



C 프로그래밍 및 실습

C Programming

이지민

CHAP 01 프로그래밍 소개

학습목표

프로그램이란 무엇인지 기본개념을 알아본다.

C언어의 특징 및 활용 분야에 대해 알아본다.

C 언어로 작성된 프로그램을 컴파일 하여 실행하는 단계를 알아본다.

Visual Studio 2017을 이용한 개발과정을 알아본다.

목차



프로그래밍 소개

- 프로그램의 이해
- 프로그래밍 언어



C 언어 개요

- C 언어 소개
- C 언어 특징



C 프로그램 개발

- 통합개발환경
- Visual C++을 이용한 C 프로그램 개발과정

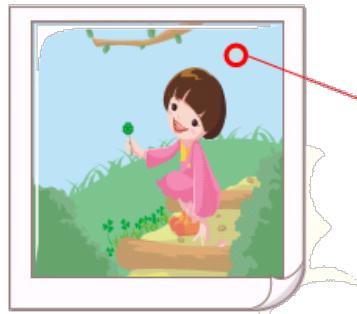
프로그램의 이해_컴퓨터란?

❖ Q) 컴퓨터(computer)는 무엇인가?

- A) 데이터를 가공하여 정보를 얻기위해 계산(compute)하는 기계

❖ Q) 컴퓨터를 이용하여 데이터를 처리하려면 반드시 데이터가 숫자 형태이어야 한다. 왜?

- A) 컴퓨터는 숫자 계산을 하기 때문에 데이터는 숫자로 표시되어야 한다.



디지털 이미지의 경우

각 화소의 밝기와 색상을
숫자로 표현한다.



디지털 음악의 경우

각 파형의 높이를
숫자로 표현한다.

프로그램의 이해_컴퓨터의 정의

❖ Q) 그렇다면 계산만 빠르게 할 수 있으면 컴퓨터인가?

- A) 현대적인 의미에서의 컴퓨터는 명령어들의 리스트에 따라 데이터를 처리하는 기계라고 할 수 있다

❖ Q) 단순한 기계인 컴퓨터가 강력한 이유?

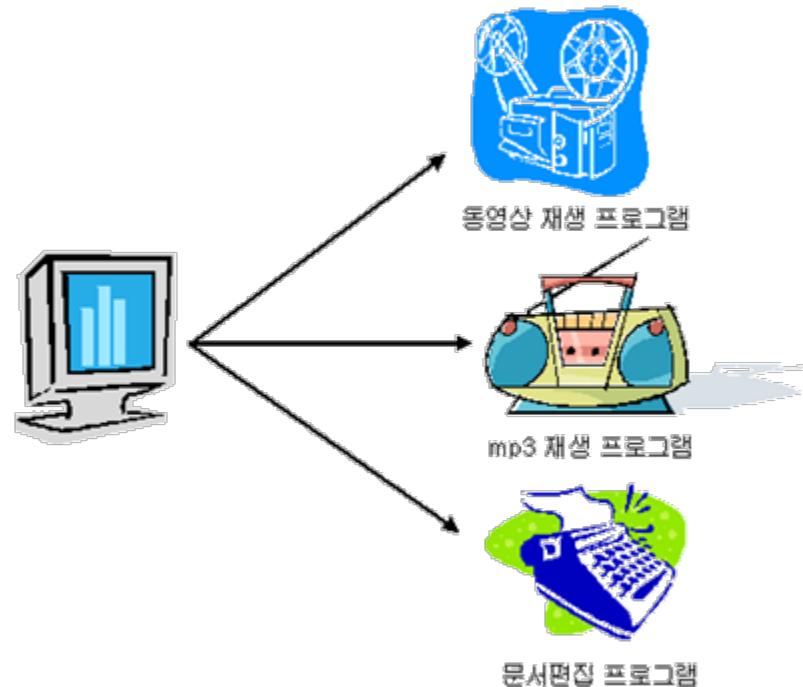
- A) 계산을 통해서 할 수 있는 일이 상상을 초월함- 음악듣기, 동영상 보기, 통신, 게임 등도 컴퓨터의 내부적인 계산을 통해 모두 처리



프로그램의 이해_컴퓨터의 장점

❖ Q) 컴퓨터의 가장 큰 장점은 무엇일까?

- A) 컴퓨터는 범용적인 기계이다. 프로그램만 바꿔주면 다양한 작업이 가능하다.



프로그램만 바꿔주면 컴퓨터는 다양한 작업을 할 수 있다.

프로그램의 이해_컴퓨터의 구성요소

❖ Q) 컴퓨터의 구성 요소를 크게 2가지로 분류하면?

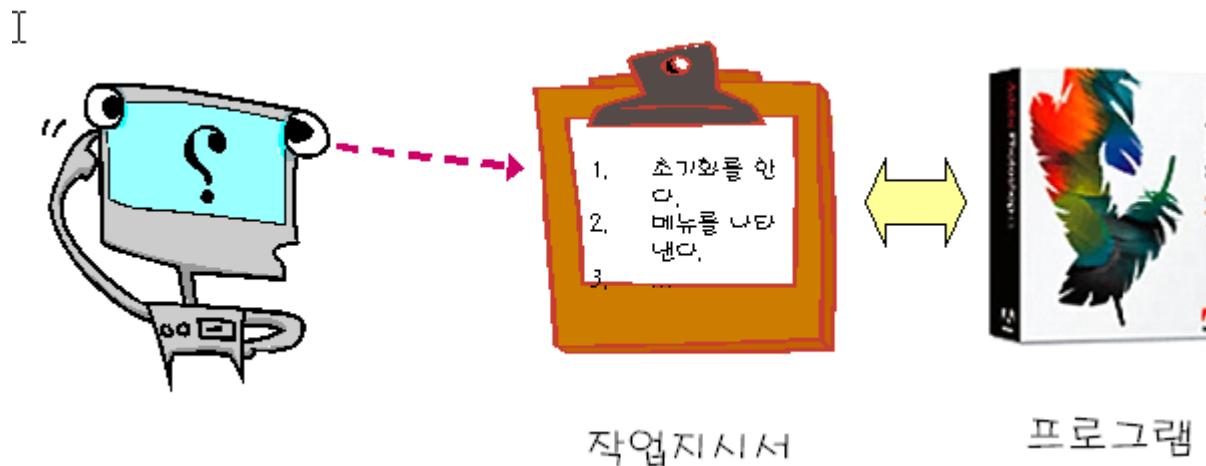
- A) 컴퓨터는 기본적으로 하드웨어와 소프트웨어로 구분



프로그램의 이해_프로그램의 역할

❖ Q) 컴퓨터에서 프로그램이 하는 일은 무엇인가?

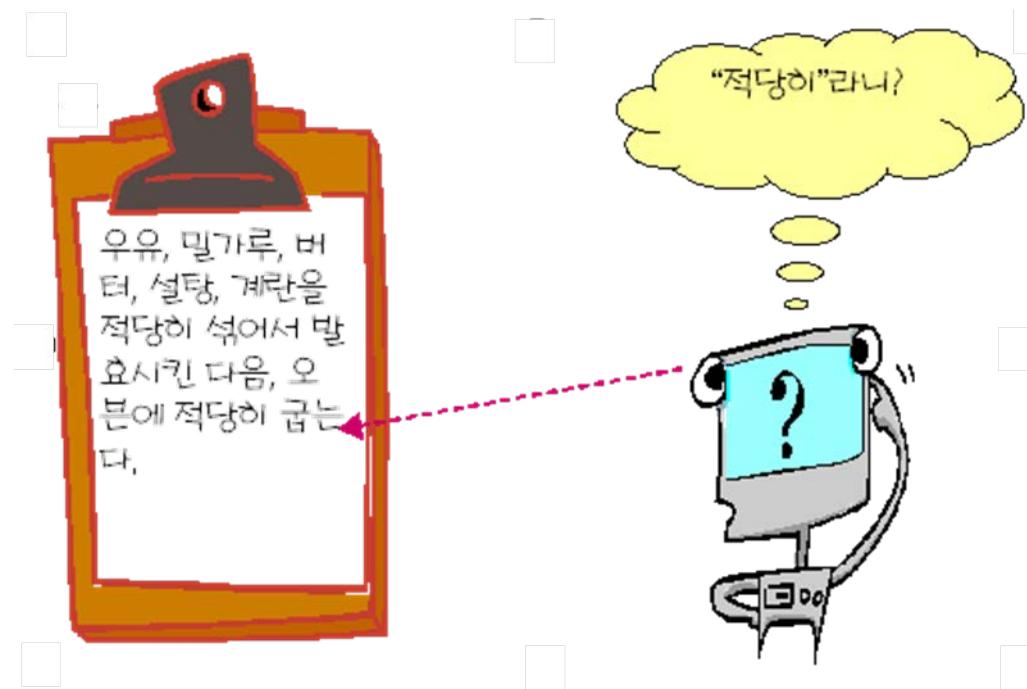
- A) 프로그램이란 우리가 하고자 하는 작업을 컴퓨터에게 전달하여 주는 역할을 한다.



프로그램의 이해_작업을 지시하는 방법

❖ Q) 컴퓨터에게 적당히 작업을 시킬 수 있을까?

- A) 상식이나 지능이 없기 때문에 아주 자세하고 구체적으로 일을 지시하여야 한다.



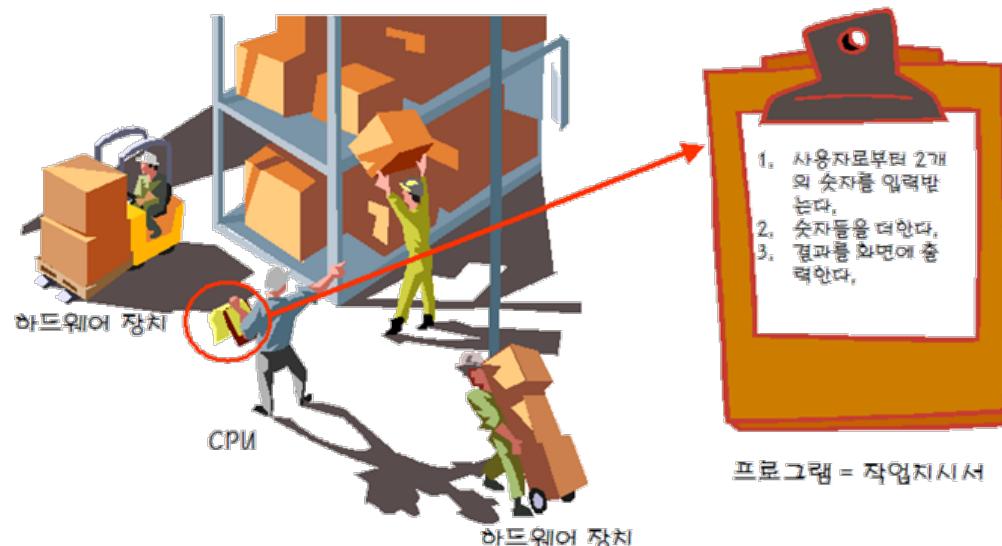
프로그램의 이해_프로그램의 구성요소

❖ 프로그램 : 컴퓨터 하드웨어가 수행할 일련의 작업을 기술하고 있는 명령어의 모임

- 명령어(instruction)

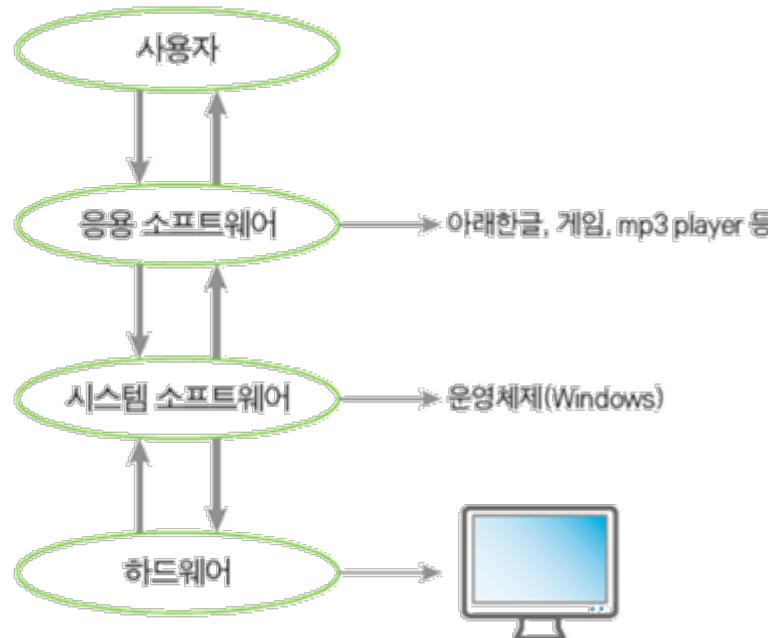
- CPU가 처리할 수 있는 2진 코드.
- CPU 명령어라고도 함.

❖ 프로그래밍 : 프로그램을 작성하는 일 또는 그 과정



프로그램의 이해

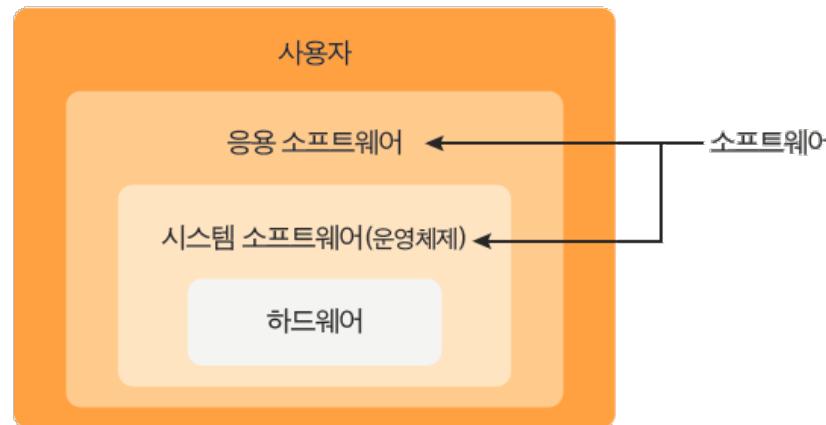
❖ 프로그램(program) = 소프트웨어



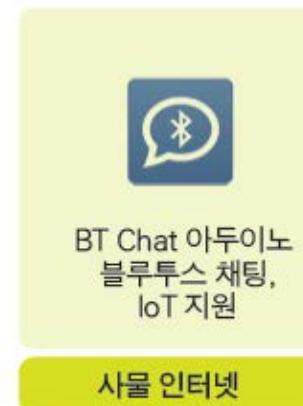
- **시스템 소프트웨어** : 하드웨어를 지시하고 통제하기 위한 프로그램
- **응용 소프트웨어** : 사용자가 원하는 일을 수행하기 위한 프로그램

프로그램의 이해

❖ 소프트웨어와 컴퓨터 시스템의 구성요소의 관계



❖ 프로그램의 종류



프로그래밍 언어_컴퓨터가 이해하는 언어

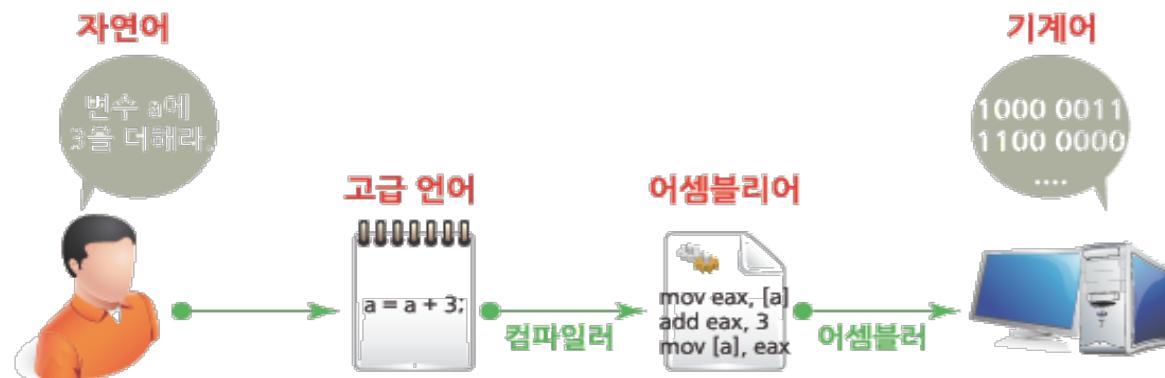
- ❖ 컴퓨터는 인간의 언어를 이해할 수 없다.
- ❖ 컴퓨터는 이진수로 된 기계어만을 이해한다.
- ❖ 프로그래밍 언어란?
 - 사람과 컴퓨터 사이에 존재하는 일종의 커뮤니케이션 수단이다.
 - 프로그램에서 컴퓨터가 수행해야 할 다양한 작업을 기술하는데 사용되는 언어



프로그래밍 언어_컴파일러의 역할

❖ 컴퓨터는 사람이 사용하는 언어를 이해할 수 없으므로, 컴퓨터 시스템이 이해할 수 있는 컴퓨터 언어가 필요하게 되었다.

- 기계어로 작성하는 경우
 - 프로그램을 작성하기가 어렵고, 실수가 자주 발생
- 어셈블리어로 작성하는 경우
 - 어셈블리어(assembler)가 기계어로 변환
 - 어셈블리어가 CPU 명령어와 1 대 1로 대등되어 있으므로 CPU가 달라지면 프로그램을 다시 작성해야 한다.
- 고급 언어로 작성하는 경우
 - 컴파일러(compiler)가 기계어로 변환



프로그래밍 언어_분류

기계어

- 컴퓨터 시스템이 사용하는 언어로 **0과 1의 집합(이진수)**
- 2진수이므로 작성하기 어렵다.

어셈블리어(CPU에 종속적)

- 컴퓨터에 이진수의 조합으로 명령을 내리기 어려우므로 **CPU 명령들을 기호(Symbolic name)로 표시한 것**.
- 사람들이 기계어를 바로 사용하는 것보다는 사용자가 기호 명령을 이용하는 것이 효율적임.
- 어셈블러는 기계어로 변환되어 처리

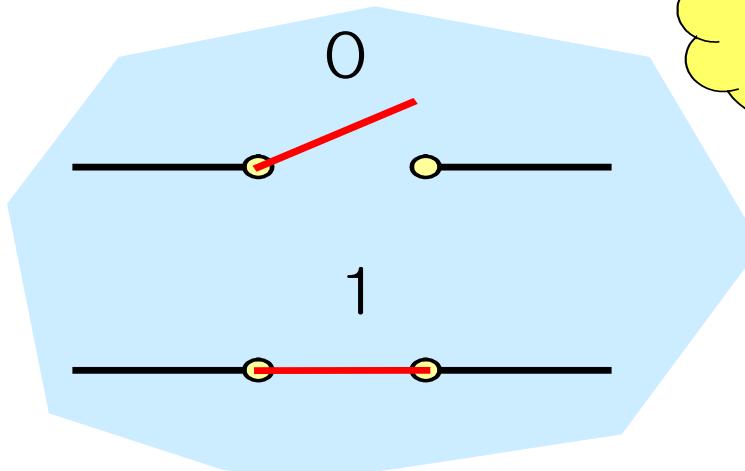
고급 언어(CPU에 독립적)

- 사람이 인식할 수 있는 문자로 컴퓨터와 대화하는 언어
- 고급언어를 기계어로 변환해주는 언어번역 프로그램인 **컴파일러(Compiler)**를 필요로 함.

프로그래밍 언어

❖ 컴퓨터에 이진수가 사용되는 이유

- 이진수는 전자회로로 구현하기가 쉽다.



0은 열린 스위치로, 1은 닫친 스위치로 표현할 수 있습니다.



❖ 고급 언어의 장점

- 특정 CPU의 동작에 대하여 **자세히 알 필요가 없다.**
→CPU에 대하여 독립적인 프로그램을 작성할 수 있다.
- 프로그램을 **개발하기 쉽고, 유지 보수하기 쉽다.**
- 고급 언어의 예 : C, C++, java, C# 등

C언어 개요_C언어의 탄생

- ❖ 1970년대 초 벨 연구소(AT & T Bell Laboratories)의
- ❖ 데니스 리치(Dennis Ritchie)에 의해 UNIX라는 운영체제(Operating System)를 설계하던 중에 C 언어 탄생.

1970년대 AT & Bell Laboratories : Dennis Ritchie

UNIX라는 운영체제(Operating System) 설계 중 개발

사용자가 더 효율적으로 컴퓨터 자원들을 다룰 수 있도록 해야 함

C 언어 탄생

하드웨어(Hardware)의 효율적 통제, 빠른 작동을 유도하는 프로그램 언어 필요

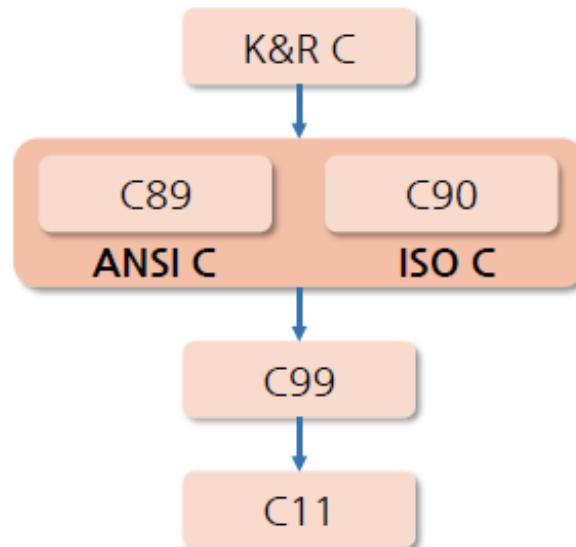
C언어 개요_C언어의 발전 과정

❖ 표준 C(standard C)

- 표준 C/C로 작성된 소스 코드는 모든 C 컴파일러에서 컴파일되고 동일하게 실행될 수 있다.
- C 컴파일러는 표준 C 기능 + 자신만의 기능을 제공

❖ 현재의 C/C++ 컴파일러는 C99버전을 표준으로 사용

- 버그발생 함수 개선하여 개정 : c11표준



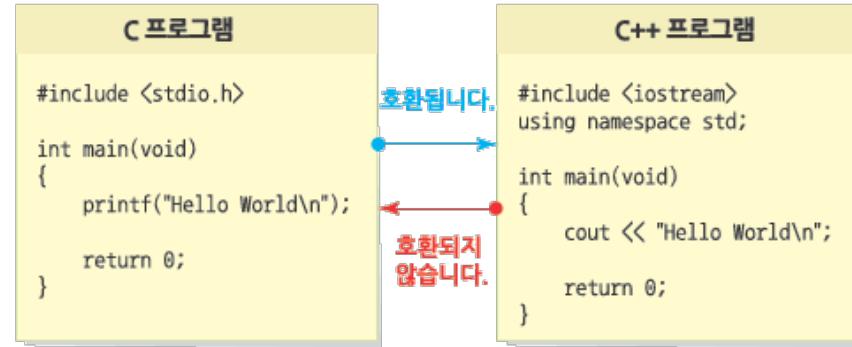
C언어 개요_C /C++ 컴파일러

❖ C/C++ 컴파일러는 C 컴파일러와 C++ 컴파일러의 역할을 동시에 제공한다.

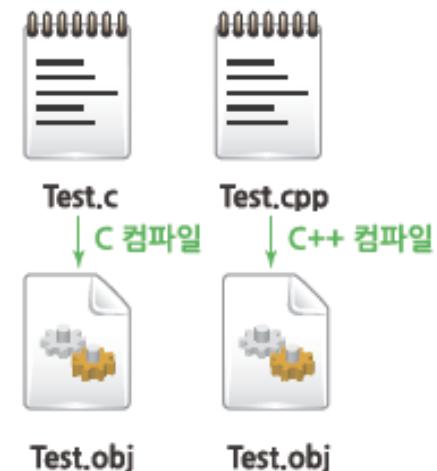
- C++은 C와 호환성이 있다.
- C 프로그램은 유효한 C++ 프로그램이지만, C++ 프로그램은 유효한 C 프로그램이 아니다.

❖ C/C++ 컴파일러는 소스 파일의 확장자가

- .c일 때는 C 컴파일을 수행하고,
- .cpp일 때는 C++ 컴파일을 수행한다.



Visual C/C++ 컴파일러



C언어 개요_C언어의 특징(1/2)

❖ 간결성

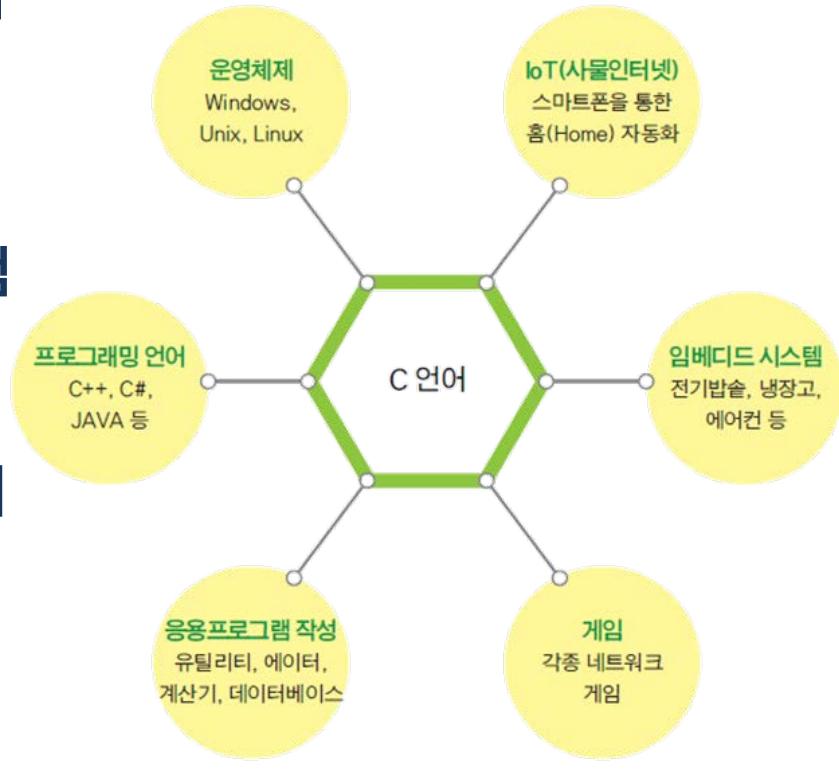
- 구문이 간결하고, 구조적인 언어이다.

❖ 이식성

- 운영체제 플랫폼에 상관없이 여러 CPU에서 실행될 수 있는 프로그램을 개발할 수 있다.

❖ 효율성

- 프로그램 크기도 작고, 프로그램의 실행 속도도 빠르다.



C언어 개요_C언어의 특징(2/2)

❖ C 언어의 단점

- 배우기도 어렵고, 사용하기도 어렵다.
- 다른 언어에 비해 사용 시 주의 사항이 많다.

❖ C 언어가 많이 사용되는 이유

- C를 배우면 C++, java, C#같은 언어를 이해하는데도 도움이 된다.
- C 언어는 프로그래밍의 기본 개념을 이해하는데도 도움이 된다.

❖ C 언어의 활용 분야

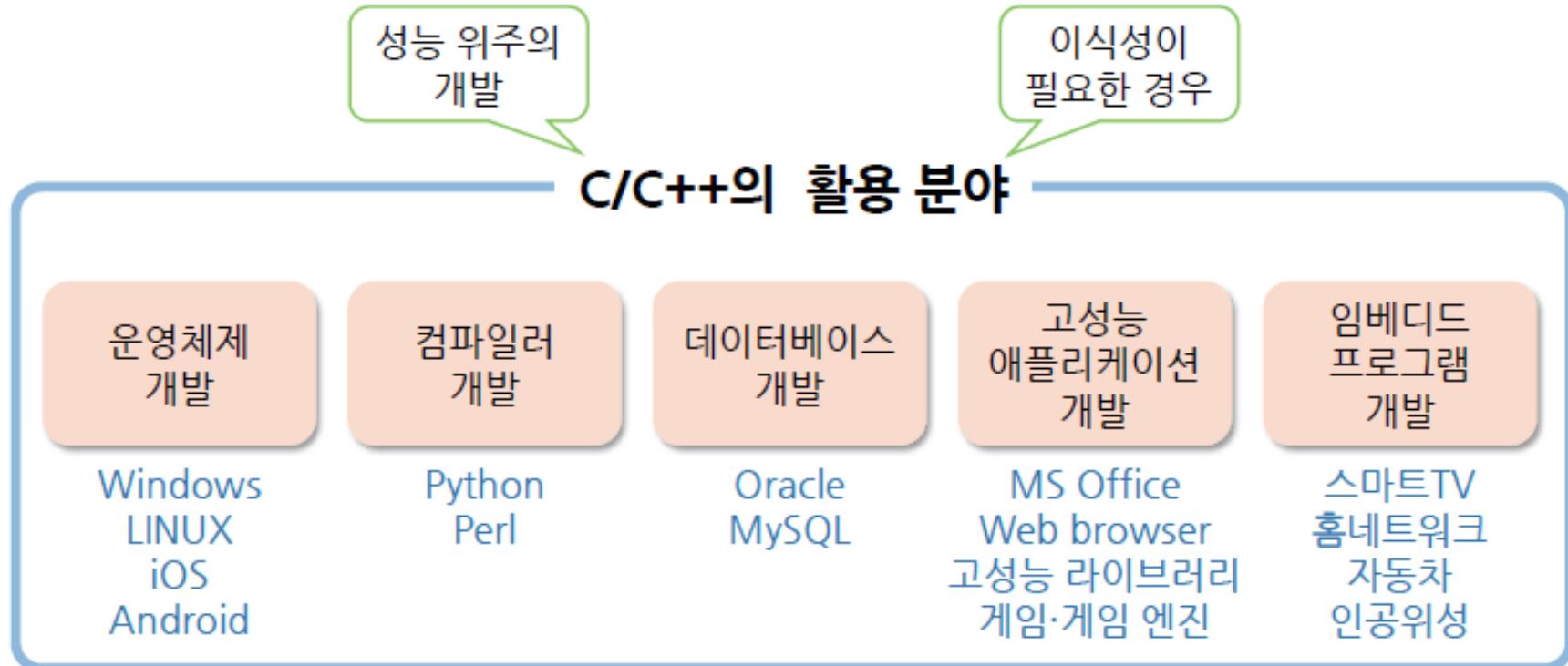
- 여러 컴퓨터 시스템에서 사용해야 할 라이브러리나 응용 프로그램 개발
- 시스템 프로그램, 서버 어플리케이션, 게임 등의 성능 위주의 프로그램 개발
- 임베디드 프로그램 개발

C언어 개요

❖ 언어 (language)

- 문법 (syntax)
 - 엄격하게 지켜야지만 컴파일 가능
- 논리 (logic)
 - 문법과 마찬가지로 논리도 중요한 요소
 - Ex) 총 10개의 연필 중 20번째 연필을 집으세요.
 - 오류 시 Runtime error 발생

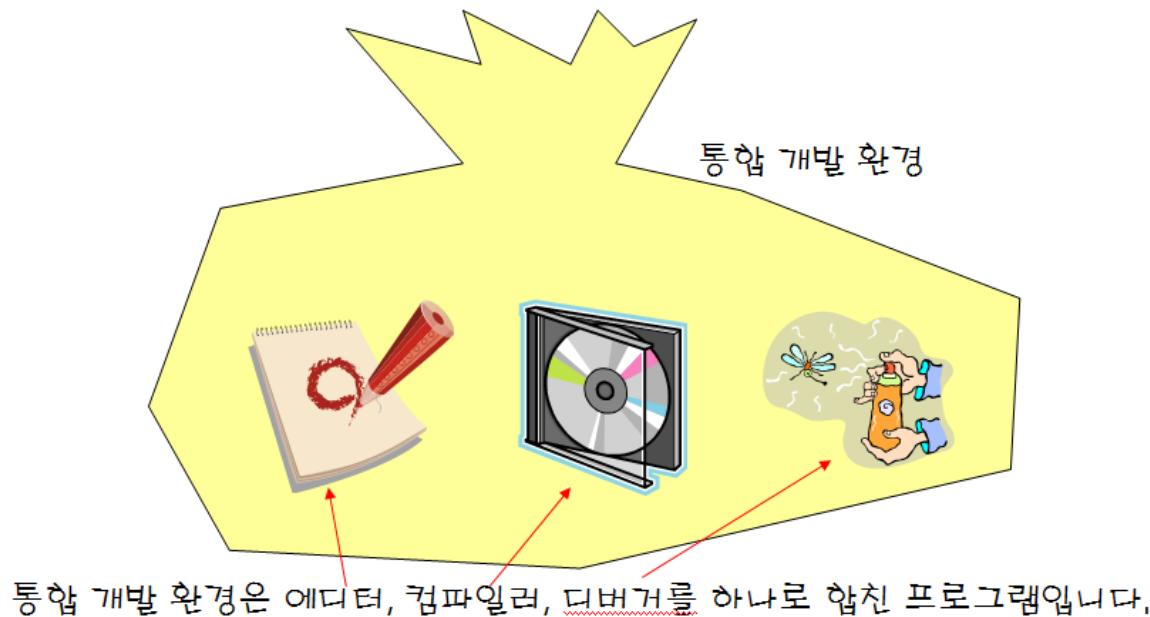
C언어 개요_C언어 활용분야



- **임베디드 시스템** : 특수 목적의 컴퓨터를 내장하고 있는 장치
- **임베디드 프로그램** : 임베디드 시스템에서 수행되는 프로그램

통합개발환경_개요

- ❖ 통합 개발 환경(IDE: integrated development environment):
 - 에디터 + 컴파일러 + 디버거



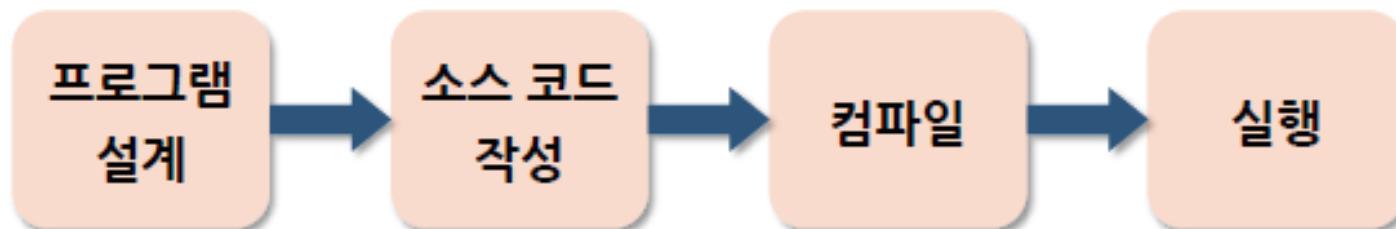
통합개발환경_프로그램 개발과정(1/4)

❖ 프로그램 설계

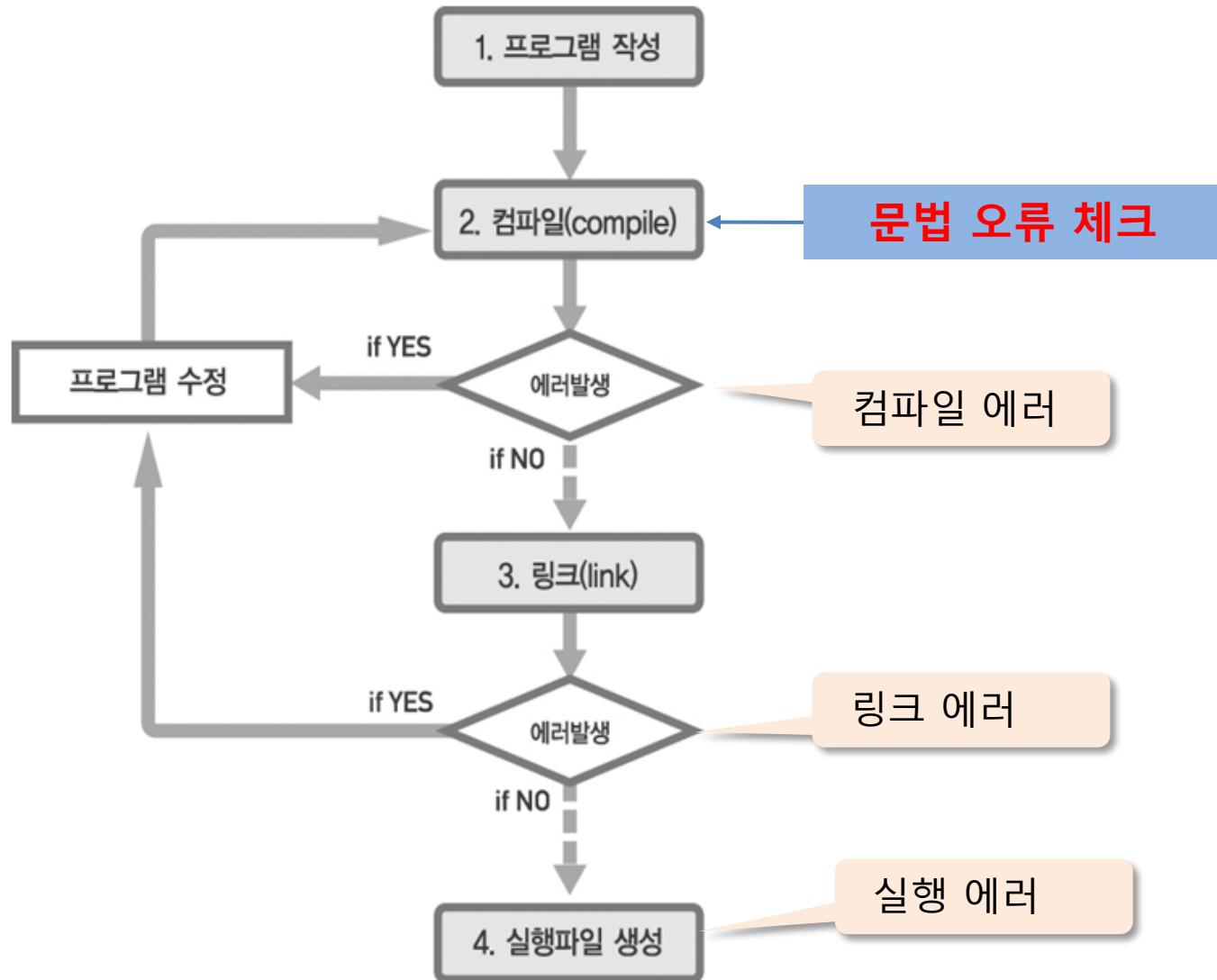
- 알고리즘(algorithm)
 - 어떤 문제를 논리적으로 해결하기 위한 절차나 방법
- 알고리즘은 자연어나 순서도, 의사코드를 이용해서 기술할 수 있다.

❖ 소스 코드 작성

- 프로그램이 처리할 작업이나 알고리즘을 특정 프로그래밍 언어로 기술
- 텍스트 편집기, 소스 코드 편집기를 이용해서 소스 파일 작성



통합개발환경_프로그램 개발과정(1/4)

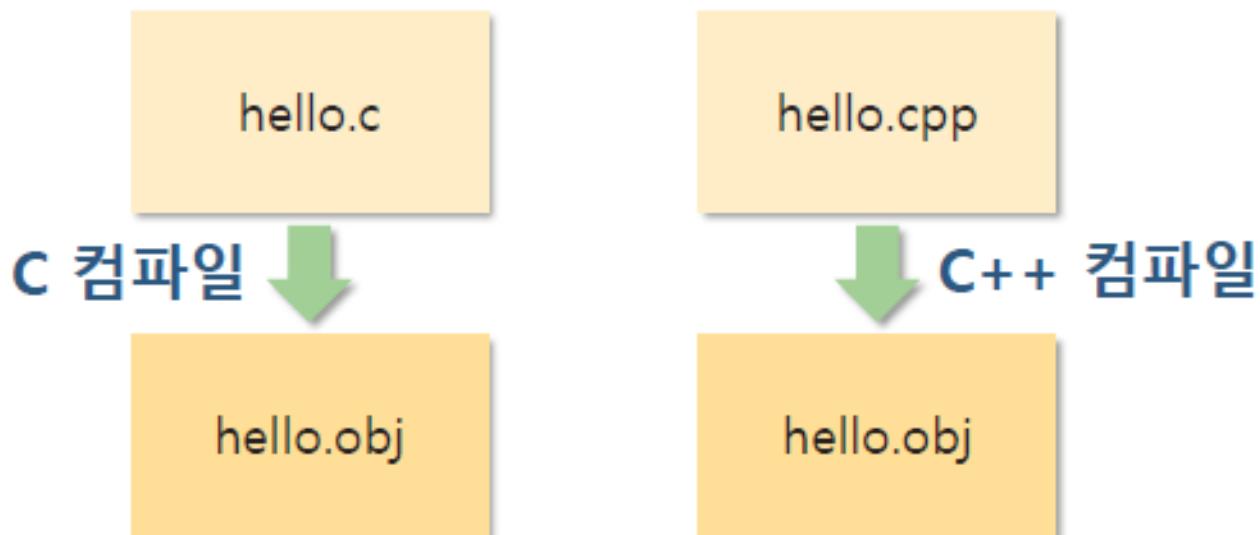


통합개발환경_프로그램 개발과정(2/4)

❖ 소스 파일 작성 → 컴파일 → 실행

❖ 소스 파일 작성

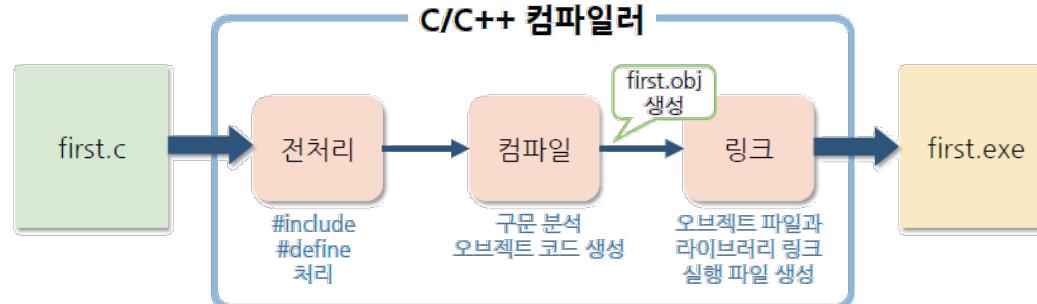
- 간단한 텍스트 편집기를 이용하거나,
- 통합 개발 환경(IDE)이 제공하는 소스 코드 편집기를 사용해서 작성
- C/C++ 컴파일러에서 C 컴파일을 하려면 .c 확장자를 가진 파일로 저장



통합개발환경_프로그램 개발과정(3/4)

❖ 컴파일

- 전처리기 → 구문 분석 → 코드 생성 → 링크
- 전처리기는 소스 파일을 컴파일하기 위해 준비
- 구문 분석에서는 소스 코드가 C 문법에 맞게 작성되었는지 검사
 - 구분 분석 시 잘못된 부분에 대하여 **컴파일 에러** 발생
- 컴파일 에러가 없으면 각각의 소스 파일마다 별개의 오브젝트 코드가 생성 → 오브젝트 파일
- 링커는 여러 개의 오브젝트 파일과 라이브러리를 링크해서 실행 파일을 생성
 - 링크하면서 문제가 있으면 **링크 에러** 발생
 - 여러 소스 파일의 상관관계를 함께 파악해서 에러를 찾아야 함
- 컴파일/링크 에러가 발생하면 소스 파일을 수정하고 컴파일하는 과정을 반복



통합개발환경_프로그램 개발과정(4/4)

❖ 실행

- 링크 결과 생성된 실행 파일을 실행해서 올바른 결과를 얻을 수 있는지 확인
 - 프로그램이 잘못된 실행 결과를 생성하거나 실행 중에 프로그램이 죽는 경우에 **실행 에러** 발생
 - 실행 에러는 프로그램의 논리가 잘못되어 발생하는 에러
- **디버깅(debugging)** : 프로그램의 실행 흐름이 올바르게 진행되는지, 프로그램 내에서 사용된 수식의 값이 맞는지 등을 살펴봄으로써 실행 에러를 찾아서 고치는 과정

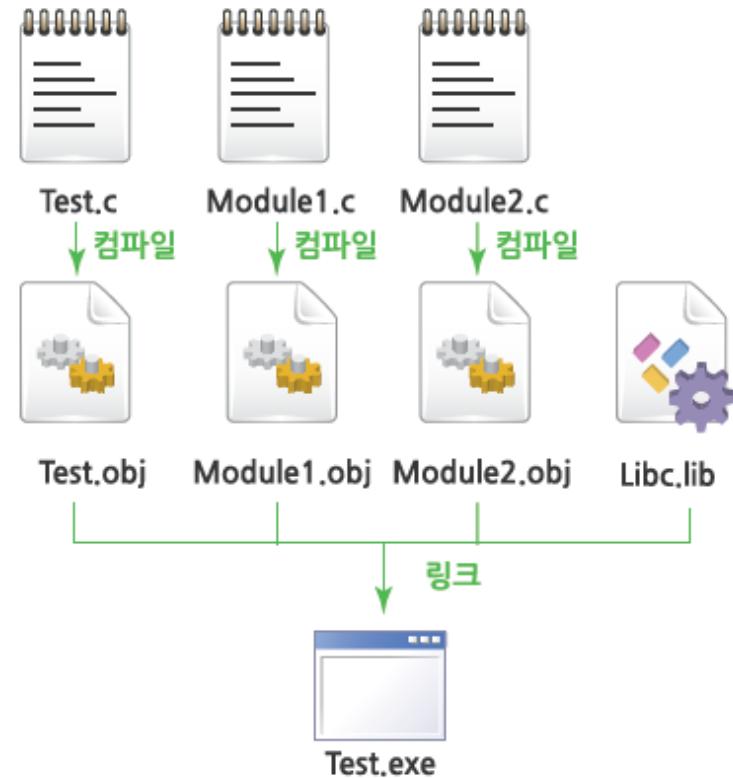
통합개발환경_종류

❖ Visual C++

- 원도 플랫폼에서 **가장 많이 사용되는 C 개발 환경**
- 우리가 사용할 버전 : Visual Studio 2022

❖ 분할 컴파일

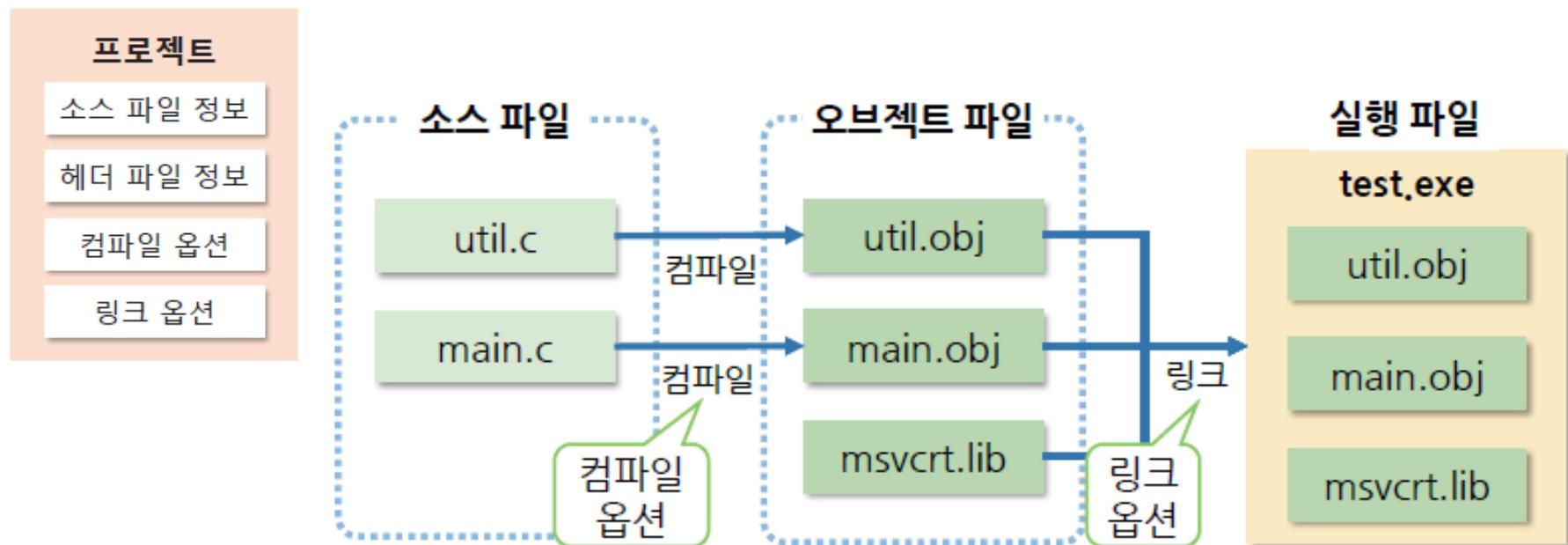
- 대규모의 프로그램을 개발하거나 여러 프로그래머가 공동 개발을 하려면 소스 파일을 여러 개로 나누어 작성해야 한다.
- 각각의 소스 파일에 대한 정보, 링크할 라이브러리 정보 등을 관리해야 한다.
- Visual C++에서는 이런 정보들을 **프로젝트**로 관리한다.



통합개발환경_프로젝트와 솔루션(1/2)

❖ 프로젝트 (*.vcxproj)

- 해당 프로젝트의 소스파일과 관련된 파일을 관리하는 파일
- 프로그램을 만들기 위해서 사용되는 **소스 파일 및 헤더 파일에 대한 정보**
- 소스 파일을 컴파일 할 때 사용되는 **컴파일 옵션**
- 오브젝트 파일이나 라이브러리를 링크할 때 사용되는 **링크 옵션**



통합개발환경_프로젝트와 솔루션(2/2)

❖ 솔루션 (*.sln)

- 여러 개의 프로젝트를 묶어서 관리하는 파일
- 서로 관련된 프로젝트들을 함께 관리하기 위해 솔루션을 사용
- 모든 프로젝트가 반드시 솔루션에 포함되어야 함

디폴트로 프로젝트
생성 시 솔루션이
함께 생성된다.

솔루션
first.sln

프로젝트
first.vcxproj

솔루션에 신경쓰지 않고
작업하는 경우

솔루션과 프로젝트를
따로 생성하고 솔루션에
프로젝트를 등록한다.

솔루션
myapp.sln

프로젝트
app.vcxproj

프로젝트
libA.vcxproj

프로젝트
libB.vcxproj

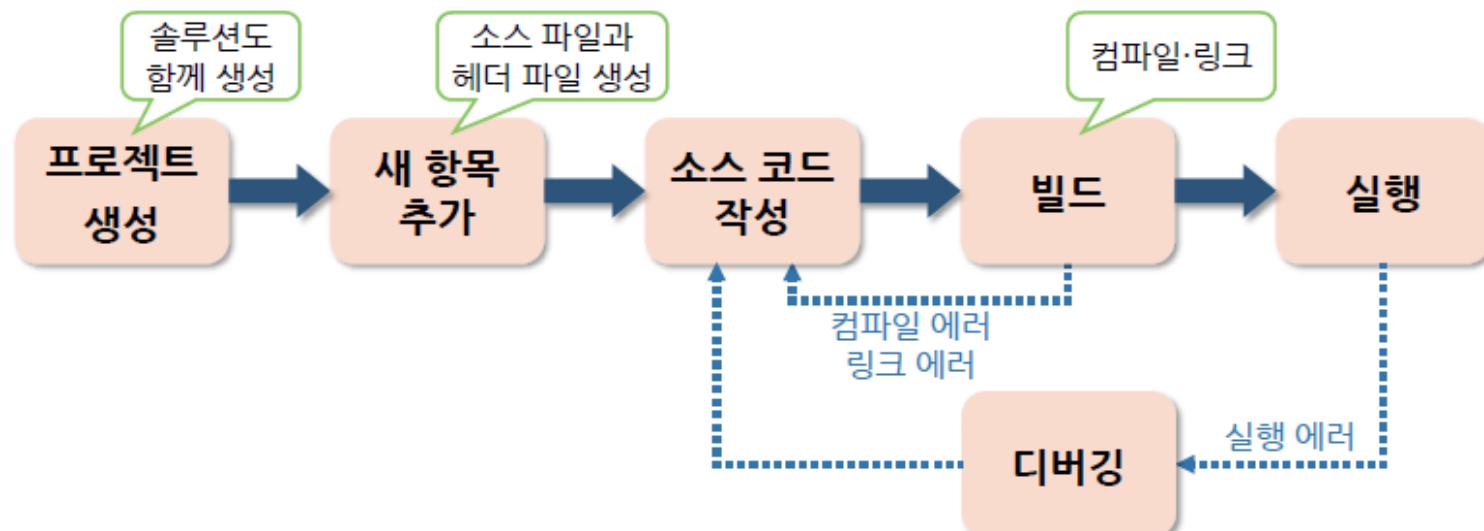
관련된 프로젝트를 모아서
솔루션을 구성하는 경우

통합개발환경_프로그램 개발과 컴파일 단계

❖ 프로그램 개발단계

- 1단계 : 프로젝트 생성
 - 프로젝트의 종류, 프로젝트명, 위치 지정
- 2단계 : 새 항목 추가(소스 파일 생성)
 - 소스 파일 확장자를 .c로 지정

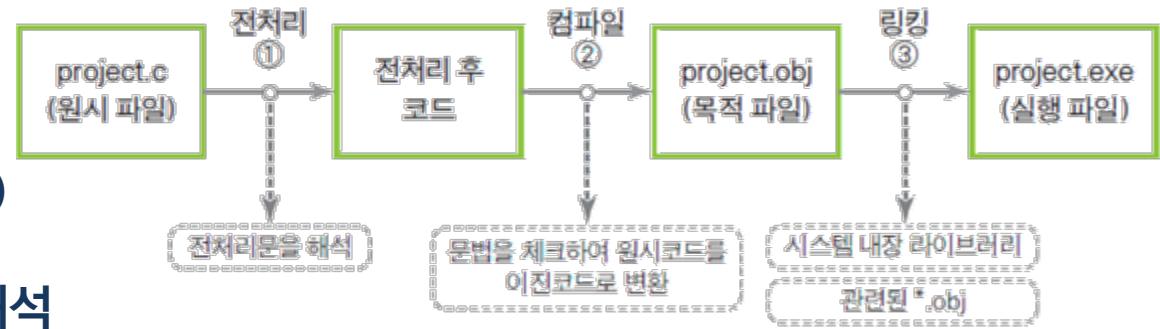
- 3단계 : 소스 파일 작성
- 4단계 : 빌드
 - 전처리기 처리 후 컴파일 및 링크 수행
- 5단계 : 실행
 - VC++ 안에서 직접 실행
- 6단계 : 디버깅
 - 프로그램의 실행 에러를 찾음



통합개발환경_프로그램 개발과 컴파일 단계

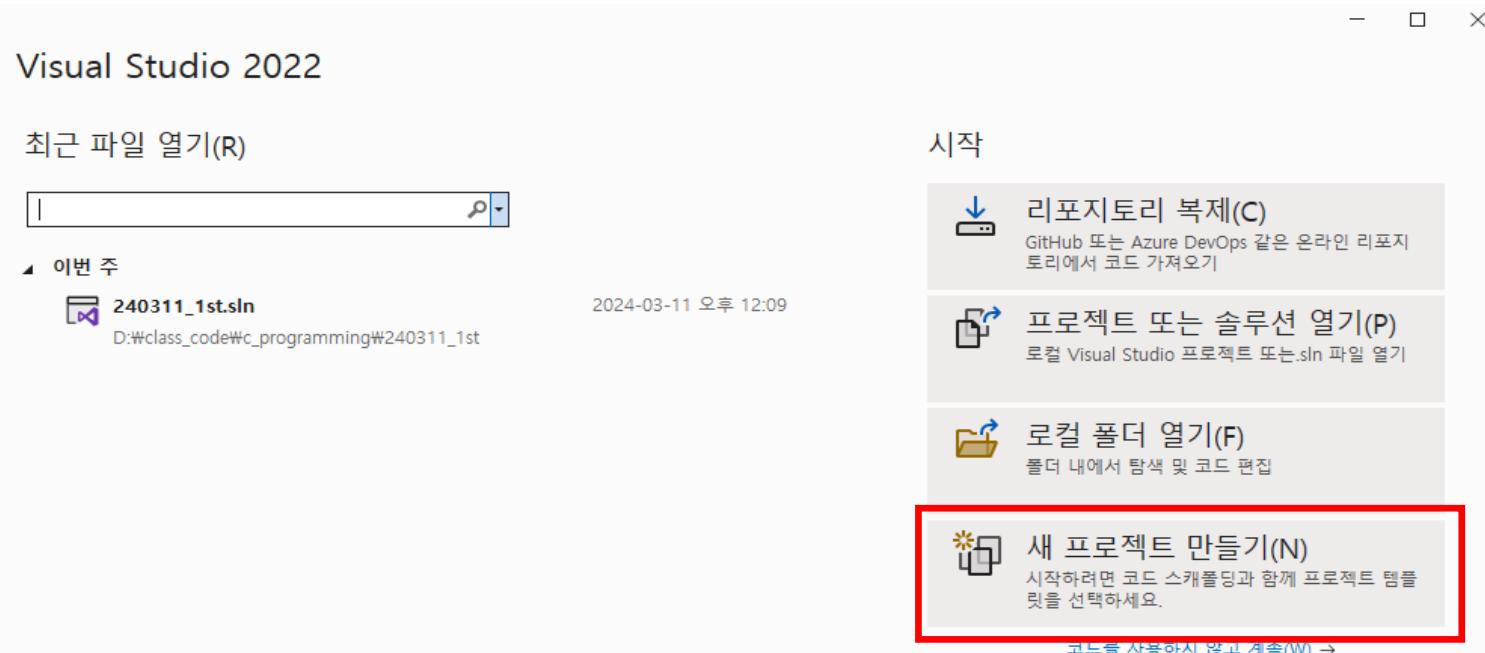
❖ 컴파일 과정

- **1단계**
 - 소스파일 작성(`project.c`)
- **2단계**
 - 전처리기가 전처리문 해석
 - ex) `#include <stdio.h>` -> 헤더파일 삽입
- **3단계**
 - 컴파일(고급언어가 기계어 코드로 번역) -> object 파일 생성
- **4단계**
 - 링크 (사용자가 생성한 object 파일과 라이브러리 함수 결합하여 실행파일 생성)
- **5단계**
 - 실행파일 실행
- **6단계**
 - 디버깅(오류가 있으면 소스코드를 수정하여 다시 컴파일 한다)



프로그램 개발과정_Hello World 제작(1/10)

❖ 새 프로젝트 생성(파일->새로만들기->프로젝트)



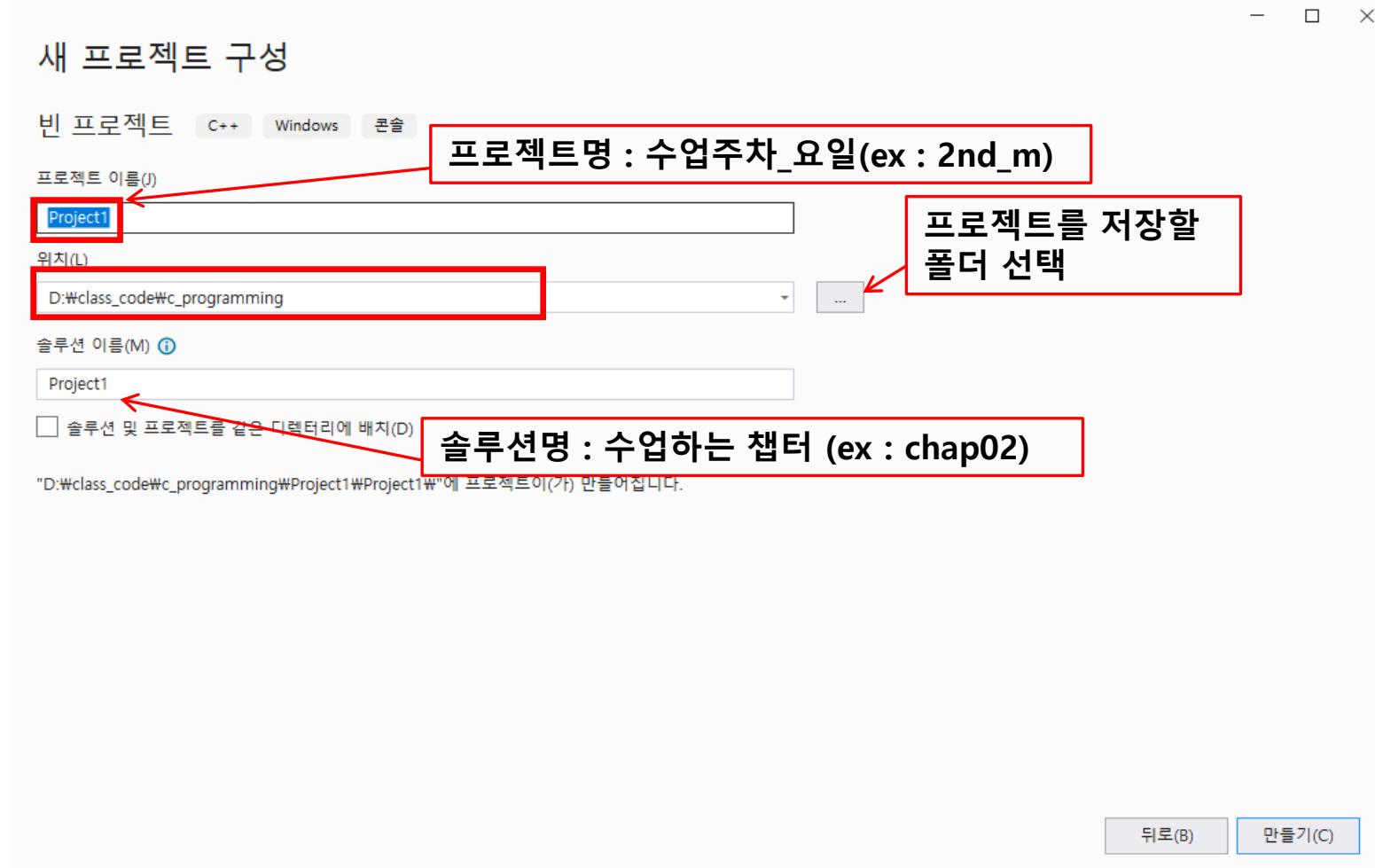
프로그램 개발과정_Hello World 제작(1/10)

❖ 새 프로젝트 생성 과정



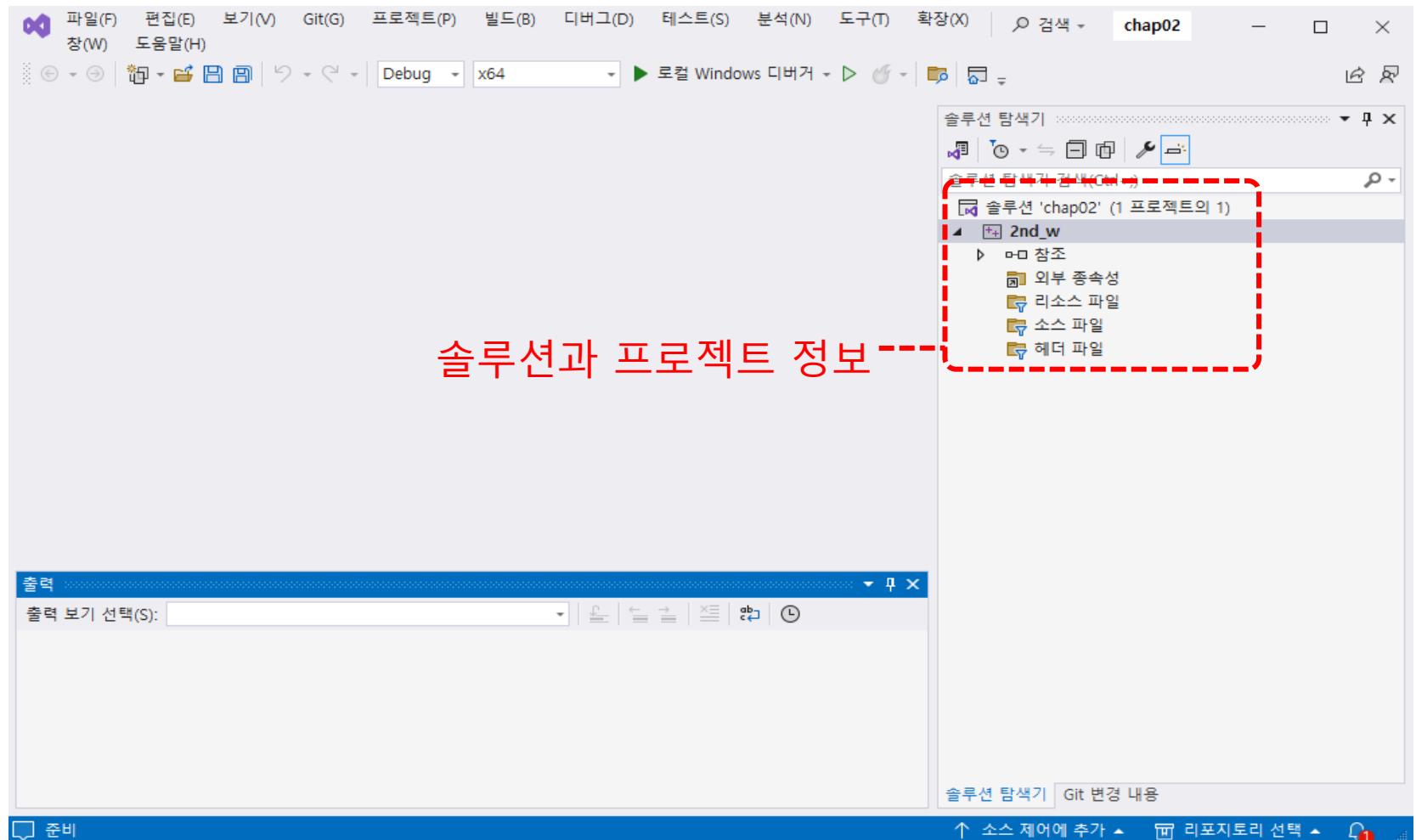
프로그램 개발과정_Hello World 제작(2/10)

❖ 프로젝트 이름 및 폴더 지정



프로그램 개발과정_Hello World 제작(4/10)

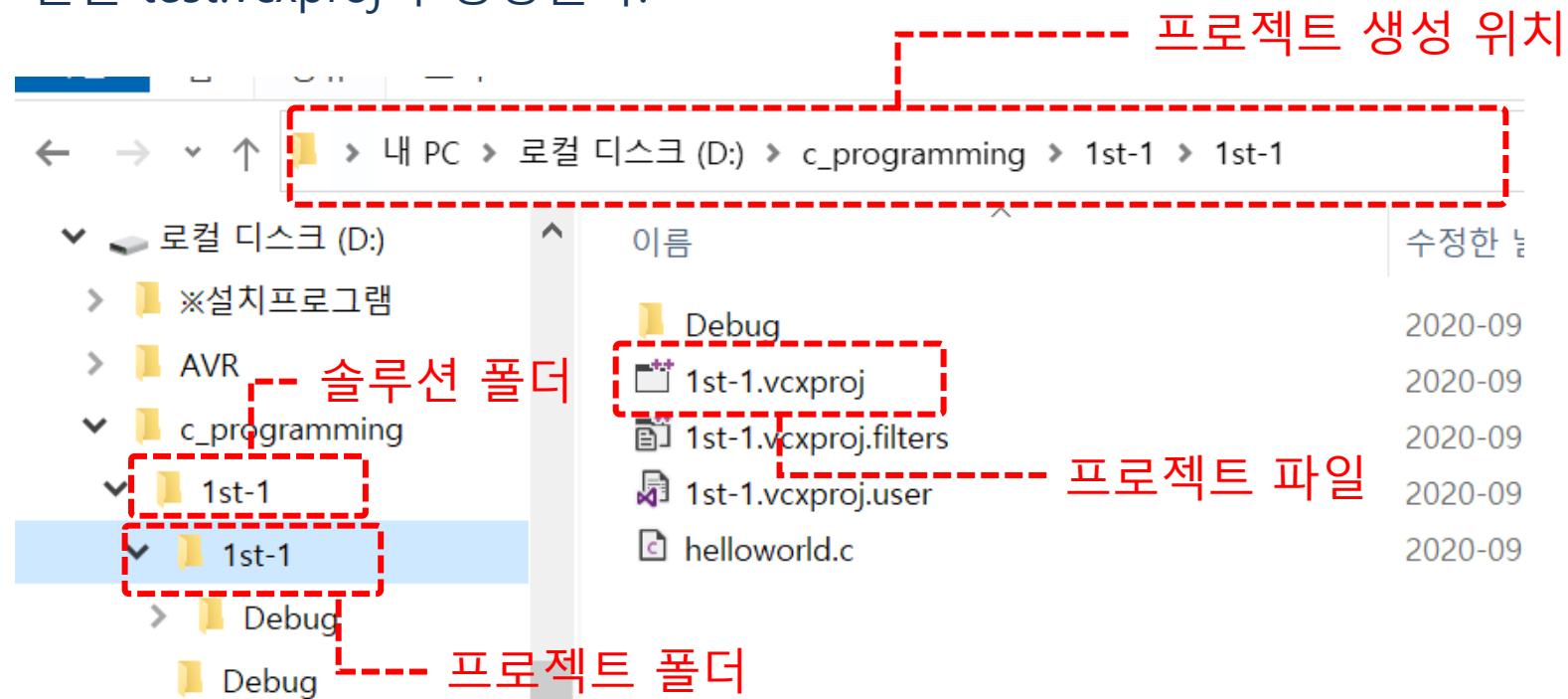
❖ 빙 프로젝트 생성



프로그램 개발과정_Hello World 제작(5/10)

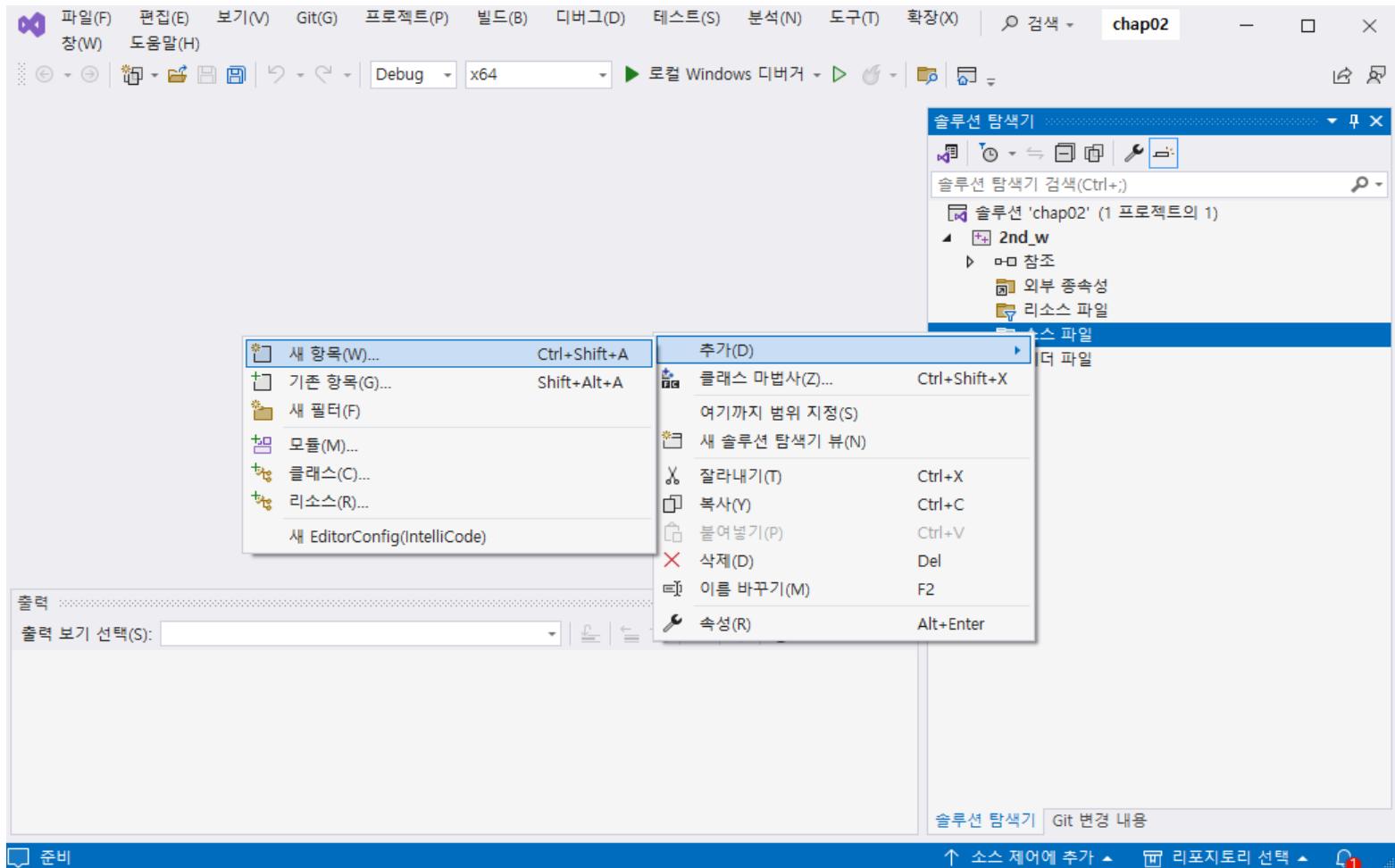
❖ 솔루션 폴더와 프로젝트 폴더

- **프로젝트를 생성하면, 디폴트로 프로젝트 이름과 같은 이름의 솔루션이 생성된다.**
 - 솔루션 폴더와 솔루션 폴더 안에 프로젝트 폴더 생성
 - 솔루션 폴더에는 솔루션 파일인 test.sln이, 프로젝트 폴더에는 프로젝트 파일인 test.vcxproj가 생성된다.



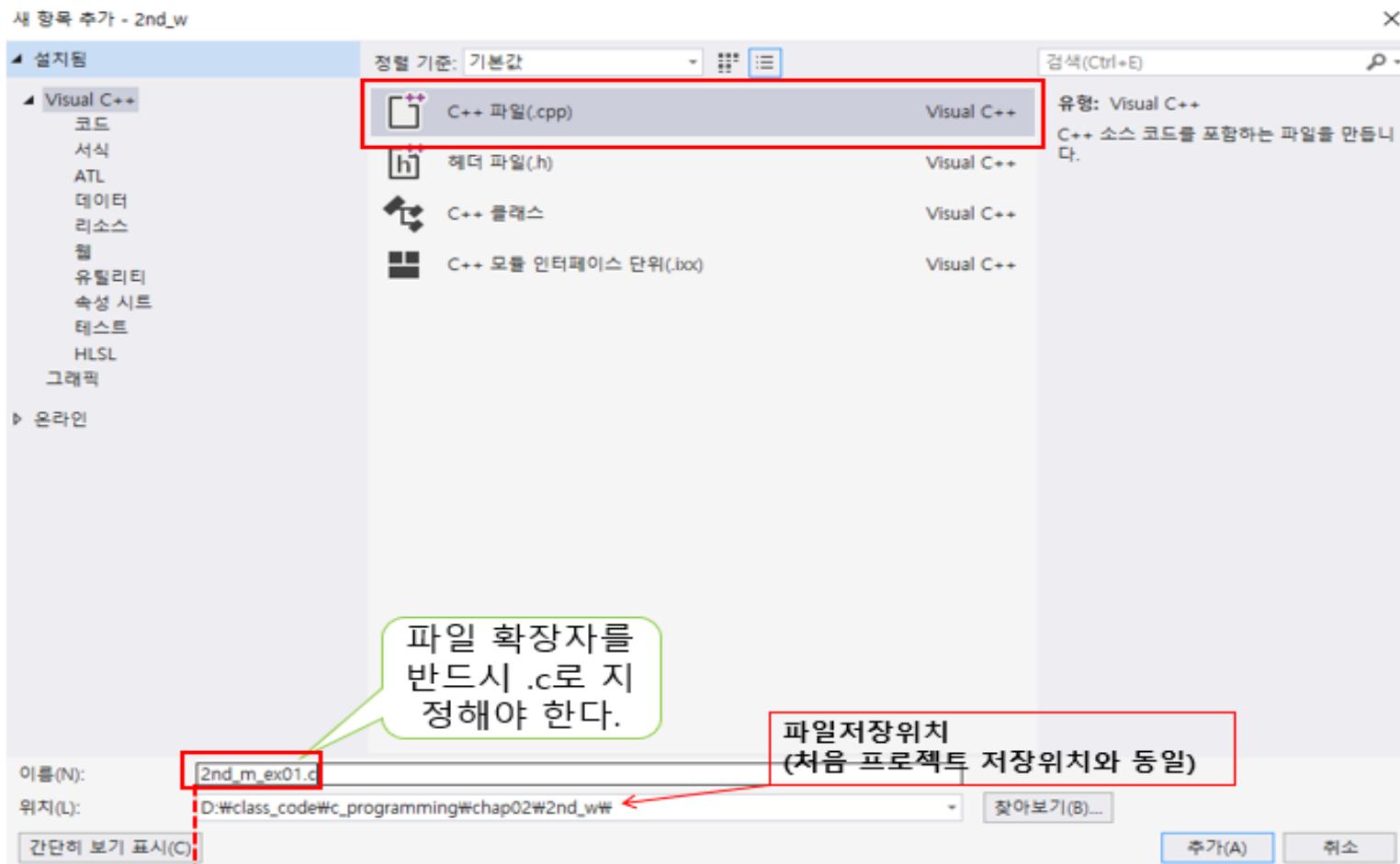
프로그램 개발과정_Hello World 제작(6/10)

❖ 새 항목 추가(소스코드 생성)



프로그램 개발과정_Hello World 제작(7/10)

❖ 새 항목 추가(소스코드 생성)



프로그램 개발과정_Hello World 제작(8/10)

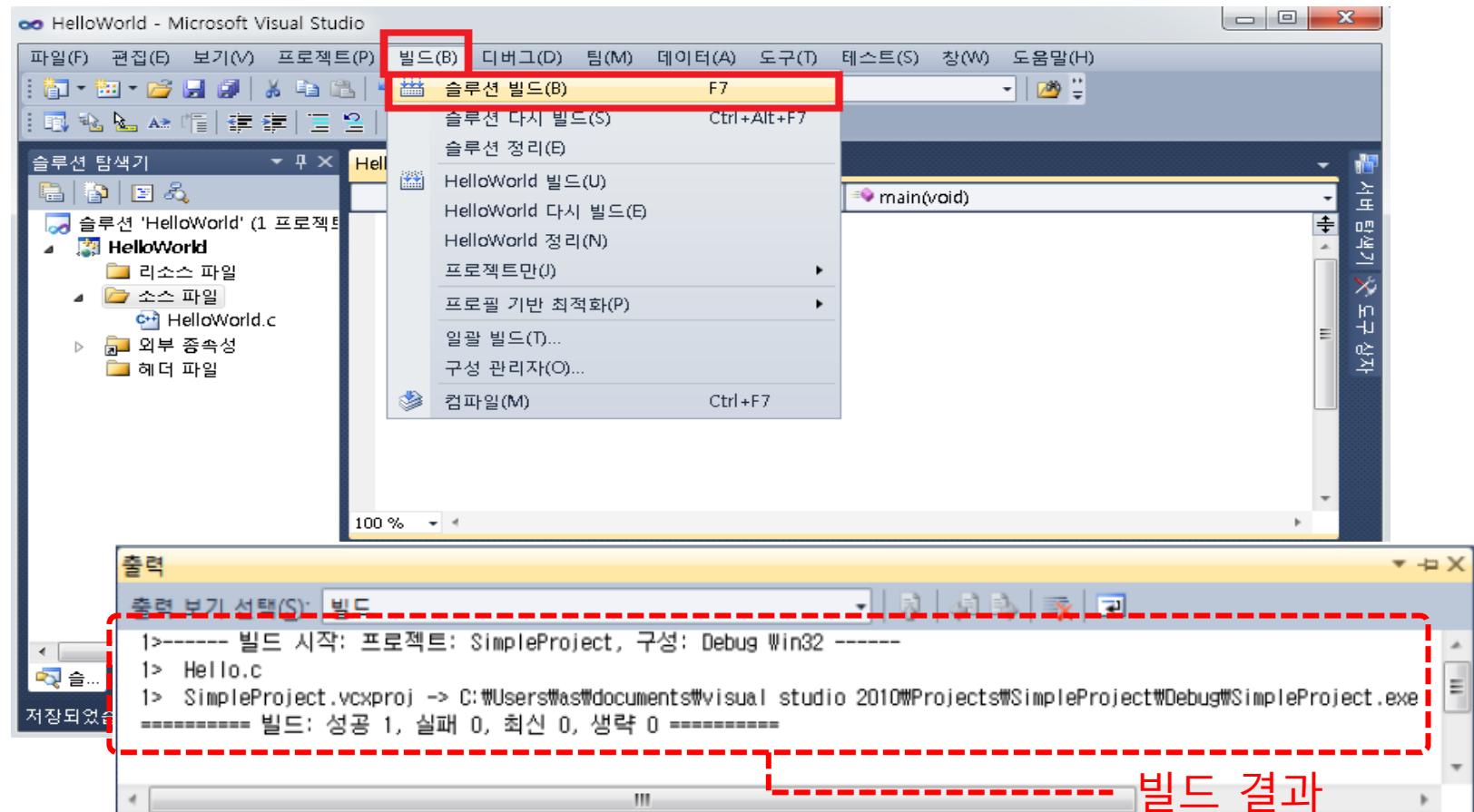
❖ 소스코드 작성

```
└/* HelloWorld.c */
  #include <stdio.h>

  int main(void)
  {
      printf("Hello, World! \n");
      return 0;
  }
```

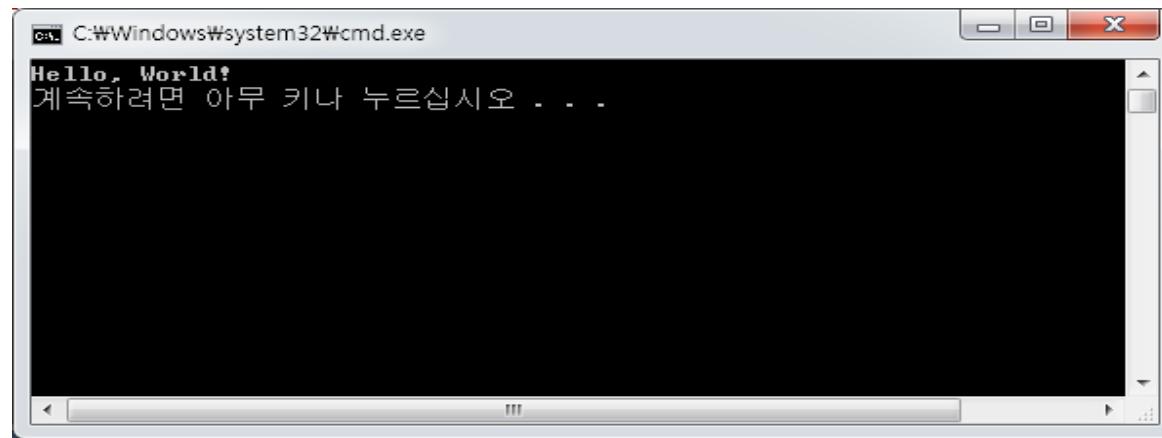
프로그램 개발과정_Hello World 제작(9/10)

❖ 프로그램 컴파일(빌드->솔루션빌드(F7))



프로그램 개발과정_Hello World 제작(10/10)

- ❖ 실행(CTRL+F5)
- ❖ 컴파일 에러와 링크 에러가 없으면 프로젝트 빌드 결과 실행 파일이 생성된다.
- ❖ [디버그] → [디버그하지 않고 시작]



- ❖ 직접 명령 프롬프트에 실행 파일의 완전 경로명을 입력한다.
 - D:\c_programming\1st-1\Debug\1st-1.exe

Visual Studio 기능

기능	메뉴 항목	단축키
새 프로젝트 생성	[파일] → [새로 만들기] → [프로젝트]	Ctrl+Shift+N
새 항목 추가	솔루션 탐색기에서 프로젝트 이름 선택 후 [프로젝트] → [새 항목 추가]	Ctrl+Shift+A
기존 항목 추가	솔루션 탐색기에서 프로젝트 이름 선택 후 [프로젝트] → [기존 항목 추가]	Shift+Alt+A
빌드	[빌드] → [솔루션 빌드]	F7
실행	[디버그] → [디버그하지 않고 시작]	Ctrl+F5
솔루션/프로젝트 열기	[파일] → [열기] → [프로젝트/솔루션]	Ctrl+Shift+O
소스 파일 전체 자동 들여쓰기/띄어쓰기	소스 파일 편집기 클릭 후 [편집] → [고급] → [문서 서식]	Ctrl+K, Ctrl+D
블록 자동 들여쓰기/띄어쓰기	블록 선택 후 [편집] → [고급] → [선택 영역 서식]	Ctrl+K, Ctrl+F
디버깅 시작	[디버그] → [디버깅 시작]	F5
한 단계씩 코드 실행	[디버그] → [한 단계씩 코드 시작]	F11
프로시저 단위 실행	[디버그] → [프로시저 단위 실행]	F10
중단점 설정/해제	[디버그] → [중단점 설정/해제]	F9

학습정리

❖ 프로그래밍 언어

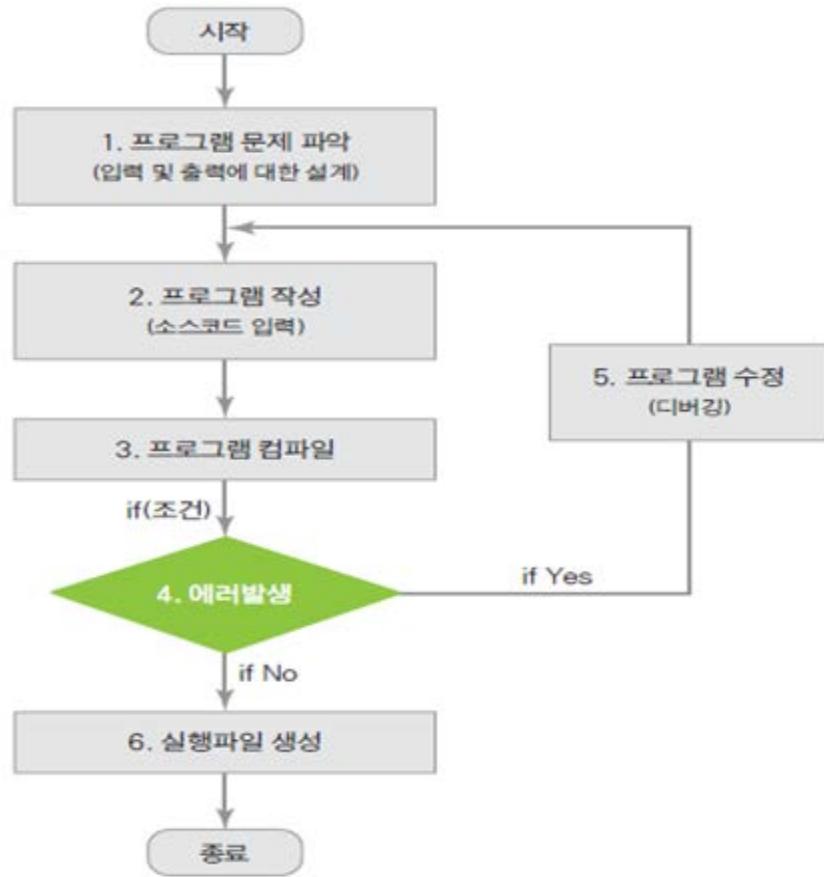
- 사람과 컴퓨터 사이에 존재하는 일종의 커뮤니케이션 수단이다.
- 사람은 기계어나 어셈블리어 대신 고급 언어를 이용해서 프로그램을 작성하고, 컴파일러가 이 프로그램을 기계어로 번역한다.
- C, C++, java, C#과 같은 고급 언어를 이용하면 프로그램을 개발하기도 쉽고, 유지 보수하기도 쉬워진다.

❖ C 언어의 특징

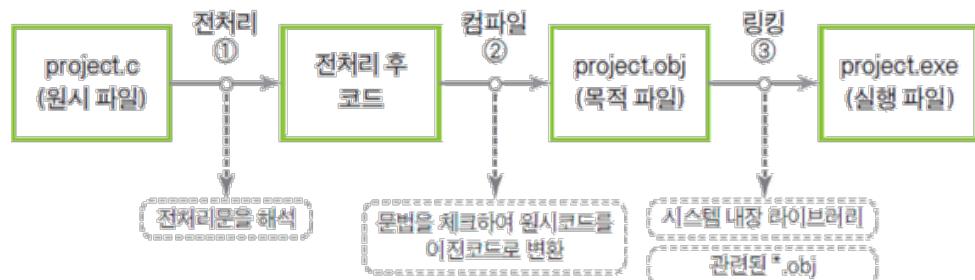
- 1972년에 데니스 리치(Dennis Ritchie)와 켄 톰슨(Ken Thomson)에 의해서 처음 만들어졌다.
- 특징은 간결성, 이식성, 효율성이다.
- 배우기나 사용하기는 어렵지만 프로그래머에게 최대한의 자유를 보장한다.

학습정리

프로그래밍 개발단계



프로그래밍 컴파일단계





CHAP 02 기초사항

학습목표

첫 번째 C 프로그램을 작성하면서 C 프로그램의 구성 요소에 대하여 알아본다.

C언어의 함수란 무엇이며, main()함수의 역할에 대하여 알아본다.

C 프로그램에서 출력함수 printf와 입력함수 scanf에 대하여 알아본다.

오류 수정 및 디버깅에 관한 내용을 알아본다.

목차



첫번째 C 프로그램

- 프로그램 코드
- 주석/표준라이브러리
- main 함수
- 출력



두번째 C 프로그램

- 프로그램 코드
- 변수
- print 함수를 이용한 출력
- scanf 함수를 이용한 입력



오류 수정 및 디버깅

- 오류의 종류
- 오류 수정 과정
- 디버깅

첫번째 C 프로그램_프로그램 코드

❖ “Hello World” 문자열을 출력하는 프로그램

```
/* HelloWorld.c */           → 주석
#include <stdio.h>           → 헤더파일 포함
                                → 입출력 라이브러리를 사용하기 위한 준비
int main(void)               → main() 함수 : 프로그램의 진입점 함수
{
    printf("Hello, World! \n"); → 화면 출력
    return 0;                → 0을 반환하며 main()함수 종료
}
```

실행 결과

- / * / *

Hello World

첫번째 C 프로그램_주석

- ❖ 소스 코드에 대한 정보를 제공하거나 소스 코드의 일부분을 컴파일하지 않게 만들 수 있다.
 - 컴파일러에 의해서 무시된다.
- ❖ C의 주석은 /*로 시작하고, */로 끝남
- ❖ //을 이용한 한줄 주석
 - C99에 추가된 기능 → 대부분의 C 컴파일러에서 사용 가능
- ❖ 주석 처리(comment out)
 - 이미 작성한 소스 코드를 컴파일하지 않게 만드는 것

```
printf("First C Program\n");
//printf("Comment out\n");
```

이 줄 전체를
주석 처리

코드 중간에 주석을
넣을 수 있다.

```
printf(/*출력할 내용*/"First C Program\n");
```

첫번째 C 프로그램_주석

❖ 주석예제

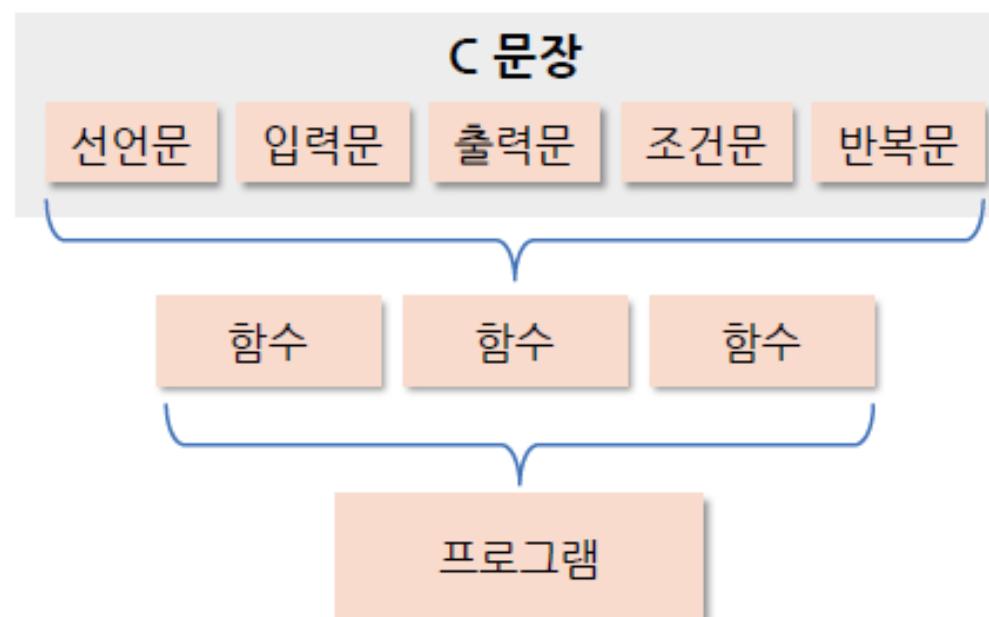
```
1. /*  
2.      주석처리 된 문장1  
3.      /* 단일 행 주석처리 */  
4.      주석처리 된 문장2  
5. */
```

```
1. /*  
2.      주석처리 된 문장1  
3.      // 단일 행 주석처리  
4.      주석처리 된 문장2  
5. */
```

```
/*  
제 목: Hello world 출력하기  
기 능: 문자열의 출력  
파일이름: HelloComment.c  
수정날짜: 2014. 07. 15  
작성자: 윤성우  
*/  
#include <stdio.h> // 헤더파일 선언  
  
int main(void) // main 함수의 시작  
{  
    /*  
        이 함수 내에서는 하나의 문자열을 출력한다.  
        문자열은 모니터로 출력된다.  
    */  
    printf("Hello world! \n"); // 문자열의 출력  
    return 0; // 0의 반환  
} // main 함수의 끝
```

첫번째 C 프로그램_main함수

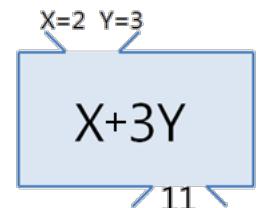
- ❖ C 프로그램은 main 함수 또는 main 함수와 여러 함수들로 구성
- ❖ 각각의 함수는 세미콜론(;)으로 끝나는 문장으로 구성
 - 선언문, 입력문, 출력문, 조건문, 반복문 등



첫번째 C 프로그램_main함수

❖ 함수(Function)

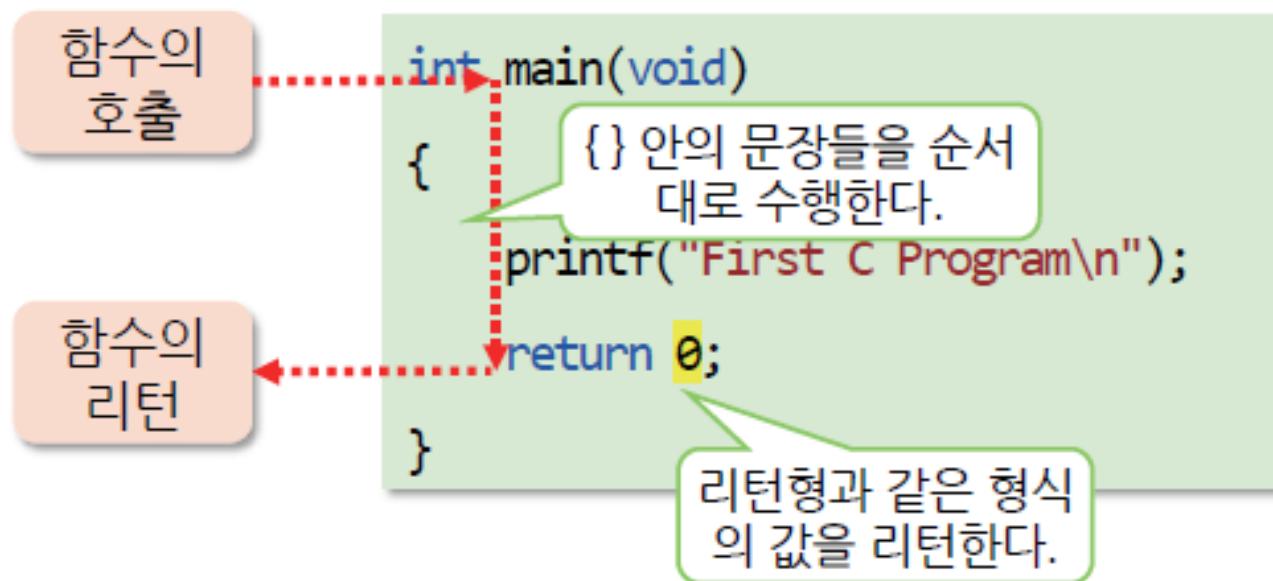
- 프로그램에서 처리할 내용을 모아두는 기본 단위
- 입력과 출력이 존재
- 함수를 만들 때는 **함수의 리턴형, 함수 이름, 매개 변수가 필요하다.**
- 함수가 처리할 내용은 {} 안에 써준다.
- 함수는 문장들로 구성된다.



첫번째 C 프로그램_main함수

❖ 함수의 호출 및 리턴 과정

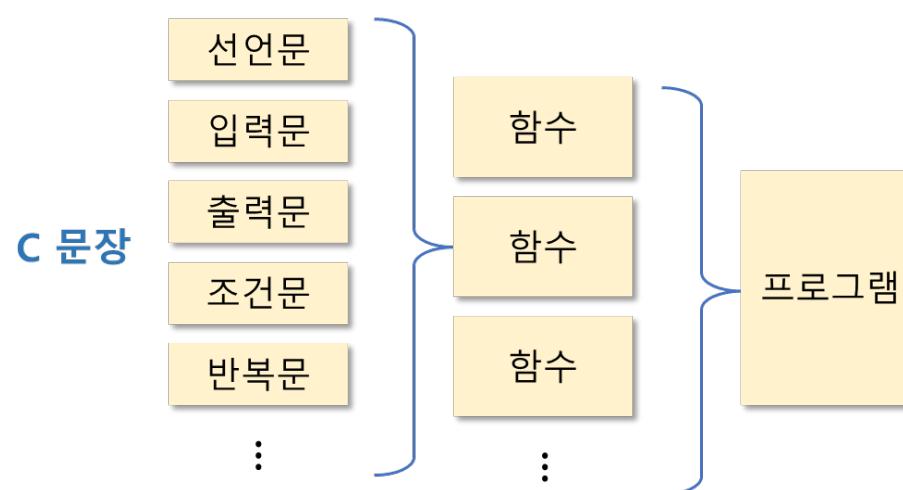
- 함수가 호출되면 함수 안에 있는 문장들이 순차적으로 수행된다.
- 함수의 끝()을 만나거나 리턴문을 만나면, 함수를 호출한 곳으로 되돌아간다.
- 리턴문에서 return 다음에 리턴할 값을 써준다.



첫번째 C 프로그램_main함수

❖ 문장(Statement)

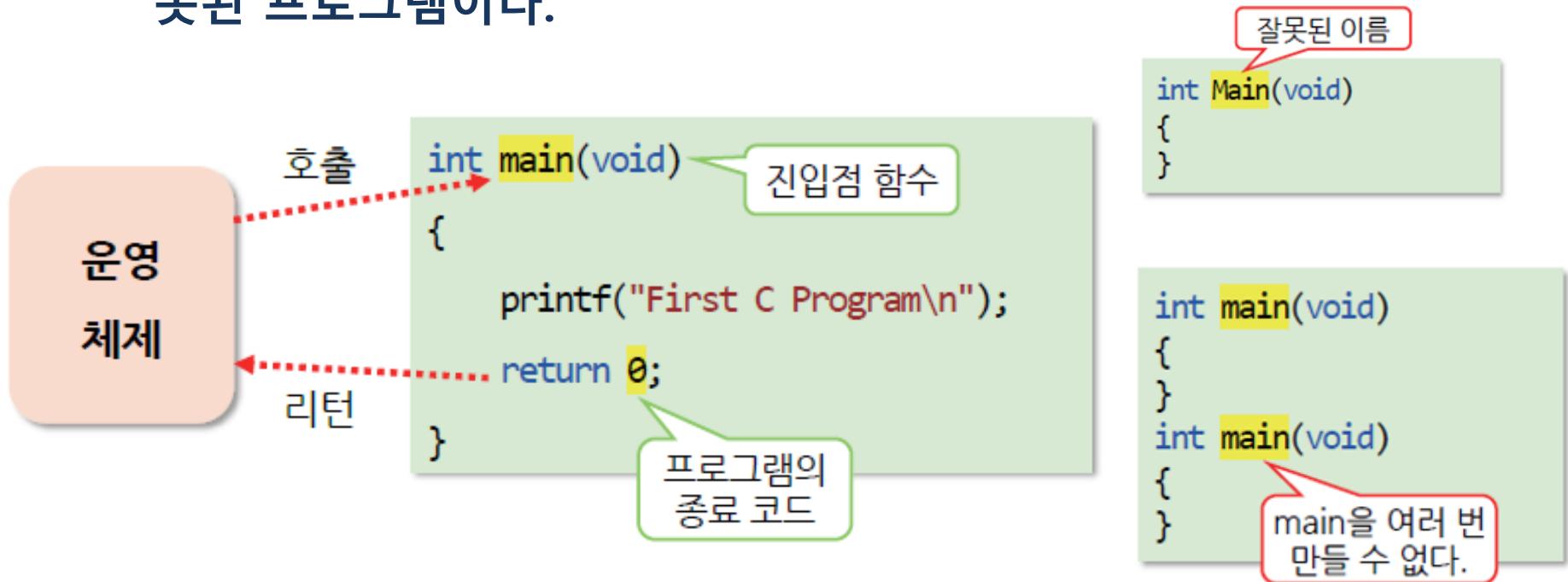
- **함수를 구성하는 기본 단위**
- C 프로그램의 각 문장은 세미콜론(;)으로 끝난다.
 - 문법 체크의 주요 단위
 - 초심자의 가장 큰 에러 요인
 - 세미콜론
 - 오타
- ~문 : 선언문, 입력문, 출력문, 조건문, 반복문 등



첫번째 C 프로그램_main함수

❖ 진입점 함수(entry-point function)

- C 프로그램이 처음 시작될 때 호출되도록 약속된 함수
- **main** 함수가 진입점 함수이므로, C 프로그램에는 반드시 main이 필요하다. (함수들 중 main() 함수를 찾아 자동으로 실행시킨다)
- main 함수가 없거나, 여러 개의 main 함수를 가진 프로그램은 잘못된 프로그램이다.



첫번째 C 프로그램_main함수

❖ main 함수의 원형

- main 함수는 void형을 리턴하거나 int형을 리턴한다.
 - void main(void)
 - int main(void)

❖ main 함수의 리턴 값

- 프로그램의 종료 코드(exit code)를 리턴함
- 프로그램이 종료될 때 운영체제에게 넘겨주는 값

```
int main(void)
{
    ...
    return 0;      정상 종료
}
```

```
int main(void)
{
    if (...) //메모리 할당 실패
        return 1;  비정상 종료
    ...
}
```

❖ 예외적으로 main 함수에서만 return 생략 가능

- 0 리턴

첫번째 C 프로그램_main함수

❖ 들여쓰기(Indentation)

- C 프로그램의 각 문장은 세미콜론(:)으로 끝난다.

```
printf(  
    "Hello World\n"  
);
```

한 문장을 여러 줄에 걸쳐서 작성할 수 있다.

```
printf("Hello"); printf("World\n");
```

여러 문장을 한 줄에 작성할 수 있다.

- 알아보기 쉽도록 한 줄에 한 문장씩 작성한다.
- 같은 블록에 속한 문장들을 들여쓰기를 하는 것이 좋다. 들여 쓴 문장은 이전 명령에 포함된다는 의미로 개발자들은 이해하게 된다

```
int main()  
{  
    printf("Good\n");  
    return 0;  
}
```

들여 쓰기를 하면
알아보기 쉽다.

```
int main() {  
    printf("Not Good\n");  
    return 0; }
```

들여 쓰기를 안 하면
알아보기 어렵다.

첫번째 C 프로그램_입출력

❖ 콘솔 프로그램

- 콘솔(명령 프롬프트)에서 실행되는 프로그램
- 키보드로부터 입력을 받아서 처리 결과를 콘솔에 텍스트로 출력

❖ 입출력 라이브러리

- C 프로그램마다 공통적으로 필요한 입력과 출력 기능 제공

표준 입력



표준 출력

```
C:\선택 C:\Windows\system32\cmd.exe
C:\work\chap02\ex02_01\Debug>ex02_01.exe
First C Program
C:\work\chap02\ex02_01\Debug>
```

콘솔 프로그램

첫번째 C 프로그램_표준라이브러리

- ❖ 기본적으로 제공되는 함수들을 표준 라이브러리라 함.
- ❖ 표준 라이브러리의 특징
 - 파일 입·출력, 문자열 조작, 수학 계산 등을 처리하기 위한 함수들을 제공
 - 라이브러리 함수는 컴파일 과정에서 프로그램 코드에 적재된다
 - C 컴파일러는 라이브러리 함수들을 처리하기 해서 헤더파일에 있는 정보를 사용한다. 따라서 소스코드에 헤더파일을 포함시켜야 한다.
 - #include 지시자를 사용하여 헤더 파일 삽입한다.

④ 형식

#include <헤더 파일명>

④ 예

#include <stdio.h> ← #include 지시자는 마지막에 세미콜론으로 끝나지 않는다.

첫번째 C 프로그램_표준라이브러리

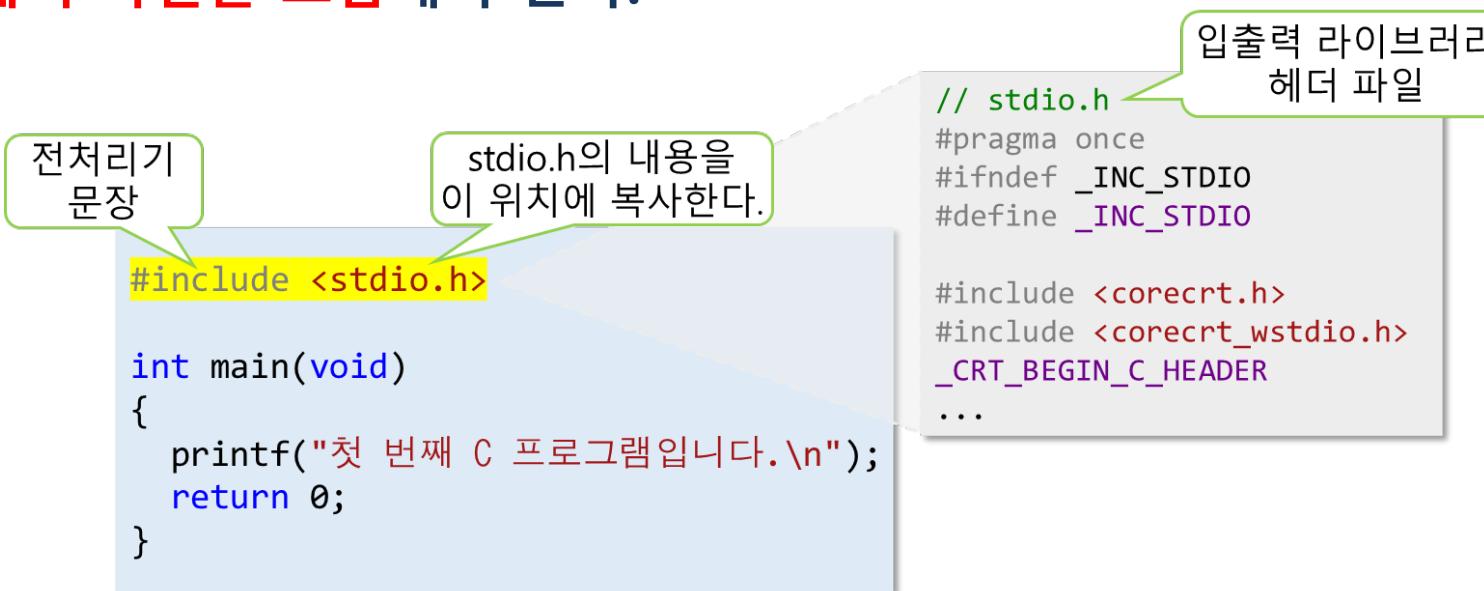
❖ #include <stdio.h>

- 헤더 파일(*.h) : 라이브러리 함수에 대한 정보(함수명, 리턴형, 매개 변수)를 제공하는 파일
 - 확장자 : *.h (header의 약자)
- 라이브러리를 사용하려면 헤더 파일을 #include로 포함해야 함
 - 헤더파일의 내용을 현재 소스 파일로 복사 → 컴파일 단위에서 실행
- <stdio.h>는 표준 라이브러리 함수 중 입·출력을 위한 함수가 호출될 때 필요한 정보를 담고 있는 헤더 파일
 - printf() 함수가 사용되기 위해 컴파일러는 헤더파일 정보를 참조

첫번째 C 프로그램_콘솔 출력

❖ 출력을 위한 준비

- 입출력 라이브러리를 사용하려면 먼저 입출력 라이브러리에 대한 헤더 파일을 포함해야 한다.



- 헤더 파일 없이 `printf` 함수를 호출하면, 컴파일 경고(warning)가 발생한다.

첫번째 C 프로그램_콘솔 출력

❖ printf 함수

- 출력할 내용을 " "로 묶어서 printf 함수의 () 안에 써준다.
- 줄바꿈 문자('`\\n`')

```
printf("First C Program\\n");
```

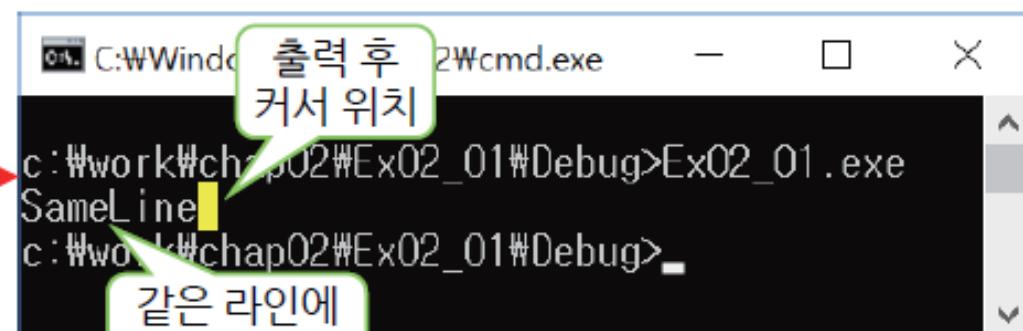
텍스트를 출력하고
커서를 다음 줄로 이동

- 이전 출력의 마지막 커서 위치부터 연속해서 출력한다.

```
int main()
{
    printf("Same");
    printf("Line");
}
```

줄바꿈 문자를
사용하지 않는 경우

실행

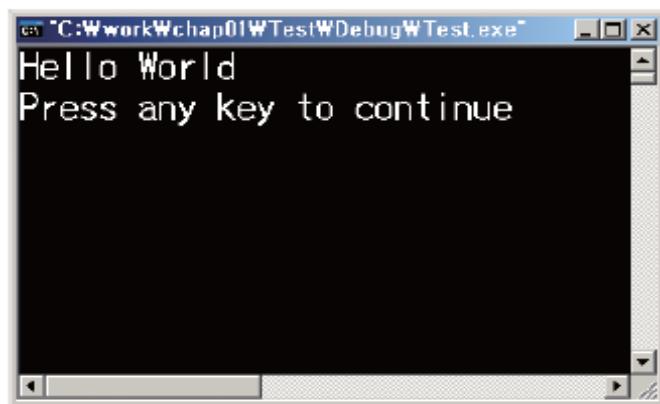


같은 라인에
출력한다.

첫번째 C 프로그램_출력

❖ 콘솔 프로그램

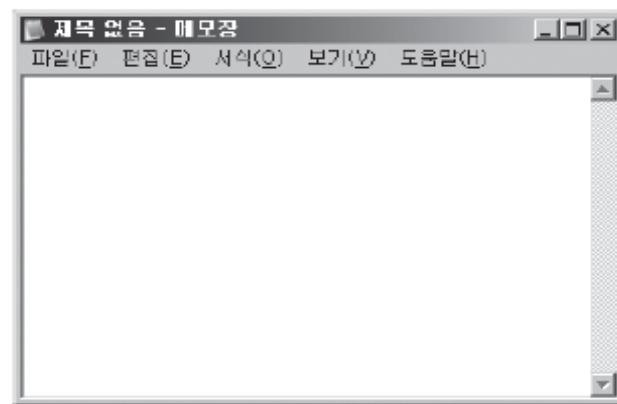
- 콘솔 창에서 실행되는 프로그램
- 텍스트 기반의 입출력만 처리
- 한 번에 하나의 프로그램만 실행



〈콘솔 프로그램〉

❖ 윈도우 프로그램

- 일반적인 윈도를 띄우고, 윈도에서 입출력을 처리하는 프로그램
- 그래픽 출력이 가능
- Win32 API 라이브러리나 MFC 라이브러리가 추가로 필요

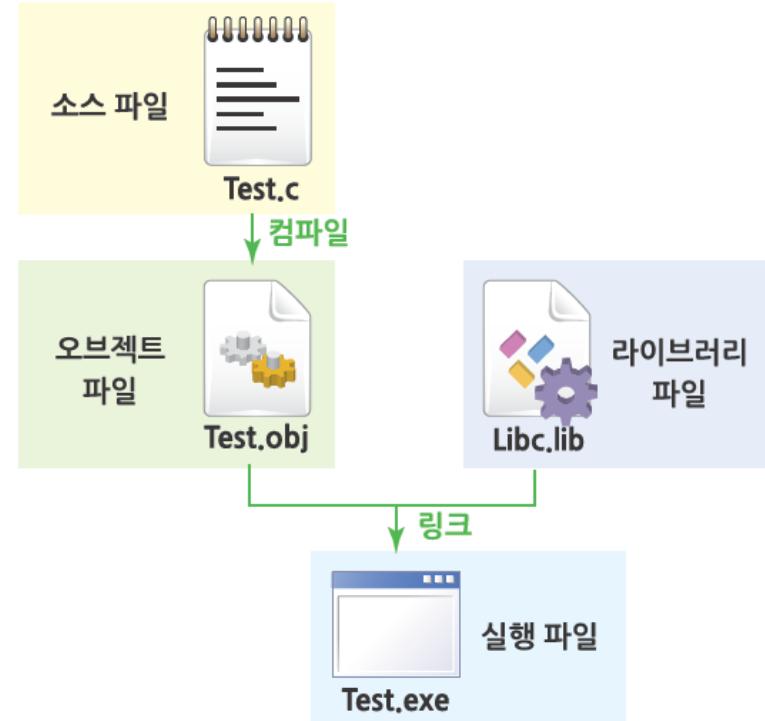


〈윈도 프로그램〉

첫번째 C 프로그램_출력

❖ 라이브러리

- 자주 사용되는 기능을 미리 준비해둔 것
- 컴파일된 오브젝트 코드를 묶어놓은 .lib 확장자를 가진 바이너리 파일



변수

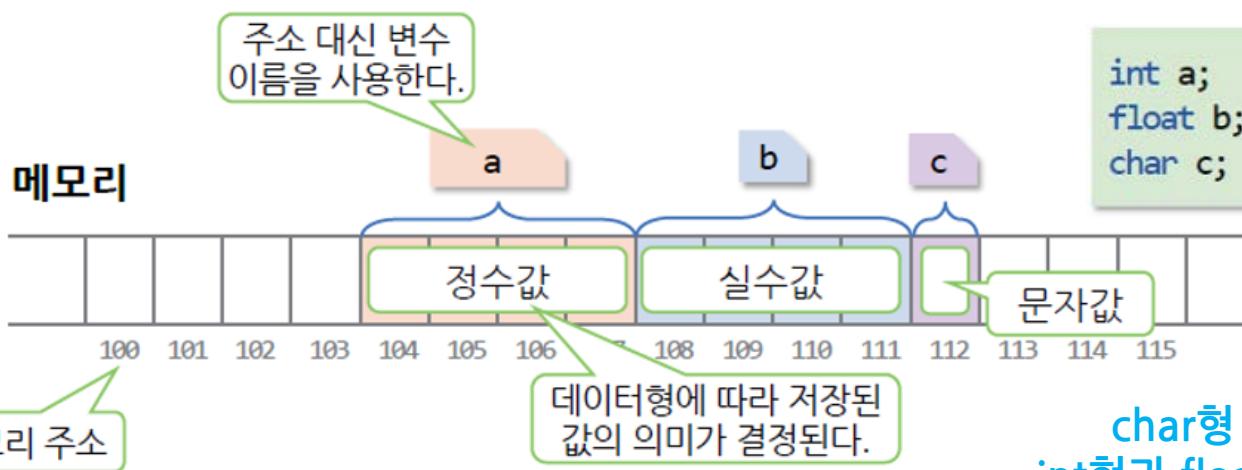
❖ 변수

- 어떤 값을 저장하기 위한 공간 → 이름으로 접근
- 컴퓨터 시스템에서는 값을 저장하기 위해서 메모리 사용
 - 메모리 공간에 대하여 이름을 붙여 두고 이름으로 접근

❖ 메모리

- 연속된 바이트의 모임
- 메모리에 접근할 때는 1 바이트 단위로 접근
- 저장할 값의 형식에 따라서 사용되는 공간의 크기가 결정된다.

변수는 선언 후
사용해야 한다.



```
int a; // 정수형 변수 선언
float b; // 실수형 변수 선언
char c; // 문자형 변수 선언
```

char형 변수는 1바이트 크기
int형과 float형 변수는 4바이트 크기

변수

❖ 변수의 선언

- 변수를 선언할 때는 **변수의 데이터형과 변수의 이름이 필요하다.**
- **C의 데이터형**
 - char, int, float 등

형식

데이터형 변수명;

사용예

```
int a;      // 정수형 변수 선언  
float b;    // 실수형 변수 선언  
char c;     // 문자형 변수 선언
```

❖ 변수의 이름

- 영문자와 숫자, 밑줄 기호(_)를 이용
- 변수의 이름 중간에는 빈칸을 사용할 수 없다.
- 첫 글자로는 반드시 영문자나 밑줄 기호가 와야 한다.



int 2022income; // 숫자로 시작하면 안된다. (컴파일 에러)



float tax rate; // 빈칸이 들어가면 안된다. (컴파일 에러)

변수

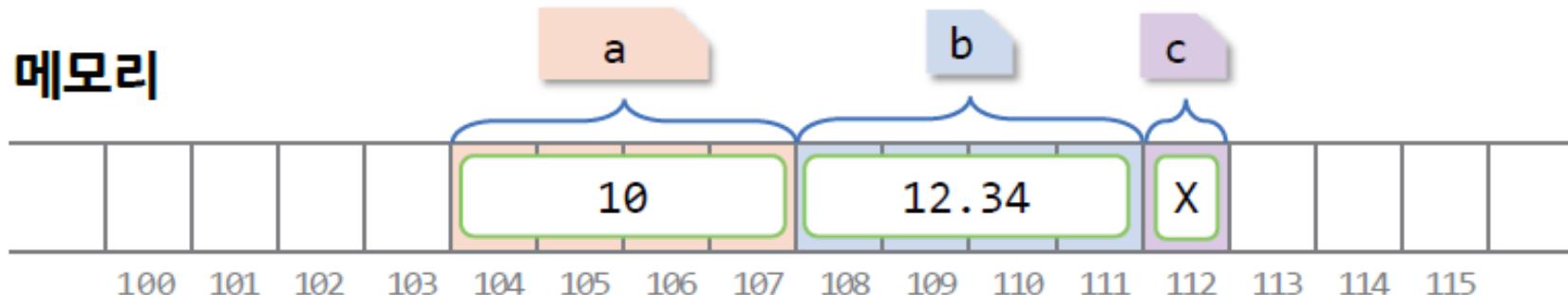
❖ 변수의 사용

- 변수명을 이용하면 변수의 값을 읽어올 수 있다.
- 변수에 값을 대입하려면 =를 이용한다.
- 변수의 데이터형과 같은 형식의 값을 대입해야 한다

```
a = 10;  
b = 12.34;  
c = 'X';
```

변수의 데이터형과 같은
형의 값을 대입해야 한다.

메모리



printf함수를 이용한 출력

❖ printf함수

- 문자열을 출력하는 기능 외에 값을 서식에 맞춰 출력하는 기능을 제공
- 형식 문자열
 - " "안에 %와 약속된 문자로 서식 지정

형식

`printf(형식문자열, 출력할값, …);`

사용예

```
printf("%d", a);
printf("%f", b);
printf("%c", c);
printf("%d %x", a, a);
```

printf함수를 이용한 출력

❖ printf 함수의 서식 지정자(1/3)

서식 지정자	의미	사용 예	실행 결과
%d	정수를 10진수로 출력	<code>int a = 10; printf("%d", a);</code>	10
%x	정수를 16진수로 출력	<code>int a = 10; printf("%x", a);</code>	a
%X	정수를 16진수로 출력	<code>int a = 10; printf("%X", a);</code>	A
%f, %F	실수를 부동소수점 표기 방식으로 출력	<code>float b = 1.23; printf("%f", b);</code>	1.230000
%e, %E	실수를 지수 표기 방식으로 출력	<code>float b = 1.23; printf("%e", b);</code>	1.230000e+00
%c	문자 출력	<code>char c = 'A'; printf("%c", c);</code>	A

printf함수를 이용한 출력

◆ printf 함수의 서식 지정자(2/3)

- 서식 지정자와 출력할 값은 순서대로 대응되며, 서식 지정자의 개수와 출력할 값의 개수가 일치해야 한다.

서식 지정자와 출력할 값이 순서대로 대응된다.

```
printf("%d %x\n", num, num);
```

123 7b

```
printf("%f %e\n", x, x);
```

1.230000 1.230000e+00

서식 지정자와 출력할 값의 개수가 일치해야 한다.

printf함수를 이용한 출력

❖ 서식 지정자를 이용해서 출력하기

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int a;
06     float b;
07     char c;
08
09     a = 10;
10     b = 1.23;
11     c = 'A';
12
13     printf("%d %x\n", a, a);
14     printf("%f %e\n", b, b);
15     printf("%c\n", c);
16 }
```

변수의 선언

변수의 대입

10진수, 16진수로 정수 출력

부동소수점, 지수 표기로 실수 출력

문자 출력

실행 결과

```
10 a
1.230000 1.230000e+00
A
```

printf함수를 이용한 출력

❖ printf 함수의 서식 지정자(3/3)

■ 문자 폭 지정

- %와 d 사이에 정수로 써준다.

6문자 폭에 맞춰 오른쪽
으로 정렬해서 출력

```
printf("%6d\n", x);  
printf("%-6d\n", x);
```

6문자 폭에 맞춰 왼쪽
으로 정렬해서 출력

■ 정밀도 지정

- 실수의 정밀도
 - 소수점 이하 자릿수
 - %와 f 사이에 .과 정수로 지정

소수점 이하 2자리로
실수를 출력

```
printf("%.2f\n", y);  
printf("%6.2f\n", y);
```

폭과 정밀도 지정

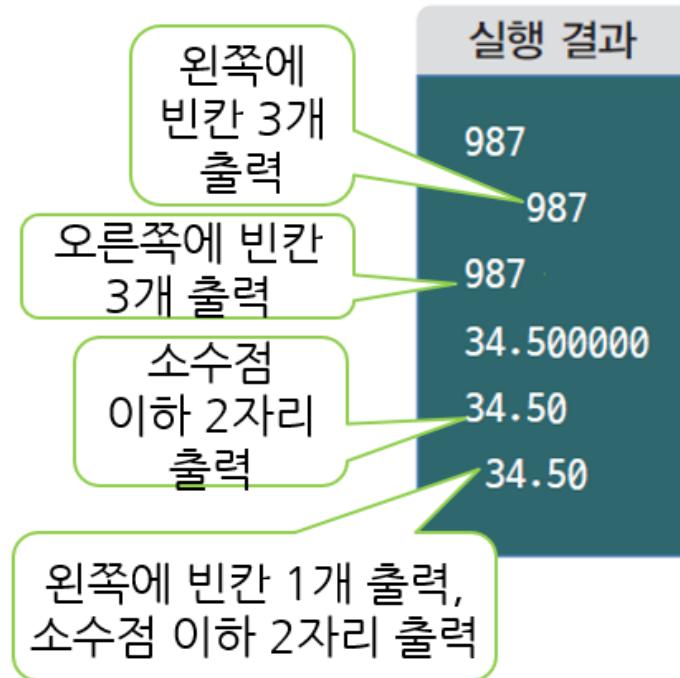
printf함수를 이용한 출력

❖ 문자 폭 지정하기

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int x = 987;
06     float y = 34.5;
07
08     printf("%d\n", x);
09     printf("%6d\n", x);
10     printf("%-6d\n", x);
11
12     printf("%f\n", y);
13     printf("%.2f\n", y);
14     printf("%6.2f\n", y);
15 }
```

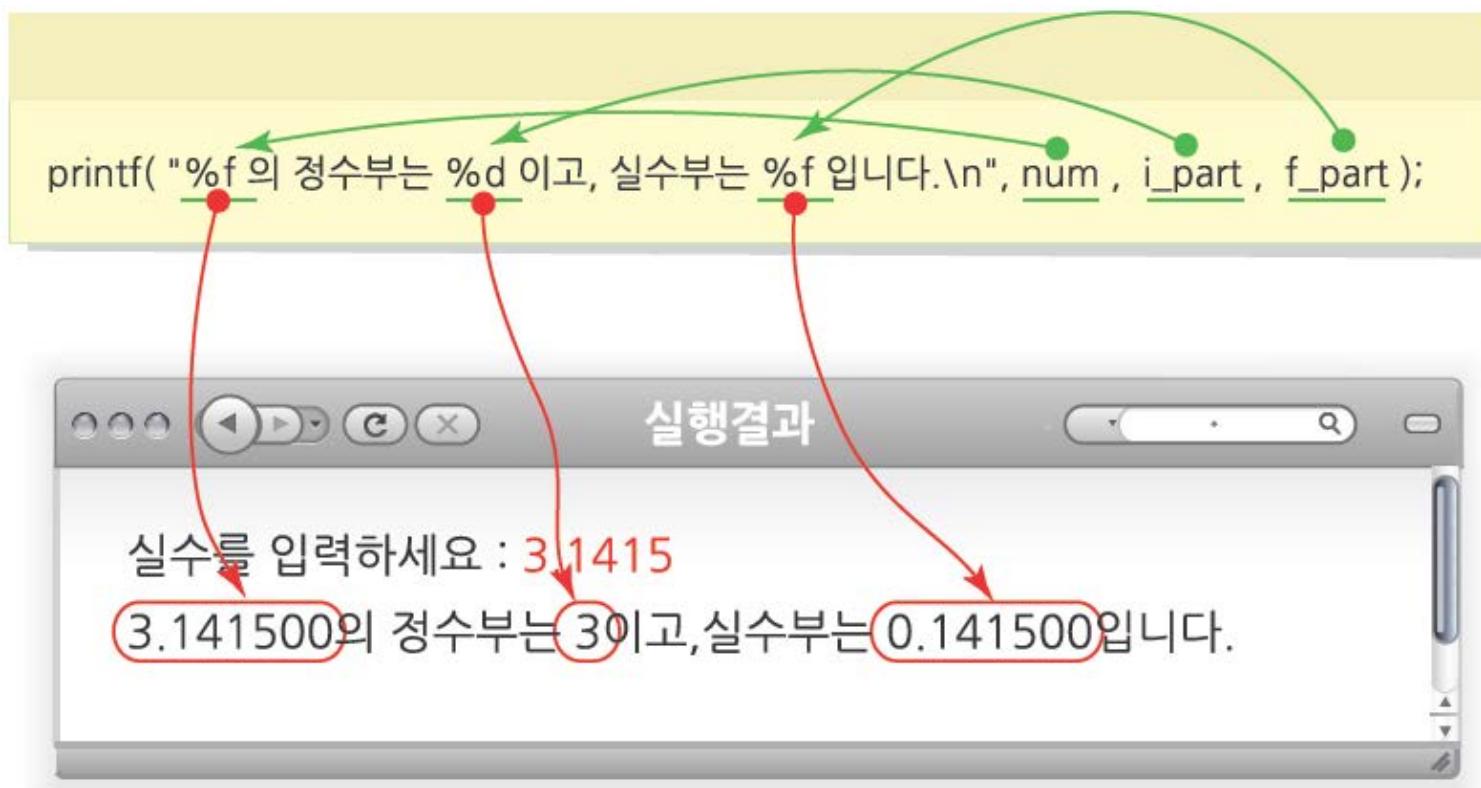
문자 폭을
지정하지
않으면
왼쪽부터
출력

정밀도를
지정하지
않으면
소수점 이하
6자리를
출력



두번째 C 프로그램 printf함수를 이용한 출력

❖ printf 함수의 사용 예



scanf 함수를 이용한 입력

- ❖ 콘솔에서 키보드로 입력한 값을 변수로 읽어온다.
 - 형식 문자열과 변수 이름을 지정
 - 변수 이름 앞에는 &를 써주어야 한다.

형식

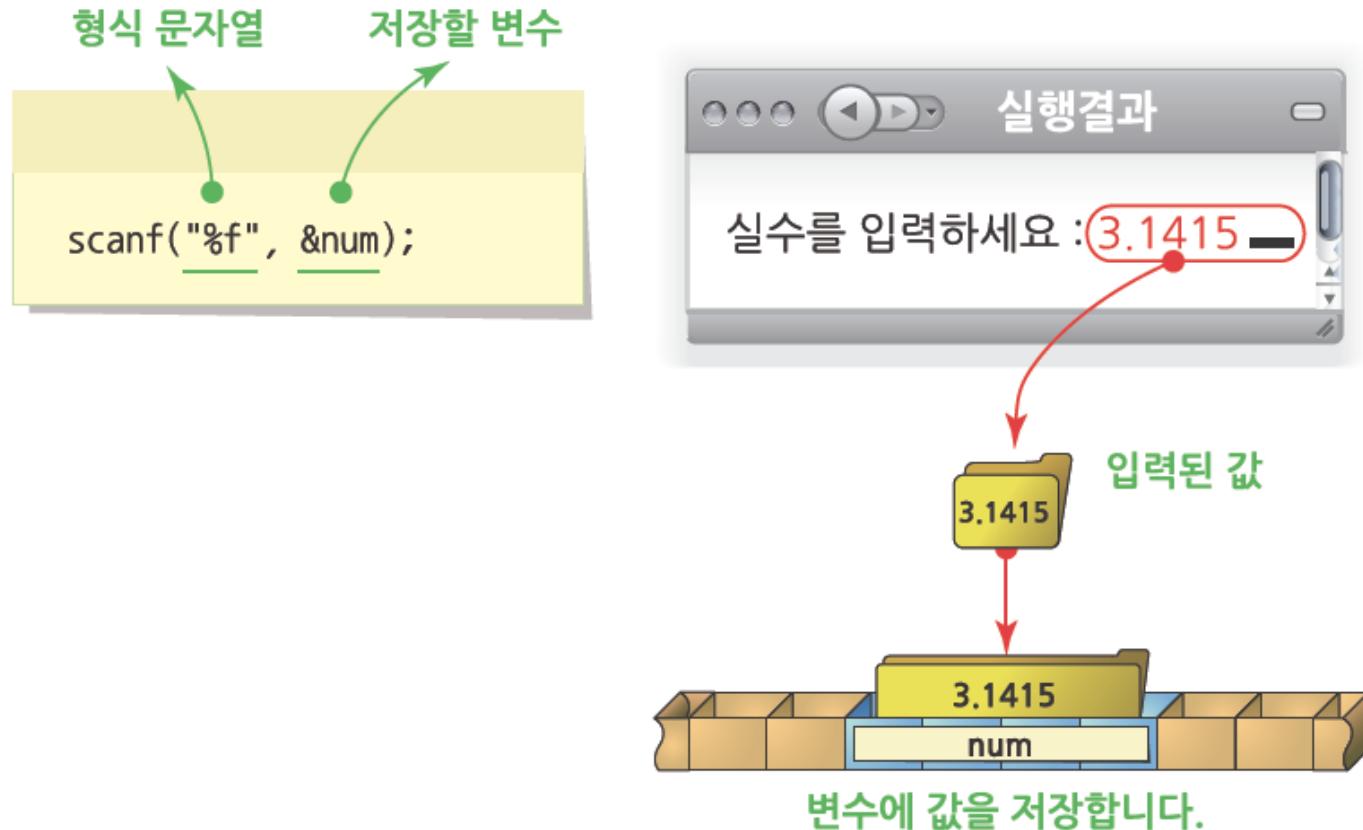
```
scanf(형식문자열, &변수명, … );
```

사용예

```
scanf("%d", &num);  
scanf("%d %f %f", &age, &height, &weight);
```

scanf 함수를 이용한 입력

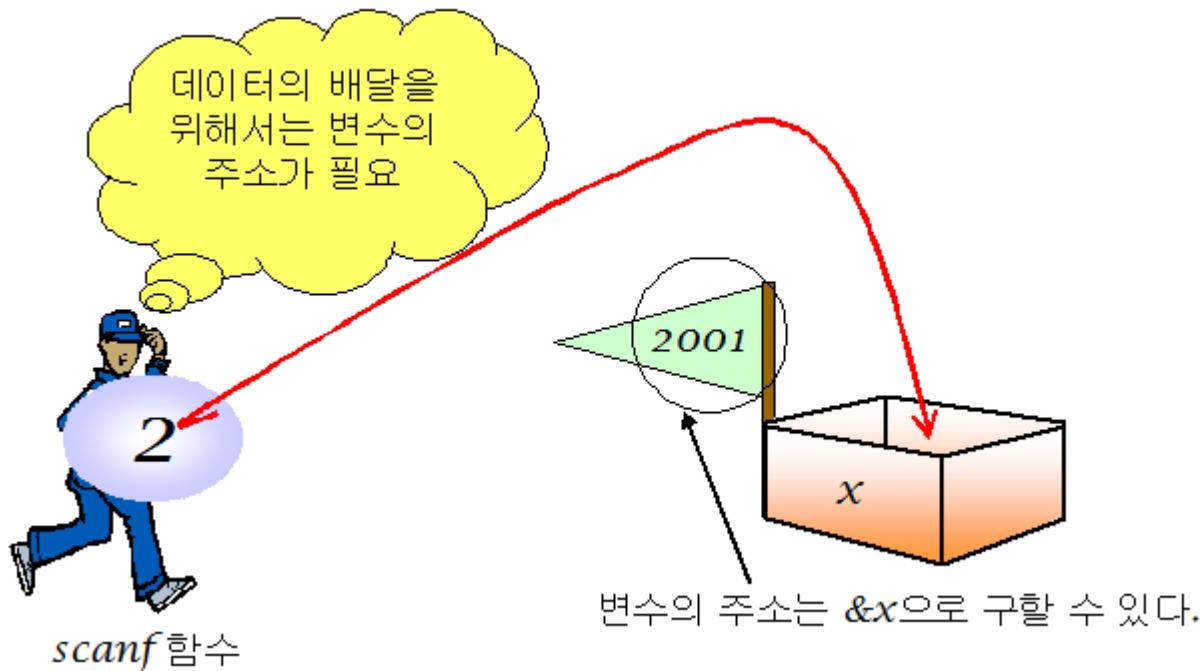
❖ scanf 함수의 사용 예



scanf 함수를 이용한 입력

❖ &의 의미

- & 연산자: 변수의 주소를 계산하는 연산자
- 변수에 값을 저장하려면 변수의 주소가 필요



scanf 함수를 이용한 입력

❖ scanf 함수의 형식 문자열(1/3)

서식지정자	의미
%d	정수를 10진수로 입력
%x	정수를 16진수로 입력
%i	정수를 10진수, 8진수, 16진수로 입력
%f	float형 실수 입력
%c	문자 입력

사용 예
<code>int num; scanf("%d", &num);</code>
<code>int num; scanf("%x", &num);</code>
<code>int num; scanf("%i", &num);</code>
<code>float x; scanf("%f", &x);</code>
<code>char gender; scanf("%c", &gender);</code>

012는 8진수 입력,
0x12는 16진수 입력

scanf 함수의 안전성 문제

- ❖ Visual Studio 2022에서 scanf 함수 사용 시 컴파일 에러 발생
 - C11에서는 scanf_s를 대신 사용하도록 권고
- ❖ scanf 함수의 의도적인 사용
 - ANSI C를 기준으로 C 프로그램을 작성하는 경우, 컴파일 에러가 발생하지 않도록 처리 필요

```
#define _CRT_SECURE_NO_WARNINGS  
#include <stdio.h>
```

안전성 관련 컴파일 경고/에러 메시지를 내지
않게 만드는 매크로를 정의한다.

또는

```
#pragma warning(disable:4996)  
#include <stdio.h>
```

4996번 컴파일 경고/에러 메시지를 내지
않도록 컴파일러에게 지시한다.

scanf 함수를 이용한 입력

❖ 입력받은 10진수 정수를 16진수로 변환해서 출력하기

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int num;           // 정수형 변수 선언
06
07     printf("정수? "); // 정수를 입력을 하도록 사용자에게 알려주기 위한 출력문
08     scanf("%d", &num); // num에 10진수로 정수 입력
09
10    printf("10진수 %d는 16진수 %x입니다.\n", num, num);
11
12    return 0;
13 }
```

10진수 정수 출력 16진수 정수 출력

실행결과

정수? 10 10진수 정수 입력
10진수 10는 16진수 a입니다.

scanf 함수를 이용한 입력

❖ scanf 함수의 형식 문자열(2/3)

- 문자 배열에 입력받을 때는 &를 지정하지 않는다.

```
char name[20];      // 문자 20개를 연속으로 저장하는 변수  
scanf("%s", name); // name에는 &가 필요 없다.
```

- 서식 지정자를 여러 개 사용할 수도 있다.

- 서식 지정자와 입력받을 변수는 순서대로 대응한다.
- 서식 지정자와 변수의 개수가 일치해야 한다.

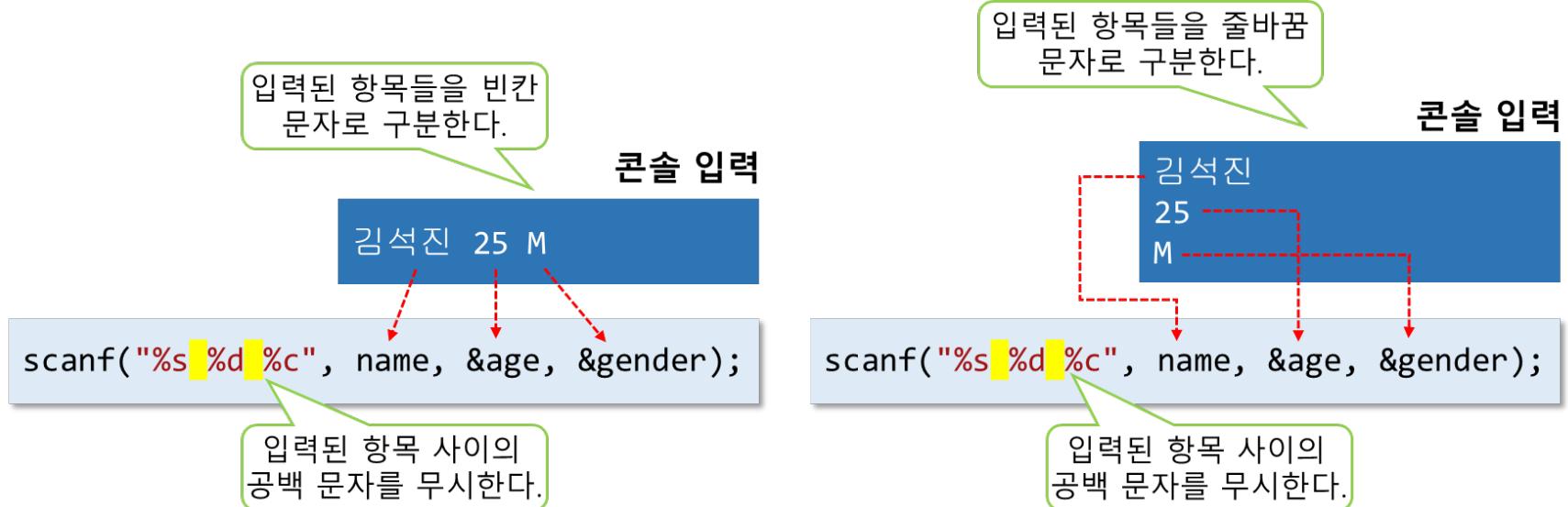
```
scanf("%: 3개", name, &a 3개 gender);
```

scanf 함수를 이용한 입력

❖ scanf 함수의 형식 문자열(3/3)

▪ 형식 문자열의 공백 문자

- 서식 지정자 사이에 빈칸을 써주면, 이전 입력 이후의 **공백 문자(빈칸, 탭, 줄바꿈 등)**를 모두 무시하고 다음 입력을 읽어온다.
- 이전 입력 이후의 공백을 모두 무시하고 다음 입력을 읽어 오게 한다.



scanf 함수를 이용한 입력

❖ 여러 개의 형식 문자열 사용하기(1/3)

```
01 #define _CRT_SECURE_NO_WARNINGS  
02 #include <stdio.h>  
03  
04 int main()  
05 {  
06     int age;  
07     float height;  
08     float weight;
```

Visual Studio 2022에서
scanf 사용 시 필요

변수의 선언

scanf 함수를 이용한 입력

❖ 여러 개의 형식 문자열 사용하기(2/3)

```
10   printf("나이, 키, 몸무게를 입력하세요: ");
11   scanf("%d %f %f", &age, &height, &weight);
12
13   printf("나이 : %5d\n", age);
14   printf("키 : %5.1f\n", height);
15   printf("몸무게: %5.1f\n", weight);
16 }
```

소수점 이하
1자리 출력

3개의 변수에
순서대로 입력

쪽을 5칸에
맞춰 출력

실행 결과

```
나이, 키, 몸무게를 입력하세요: 21 167.2 53.5
나이 : 21
키 : 167.2
몸무게: 53.5
```

세 가지 항목을
빈칸으로 구분해서 입력

scanf 함수를 이용한 입력

❖ 여러 개의 형식 문자열 사용하기(3/3)

```
03 int main()
04 {
05     char name[20];
06     int age;
07     char gender;
08
09     printf("이름, 나이, 성별(M/F) 순으로 입력하세요.\n");
10     scanf("%s %d %c", name, &age, &gender);    // 3개의 변수 입력
11     printf("이름: %s\n", name);
12     printf("나이: %d\n", age);
13     printf("성별: %c\n", gender);
14
15     return 0;
16 }
```

실행결과

이름, 나이, 성별(M/F) 순으로 입력하세요.

김석진 25 M

세 가지 항목을 빈칸으로 구분해서 입력

이름: 김석진

나이: 25

성별: M

name은 문자 배열이므로
&가 필요 없다.

오류 처리 및 디버깅_오류의 종류

❖ 컴파일이나 실행 시에 오류가 발생할 수 있다.

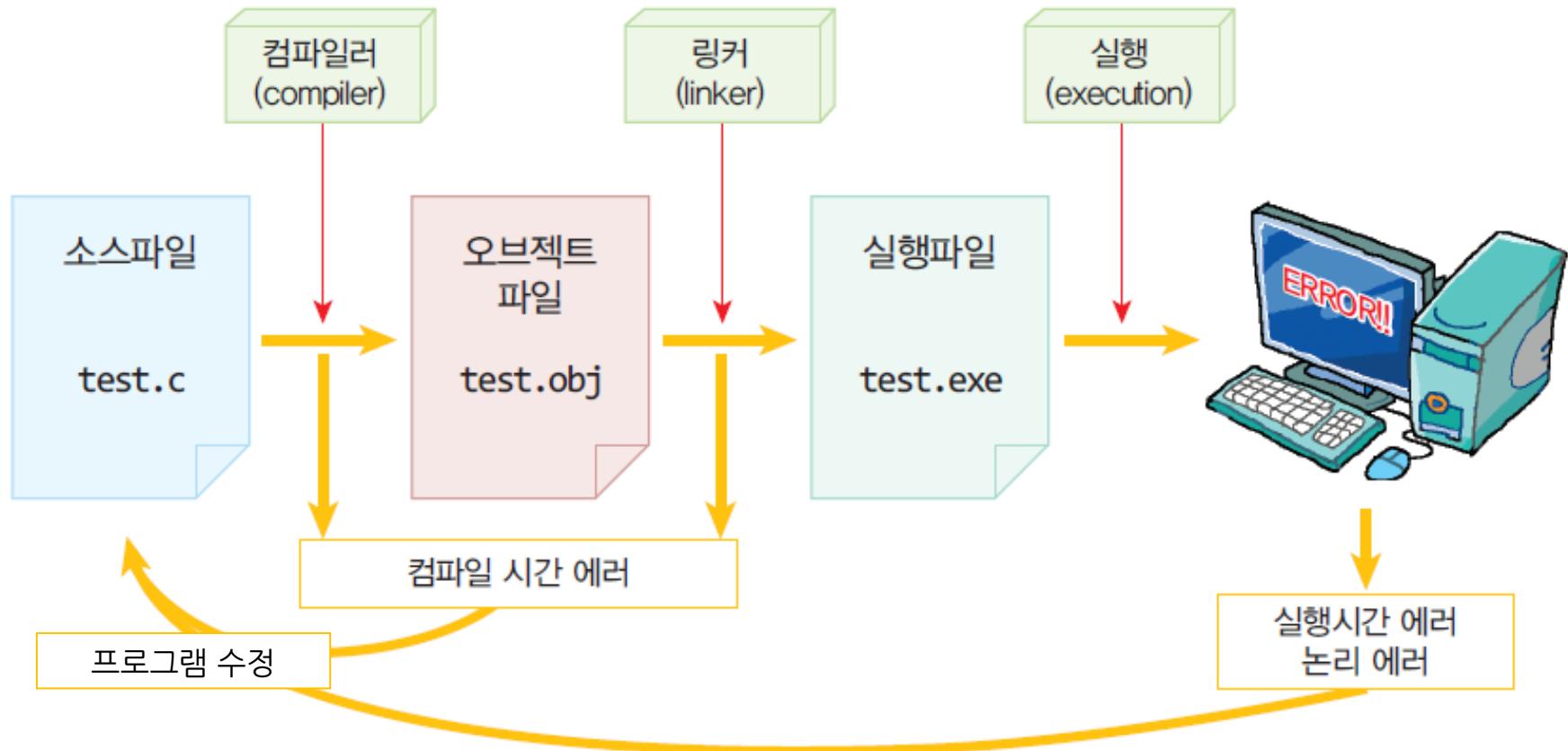
❖ 에러와 경고

- **에러(error)**: 심각한 오류
- **경고(warning)**: 경미한 오류(컴파일과 링크는 가능하지만 잠재적 문제 보유)
 - 나중에 원인을 알 수 없는 어려운 오류로 이어질 수 있음

❖ 오류의 종류

- **문법 오류**: 컴파일 과정에서 발생, 컴파일 시간 오류라고도 함
- **실행 시간 오류**: 실행 중에 잘못된 메모리 주소에 접근하거나, 0으로 나누는 연산 같은 오류
- **논리 오류** : 실행 과정에서 의도한 것과 다른 실행결과를 보여주는 오류
 - 복잡한 프로그램을 작성하다 보면 의도하지 않게 논리 오류가 발생할 수 있음

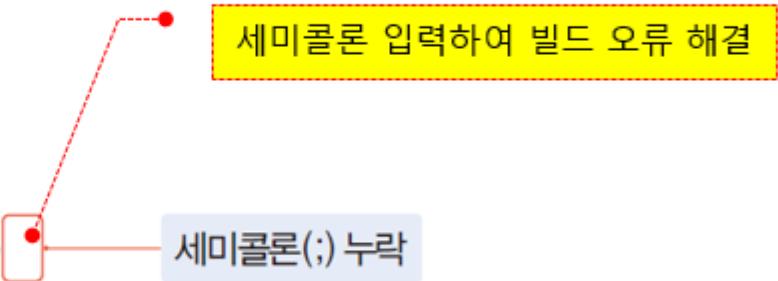
오류 처리 및 디버깅_오류 수정 과정



오류 처리 및 디버깅_문법 오류

❖ 세미콜론 누락

```
01 /* 문법 오류 첫 번째 프로그램 */
02 #include <stdio.h>
03
04 int main(void)
05 {
06     printf("문법 오류 : 첫 번째 테스트\n")
07     return 0;
08 }
```



```
출력
01 출력 보기 선택(S): 빌드
02 1>----- 빌드 시작: 프로젝트: ex02_02, 구성: Debug Win32 -----
03 1>ex02_02.c
04 1>D:\WORK\workspace\Chap_02\ex02_02\ex02_02\ex02_02.c(7,3): error C2143: 구문 오류: ';'이(가) 'return' 앞에 없습니다.
05 1>"ex02_02.vcxproj" 프로젝트를 빌드했습니다. - 실패
06 ----- 빌드: 성공 0, 실패 1, 최신 0, 생략 0 -----
```

오류 발견 소스 파일

오류 발견 줄 번호

return 앞에 ;이 누락되었음을 알려주는 메시지

오류 처리 및 디버깅_문법 오류

❖ 형식에 맞지 않는 주석문 사용

```
01 /* 문법 오류 두 번째 프로그램 /* 주석문 형식 오류
02 #include <stdio.h>
03
04 int main(void)
05 {
06     printf("문법 오류 : 두 번째 테스트\n");
07     return 0;
08 }
```

*/으로 수정하면 문법오류 해결

The screenshot shows a command-line interface window titled 'Output' (출력). The status bar at the bottom says '빌드: 성공 0, 실패 1, 최신 0, 생략 0'. The error message is: '1>----- 빌드 시작: 프로젝트: ex02_03, 구성: Debug Win32 -----
1>ex02_03.c
1>D:\WORK\workspace\Chap_02\ex02_03\ex02_03\ex02_03.c(9,1): error C1071: 주석에서 예기치 않은 파일의 끝이 나타났습니다.
1>"ex02_03.vcxproj" 프로젝트를 빌드했습니다. - 실패
===== 빌드: 성공 0, 실패 1, 최신 0, 생략 0 ====='

오류 발견 소스 파일

오류 발견 줄 번호

주석문에서 오류가 발생했음을 알려주는 메시지

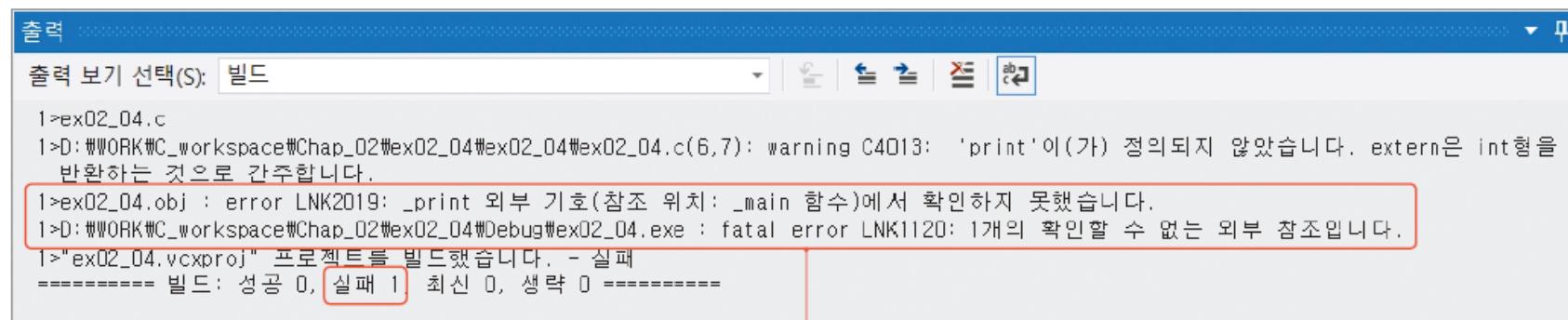
오류 처리 및 디버깅_문법 오류

❖ 틀린 소스코드 입력

```
01 /* 문법 오류 세 번째 프로그램 */
02 #include <stdio.h>
03
04 int main(void)
05 {
06     printf("문법 오류 : 세 번째 테스트\n");
07     return 0;
08 }
```

printf로 수정하면 문법오류 해결

printf()를 사용해야 함



- 링크 과정에서 오류 발생

오류 처리 및 디버깅_논리 오류

❖ 개행문자 누락-프로그램은 정상적으로 실행됨

```
01 /* 논리 오류 프로그램 */
02 #include <stdio.h>
03
04 int main(void)
05 {
06     printf("논리 오류 발생");
07     printf("소스 코드 확인");
08     return 0;
09 }
```

강제 개행 `\n` 추가하여 논리오류 해결

`\n`을 선언해야 함

논리 오류 발생소스 코드 확인

프로그래머가 의도한 실행 결과

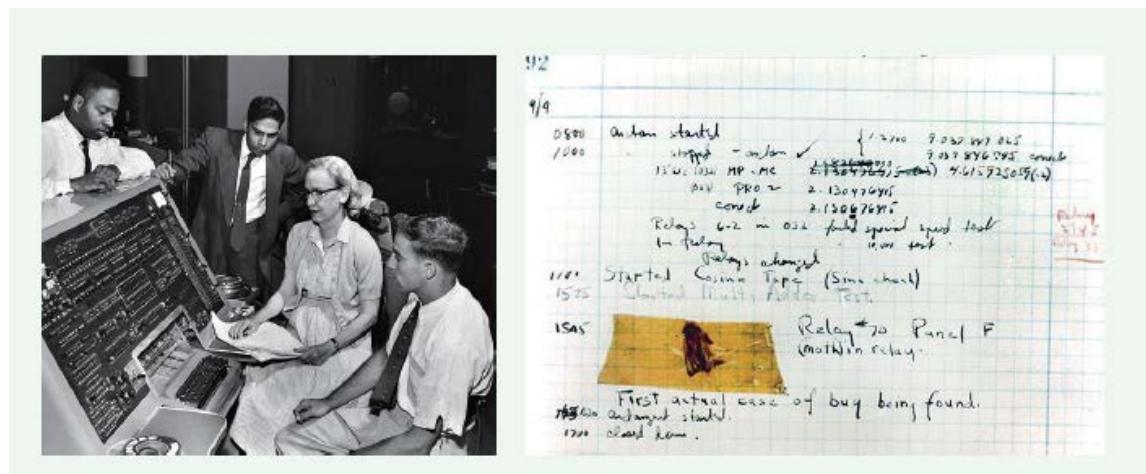
논리 오류 발생
소스 코드 확인

오류 처리 및 디버깅_디버깅

❖ 디버깅 : 논리 오류를 찾는 과정

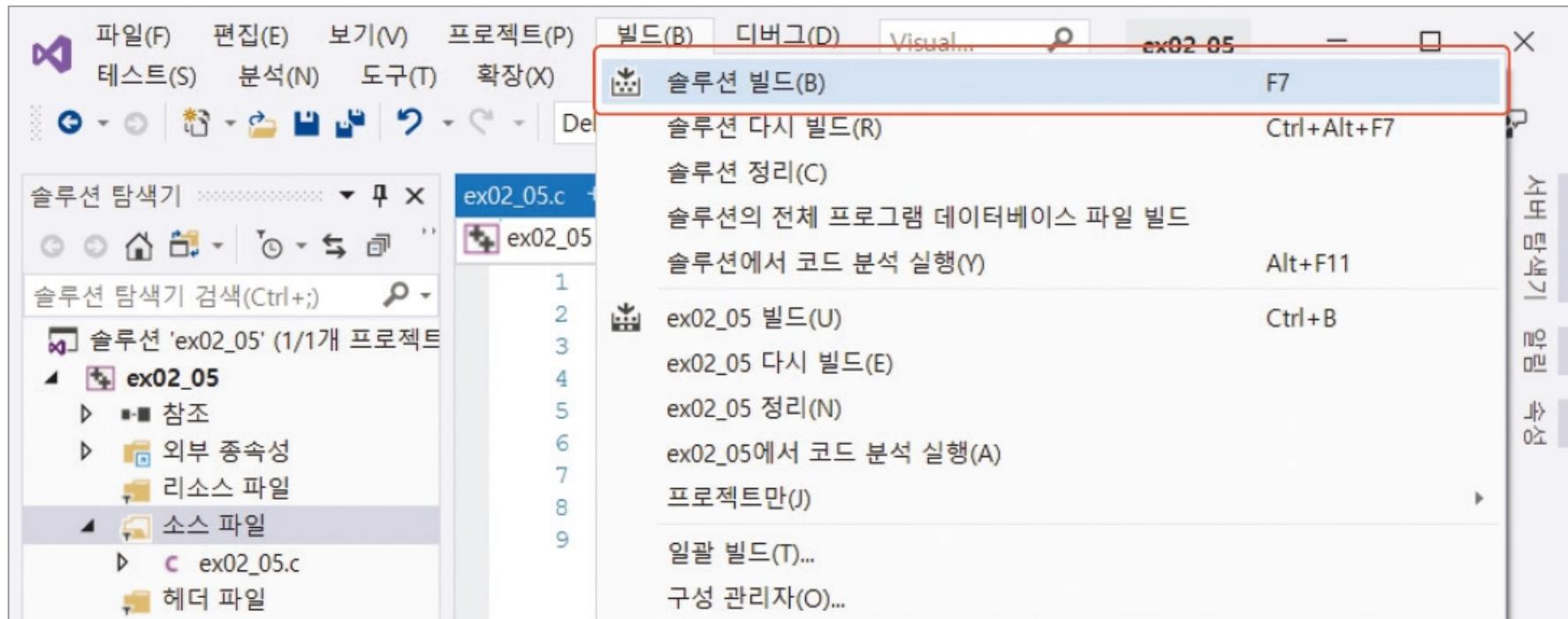
❖ 디버깅의 유래

- 1945년 마크 II 컴퓨터가 릴레이 장치에 날아든 나방 때문에 고장을 일으켰고 이것을 “컴퓨터 버그(bug: 벌레)”
- 라고 불렀다. 여성 컴퓨터 과학자인 그레이스 호퍼가 나방을 채집해 기록에 남기고 이를 “디버깅(debugging)”작업이라고 보고하였다



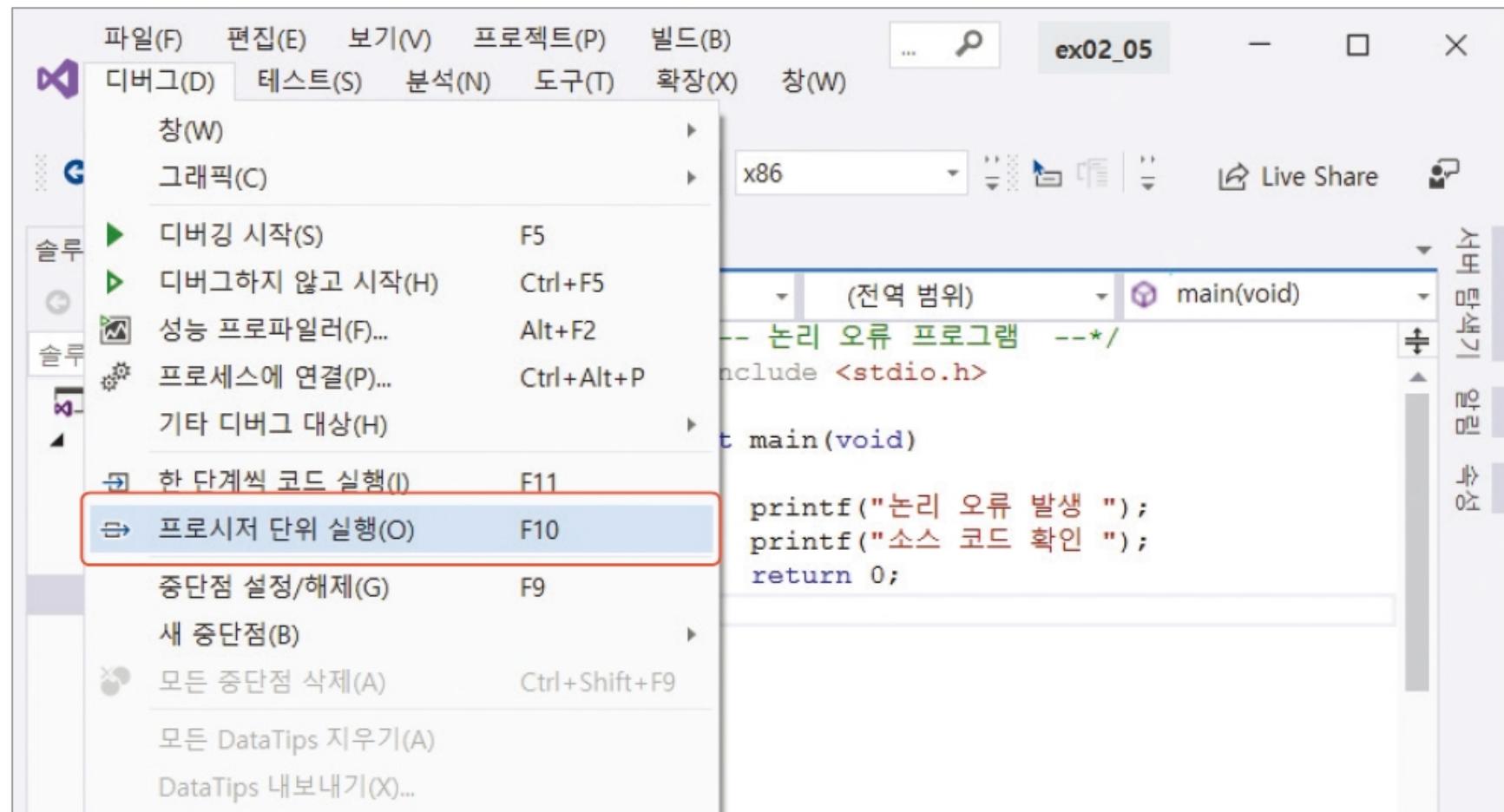
오류 처리 및 디버깅_디버깅

1. 실행단계를 먼저 수행 : [빌드]-[솔루션 빌드]



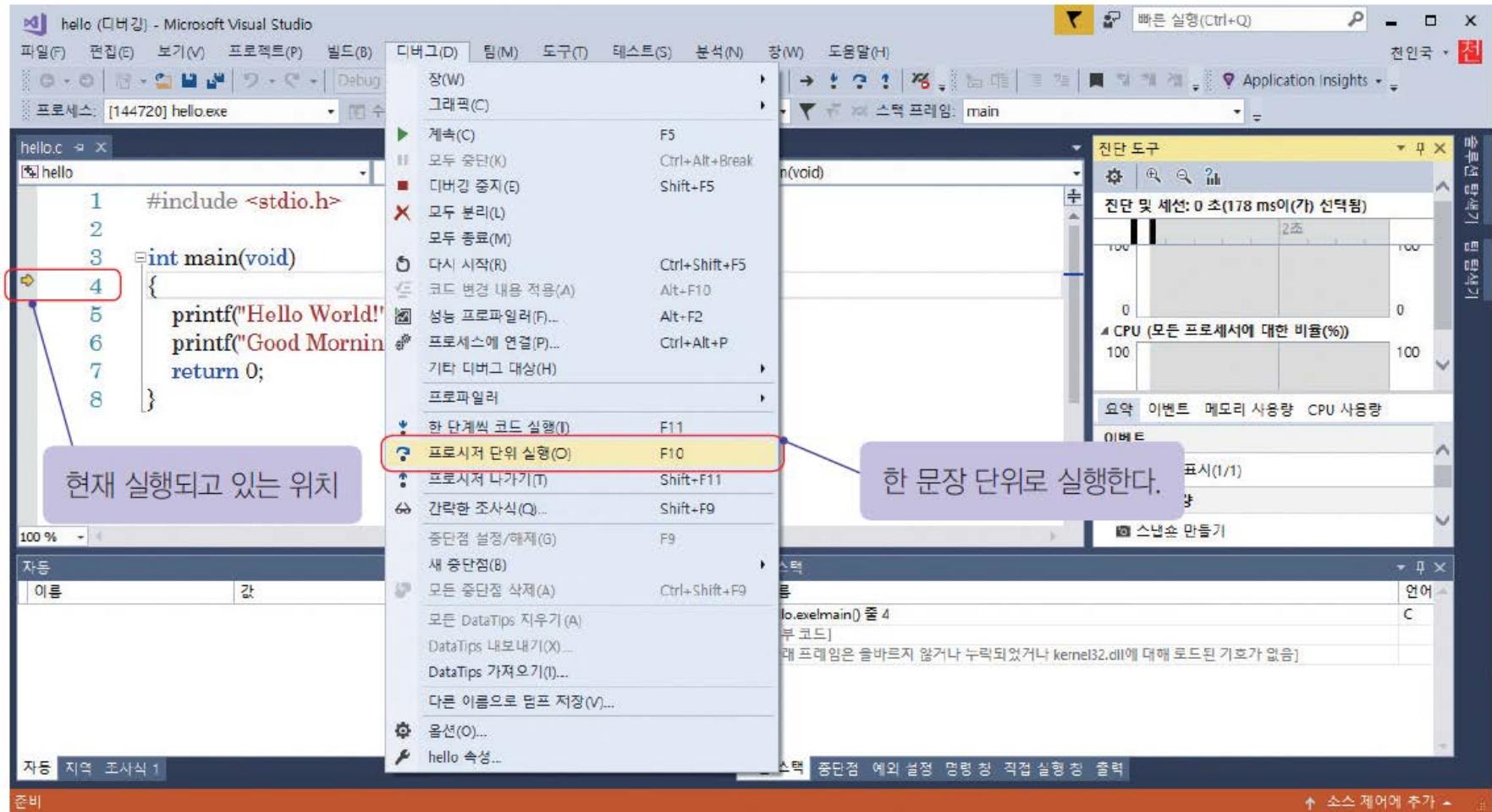
오류 처리 및 디버깅_디버깅

2. 한 문장씩 디버깅 : [디버그]-[프로시저 단위 실행]



오류 처리 및 디버깅_디버깅

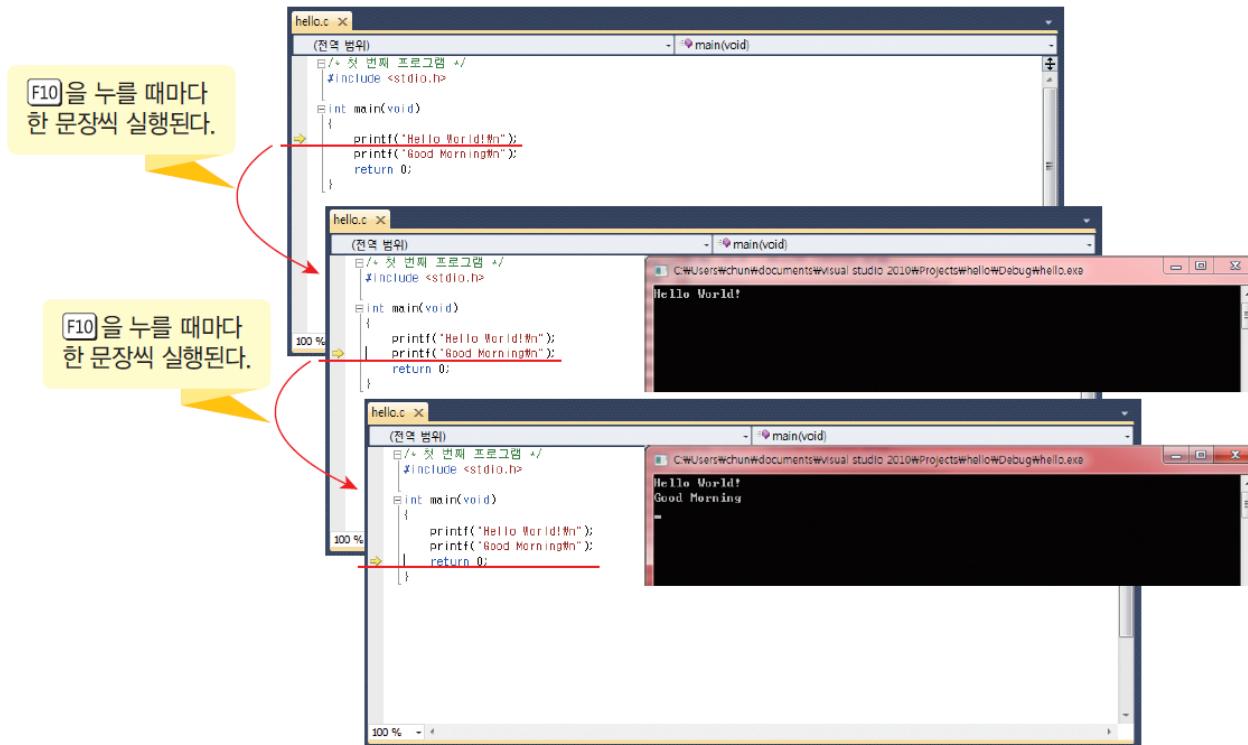
- 중단점 설정(F9) 후 한 문장씩 디버깅



오류 처리 및 디버깅_디버깅

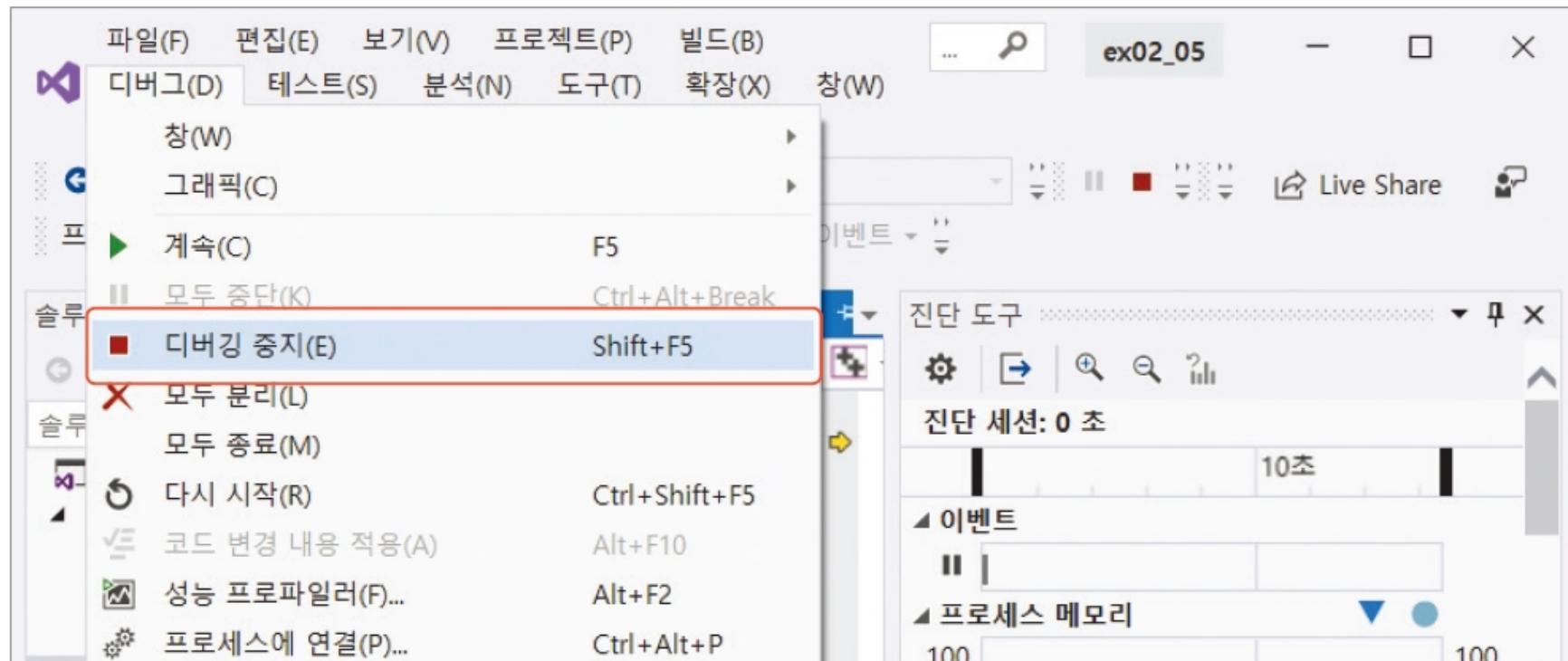
■ 디버거의 명령어 정의

- F5 (Go): 실행
- F10 (Step Over): 한 문장씩 실행(함수도 하나의 문장 취급)
- F11 (Step Into): 한 문장씩 실행(함수 안으로 진입)
- F9 (Breakpoint): 현재 문장에 중단점을 설정



오류 처리 및 디버깅_디버깅

3. 디버깅 작업 중지 : [디버그]-[디버깅 중지]



학습정리

❖ 첫 번째 C 프로그램

- **주석 : 프로그램에 대한 설명으로 실제로 수행되지 않는 부분**
 - 주석은 /*로 시작해서 */로 끝난다.
 - //를 이용해서 한 줄 주석을 만들 수 있다.
- **main 함수 : C 프로그램의 진입점 함수**
 - 프로그램을 시작할 때 운영체제가 호출해주는 함수
 - 함수가 처리할 내용은 {} 안에 써준다.
- **문장 : 함수를 구성하는 기본 단위**
 - 문장의 끝에는 세미콜론(;)을 써준다.
- **들여쓰기**
 - 한 줄에 한 문장을 작성하고, 각 문장은 블록 단위로 들여 쓰는 것이 좋다.
- **출력 : printf 함수를 이용한다.**
 - <stdio.h>라는 헤더 파일을 포함한다.
 - ()안에 출력할 문자열을 지정한다.

학습정리

❖ 두 번째 C 프로그램

- **변수** : 프로그램에서 사용되는 값을 저장
 - 변수를 선언하려면 변수의 데이터 형과 이름이 필요하다.
- **변수의 데이터 형** : char, int, float 등이 있다.
- **변수의 사용** : 메모리에 저장된 변수의 값을 읽어오거나 값을 저장하려면 변수의 이름을 사용한다.
 - 변수의 값을 변경하려면 =를 이용한다.
- **scanf 함수** : 변수의 값을 입력 받을 때는 scanf 함수를 이용한다.
 - 형식 문자열을 이용해서 입력 받을 값의 형식을 알려준다.
 - %d는 정수 입력, %f는 실수 입력
 - 입력된 값을 저장할 변수 이름 앞에 &를 써준다.
- **printf 함수** : printf 함수를 이용해서 변수의 값을 출력할 때도 형식 문자열을 이용한다.
 - %d는 정수 출력, %f는 실수 출력