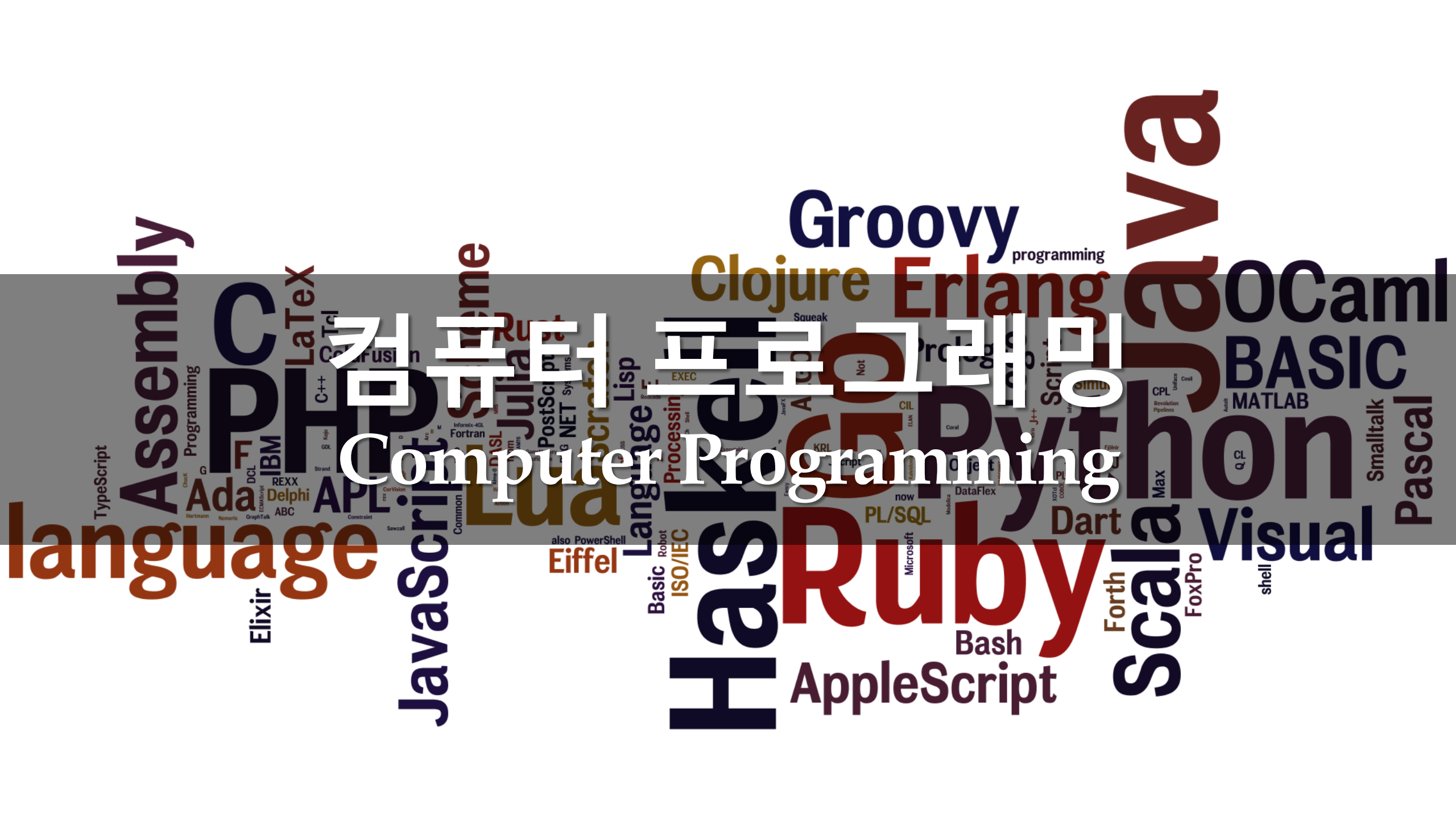
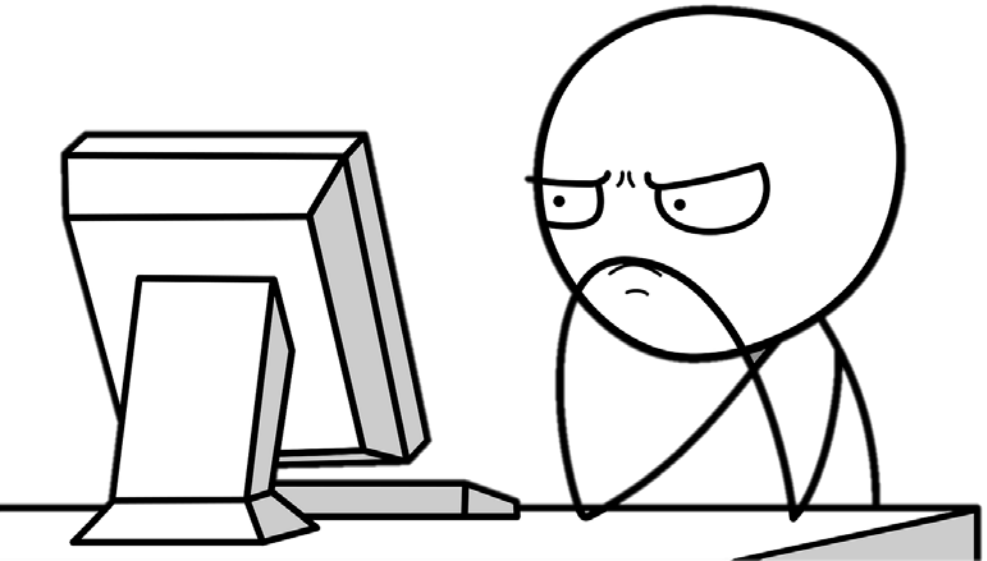

컴퓨터 프로그래밍
Computer Programming

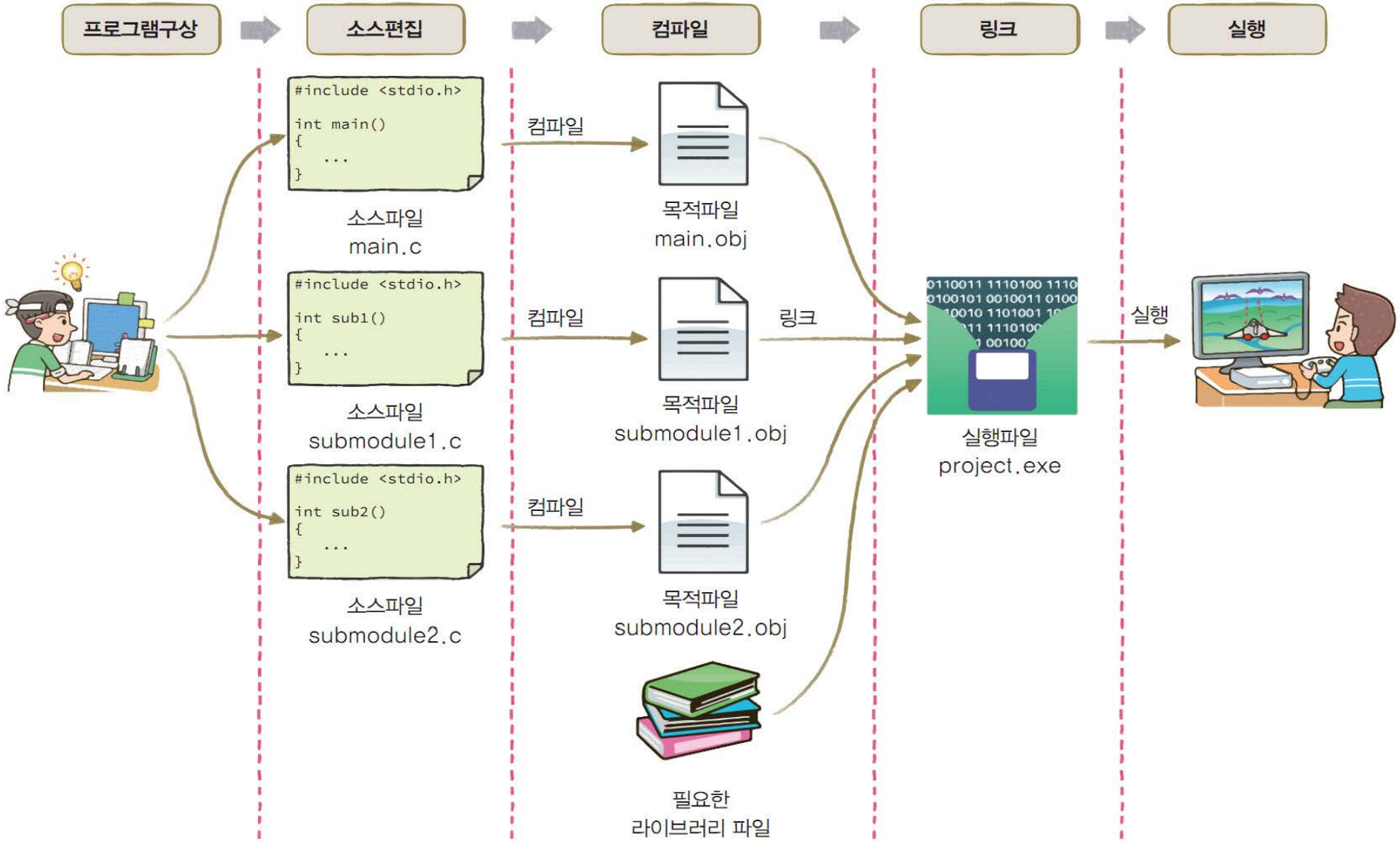
02
C 프로그래밍 첫걸음

목차

1. 프로그램 구현 과정과 통합 개발 환경



프로그램 구현 과정 5단계



프로그램 구상과 소스편집

- 소스코드는 선정된 프로그래밍 언어인 C 프로그램 자체로 만든 일련의 명령문을 의미
- 소스파일source file: C와 같은 프로그래밍 언어로 원하는 일련의 명령어가 저장된 파일, 텍스트 파일로 저장

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

```
# include <stdio.h>
int main()
{
    puts("첫 C 프로그램!");
    return 0;
}
```

C 소스파일: *.c

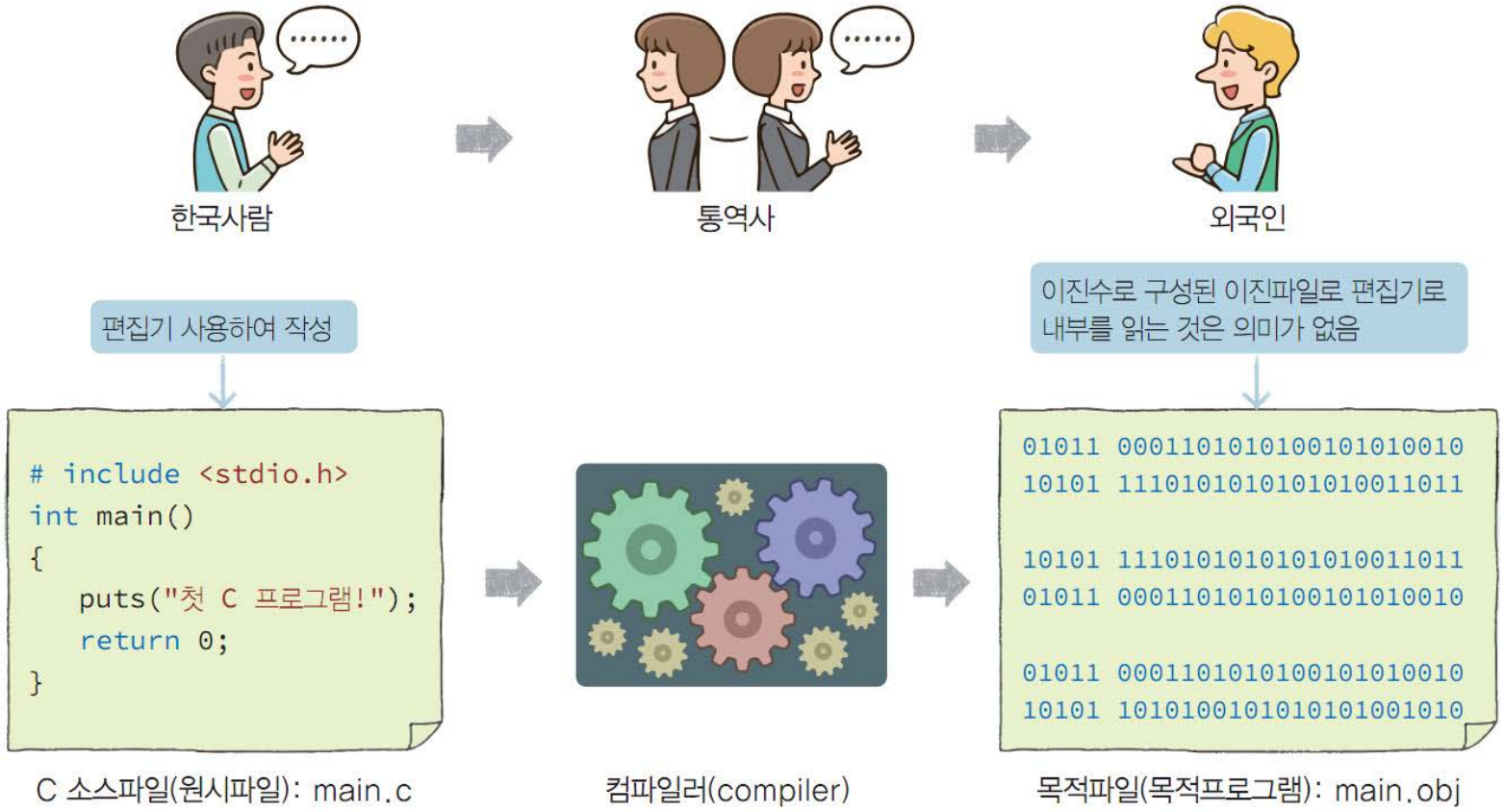
소스파일은 모든 편집기로 읽을 수 있으나 아래한글이나 워드로 작성된 일반 문서는 내부 형식이 다르므로 작성한 편집기로만 읽을 수 있다.

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

아래한글 파일: *.hwp

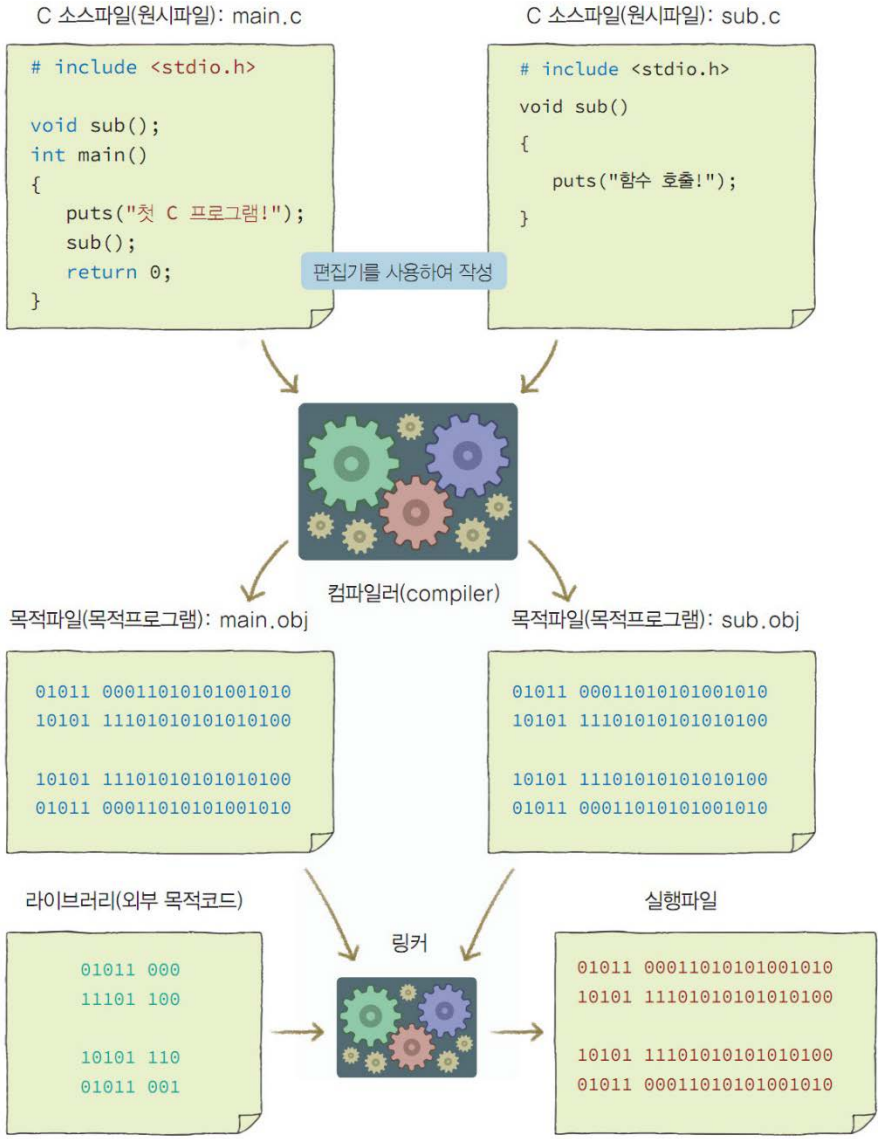
컴파일러

- 소스파일에서 기계어로 작성된 목적파일object file을 만들어내는 프로그램
- 컴파일러에 의해 처리되기 전의 프로그램을 소스코드source code라면 컴파일러에 의해 기계어로 번역된 프로그램은 목적코드object code



링크와 실행

- 링커linker
  - 하나 이상의 목적파일을 하나의 실행파일execute file로 만들어 주는 프로그램
  - 여러 개의 목적파일을 연결하고 참조하는 라이브러리를 포함시켜 하나의 실행파일을 생성
- 라이브러리library
  - 자주 사용하는 프로그램들은 프로그램을 작성할 때, 프로그래머마다 새로 작성할 필요 없이 개발환경에서 미리 만들어 컴파일해 저장해 놓는데, 이 모듈을 라이브러리library라 칭함



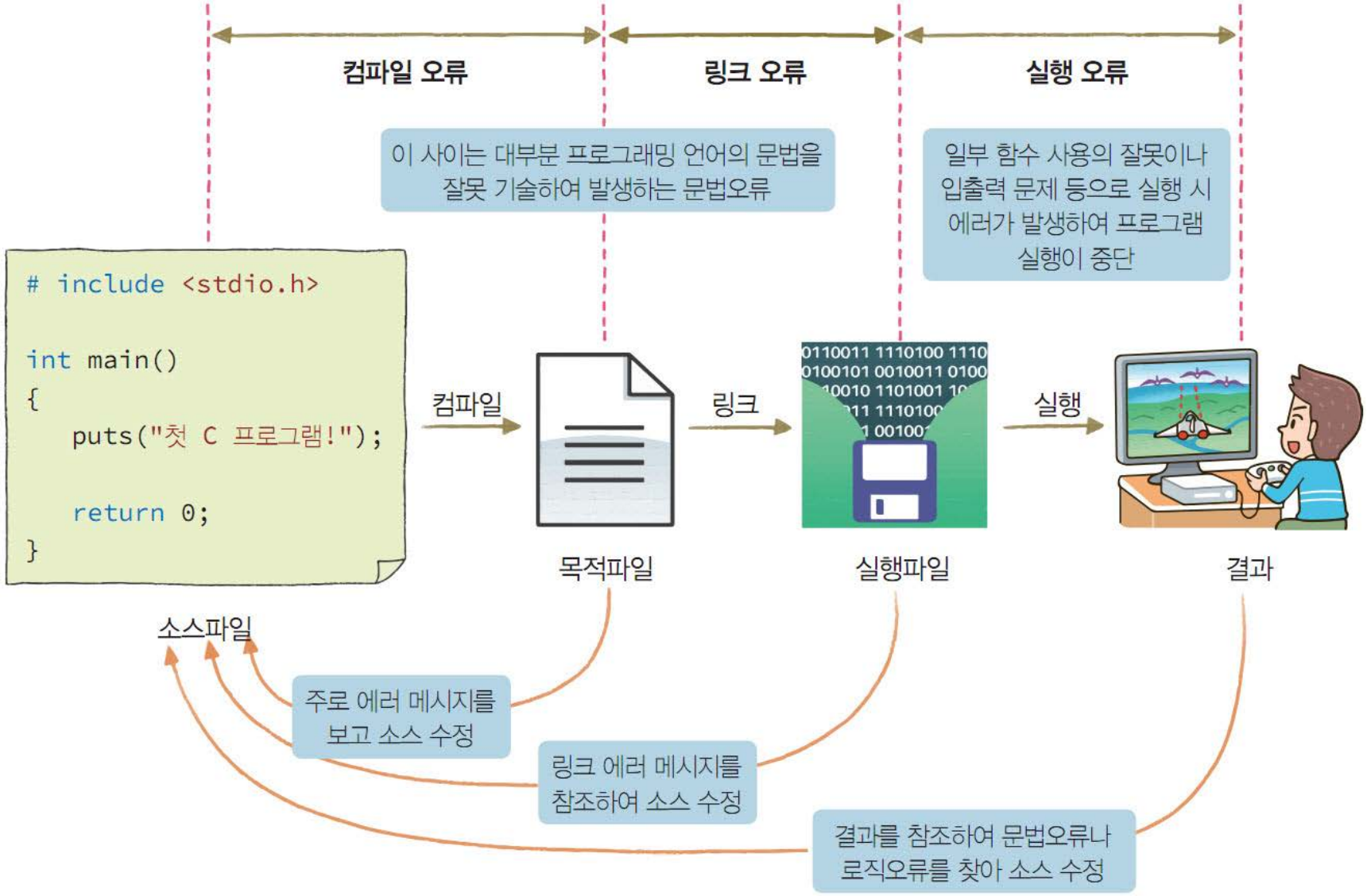
오류 또는 에러error

- 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 오류syntax error: 프로그래밍 언어 문법을 잘못 기술
    - 논리 오류logic error: 내부 알고리즘이 잘못되거나 원하는 결과가 나오지 않은 등의 오류

디버깅과 디버거

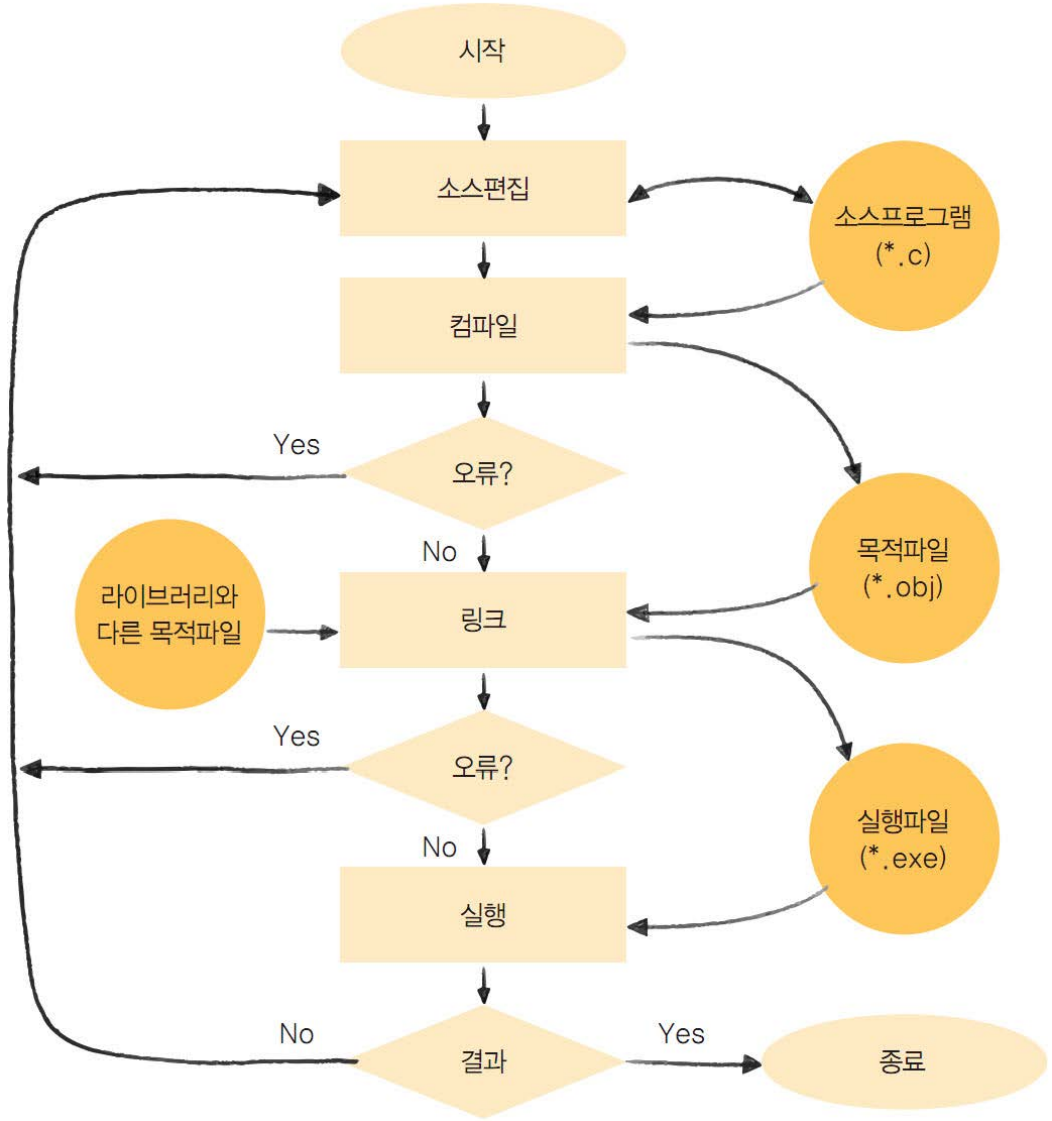
- 디버깅debugging
  - 프로그램 개발 과정에서 발생하는 오류를 찾아 소스를 수정하여 다시 컴파일, 링크, 실행하는 과정
- 디버거debugger
  - 디버깅을 도와주는 프로그램
  - 벌레라는 단어의 버그bug란 바로 오류

디버깅 과정


컴파일 오류
링크 오류
실행 오류
이 사이는 대부분 프로그래밍 언어의 문법을 잘못 기술하여 발생하는 문법오류
일부 함수 사용의 잘못이나 입출력 문제 등으로 실행 시 에러가 발생하여 프로그램 실행이 중단
include <stdio.h>
int main()
{
puts("첫 C 프로그램!");
return 0;
}
컴파일
링크
실행
목적파일
실행파일
결과
소스파일
주로 에러 메시지를 보고 소스 수정
링크 에러 메시지를 참조하여 소스 수정
결과를 참조하여 문법오류나 로직오류를 찾아 소스 수정

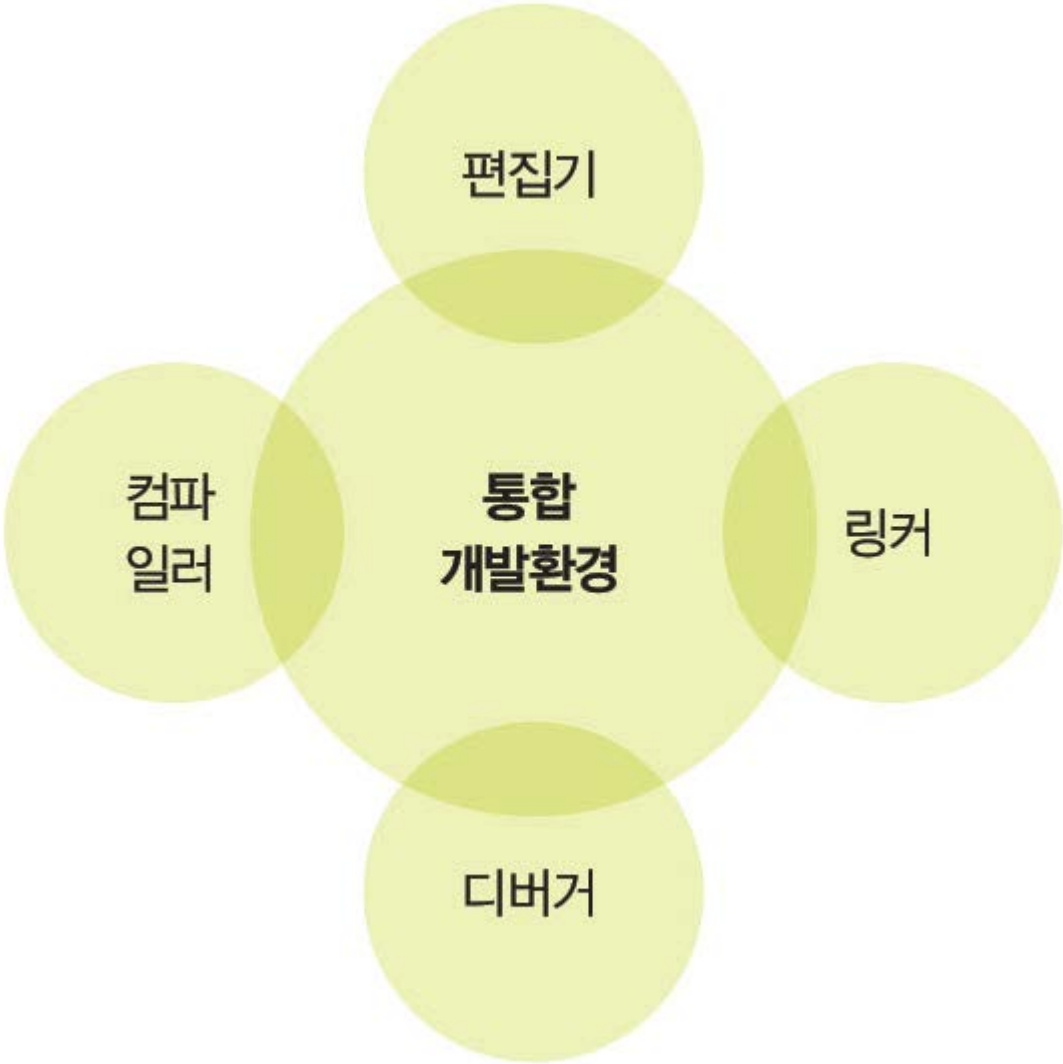
프로그램 구현 과정 순서도

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



통합개발환경 IDE Integrated Development Environment

- 프로그램 개발에 필요한 편집기editor, 컴파일러compiler, 링커linker, 디버거debugger 등을 통합하여 편리하고 효율적으로 제공하는 개발환경



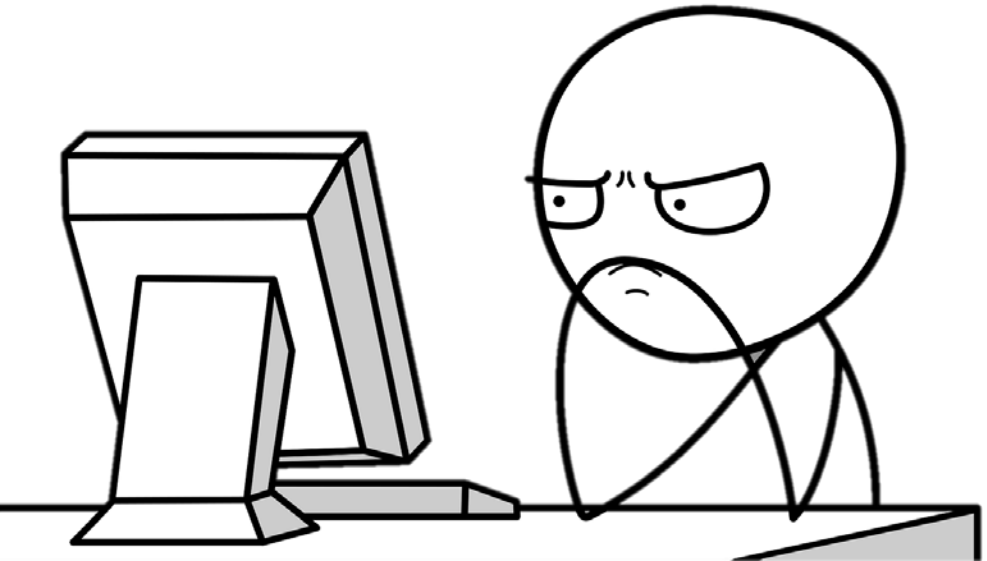
통합개발환경IDE - 마이크로소프트(MS) 비주얼 스튜디오

- 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 프로페셔널professional
  - 비주얼 스튜디오 엔터프라이즈enterprise
  - 무료버전: 비주얼 스튜디오 커뮤니티community

통합개발환경IDE - 이클립스 C/C++ 개발자용 IDE

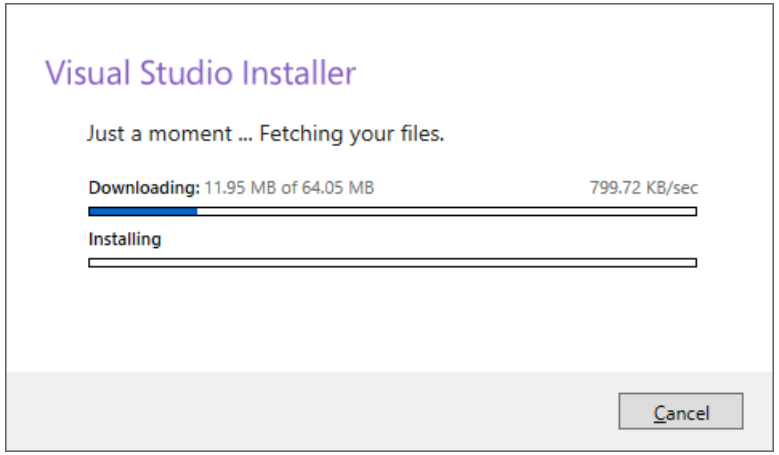
- IBM이 주도하는 이클립스 컨소시엄이 개발
- 모든 부분에 대해 개방형
  - PDEPlug-in Development Environment 환경을 지원하여 확장 가능한 통합개발환경
- C/C++ 개발자용 IDE(Eclipse IDE for C/C++ Developers)
  - C/C++를 개발하기 위한 개발도구로 컴파일러는 따로 설치
  - C/C++ 컴파일러로는 주로 공개 모듈인 GNU의 GCCGNU Compiler Collection를 이용

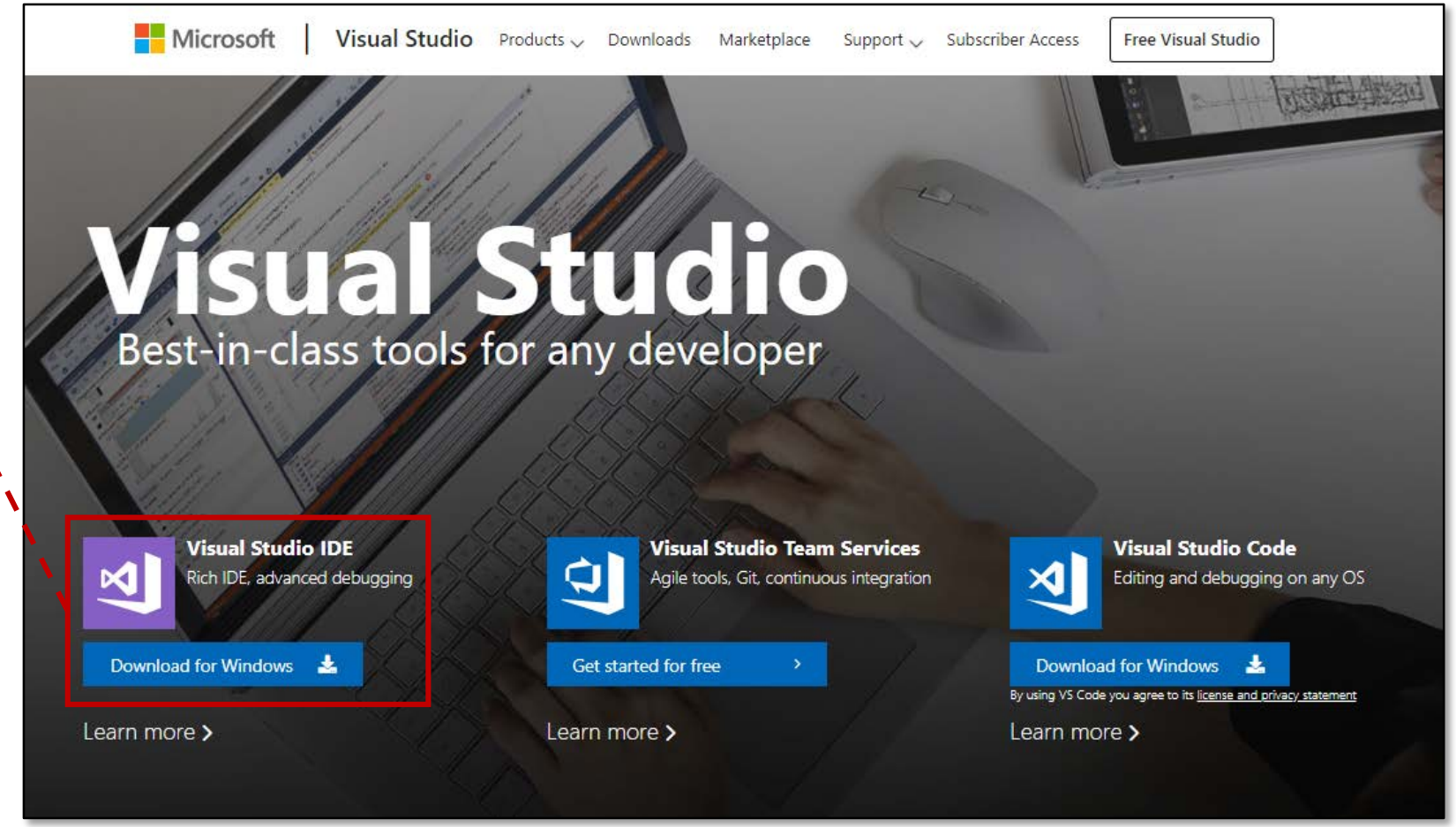
2. 비주얼 스튜디오 설치와 C 프로그램의 첫 개발



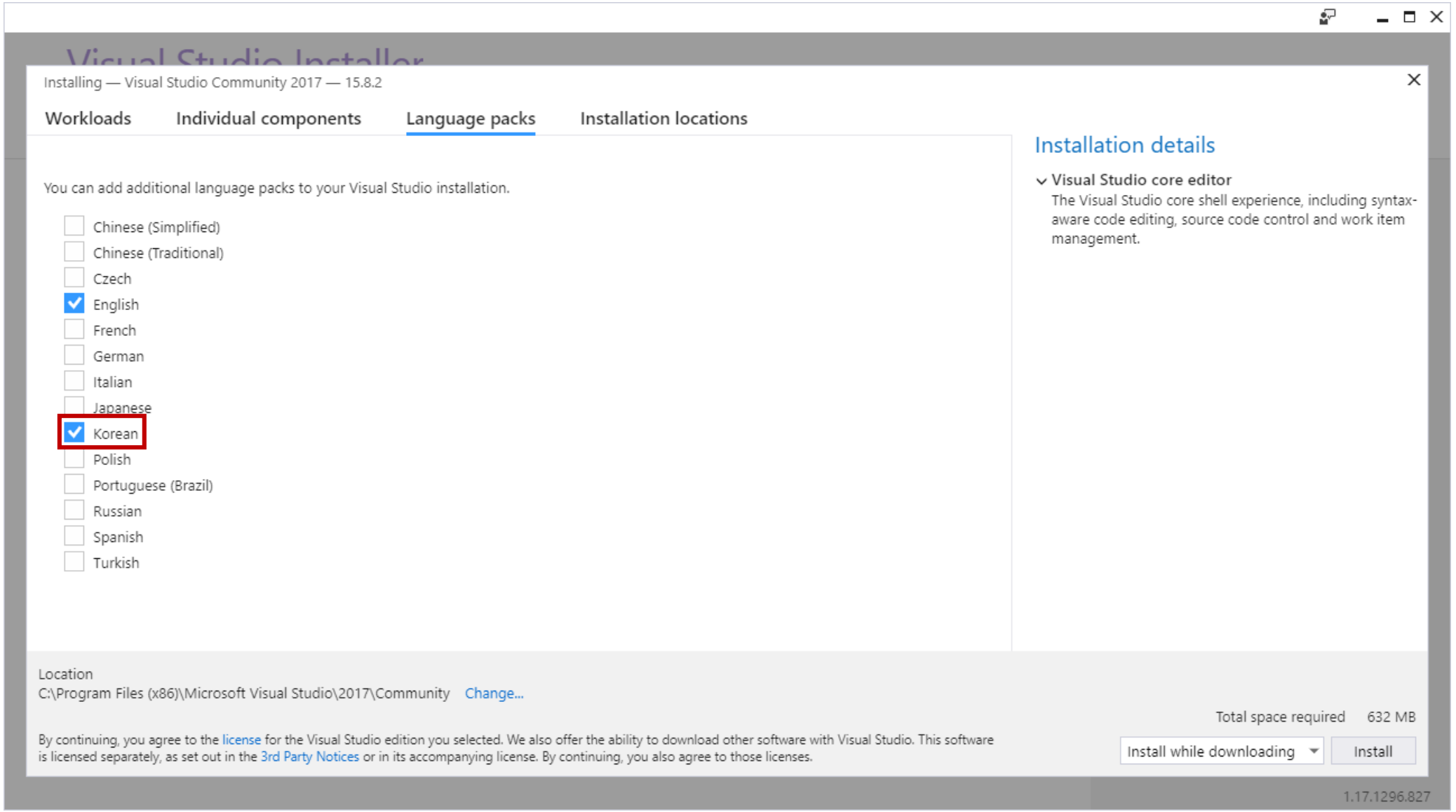
비주얼 스튜디오 커뮤니티 Visual Studio Community

- http://visualstudio.com (https://visualstudio.microsoft.com)

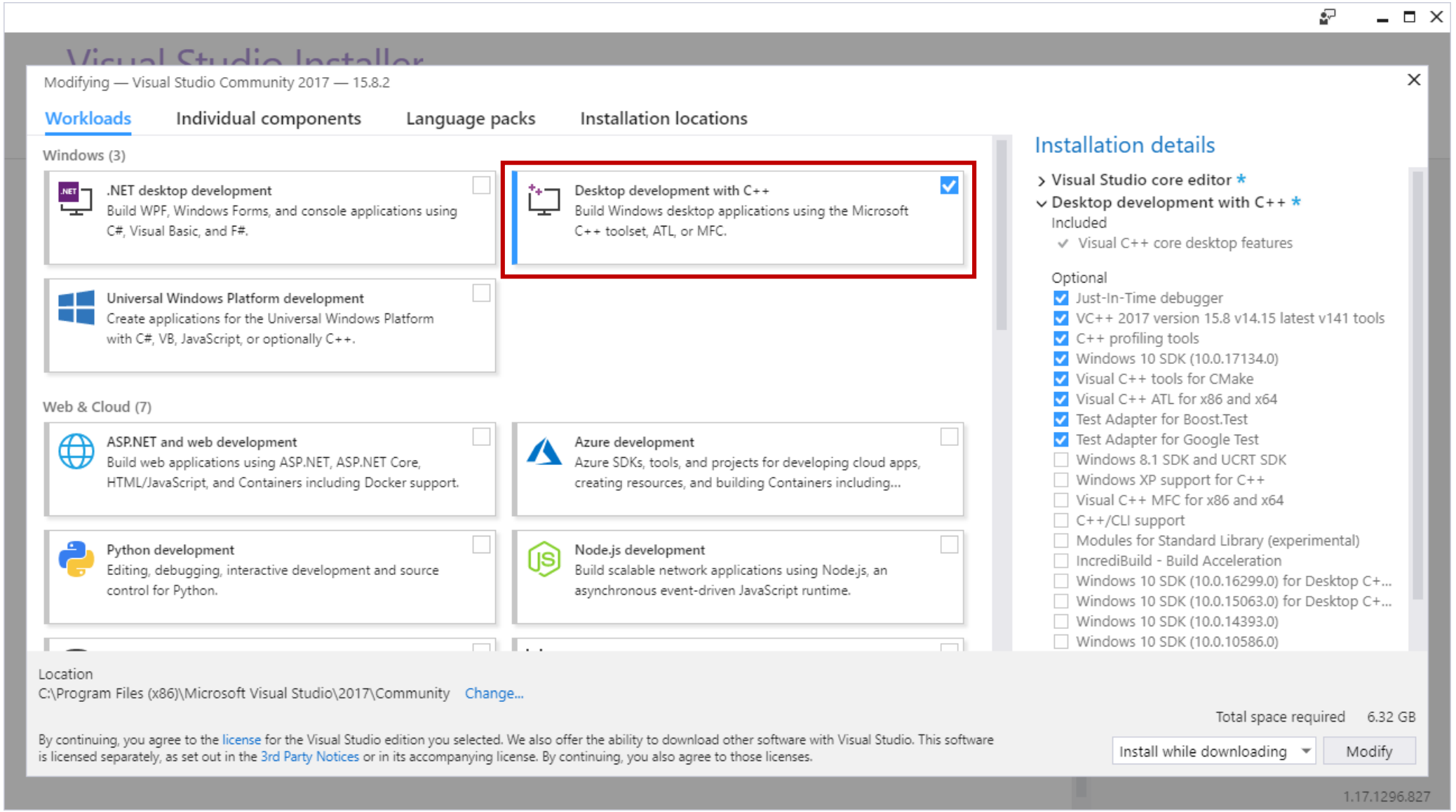




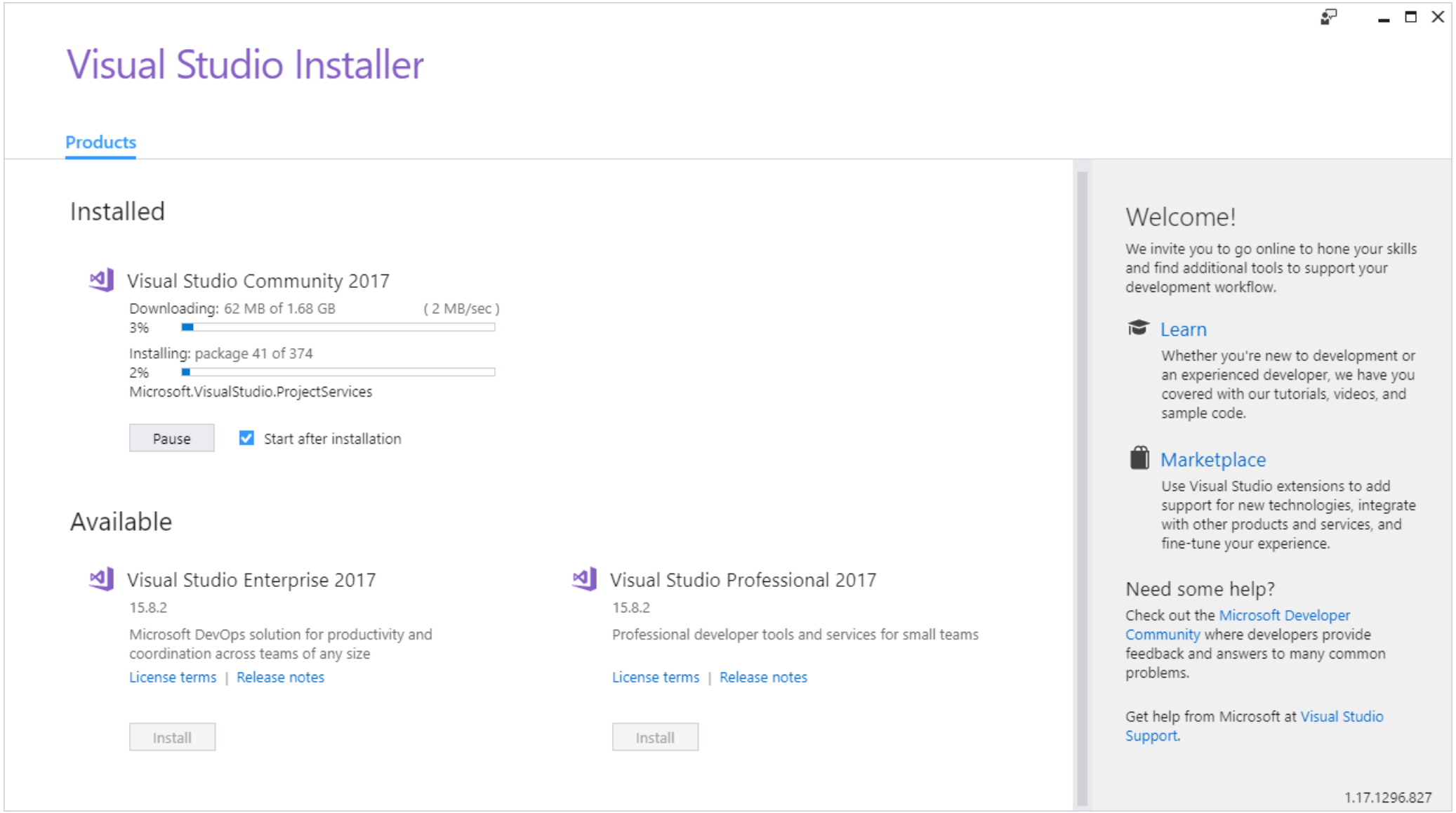
비주얼 스튜디오 커뮤니티Visual Studio Community



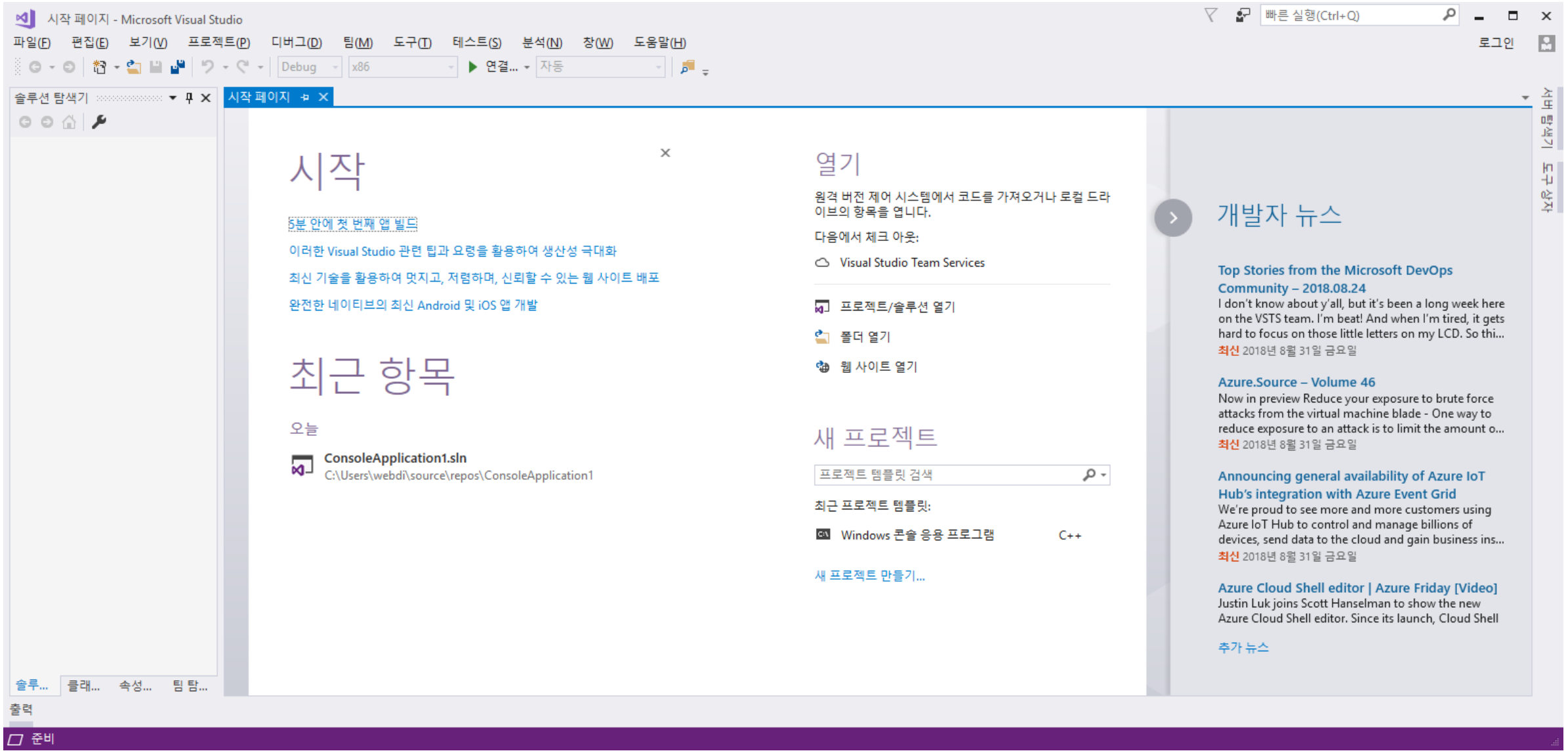
비주얼 스튜디오 커뮤니티 Visual Studio Community



비주얼 스튜디오 커뮤니티Visual Studio Community

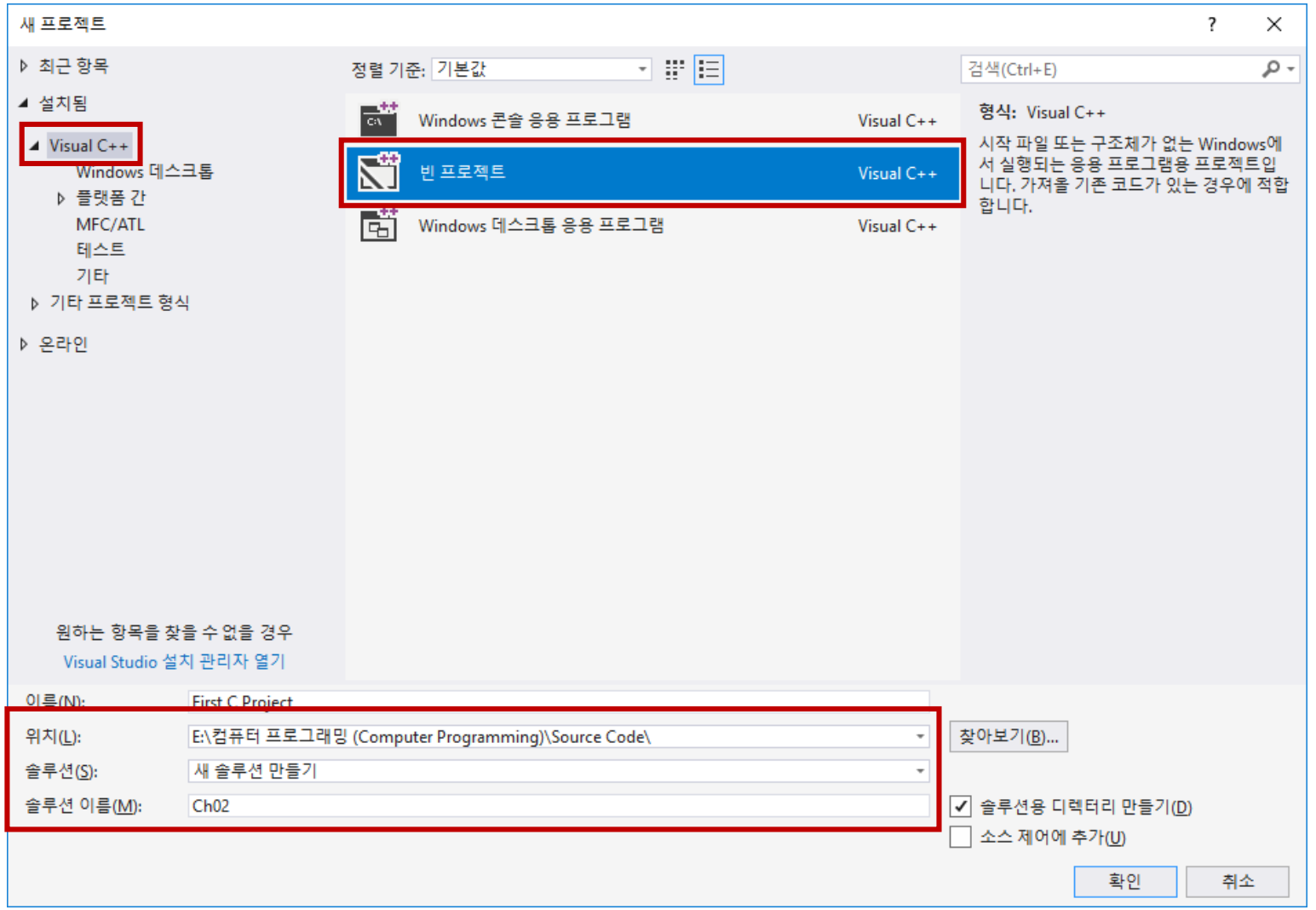


비주얼 스튜디오 커뮤니티Visual Studio Community



비주얼 스튜디오의 솔루션과 프로젝트 생성

- 메뉴 [파일] → [새로 만들기] → [프로젝트]를 선택
- 프로젝트 형식
  - 선택된 템플릿 'Visual C++'에서 'Win32 콘솔 응용프로그램'으로 선택
- 이름: 'First C Project'로 지정
- 위치: 솔루션과 프로젝트 관련 폴더와 여러 파일이 저장될 상위 폴더, 위치에 지정되는 폴더는 없는 경우 자동으로 생성
- 솔루션 이름: 단원이름 'Ch02'를 지정

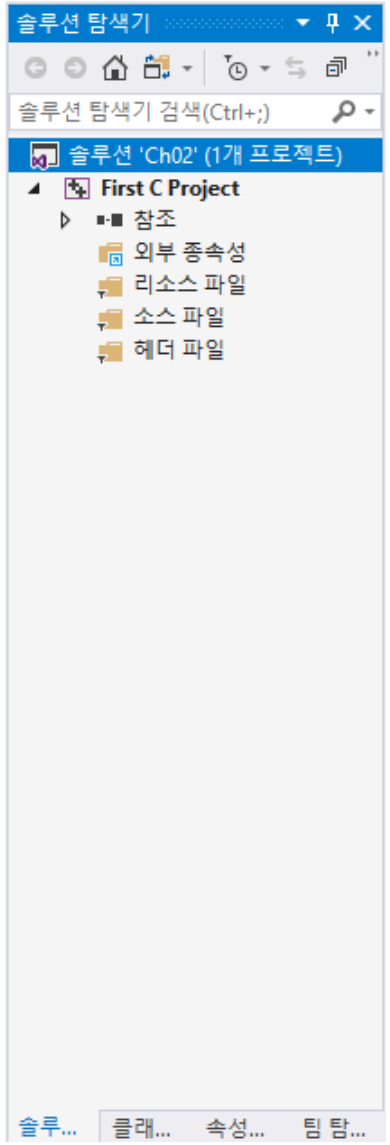


비주얼 스튜디오의 솔루션과 프로젝트 생성

프로젝트 생성을 위한 여러 설정

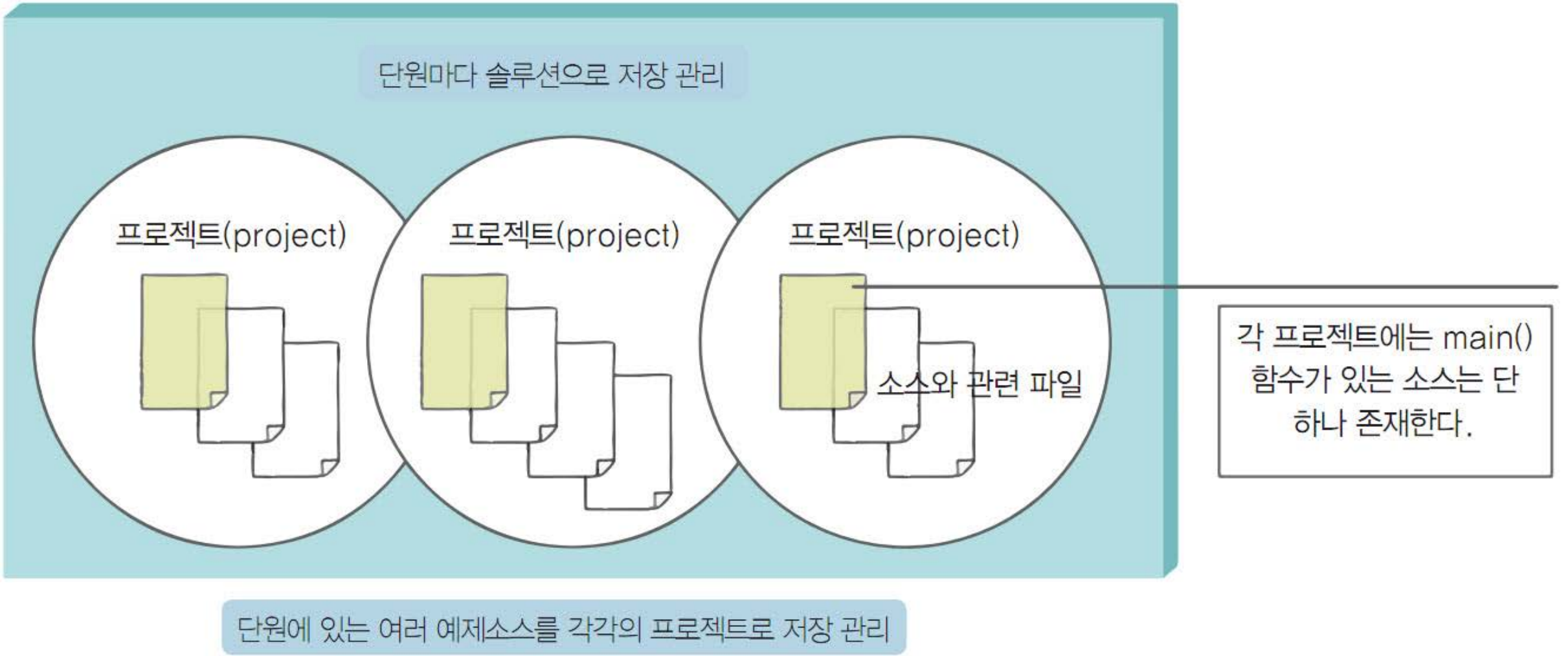
주요 설정	설명	설정 내용
템플릿	개발하려는 환경	Visual C++
프로젝트 형식	다양한 프로젝트 형식 중 하나 선택	Windows 콘솔 응용 프로그램
이름	만들려는 프로젝트 이름을 선택	First C Project
위치	솔루션과 프로젝트가 저장되는 폴더	D:₩CP
솔루션 이름	만들려는 솔루션 이름을 입력	Ch02
솔루션을 디렉토리로 만들기	솔루션을 폴더로 지정하려면 체크	체크

솔루션 탐색기

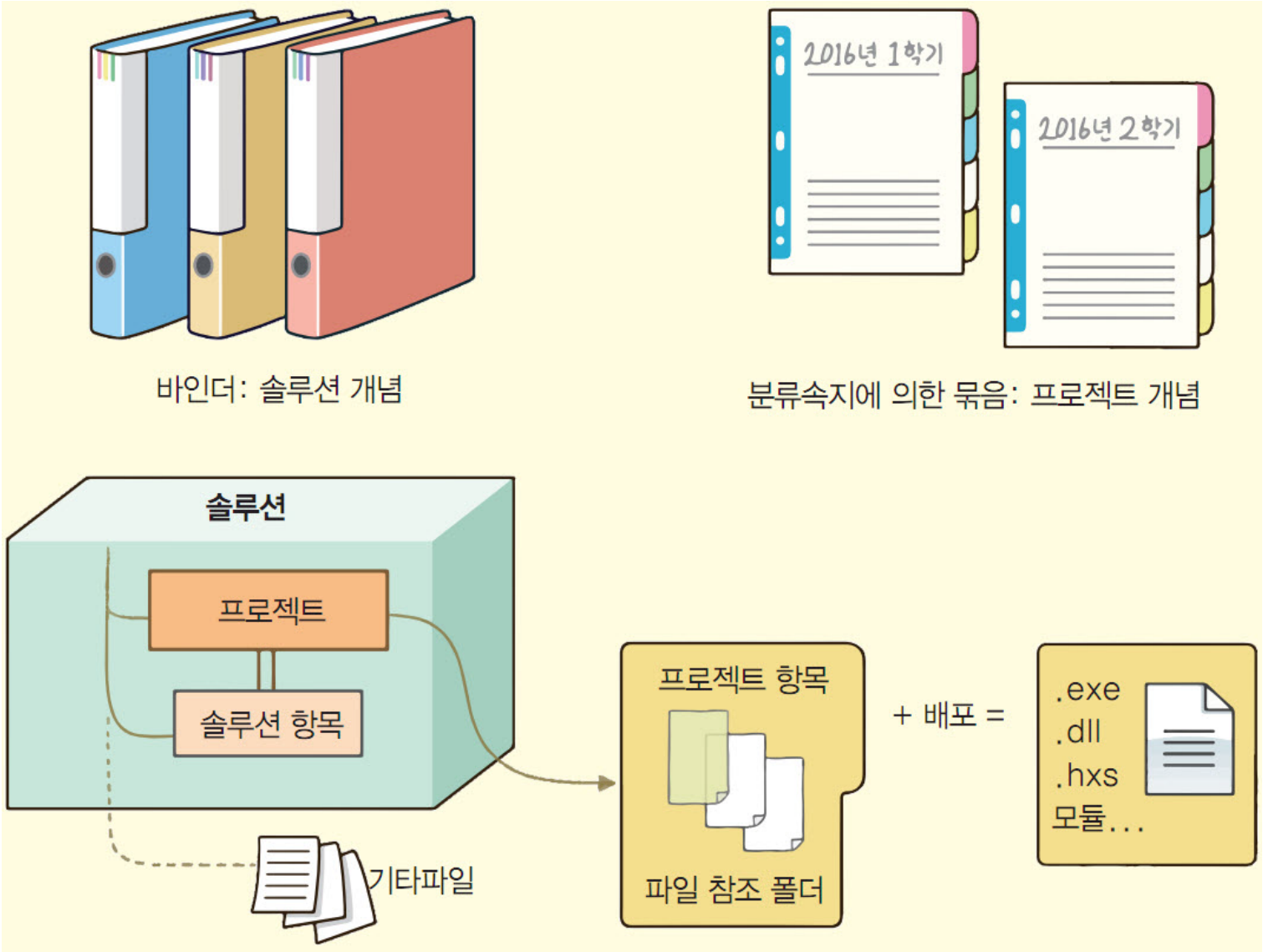


- 생성된 솔루션과 프로젝트 표시
  - 전체 솔루션의 그래픽 뷰를 제공하여 응용 프로그램을 개발할 때 솔루션의 프로젝트와 파일을 쉽게 관리할 수 있도록 도움
- 프로젝트 하단부
  - 관련 폴더인 리소스 파일, 소스 파일, 외부 종속성, 참조, 헤더 파일로 나뉨

솔루션과 프로젝트 관리

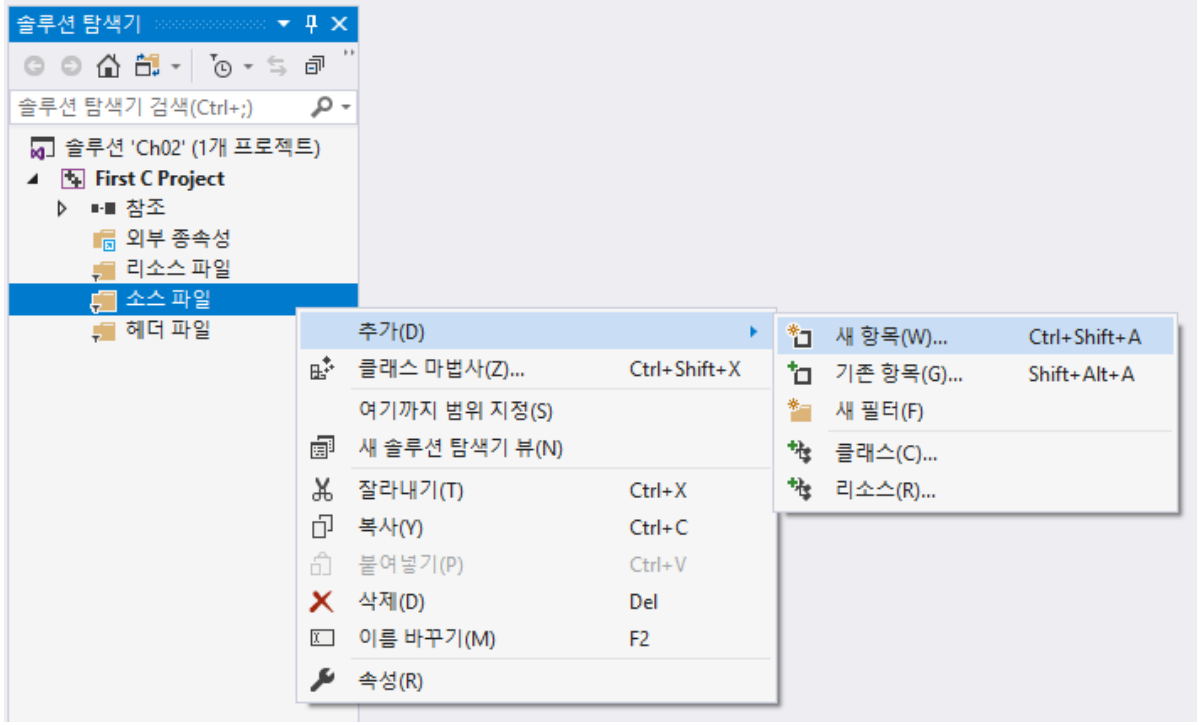

단원마다 솔루션으로 저장 관리
프로젝트(project)
프로젝트(project)
프로젝트(project)
소스와 관련 파일
각 프로젝트에는 main() 함수가 있는 소스는 단 하나 존재한다.
단원에 있는 여러 예제소스를 각각의 프로젝트로 저장 관리

솔루션과 프로젝트의 이해

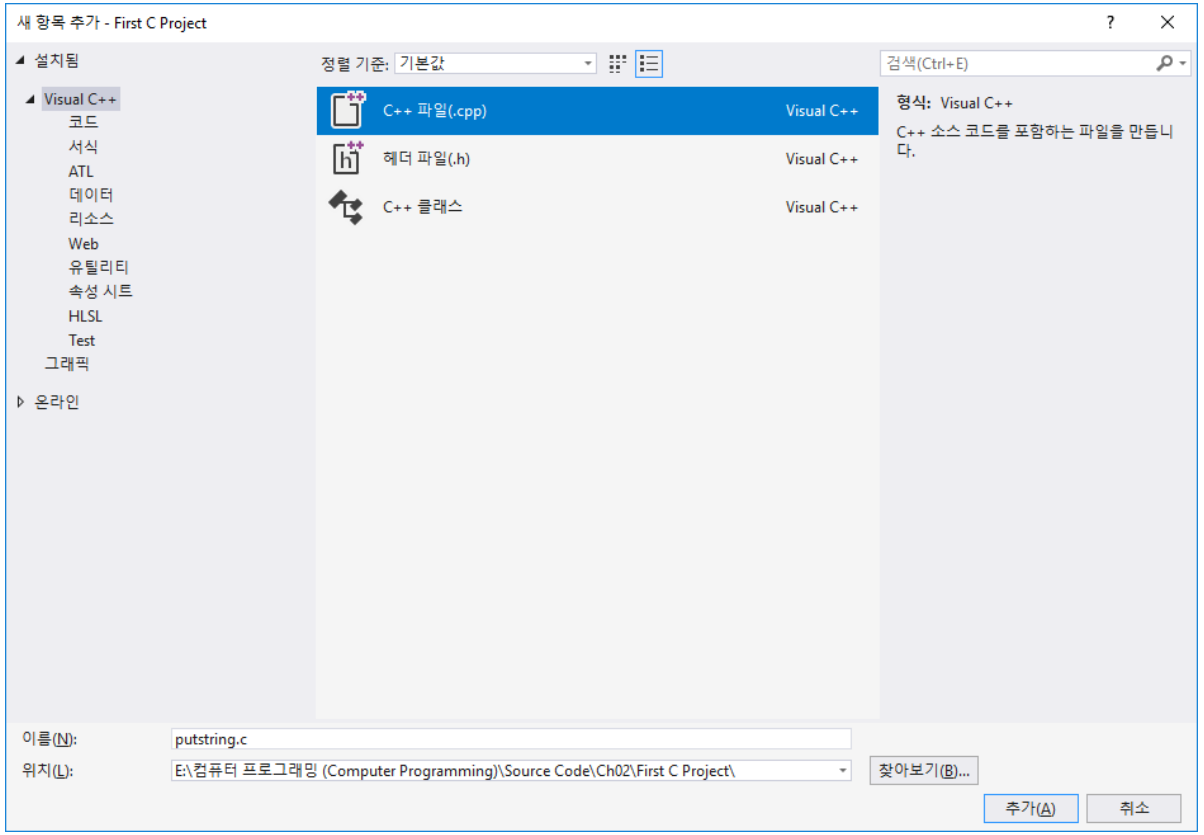


소스 파일 추가

- 메뉴 [프로젝트] → [새 항목 추가]를 선택
- '솔루션 탐색기'의 '소스파일' 폴더, 마우스 오른쪽을 클릭, 메뉴 [추가] → [새 항목]을 선택



소스 파일 추가



- 각각 'Visual C++'와 'C++ 파일 (cpp)'을 선택
- '이름'에 소스파일 이름 putstring.c를 입력
- 파일이름에 반드시 확장자 .c를 입력
- 위치: '솔루션 폴더/프로젝트 폴더'인 'Ch02/First C Project'을 확인한 후 [추가]를 누름

Source Code #01: putstring.c

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

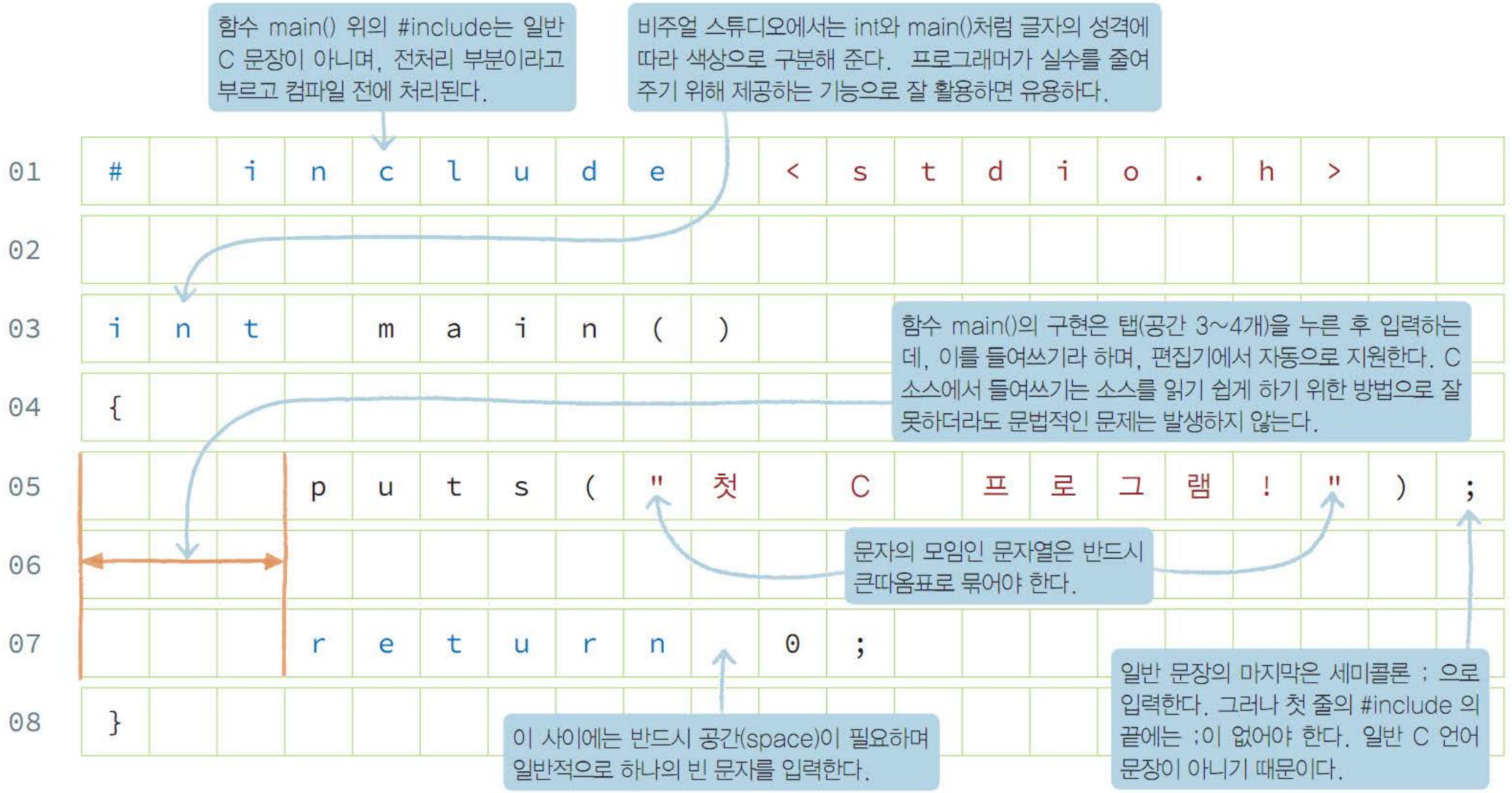
```c
#include <stdio.h>

int main()
{
    puts("첫 c 프로그램!");

    return 0;

}

```

```
첫 C 프로그램!
```

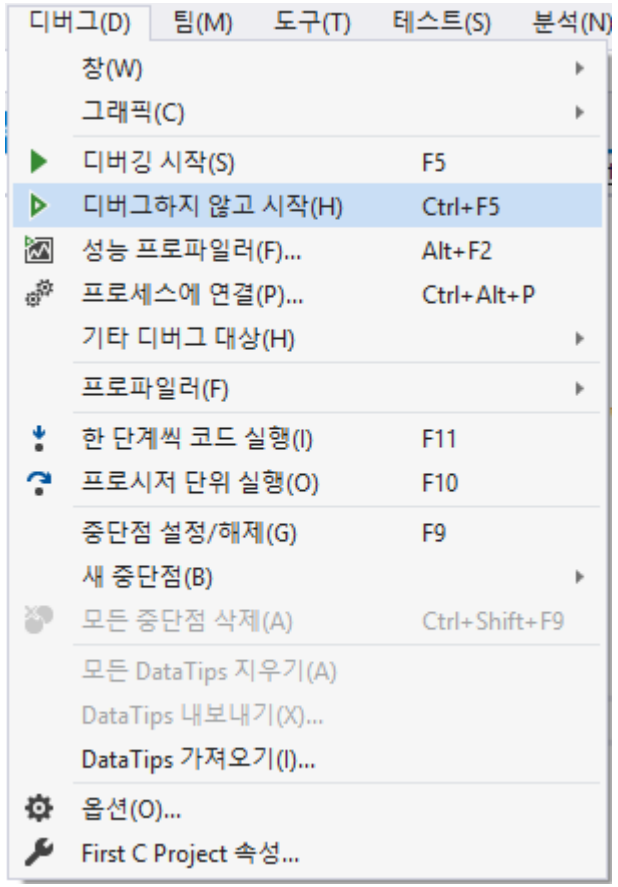
프로그래밍 주의점

- 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
  - # < > ( ) { } ; ""
- 컴파일러
  - 컴파일러는 하나의 오타도 허용하지 않음
  - 편집기에서 주의를 기울여, 행과 열을 맞추어 정확히 소스를 입력
  - 문장의 종료를 표시하는 세미콜론 ;을 콜론 :으로 잘못 입력하면 컴파일에 문제가 발생

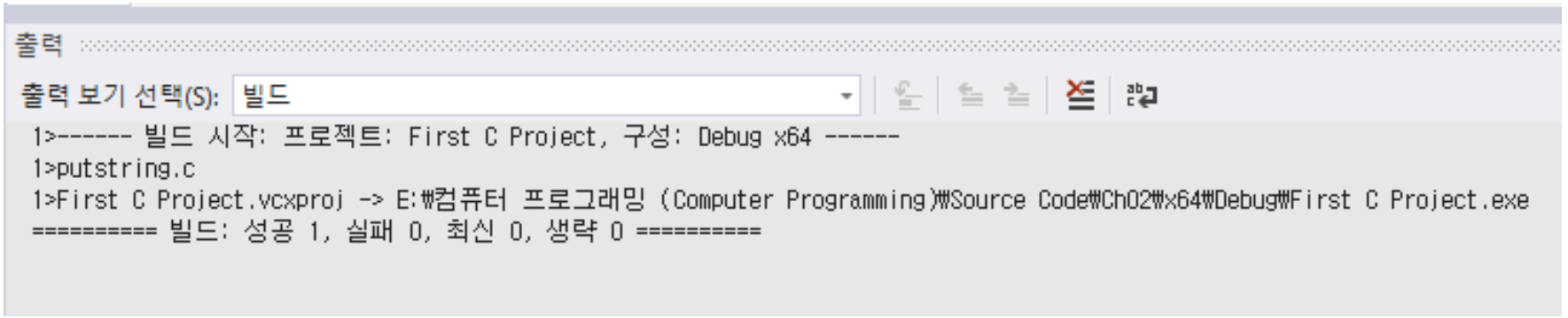
프로그래밍 주의점


함수 main() 위의 #include는 일반 C 문장이 아니며, 전처리 부분이라고 부르고 컴파일 전에 처리된다.
비주얼 스튜디오에서는 int와 main()처럼 글자의 성격에 따라 색상으로 구분해 준다. 프로그래머가 실수를 줄여 주기 위해 제공하는 기능으로 잘 활용하면 유용하다.
01 # include <stdio.h>
02
03 int main()
04 {
05 puts("첫 C 프로그램!");
06
07 return 0;
08 }
함수 main()의 구현은 탭(공간 3~4개)을 누른 후 입력하는데, 이를 들여쓰기라 하며, 편집기에서 자동으로 지원한다. C 소스에서 들여쓰기는 소스를 읽기 쉽게 하기 위한 방법으로 잘 못하더라도 문법적인 문제는 발생하지 않는다.
문자의 모임인 문자열은 반드시 큰따옴표로 묶어야 한다.
이 사이에는 반드시 공간(space)이 필요하며 일반적으로 하나의 빈 문자를 입력한다.
일반 문장의 마지막은 세미콜론 ; 으로 입력한다. 그러나 첫 줄의 #include 의 끝에는 ;이 없어야 한다. 일반 C 언어 문장이 아니기 때문이다.

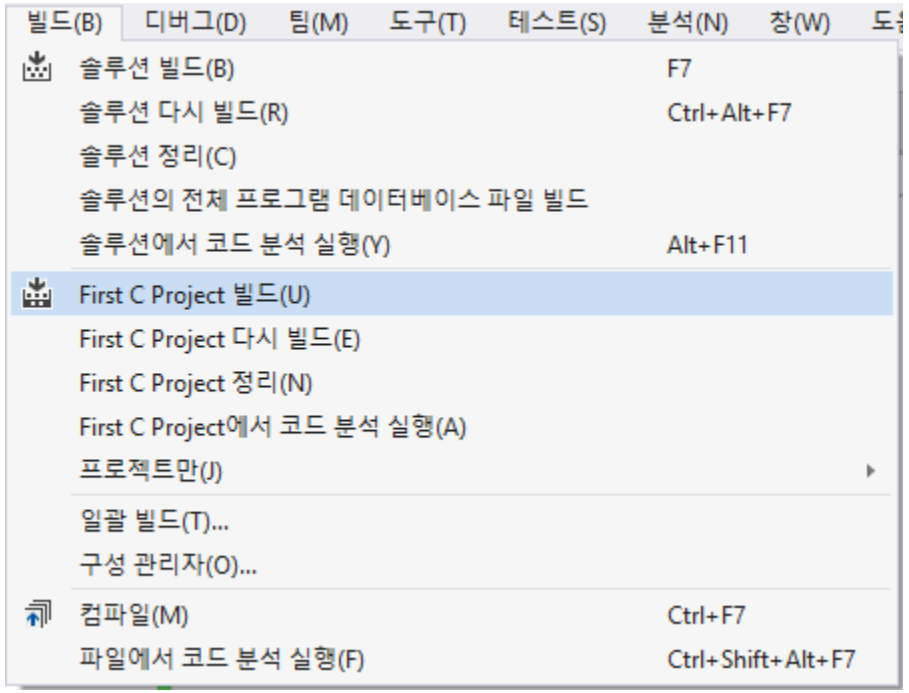
프로젝트 실행



- 작성된 소스에 문제가 없는지 확인
- 메뉴 [디버그] → [디버그하지 않고 시작]을 선택 (Ctrl + F5)
- 빌드를 묻는 대화상자에서 [예]를 눌러 실행
- 출력에 빌드 과정이 표시되고, 성공 1, 실패 0과 같이 표시

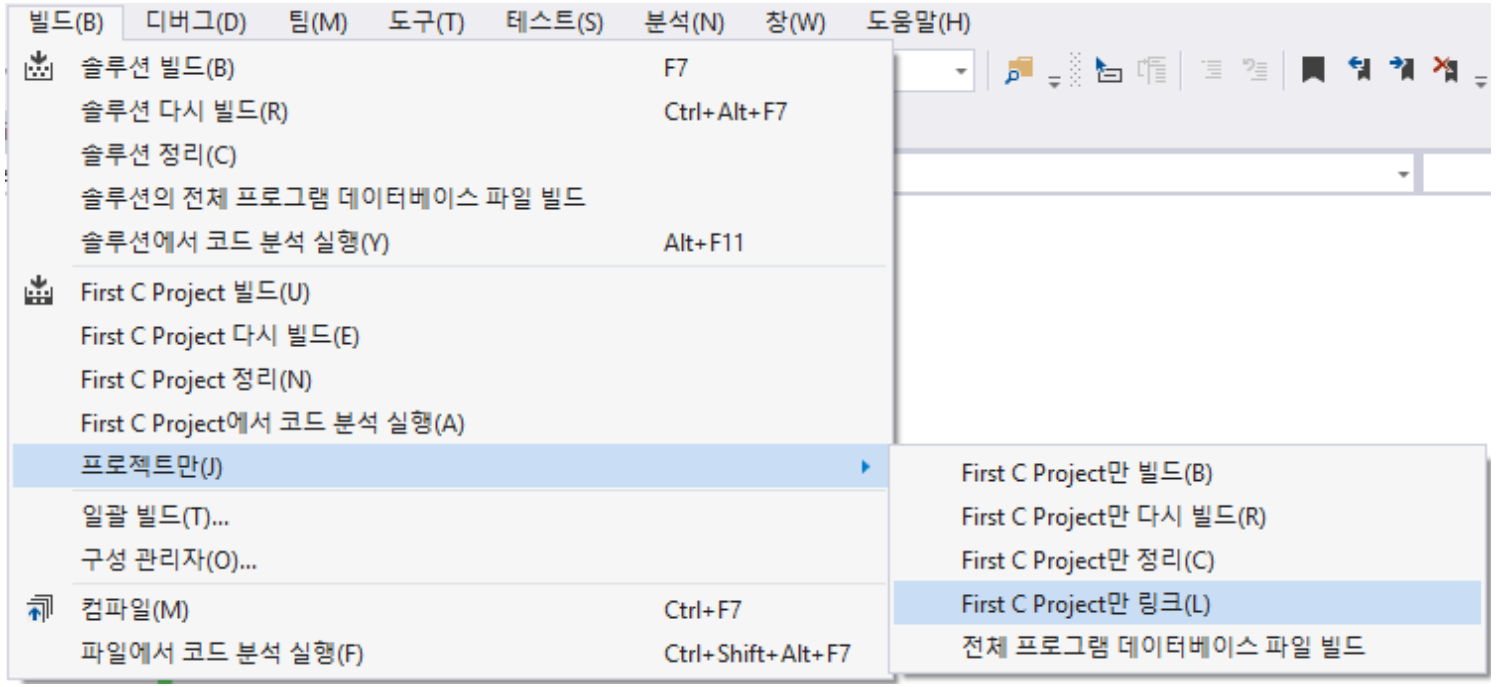


빌드와 컴파일



- 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

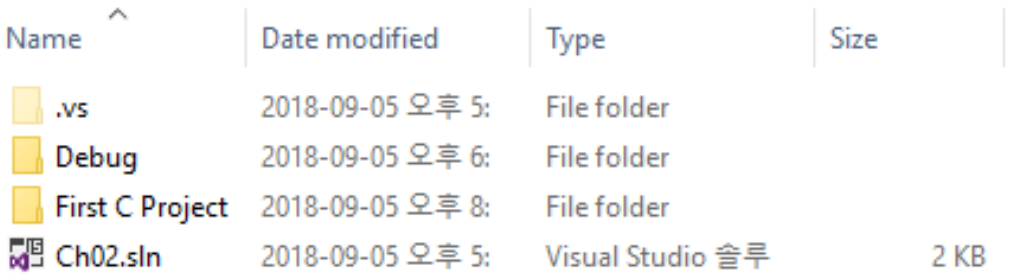
링크



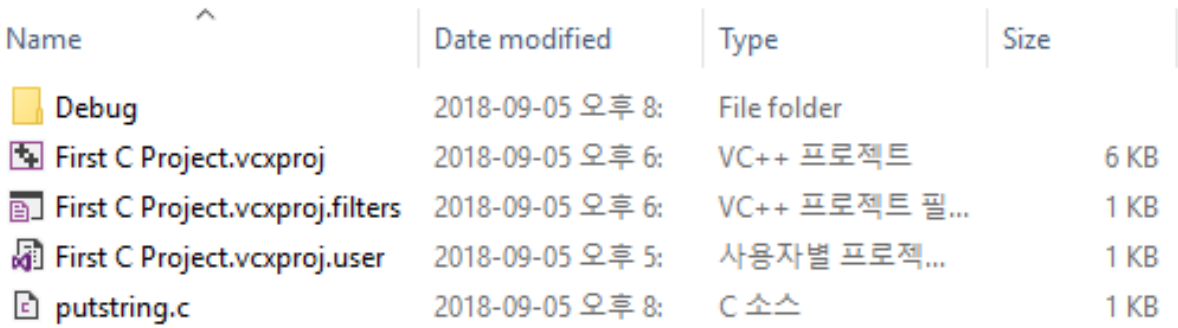
- 컴파일 후 메뉴 [빌드] → [프로젝트만] → [First C Project만 링크]를 선택
  - 링크만 구분하여 실행

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'



Name	Date modified	Type	Size
.vs	2018-09-05 오후 5:	File folder	
Debug	2018-09-05 오후 6:	File folder	
First C Project	2018-09-05 오후 8:	File folder	
Ch02.sln	2018-09-05 오후 5:	Visual Studio 솔루	2 KB

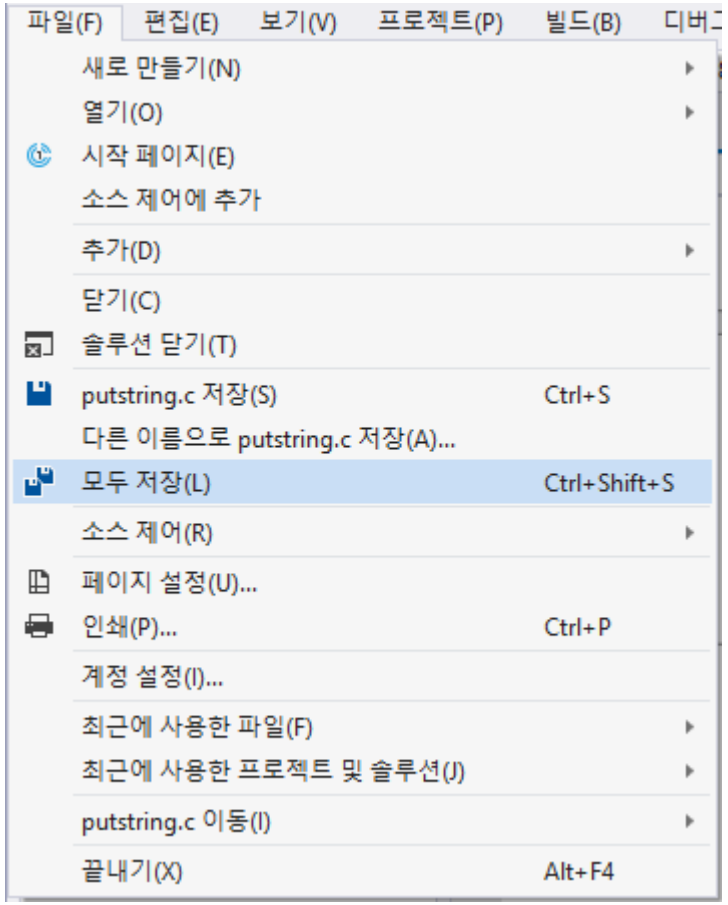


Name	Date modified	Type	Size
Debug	2018-09-05 오후 8:	File folder	
First C Project.vcxproj	2018-09-05 오후 6:	VC++ 프로젝트	6 KB
First C Project.vcxproj.filters	2018-09-05 오후 6:	VC++ 프로젝트 필...	1 KB
First C Project.vcxproj.user	2018-09-05 오후 5:	사용자별 프로젝...	1 KB
putstring.c	2018-09-05 오후 8:	C 소스	1 KB

비주얼 스튜디오 생성 파일

파일명.확장자명	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	Ch02₩First C Project
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	Ch02₩Debug
First C Proejct.pdb	디버그	프로그램 디버그를 위한 데이터베이스 파일	Ch02₩Debug

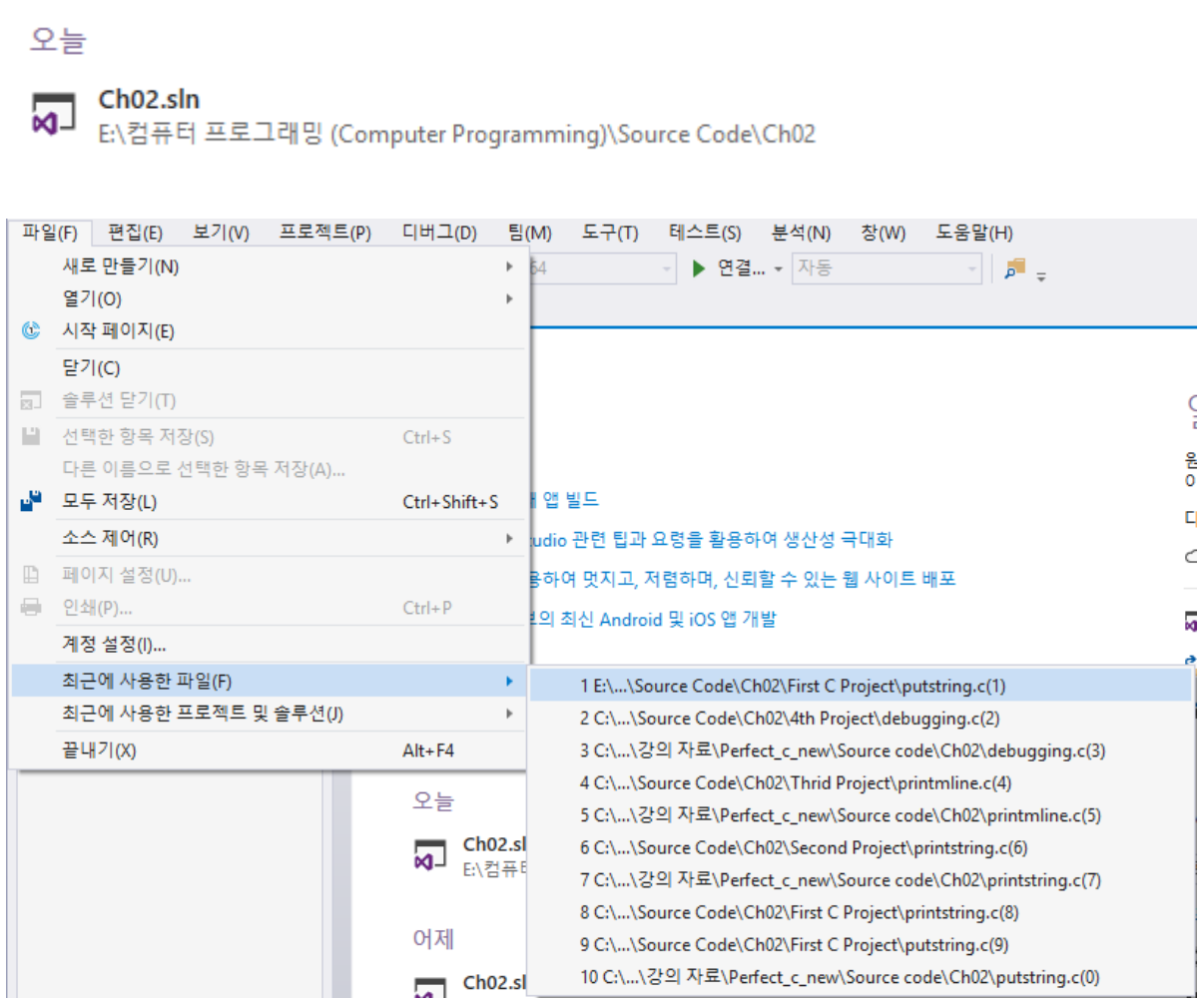
솔루션 저장 및 비주얼 스튜디오 종료



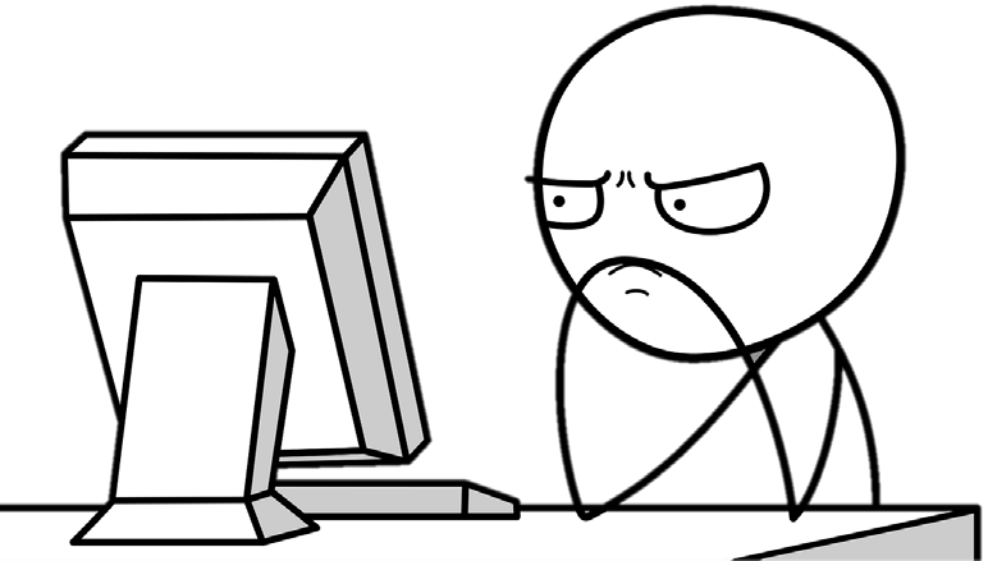
- 솔루션 저장
  - 프로젝트를 마치려면 메뉴 [파일] → [모두 저장]을 누름
- 비주얼 스튜디오 종료
  - [파일] → [끝내기]를 선택

생성된 솔루션/프로젝트 열기

- 메뉴 [파일] → [최근에 사용한 프로젝트 및 솔루션]을 선택
- 일반적으로 메뉴 [파일] → [열기] → [프로젝트/솔루션]을 선택
- 솔루션 파일은 확장자가 .sln이며, 프로젝트 파일은 .vcxproj



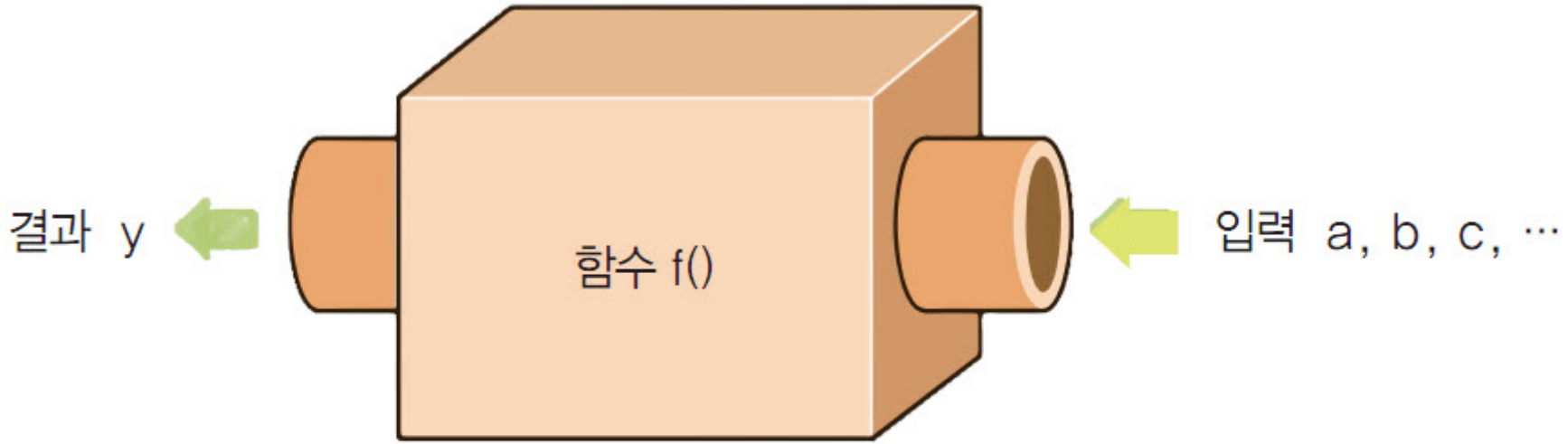
3. C 프로그램의 이해와 디버깅 과정



함수의 이해

- 함수 개요와 시작함수 main()
  - C 프로그램의 시작과 끝은 함수
  - C 프로그램과 같은 절차지향 프로그램은 함수function로 구성
  - 함수 하나하나가 프로그램 단위
- 함수: 입력과 출력
  - 함수는 'a, b, c...'와 같은 입력input을 받아
  - 'y'와 같은 결과output 값을 만들어 내는 기계장치와 유사
  - '입력'은 여러 개 사용될 수 있지만 결과값은 꼭 하나여야 한다는 점
- 사용자 정의 함수user defined function
  - 프로그래머가 직접 만드는 함수
- 라이브러리 함수library function
  - 시스템이 미리 만들어 놓은 함수

함수의 이해



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

함수관련 용어

- 함수 정의function definition: 사용자 정의 함수를 만드는 과정
- 함수 호출function call: 라이브러리 함수를 포함해서 만든 함수를 사용하는 것
- 매개변수parameters: 함수를 정의할 때 나열된 여러 입력 변수
- 인자argument: 함수 호출 과정에서 전달되는 여러 입력값

함수관련 용어

- 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

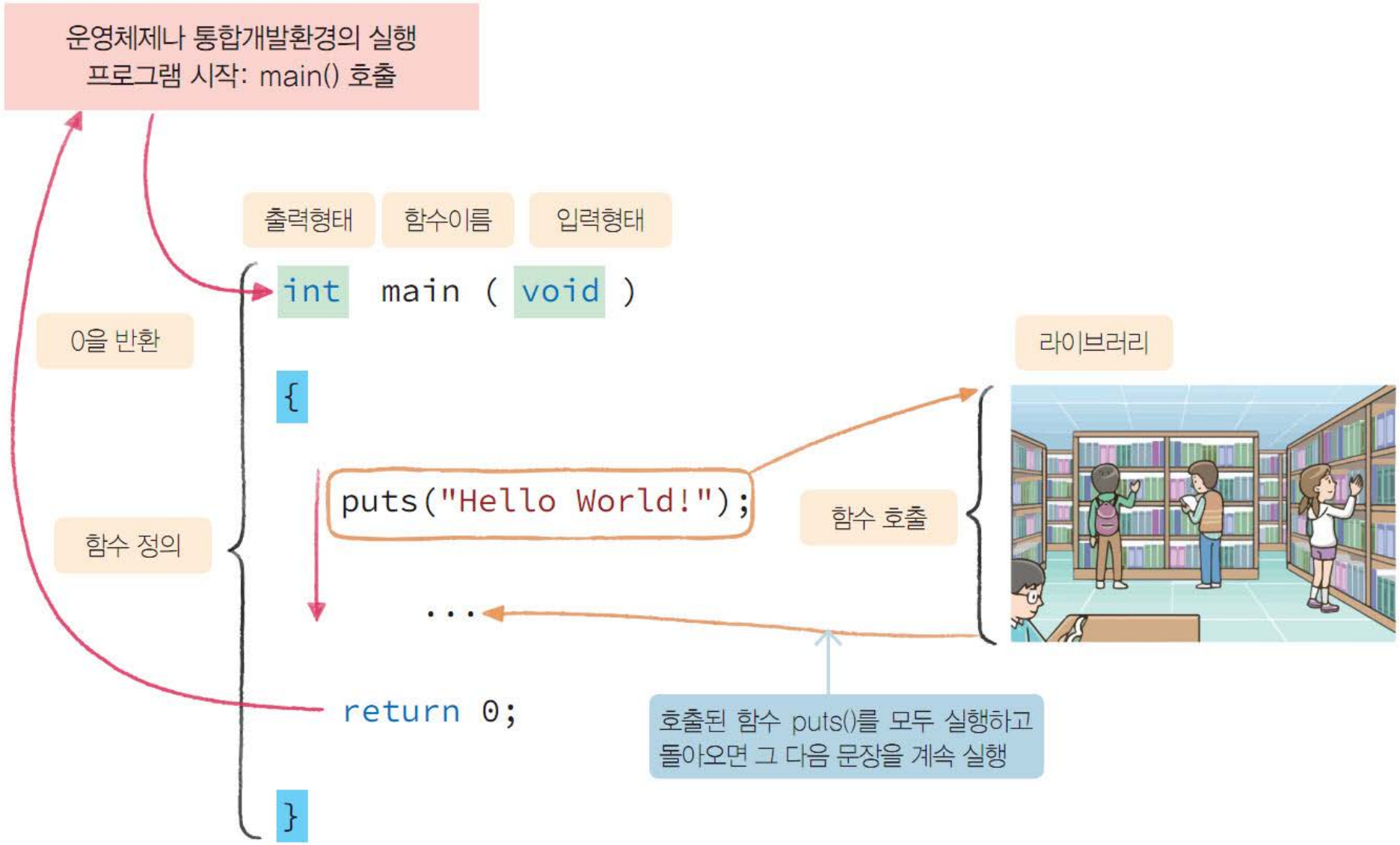
main()함수의 정의

- 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 전달
  - puts()를 실행한 후 다시 main()으로 돌아옴
  - 그 다음 줄인 return 0;을 실행
- CRT C Runtime Startup function 시작함수
  - 프로그램 실행 시 가장 먼저 호출되는 특별한 함수
  - 반환값: 함수 main()은 정상적인 작업을 마치면 정수 0을 반환

시작함수 main() 예제


운영체제나 통합개발환경의 실행
프로그램 시작: main() 호출
출력형태
함수이름
입력형태
int main (void)
{
puts("Hello World!");
...
return 0;
}
0을 반환
함수 정의
함수 호출
라이브러리
호출된 함수 puts()를 모두 실행하고
돌아오면 그 다음 문장을 계속 실행

Source Code #02: printstring.c

- 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

```c
#include <stdio.h>

int main(void)
{
    printf("Hello World!\n");

    return 0;
}

```

```
Hello World!
```

함수의 머리와 몸체

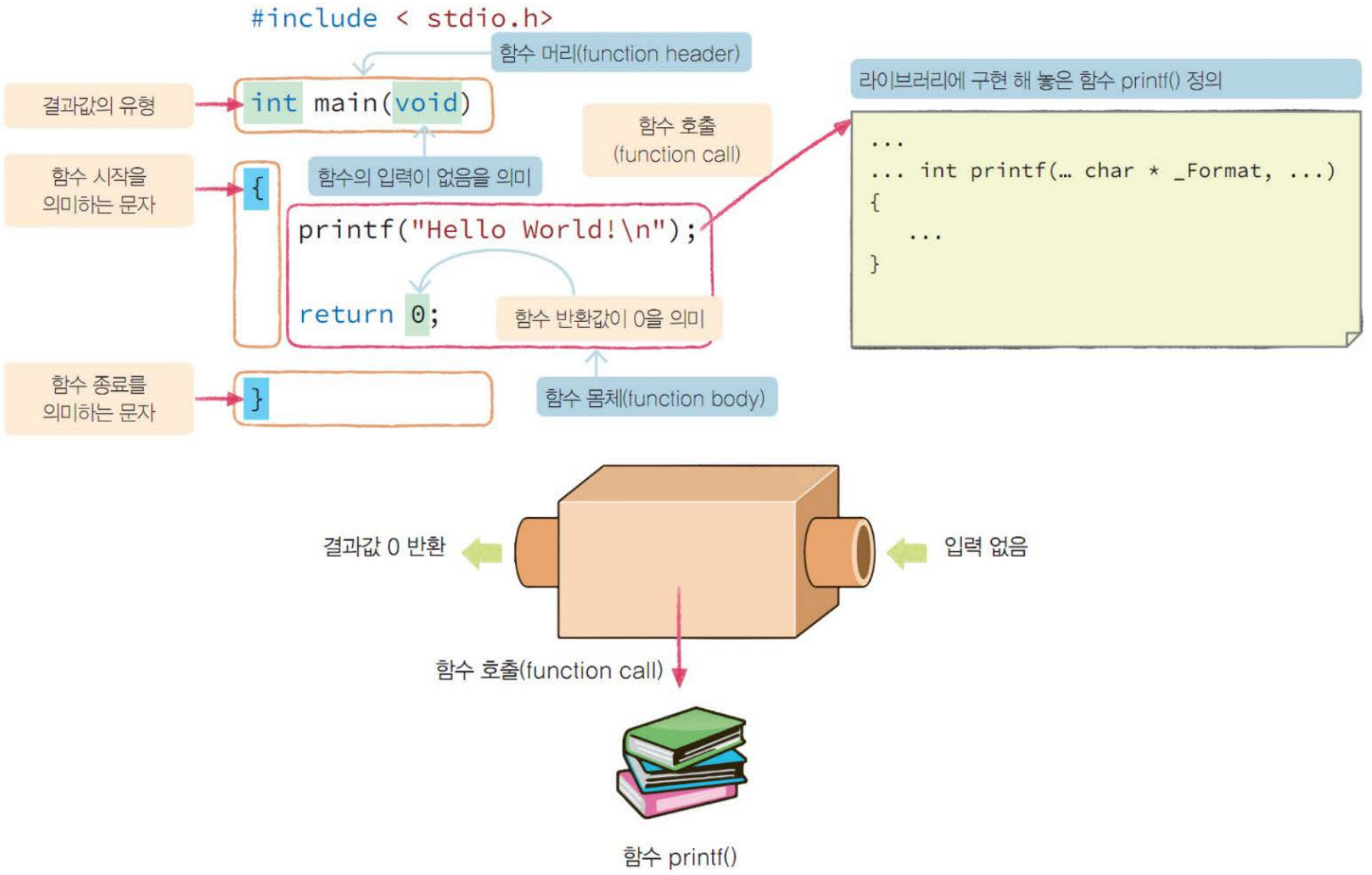
- C프로그램에서 main() 함수
  - 자동차에 시동을 켜는 열쇠와 같은 역할
  - 반드시 정의되어야 함
- 함수 구현(정의)
  - 함수 머리function header
    - int main(void)와 같이 함수에서 제일 중요한 결과값의 유형, 함수이름, 매개변수인 입력 변수 나열을 각각 표시
  - 함수 몸체function body
    - 함수 머리 이후 {...}의 구현 부분

순차실행과 들여쓰기

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

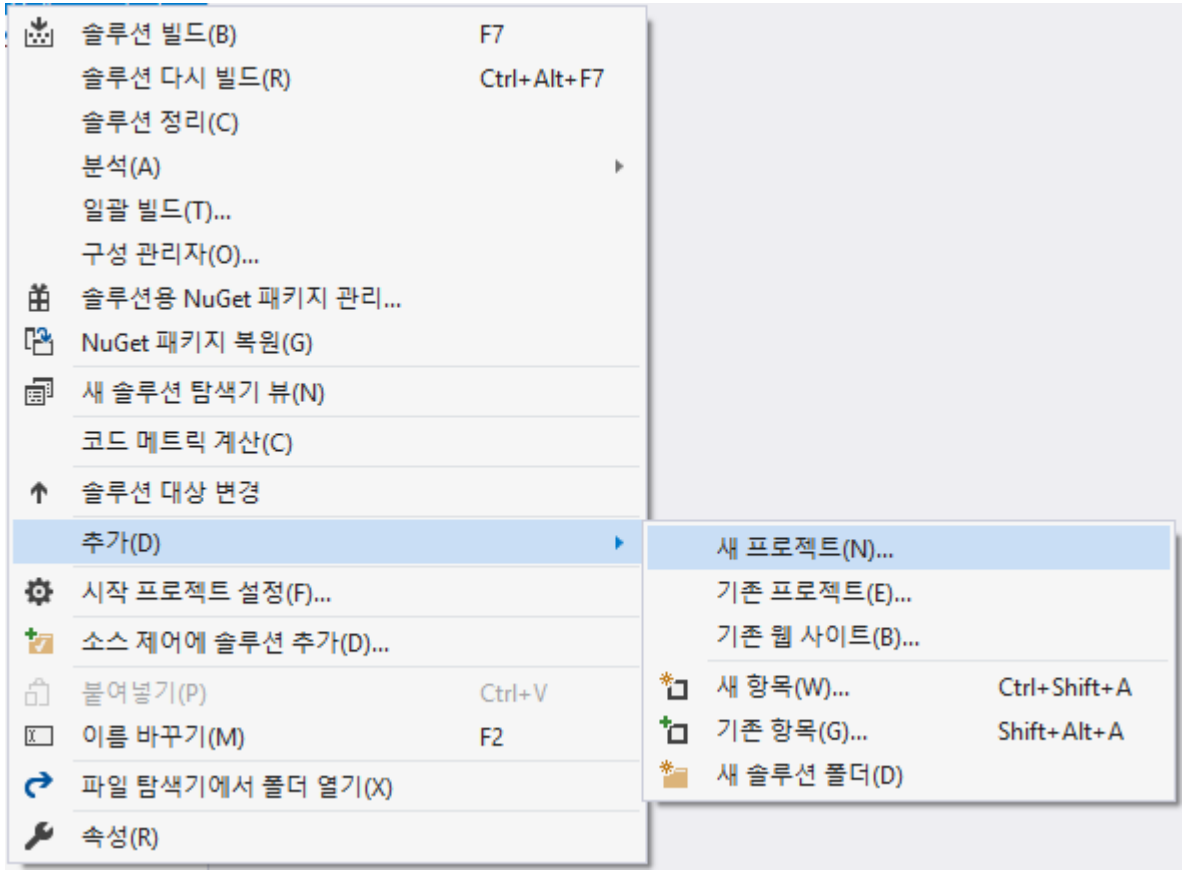
- 들여쓰기|indentation
  - 함수 몸체는 프로그래머가 소스를 쉽게 읽고 빠르게 이해하기 위해
  - 블록 시작 { 다음 줄을 탭(tab)만큼 오른쪽으로 이동하여 기술

순차실행과 들여쓰기


#include < stdio.h>
함수 머리(function header)
결과값의 유형
int main(void)
함수 시작을
의미하는 문자
{
함수의 입력이 없음을 의미
함수 호출
(function call)
printf("Hello World!\n");
return 0;
함수 반환값이 0을 의미
함수 종료를
의미하는 문자
}
함수 몸체(function body)
라이브러리에 구현 해 놓은 함수 printf() 정의
...
... int printf(… char * _Format, ...)
{
...
}
결과값 0 반환
입력 없음
함수 호출(function call)
함수 printf()

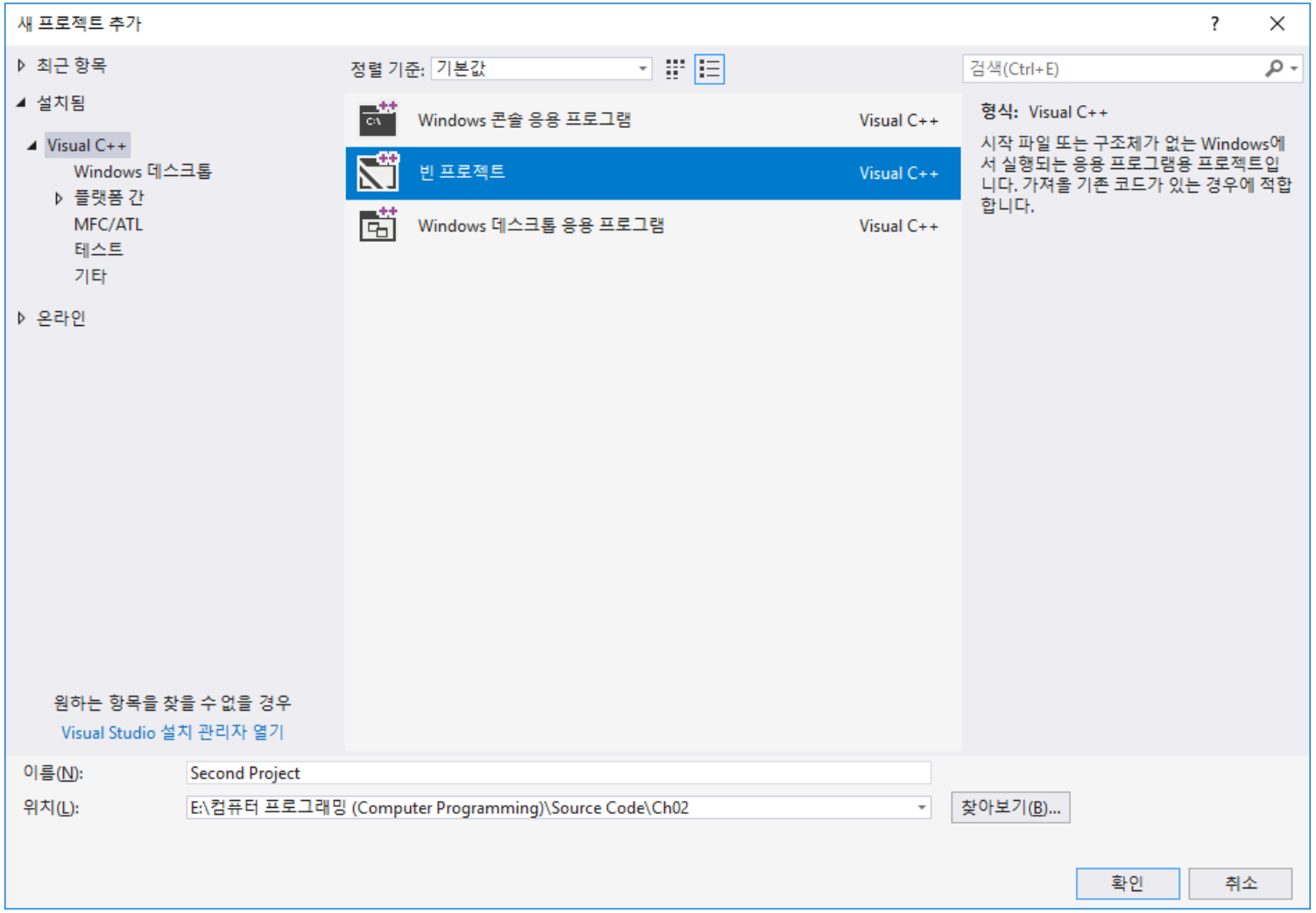
기존 솔루션에 프로젝트 추가

- 메뉴 [파일] → [추가] → [새 프로젝트]를 선택
- 오른쪽 마우스를 클릭해 [추가] -> [새 프로젝트]를 선택



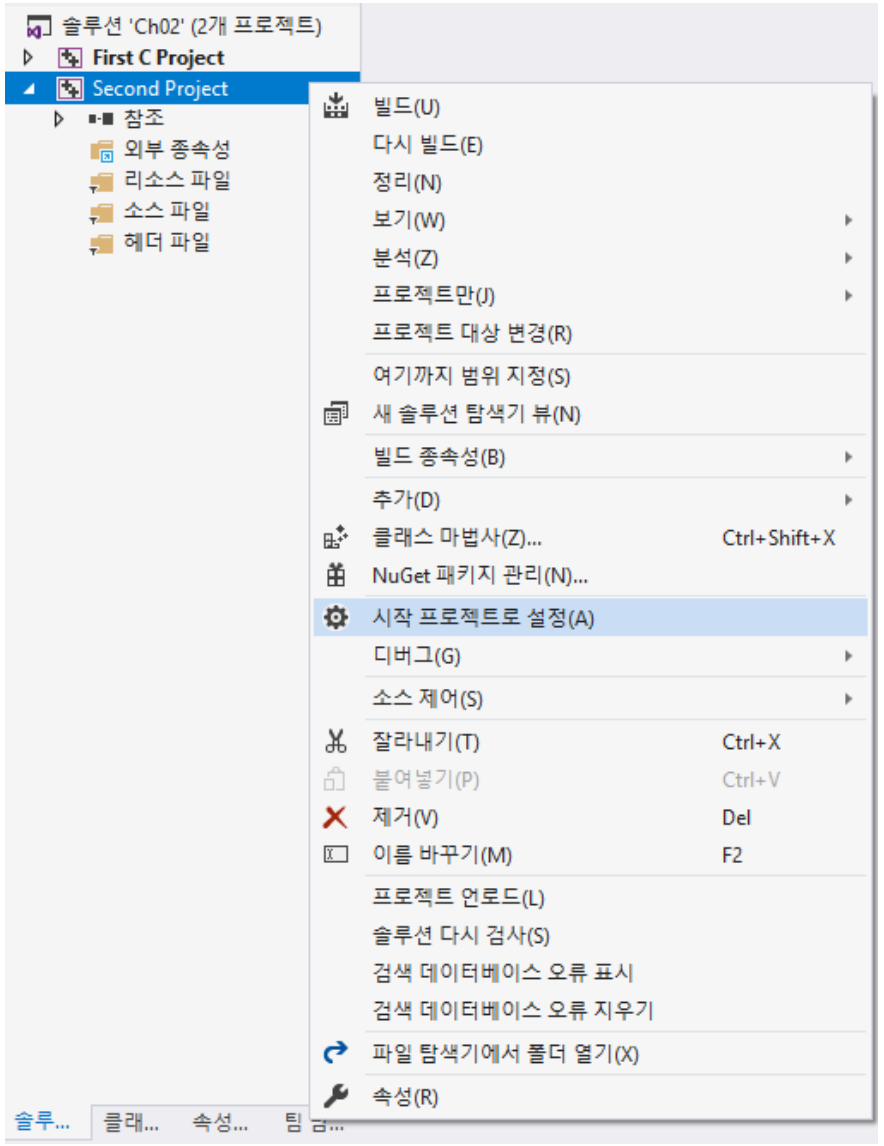
기존 솔루션에 프로젝트 추가

- Visual C++에서 '빈 프로젝트'를 선택
- 항목 '위치'
  - 솔루션 'Ch02'의 폴더임을 확인
- 프로젝트 이름
  - 'Second Project'을 입력



두번째 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



세번째 실습 예제 – 여러 줄에 문자열을 출력

- 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

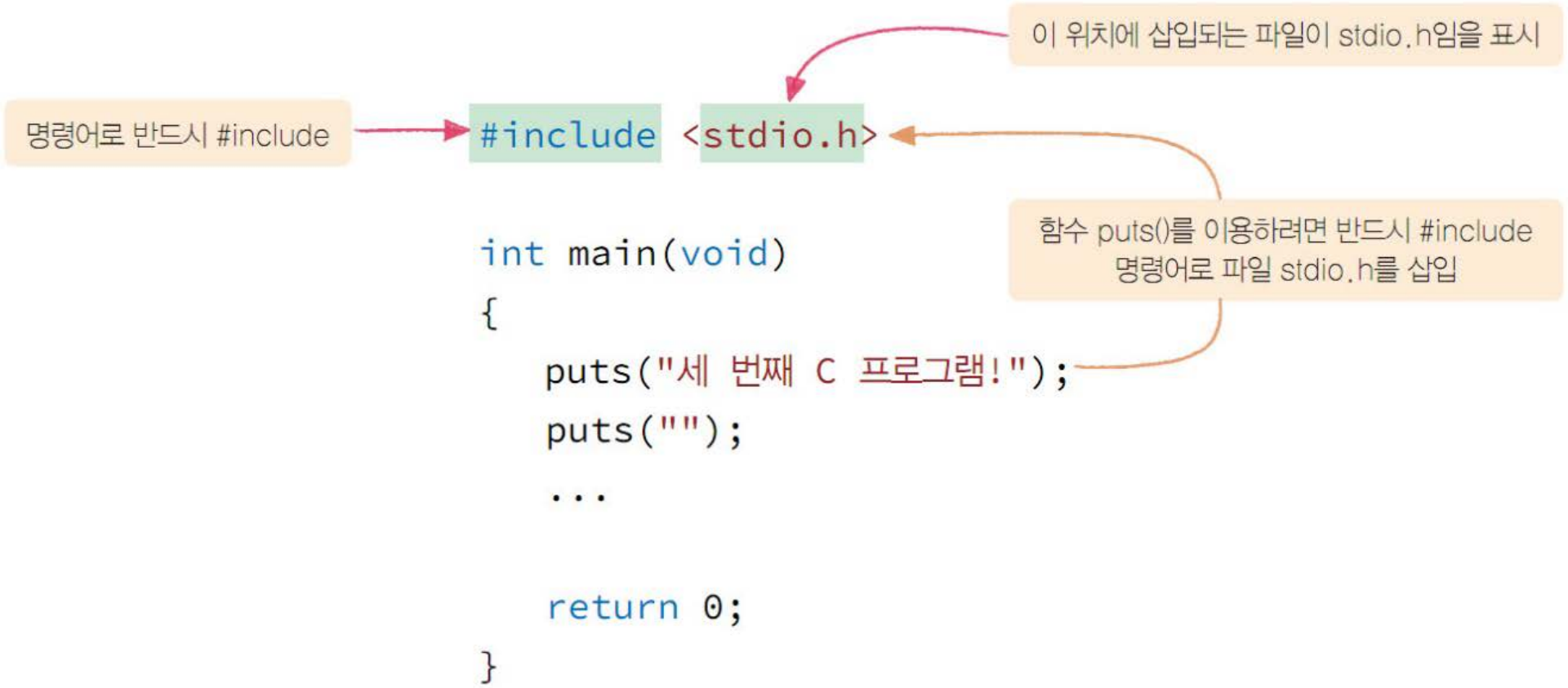
Source Code #03: printmline.c

- #include <stdio.h>
  - 라이브러리 함수 puts()와 printf()를 사용
  - #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

```c
#include <stdio.h>

int main(void)
{
    puts("세 번째 C 프로그램!");
    puts("");
    printf("-- 비주얼 스튜디오 C 프로그래밍 과정 --\n");
    printf("\n");
    puts("1. 솔루션과 프로젝트 만들기");
    printf("2. 소스파일 편집\n");
    printf("3. 실행\n");

    return 0;
}
```

```
세 번째 C 프로그램!

-- 비주얼 스튜디오 C 프로그래밍 과정 --

1. 솔루션과 프로젝트 만들기
2. 소스파일 편집
3. 실행
```

함수 puts()


이 위치에 삽입되는 파일이 stdio.h임을 표시
명령어로 반드시 #include
#include <stdio.h>
함수 puts()를 이용하려면 반드시 #include
명령어로 파일 stdio.h를 삽입
int main(void)
{
puts("세 번째 C 프로그램!");
puts("");
...
return 0;
}

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

함수 printf()

```
puts("세 번째 C 프로그램!");
puts("");
printf("-- 비주얼 스튜디오 C 프로그래밍 과정 --\n");
printf("\n");
puts("1. 솔루션과 프로젝트 만들기");
printf("2. 소스파일 편집\n");
printf("3. 실행\n");
```

함수 puts()는 문자열을 출력하고 자동으로 새로운 줄로 이동한다.

화살표가 있는 문장 실행 이후의 출력 위치를 나타낸다.

콘솔

```
세 번째 C 프로그램!

-- 비주얼 스튜디오 C 프로그래밍 과정 --

1. 솔루션과 프로젝트 만들기
2. 소스파일 편집
3. 실행
계속하려면 아무 키나 누르십시오 . . .
```

함수 printf() 인자 내부에 \n이 없으면 인자 출력 후 새로운 줄로 이동하지 않는다.

디버깅 예제

- 함수 printf()로 원하는 문자열string을 출력하는 프로그램을 작성
  - 문법 오류가 발생하도록 의도적으로 소스에 오류를 심어 놓음
  - 솔루션: 기존 솔루션 'Ch02', 프로젝트: '4th Project', 소스파일: debugging.c

Source Code #04: debugging.c

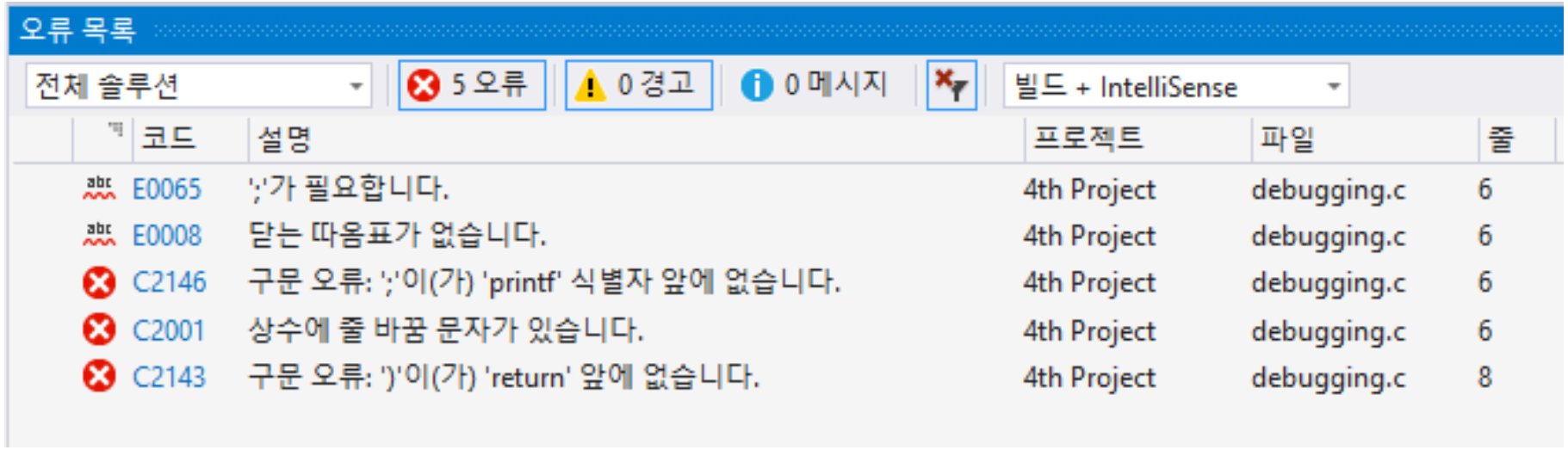
- '저장'을 하면 오류가 의심되는 소스 부분에 붉은 색 밑줄이 생김 (마우스로 이동)
- 바로 "오류: ';'가 필요합니다."와 "오류: 닫는 따옴표가 없습니다."라는 정확한 오류 원인 표시
- 만일 수정을 못하고 계속해서 컴파일이나 빌드를 수행
  - 컴파일 오류가 발생
  - 오류 목록 창에 '오류 내용'이 표시

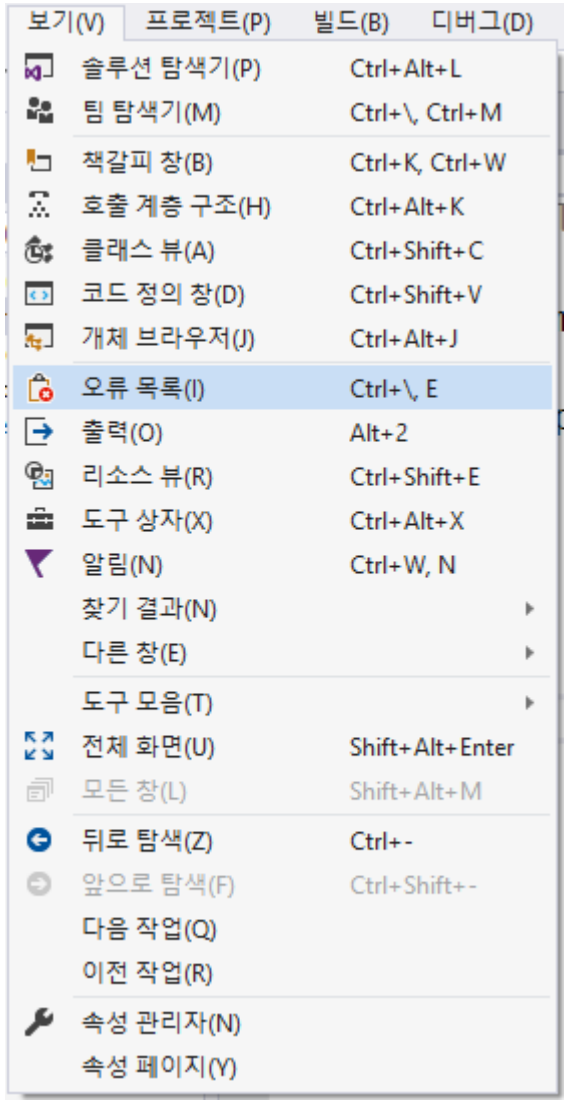
```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      printf("컴퓨터공학과 학생이며\n")
6          printf("C 프로그래밍 언어를 수강합니다.\n);
7                  ';'가 필요합니다.
8              return 0;
9  }
10
```

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      printf("컴퓨터공학과 학생이며\n")
6          printf("C 프로그래밍 언어를 수강합니다.\n);
7                                  닫는 따옴표가 없습니다.
8              return 0;
9  }
10
```

디버깅 예제

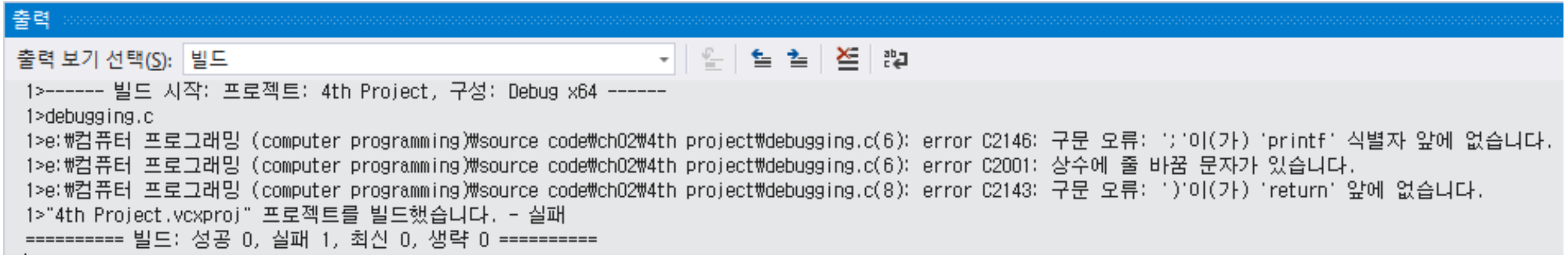
- '오류 목록' 창
  - 일목요연한 오류 목록 표시
  - 오류 코드, 설명, 프로젝트, 파일, 줄 번호 등을 자세히 표시
  - 창에는 지능적인 오류 표시인 IntelliSense 오류 표시





디버깅 예제

- 출력
  - 첫 줄에는 빌드를 시작한 프로젝트 이름
  - 두 번째 줄에는 컴파일한 소스인 debugging.c가 표시
  - 세 번째 줄부터 문제가 발생한 원인의 내용이 표시
  - 마지막으로 "= 빌드: 성공 0, 실패 1, 최신 0, 생략 0 ="와 같이 최종 빌드 결과가 표시



컴파일 오류 메시지의 이해와 소스 수정

- 오류 내용 표시와 문장 종료 문자 ;이 빠진 오류 메시지 분석
  - 4개의 요소가 3개의 콜론(:)으로 구분되어 표시
    - 오류가 발생한 파일이름(전체경로): …\Ch02\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.
  - 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 클릭하면 소스의 해당 줄로 이동
  - 다시 소스 5줄로 이동하여 마지막에 ;를 삽입하면, 이 컴파일 오류는 해결

컴파일 오류 메시지의 이해와 소스 수정

```
05    printf("컴퓨터공학과 학생이며\n")
06    printf("C 프로그래밍 언어를 수강합니다.\n);
```

문장 마지막에 ;가 빠져 발생하는 오류는 다음과 같은 메시지가 표시.

```
...\debugging.c(6): error C2146: 구문 오류 : ';'이(가) 'printf' 식별자 앞에 없습니다.
```

컴파일 오류 메시지의 이해와 소스 수정

- 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)
  - 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 효과적임
  - 소스 debugging.c에서 발생한 오류 원인 2개를 수정
  - 5줄에 ;을, 6줄에 "을 삽입하면 컴파일 오류는 사라지고 결과가 출력

컴파일 오류 메시지의 이해와 소스 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

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

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.

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 습관이 필요

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: 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

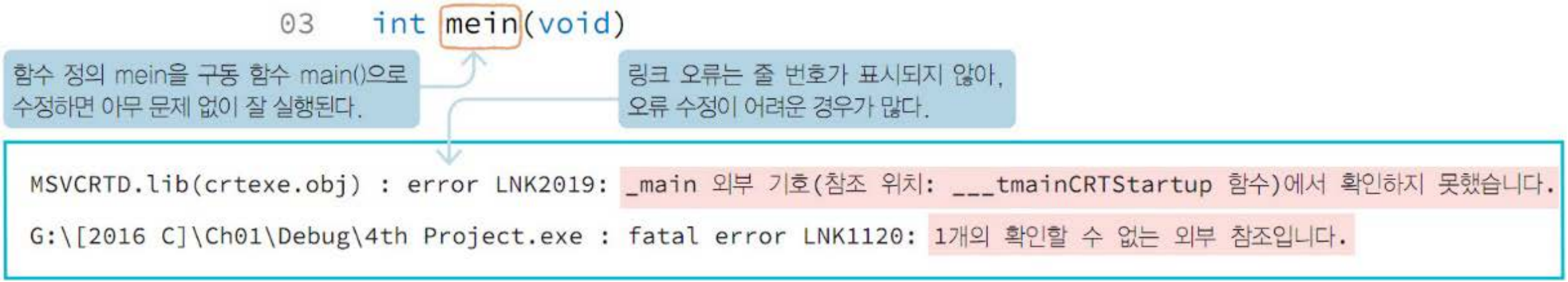
```
05    print("컴퓨터공학과 학생이며\n");
```

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.
```

링크 오류: 함수 main() 사용

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



링크 오류: 실행시간 오류

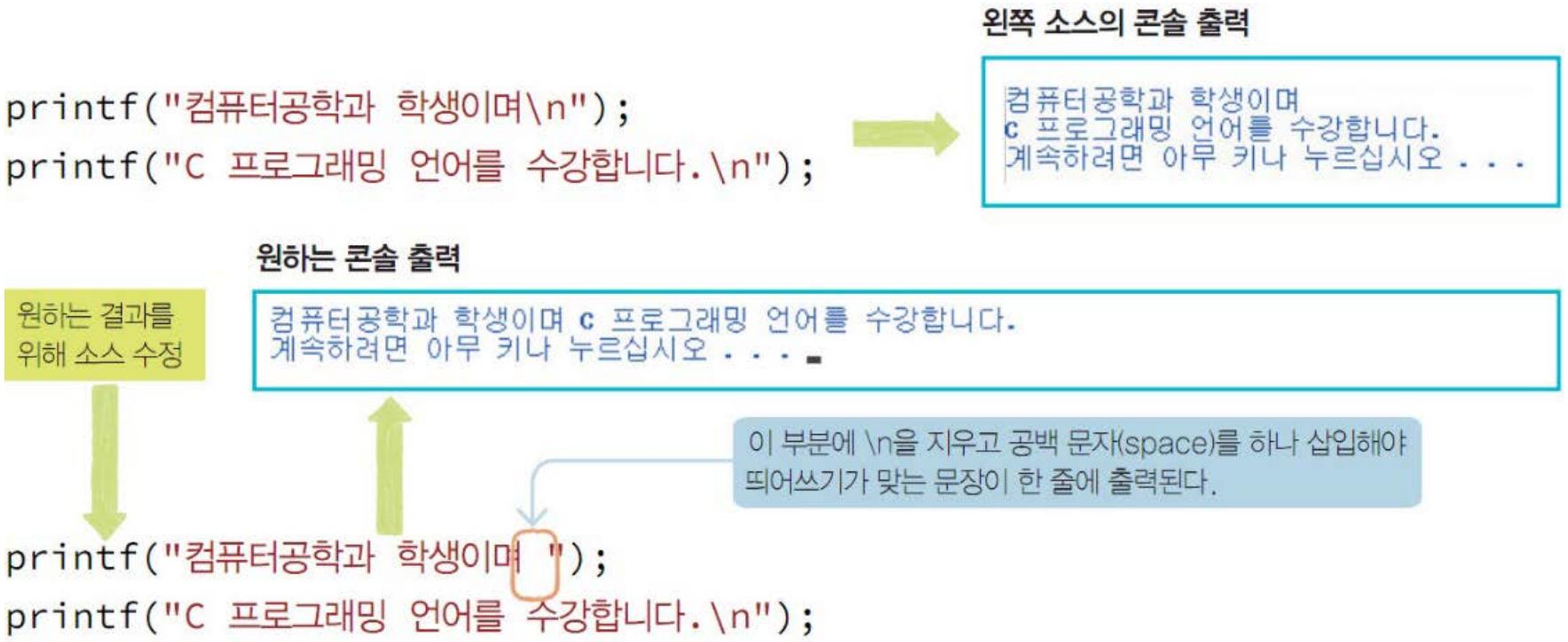
- 컴파일과 링크가 성공해도 실행시간에 오류가 발생 가능
  - 일반적으로 메모리관리를 실수하거나
  - 0으로 나누는 식을 사용하는 등
  - 프로그램의 잘못으로 발생하는 경우가 대부분
  - 간혹 기계적 결함으로도 발생

논리 오류: 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

논리오류의 디버깅

- 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




I
PROGRAMMING