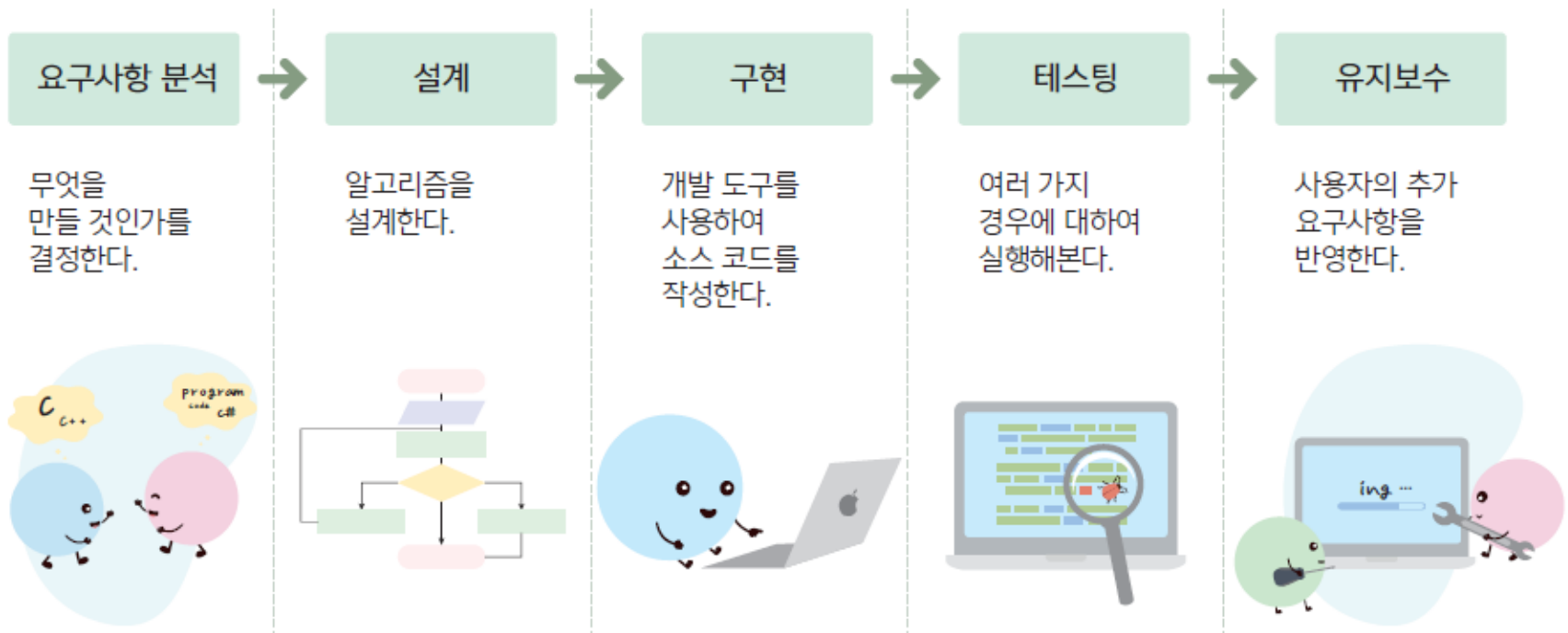


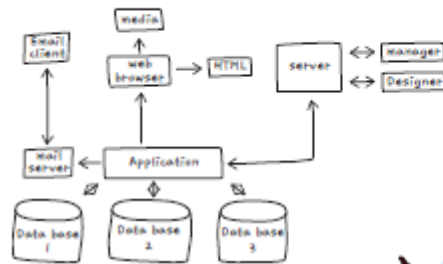
# 제 2장 프로그램 작성 과정

# 프로그램 개발 과정



# 설계

- 문제를 해결하는 알고리즘을 개발하는 단계
- 순서도와 의사 코드를 도구로 사용
- 알고리즘은 프로그래밍 언어와는 무관
- 알고리즘은 원하는 결과를 얻기 위하여 밟아야 하는 단계에 집  
중적으로 초점을 맞추는 것



흠, 이제야 좀 이해가 되는군.



# 소스 작성

- 알고리즘의 각 단계를 프로그래밍 언어를 이용하여 기술
- 알고리즘을 프로그래밍 언어의 문법에 맞추어 기술한 것을 *소스 프로그램(source program)*이라고 한다.
- 소스 프로그램은 주로 텍스트 에디터나 통합 개발 환경을 이용하여 작성한다.
- 소스 파일 이름: (예) test.c



에디터

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

# 컴파일

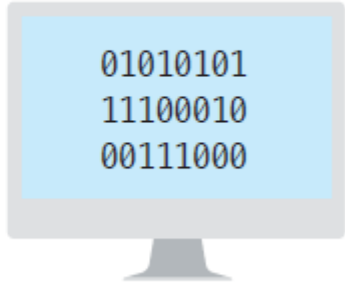
- 소스 프로그램을 오브젝트 파일로 변환하는 작업
- 오브젝트 파일 이름: (예) test.obj

이제야 좀 이해가 되는군

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



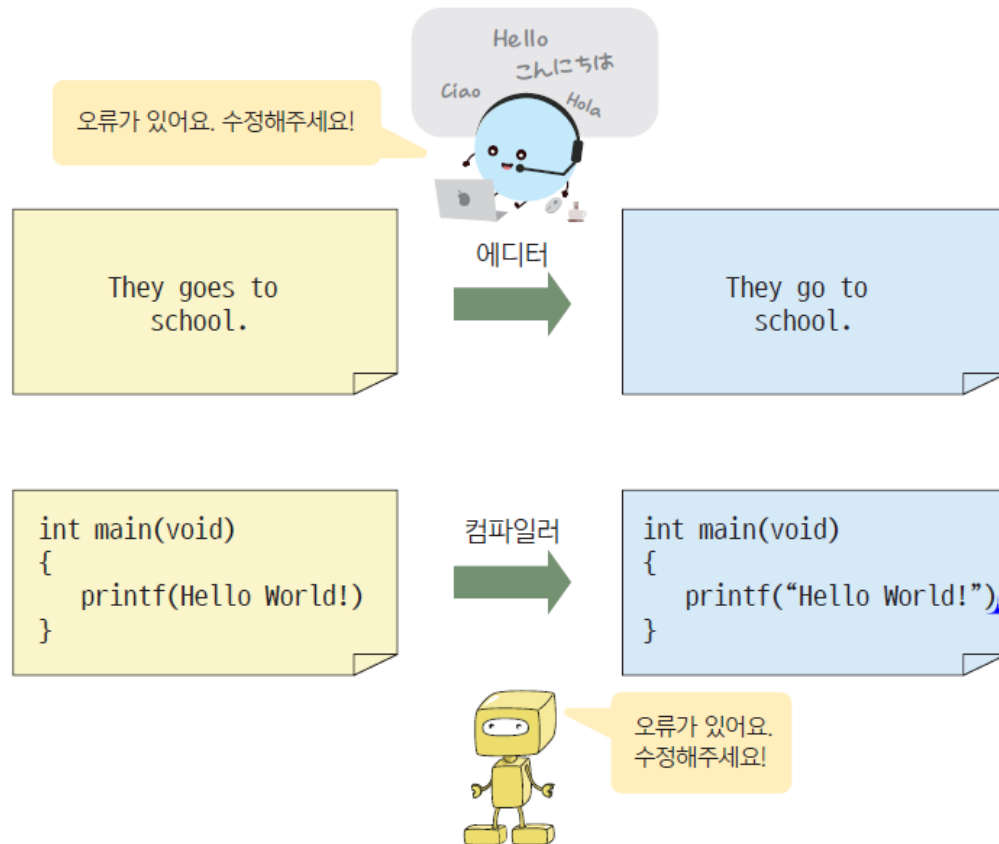
컴파일러



```
01010101
11100010
00111000
```

# 컴파일 오류

- 컴파일 오류(compile error): 문법 오류
  - (예) He go to school;



# 링크

- 컴파일된 목적 프로그램을 라이브러리와 연결하여 실행 프로그램을 작성하는 것
- 실행 파일 이름: (예) test.exe
- *라이브러리(library)*: 프로그래머들이 많이 사용되는 기능을 미리 작성해 놓은 것
  - (예) 입출력 기능, 파일 처리, 수학 함수 계산
- 링크를 수행하는 프로그램을 *링커(linker)*라고 한다.

# 오브젝트 파일

```
#include <stdio.h>

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

소스 파일

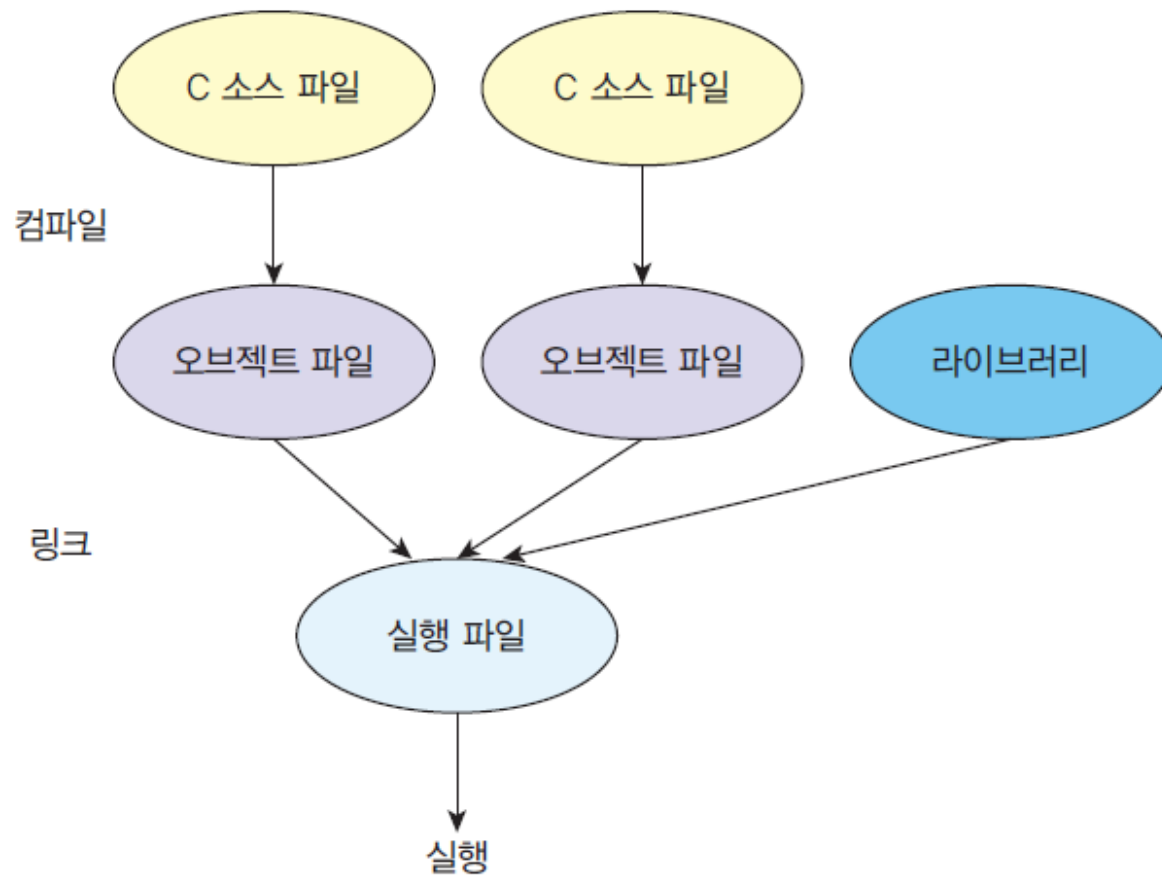


```
010101000001111101010
101010100000111110101
010101010000011111010
101010101000001111101
010101010100000111110
101001010100000111110
10101...
```

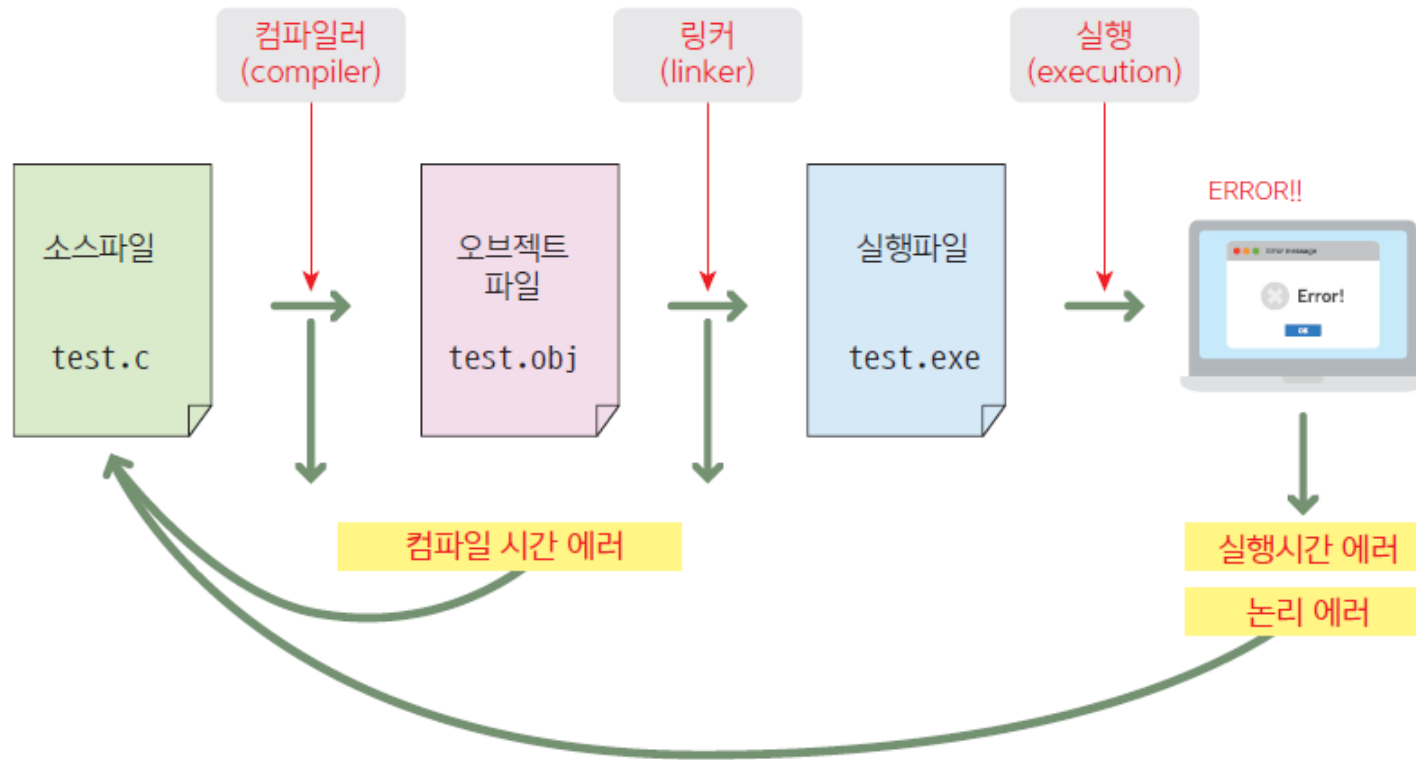
오브젝트 파일



# 링크



# 실행 및 디버깅



# 실행 및 디버깅

- 실행 시간 오류(run time error):
  - 0으로 나누는 것
  - 잘못된 메모리 주소에 접근하는 것
- 논리 오류(logical error): 문법은 틀리지 않았으나 논리적으로 정확하지 않는 것
  - (예)

- ① 그릇1과 그릇2를 준비한다.
- ② 그릇1에 밀가루, 우유, 계란을 넣고 잘 섞는다.
- ③ 그릇2를 오븐에 넣고 30분 동안 350도로 굽는다.

실수로 빈그릇을 오븐에 넣는다면  
논리적인 오류입니다.



# 디버깅

- 소스에 존재하는 오류를 잡는 것



# 디버깅의 유래

- 1945년 마크 II 컴퓨터가 릴레이 장치에 낱아든 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



9/2  
9/9

0800 Action started  
1000 stopped - action ✓  
1300 (1200) HP - MC 2.13047685  
1400 PRO - 2.13047685  
1500 correct 2.13047685  
Relays 6-2 in 0.24 fault speed speed test  
in relay 11.00 test

1100 Relays changed  
1500 Started Cassio Taps (Sine check)  
1505 Started Quality Audio Test

1545 Relay 70 Panel F  
(motion relay)

1550 First actual case of bug being found.  
1600 Action started.  
1700 closed down.

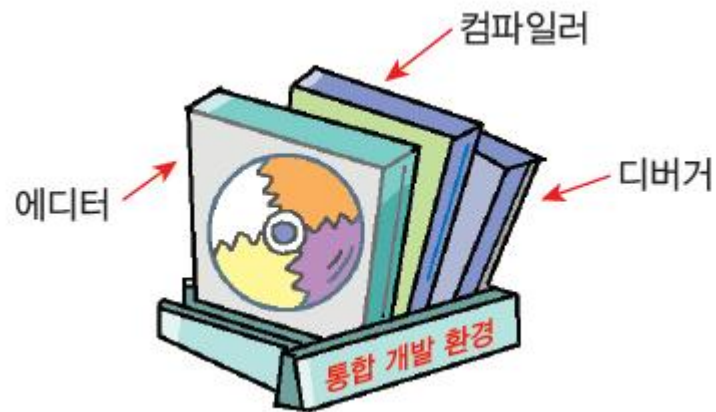
# 소프트웨어의 유지 보수

- 소프트웨어의 유지 보수가 필요한 이유
  1. 디버깅 후에도 버그가 남아 있을 수 있기 때문
  2. 소프트웨어가 개발된 다음에 사용자의 요구가 추가될 수 있기 때문
- 유지 보수 비용이 전체 비용의 50% 이상을 차지



# 통합 개발 환경

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



# 통합 개발 환경의 예

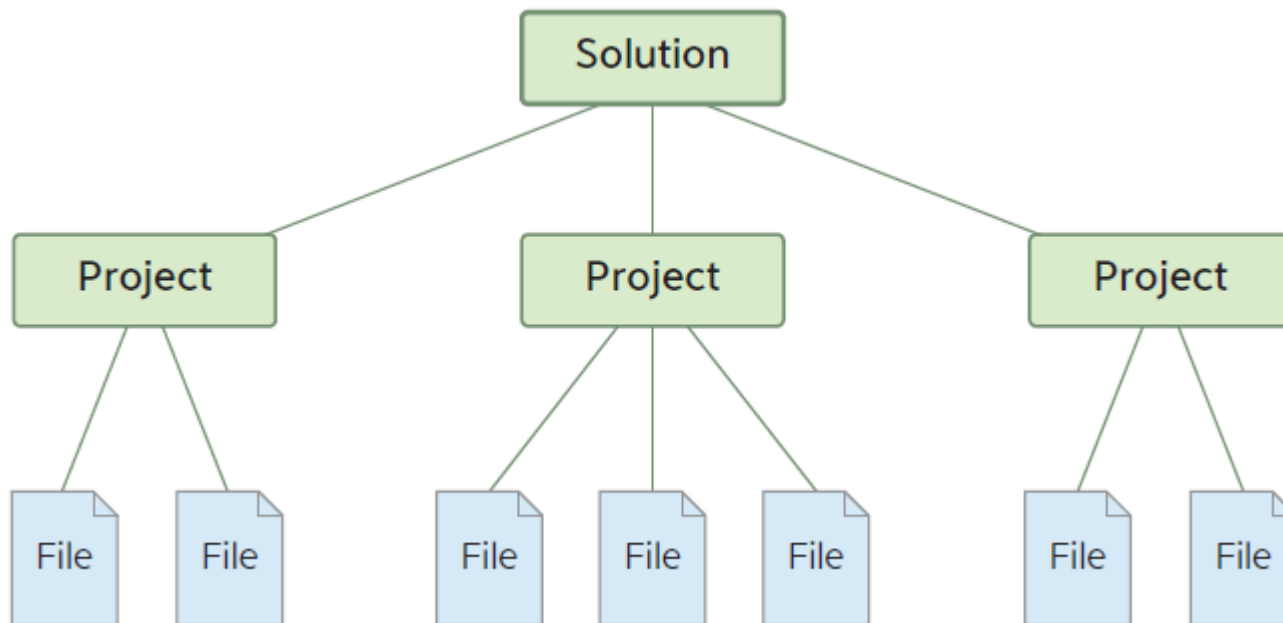
- 비주얼 스튜디오:                   마이크로소프트
- 이클립스(eclipse):               오픈 소스 프로젝트
- Dev-C++:                           오픈 소스 프로젝트





# 솔루션과 프로젝트

- **솔루션(solution)**; 문제 해결에 필요한 프로젝트가 들어 있는 컨테이너
- **프로젝트(project)**; 하나의 실행 파일을 만드는데 필요한 여러 가지 항목들이 들어 있는 컨테이너



# 프로그램 입력

include나 stdio는 붙여쓴다.

기호는 정확하게 입력

```
#include <stdio.h>

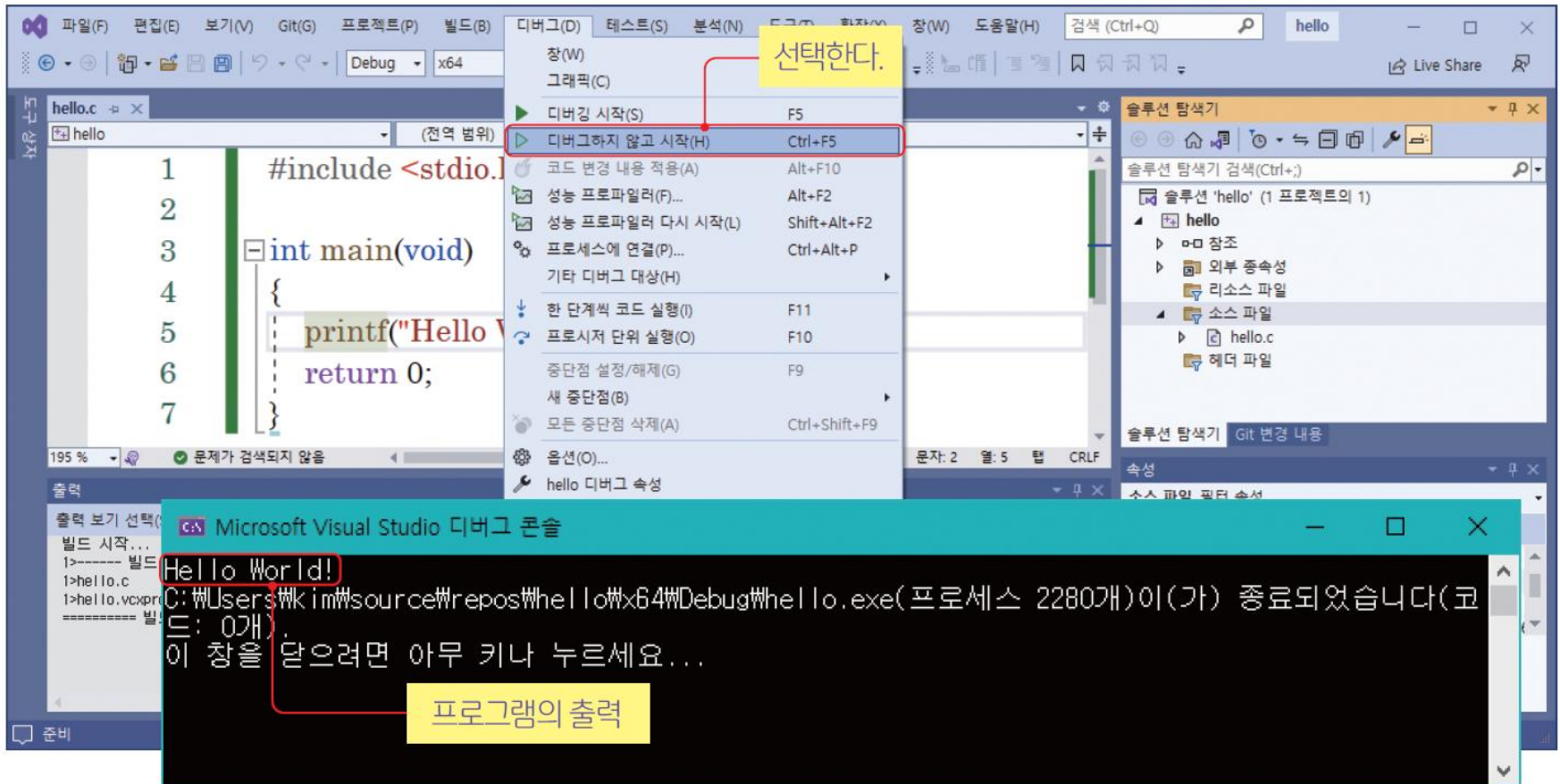
int main(void)
{
    printf( "Hello World! " );
    return 0;
}
```

들여쓴다.

문장의 끝에는 세미콜론

문장의 끝에는 세미콜론

# 프로그램 실행 하기



# 첫번째 프로그램의 설명

*hello.c*

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

```
int main(void)
```

```
{
```

```
    printf("Hello World!");
```

```
    return 0;
```

```
}
```



Hello World!

# 프로그램 == 작업 지시서




작업 지시서



프로그램

# 작업을 적어주는 위치

```
#include <stdio.h>

int main(void)
{
    
    return 0;
}
```

여기다가 원하는 작업을 수행하는 문장을 적어 준다.

프로그램

# 간략한 소스 설명



프로그램

# 헤더 파일 포함

- `#include`는 소스 코드 안에 특정 파일을 현재의 위치에 포함

- 주의!: 전처리기 지시자 문장 끝에는 세미콜론(;)을 붙이면 안 된다.

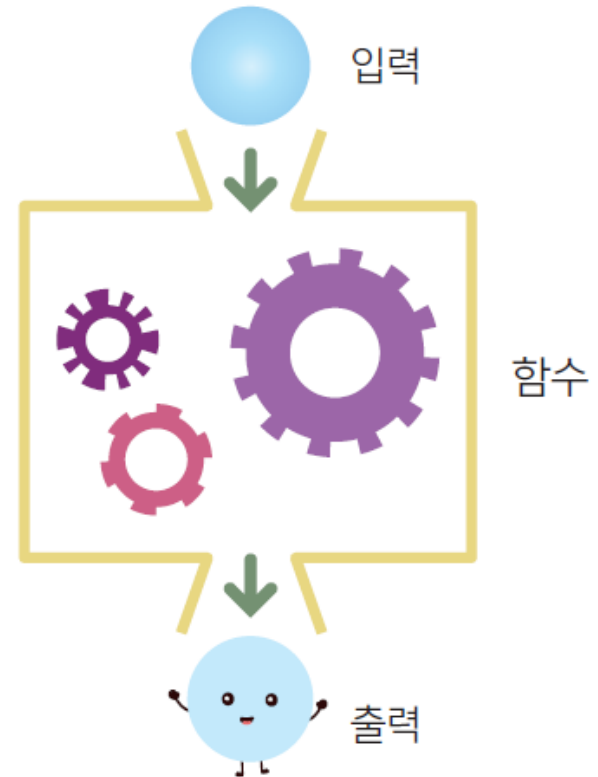
`#include <stdio.h>`

- 헤더 파일(header file): 컴파일러가 필요로 하는 정보를 가지고 있는 파일
- stdio.h: standard input output header file

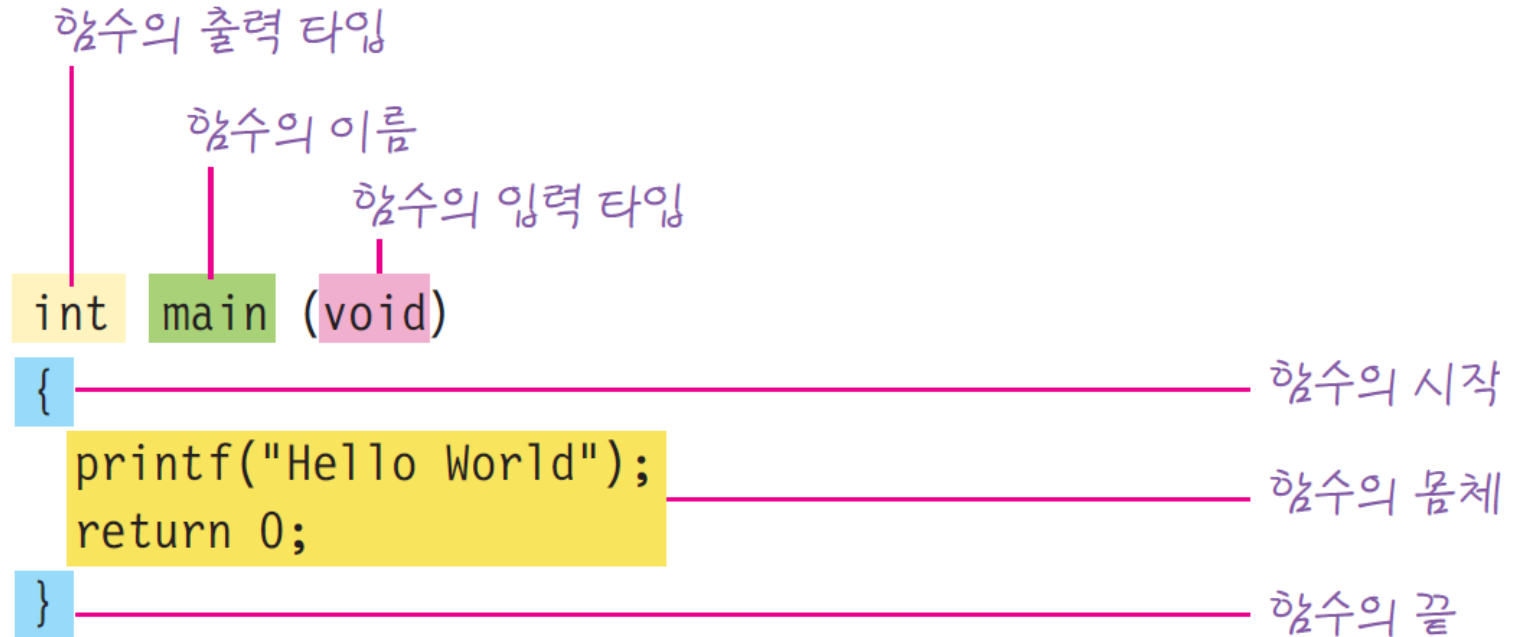


# 함수

- 함수(function): 특정한 작업을 수행하기 위하여 작성된 독립적인 코드
- (참고) 수학적 함수  $y = x^2 + 1$
- 프로그램 = 함수의 집합

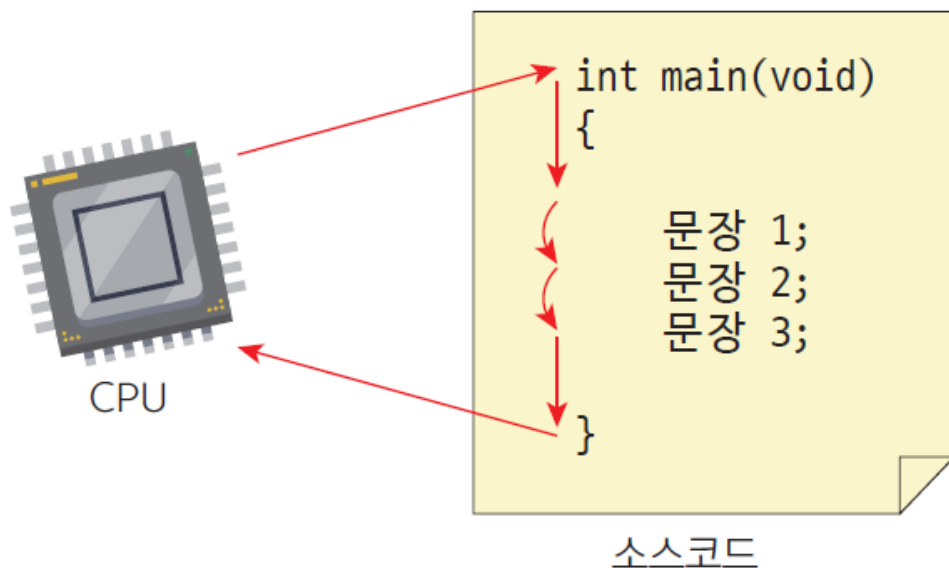


# 함수의 간략한 설명



# 문장(명령문)

- 함수는 여러 개의 문장으로 이루어진다.
- 문장들은 순차적으로 실행된다.
- 문장의 끝에는 반드시 ;이 있어야 한다.



소스 코드의 문장들은  
기본적으로 차례대로  
실행됩니다.



# printf() 호출

- printf()는 컴파일러가 제공하는 함수로서 출력을 담당한다

printf("Hello World!");



Hello World!

- 큰따옴표 안의 문자열이 화면에 출력된다.

# 함수의 반환값

- return은 함수의 결과값을 외부로 반환

return 0;

- 반환값은 0

# 응용 프로그램 #1

- 다음과 같은 출력을 가지는 프로그램을 제작하여 보자.



# 첫번째 버전

- 문장들은 순차적으로 실행된다는 사실 이용

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

```
int main(void)
```

```
{
```

```
    printf("Hello World!");
```

```
    printf("Kim ChulSoo");
```

```
    return 0;
```

```
}
```

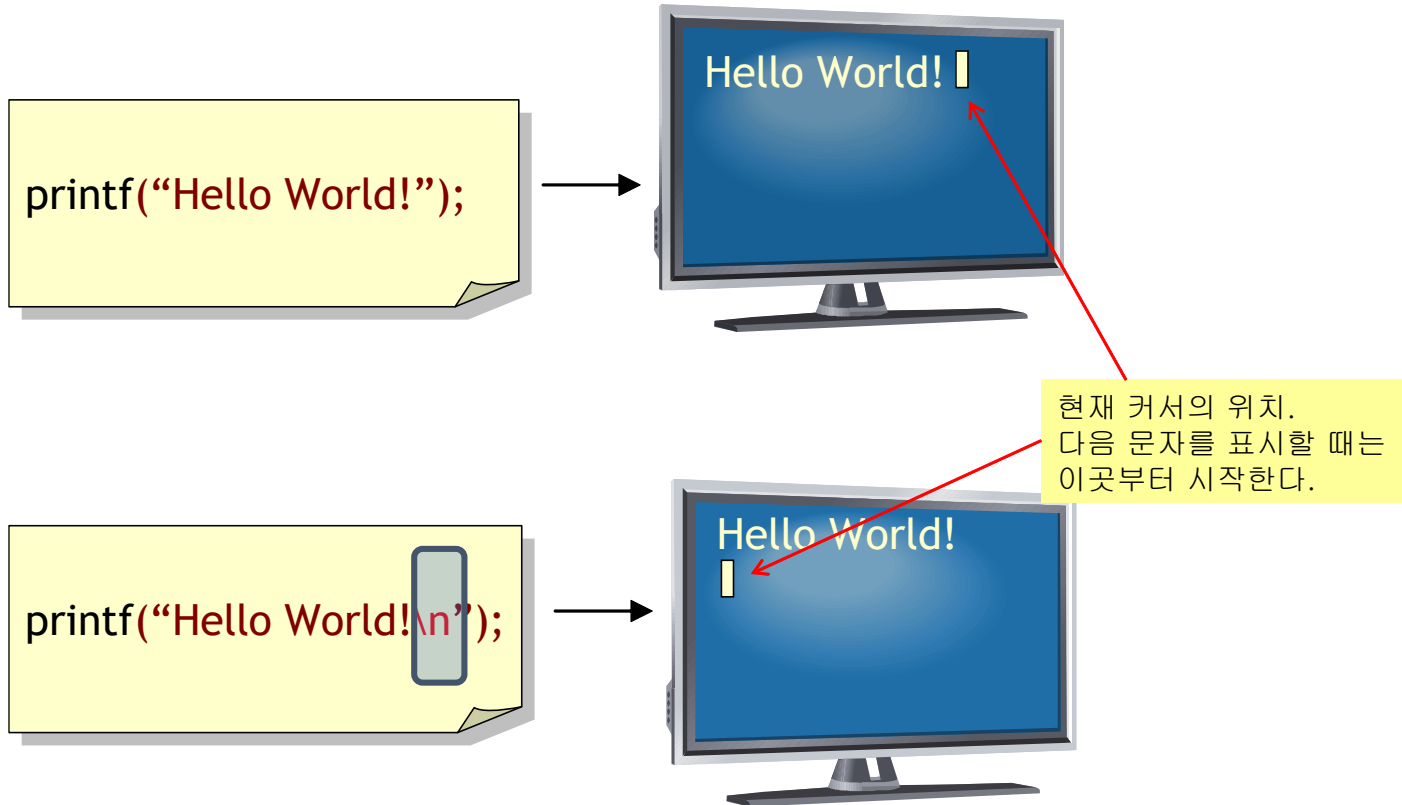
2개의 문장은 순차적으로  
실행된다



Hello World! Kim ChulSoo

# 줄바꿈 문자

- 줄바꿈 문자인 `\n`은 화면에서 커서는 다음줄로 이동하게 한다.





# 변경된 프로그램

- 줄바꿈 문자를 추가하면 우리가 원하던 결과가 된다.

```
#include <stdio.h>

int main(void)
{

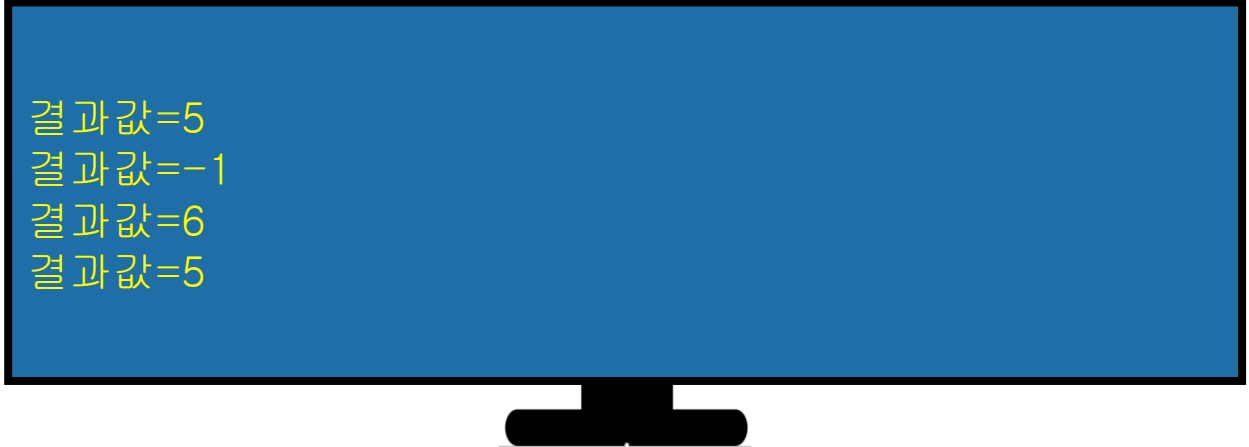
    printf("Hello World!\n");
    printf("Kim ChulSoo\n");

    return 0;
}
```



# Lab: 간단한 계산을 해보자

- 덧셈과 뺄셈, 곱셈, 나눗셈 계산을 하는 프로그램을 작성해보자.



```
결과값=5  
결과값=-1  
결과값=6  
결과값=5
```

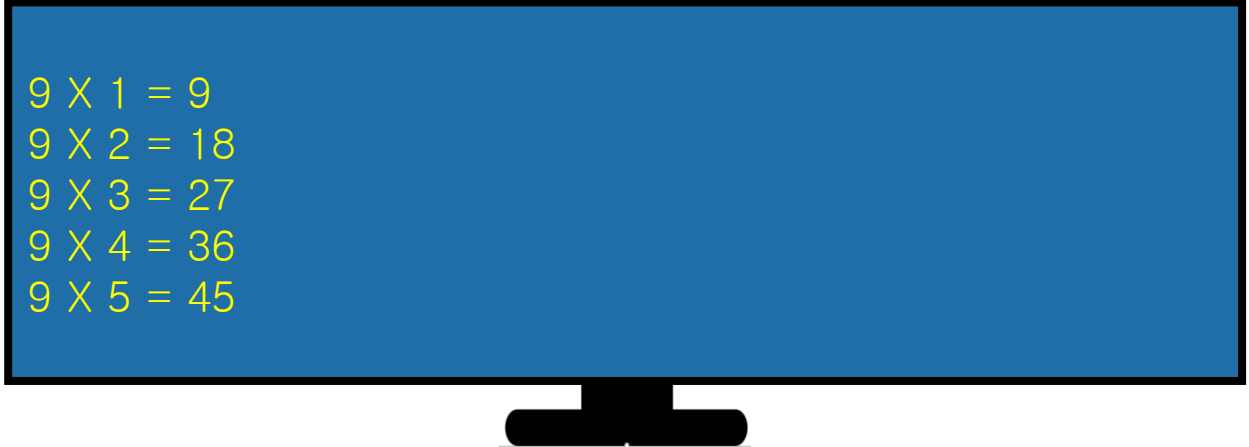
# Solution

```
#include <stdio.h>

int main(void)
{
    printf("결과값=%d\n", 2 + 3);
    printf("결과값=%d\n", 2 - 3);
    printf("결과값=%d\n", 2 * 3);
    printf("결과값=%d\n", 2 / 3);
    return 0;
}
```

# Lab: 구구단을 출력해보자.

- 구구단 중에서 9단의 일부를 출력하는 프로그램을 작성해보자.

A computer monitor with a black frame and a black stand. The screen is blue and displays five lines of yellow text, each representing a multiplication problem from the 9 times table.

9 X 1 = 9  
9 X 2 = 18  
9 X 3 = 27  
9 X 4 = 36  
9 X 5 = 45

# Solution

```
#include <stdio.h>

int main(void)
{
    printf("9 X 1 = %d\n", 9*1);
    printf("9 X 2 = %d\n", 9*2);
    printf("9 X 3 = %d\n", 9*3);
    printf("9 X 4 = %d\n", 9*4);
    printf("9 X 5 = %d\n", 9*5);
    return 0;
}
```



9단 전체를 출력하도록 코드를 수정해보자.

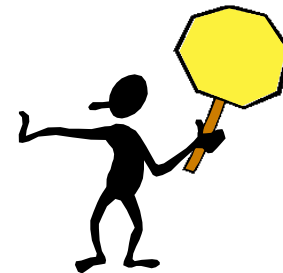
# 오류 수정 및 디버깅

- 컴파일이나 실행 시에 오류가 발생할 수 있다.
- 에러와 경고
  - 에러(error): 심각한 오류
  - 경고(warning): 경미한 오류

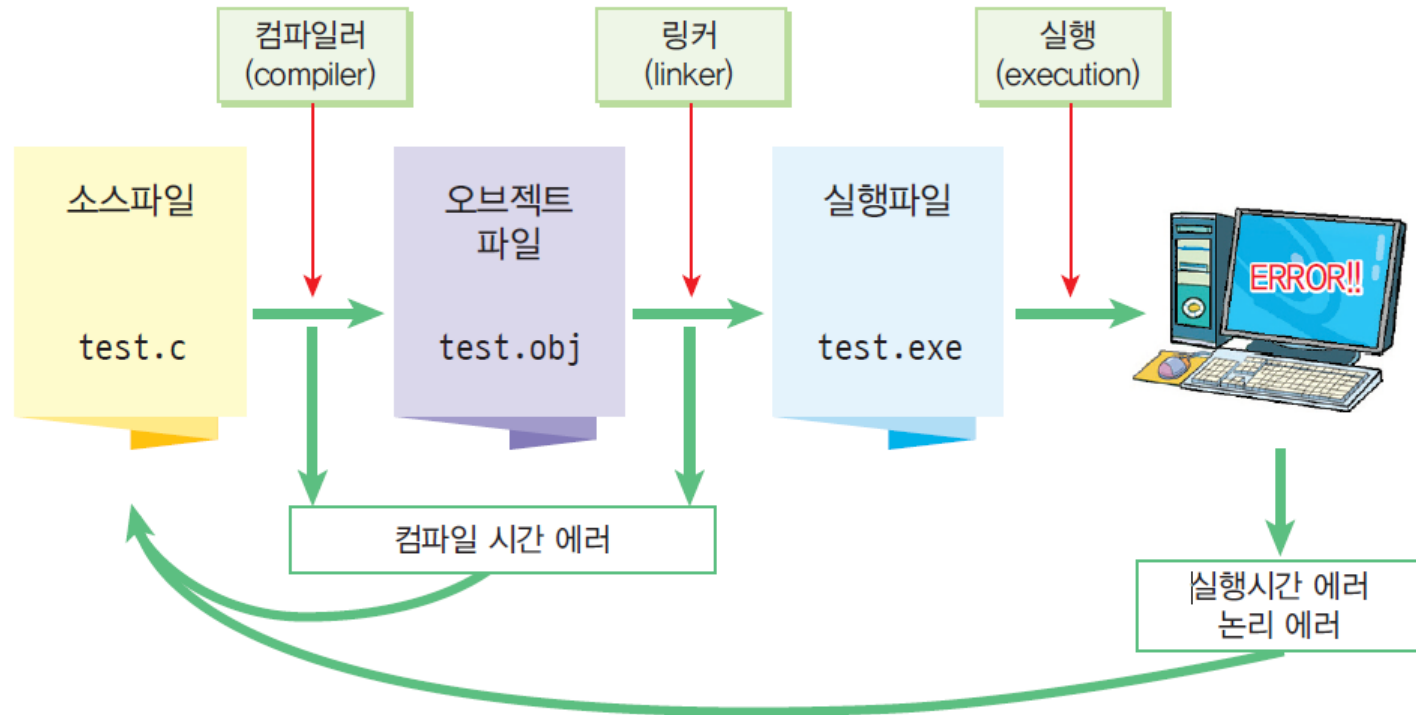


# 오류의 종류

- 오류의 종류
  - 컴파일 시간 오류: 대부분 문법적인 오류
  - 실행 시간 오류: 실행 중에 0으로 나누는 연산 같은 오류
  - 논리 오류: 논리적으로 잘못되어서 결과가 의도했던 대로 나오지 않는 오류



# 오류 수정 과정





# 오류 #1

add - Microsoft Visual Studio

빠른 실행(Ctrl+Q)

파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버그(D) 팀(M) 도구(T) 테스트(S) 분석(N) 창(W) 도움말(H) 로그인

Debug x86 로컬 Windows 디버거

error1.c\* x

add (전역 범위) main(void)

```
1 #include <stdio.h>
2
3 int main(void) ;이 생략되었다.
4 {
5     printf("Hello World!");
6     return 0;
7 }
```

100 %

오류 목록

전체 솔루션 2 오류 0 경고 0 메시지 빌드 + IntelliSense 검색 오류 목록

코드	설명	프로젝트	파일	줄	비표시 오류(Suppr...)
C2143	';'가 필요합니다. 구문 오류: ';'이(가) 'return' 앞에 없습니다.	add	error1.c	6	

오류가 발견된 소스 파일

오류가 발견된 줄번호

main VCodeFunction

(Name)	main
File	d:\sources\Wch
FullName	main

C++

준비 줄: 7 열: 2 문자: 2 INS

## 오류 #2

add - Microsoft Visual Studio

빠른 실행(Ctrl+Q)

파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버그(D) 팀(M) 도구(T) 테스트(S) 분석(N) 창(W) 도움말(H) 로그인

Debug x86 로컬 Windows 디버거

error3.c

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Hello World!");
6     return 0;
7 }
```

문자열을 표시할 때, “와 ”을 빠뜨렸다.

오류 목록

전체 솔루션 5 오류 2 경고 0 메시지 빌드 + IntelliSense 검색 오류 목록

코드	설명	프로젝트	파일	줄	비표시 오류(Suppr...)
abc	식별자 "Hello"이(가) 정의되어 있지 않습니다.	add	error3.c	5	
abc	')'가 필요합니다.	add	error3.c	5	
C2065	'Hello': 선언되지 않은 식별자입니다.	add	error3.c	5	
C4047	'함수': 'const char *const '의 간접 참조 수준이 'int'과(와) 다릅니다.	add	error3.c	5	

오류 목록 출력

솔루션 탐색기

솔루션 탐색기 검색(Ctrl+;)

솔루션 'add' (1개 프로젝트)

- add
  - 참조
  - 외부 종속성
  - 리소스 파일
  - 소스 파일
    - error3.c
  - 헤더 파일

솔루션 탐색... 팀 탐색기 클래스 뷰

속성

main VCCodeFunction

C++

속성	값
(Name)	main
File	d:\sources\Wch
FullName	main

C++

준비 줄: 6 열: 14 문자: 11 INS

↑ 게시

# 오류 #3

The screenshot shows the Visual Studio IDE with a C++ project named 'add'. The source file 'error3.c' contains the following code:

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     print("Hello World!");
6     return 0;
7 }
```

Two annotations are present:

- A yellow box with the text "printf이어야 한다." (It should be printf) points to the `print` function call on line 5.
- A yellow box with the text "함수를 찾지 못했음." (Could not find the function) points to the LNK2019 error in the error list.

The error list at the bottom shows the following errors:

코드	설명	프로젝트	파일	줄	비표시 오류(Suppr...)
C4013	'print'이(가) 정의되지 않았습니다. extern은 int형을 반환하는 것으로 간주합니다.	add	error3.c	5	
LNK2019	_print 외부 기호(참조 위치: _main 함수)에서 확인하지 못했습니다.	add	error3.obj	1	
LNK1120	1개의 확인할 수 없는 외부 참조입니다.	add	add.exe	1	

The right sidebar shows the Solution Explorer with the project 'add' and the file 'error3.c' selected. The Properties window shows the 'main' function in the 'VCCodeFunction' class.

# 논리 오류

- 다음과 같은 출력을 가지는 프로그램을 작성하여 보자.




# 논리 오류가 존재하는 프로그램

```
#include <stdio.h>

int main(void)
{
    printf("마트에서 사올 품목");
    printf("=====");
    printf("사과, 우유, 빵");
    printf("=====");
    return 0;
}
```

줄이 바뀌지  
않았음!




마트에서 사올 품목=====  
사과, 우유, 빵=====

# 논리 오류가 수정된 프로그램

```
#include <stdio.h>

int main(void)
{
    printf("마트에서 사올 품목\n");
    printf("=====\n");
    printf("사과, 우유, 빵\n");
    printf("=====\n");
    return 0;
}
```

논리 오류  
수정!!



마트에서 사올 품목  
=====  
사과, 우유, 빵  
=====

# 디버깅

- 디버깅: 논리 오류를 찾는 과정

아무래도 이 부분이  
수상해..



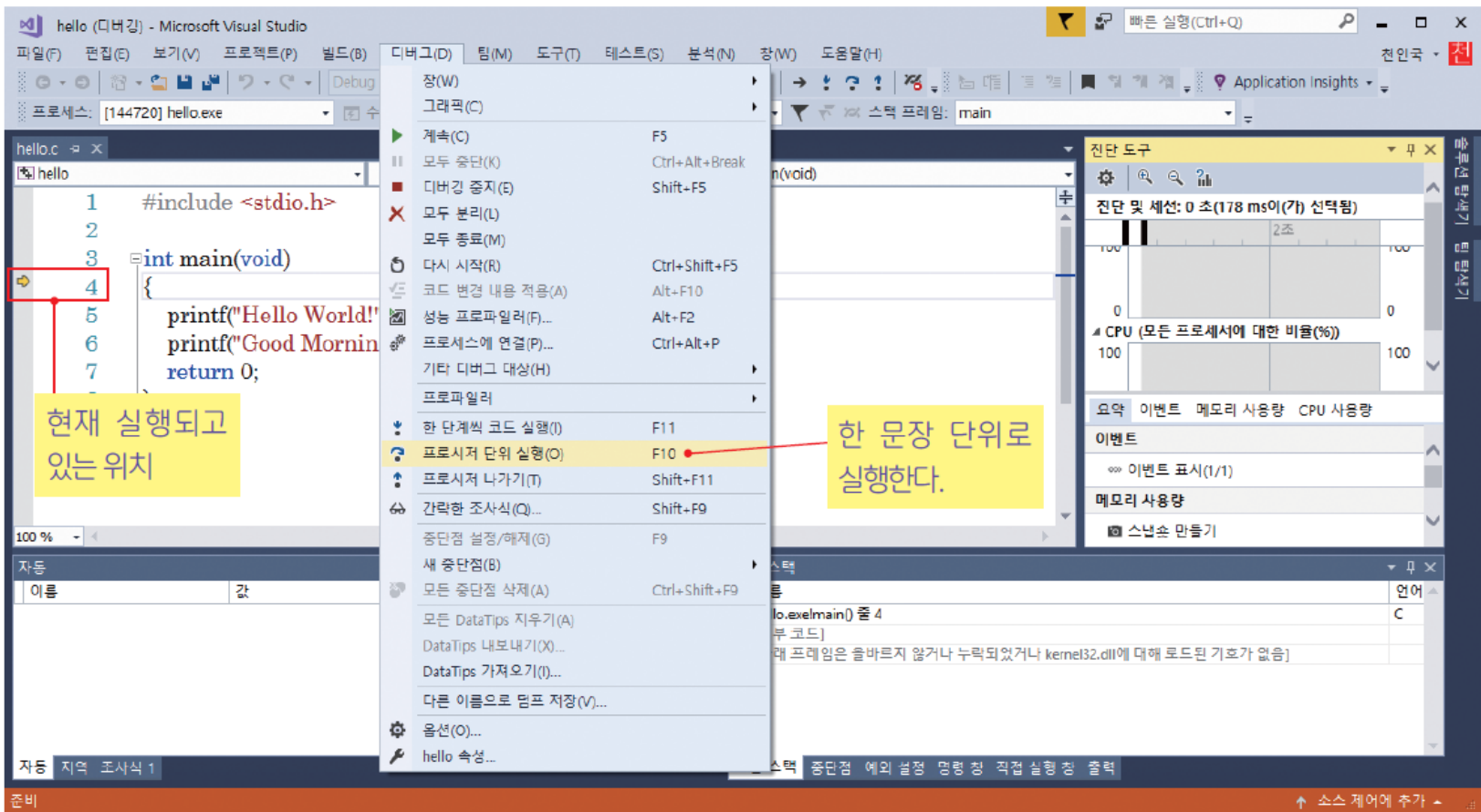
프로그램의 실행결과

논리 에러를 발견하는 것은  
수사관이 범죄 흔적을 이용하여  
범인을 찾는 것과 같습니다.



# 디버거(debugger)

- 프로그램을 한 문장씩 실행하면서 오류의 원인을 찾는 도구





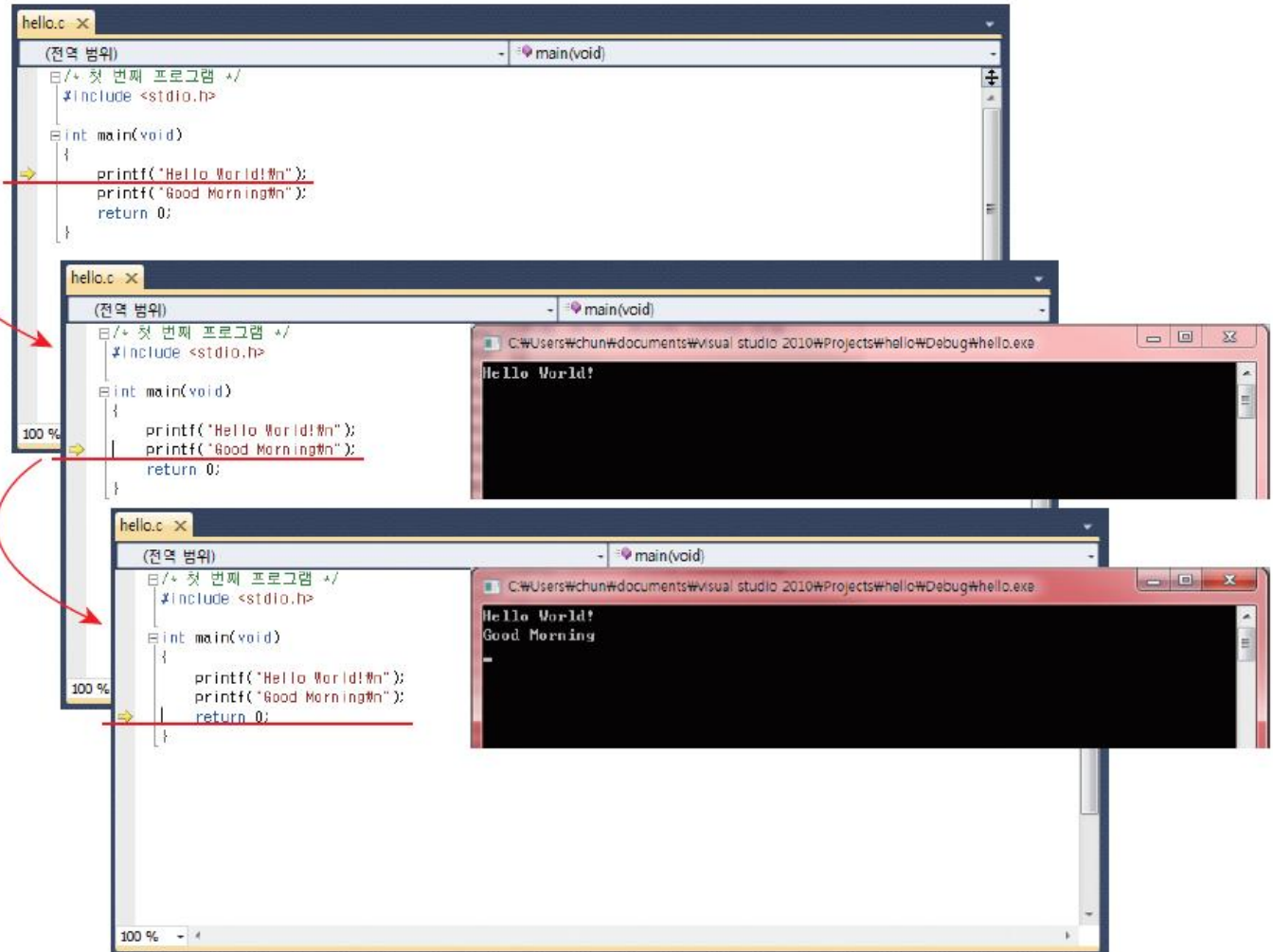
# 디버거의 명령어 정의

- 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

# 디버거의 실행 과정

F10을 누를 때마다  
한 문장씩 실행된다.

F10을 누를 때마다  
한 문장씩 실행된다.



# 참고사항: 디버거 기능

Start Debug→Go( <b>F5</b> )	디버깅 모드로 프로그램을 실행한다.
Restart( <b>Ctrl</b> + <b>Shift</b> + <b>F5</b> )	프로그램을 재실행한다.
Stop Debugging( <b>Shift</b> + <b>F5</b> )	디버깅을 중단한다.
Break Execution	프로그램 실행 중에 이 버튼을 누르면 현재 위치에서 실행이 중단된다.
Step Into( <b>F11</b> )	하나의 문장을 실행한다. 만약 문장에 함수 호출이 있으면 그 함수로 들어간다.
Step Over( <b>F10</b> )	하나의 문장을 실행한다. 만약 문장에 함수 호출이 있어도 함수도 들어가지 않는다.
Step Out( <b>Shift</b> + <b>F11</b> )	현재 실행중인 함수를 빠져 나온다.
Run to Cursor( <b>Ctrl</b> + <b>F11</b> )	현재 커서 위치까지 실행한다.
Quick Watch( <b>Shift</b> + <b>F9</b> )	현재 사용 중인 변수를 입력하여 그 변수의 값을 볼 수 있다.
Watch	보고 싶은 변수를 입력한다.
Variables	현재 사용되는 변수값이 표시된다.
Registers	CPU 안의 레지스터의 상태를 보여준다.
Memory	메모리를 16진수와 문자열로 표시한다.
Call Stack	함수의 호출 순서를 볼 수 있다.
Disassembly	변환된 어셈블리 코드를 보여준다.
<b>F9</b>	현재 위치에 중단점을 설정한다. 디버거가 중단점을 만나면 실행을 중지한다.

# Mini Project

- 오류를 수정해보자!

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

```
int Main(void)
```

```
(
```

```
    printf(안녕하세요?\n);
```

```
    printf(이번 코드에는 많은 오류가 있다네요\n)
```

```
    print(제가 다 고쳐보겠습니다.\n);
```

```
    return 0;
```

```
)
```

# Mini Project



Solution bug.c

```
1 #include <stdio.h>
2
3 int Main(void)
4 (
5     printf(안녕하세요? \n);
6     printf(이번 코드에는 많은 오류가 있다네요 \n);
7     print(제가 다 고쳐보겠습니다.\n);
8     return 0;
9 )
```

main

(가 아니라 {이어야 한다.

문장의 끝에는 ;가 있어야 한다.

문자열에는 따옴표를 붙인다.

print가 아니고 printf 이어야 한다.

# Mini Project

- 오류가 수정된 프로그램

```
#include <stdio.h>

int main(void)
{
    printf("안녕하세요 ? \n");
    printf("이번 코드에는 많은 오류가 있다네요\n");
    printf("제가 다 고쳐보겠습니다.\n");
    return 0;
}
```

# Q & A

