

쉽게 풀어쓴 C언어 Express[개정판] 천인국 저, 생능출판사 2012

3장. C 프로그램 구성 요소

성결대학교 컴퓨터공학부
임 상 순

강의 목표 및 내용

▶ 강의 목표

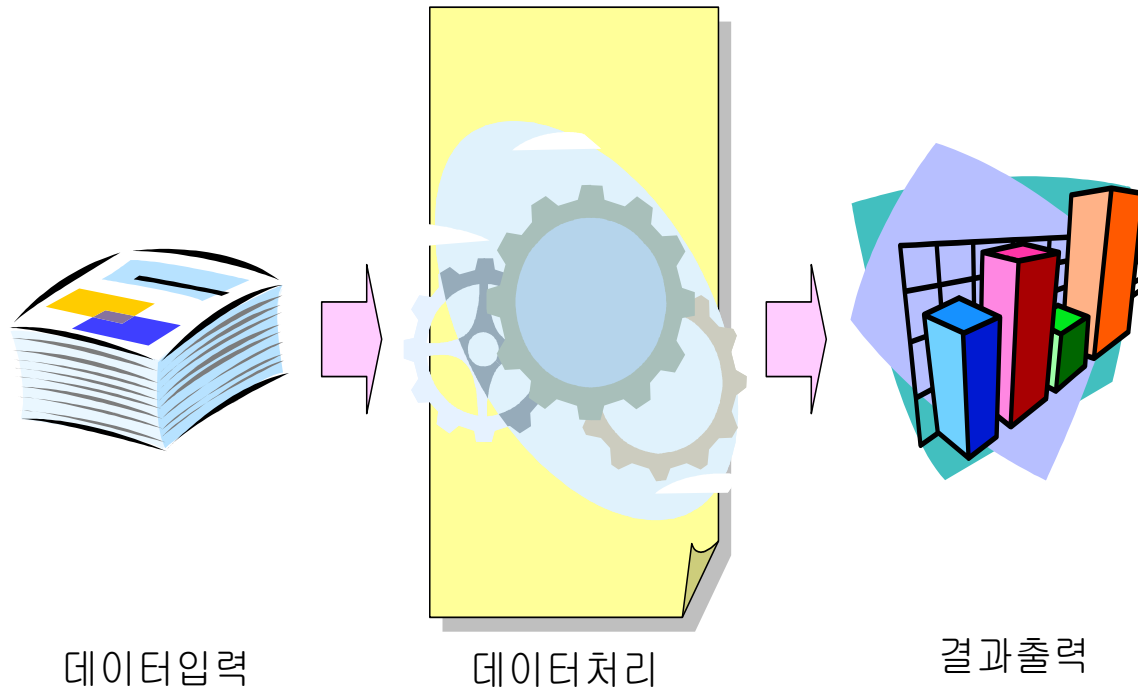
- 주석, 변수, 함수, 문장 등의 프로그램을 구성하는 요소들의 개념을 이해한다.
- printf()와 scanf() 같은 입출력 함수의 사용법을 익힌다.
- 수식과 연산의 기초적인 사항들을 학습한다.

▶ 내용

- 덧셈 프로그램 1
- 주석, 전처리기, 함수, 변수
- 수식과 연산
- printf(), scanf()
- 덧셈 프로그램 2
- 프로그램 예제 실습

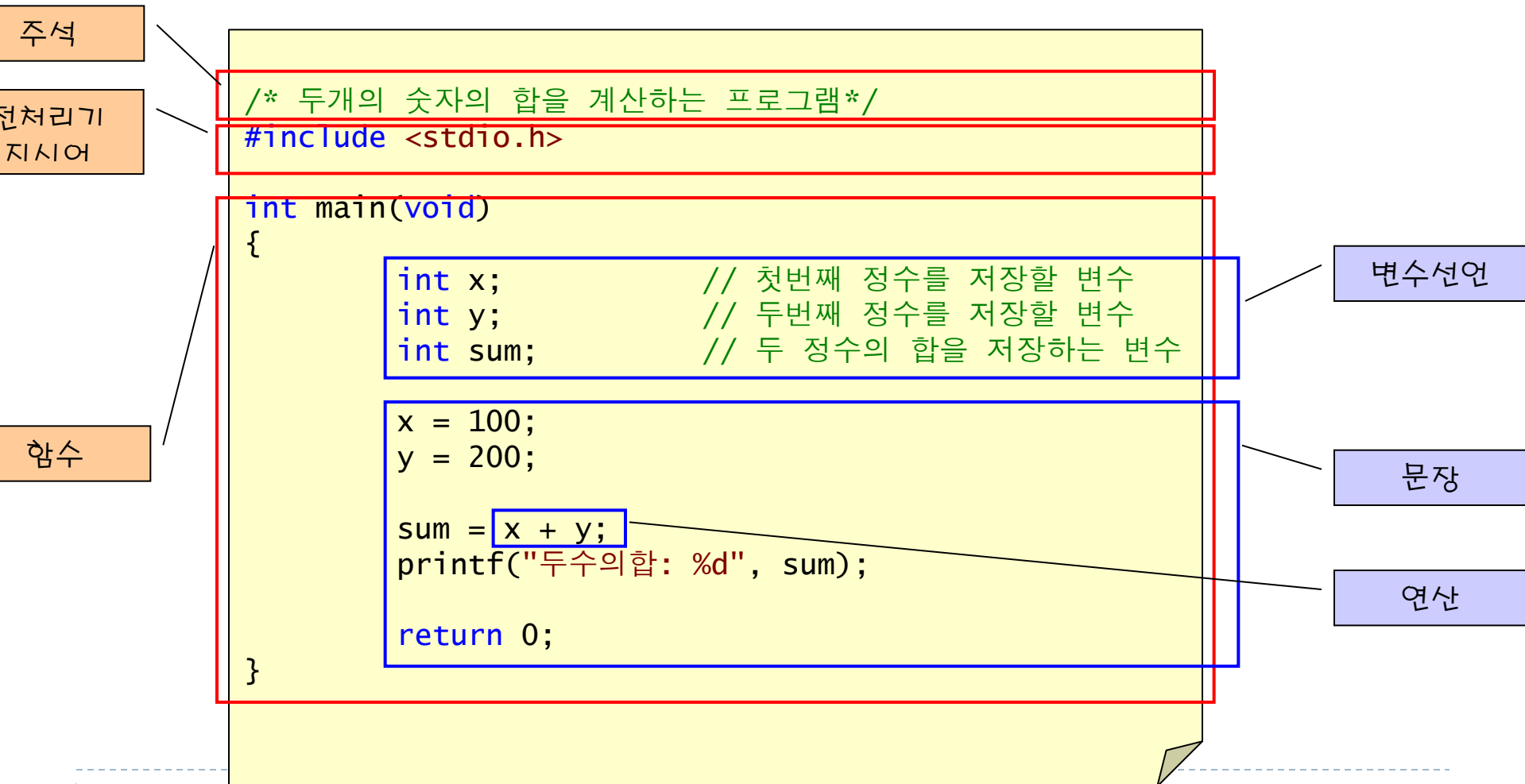
일반적인 프로그램의 형태

- ▶ 데이터를 받아서(입력단계), 데이터를 처리한 후에(처리 단계), 결과를 화면에 출력(출력단계)



덧셈 프로그램 #1 [1/2]

▶ 소스 파일 이름 : add1.c



덧셈 프로그램 #1 [2/2]

▶ 프로그램 실행 화면



주석(Comment)[1/3]

▶ 주석

- 코드를 설명하는 설명글
- 반드시 있어야 하는 부분은 아님
- 컴파일러는 주석을 무시
 - ▶ 주석에 대한 기계어 코드를 생성하지 않음

```
/* 두개의 숫자의 합을 계산하는 프로그램 */
```

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

```
int main(void)
```

```
{
```

```
...
```

```
...
```

```
...
```

```
}
```

주석은 코드를 설명
하는 글입니다.



주석

주석(Comment)[2/3]

▶ 주석의 특징

- 프로그램의 가독성을 높임
- 프로그램의 구조와 동작을 설명
- 좋은 주석은 코드를 설명하지 않고 개발자의 작성 의도를 표현

▶ 3가지 방법의 주석

- `/* 한줄로 된 주석 */`
- `/* -----`

저자: 홍길동

날짜: 2013.3.4

여러 줄로 이루어진 주석

----- */

- `// 여기서부터 줄의 끝까지 주석`

▶ 주의할 점

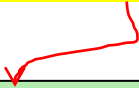
- 주석 안에 다른 주석이 들어가면 안됨
 - ▶ 예) `/* /* 이것은 잘못된 주석 방법입니다. */ */`

주석(Comment)[3/3]

▶ 주석의 예

- 프로그램 전체의 목적을 설명하는 주석의 예

• 주석



```
/* This program accepts an array of N elements and a key. *  
* Then it searches for the desired element. If the search *  
* is successful, it displays "SUCCESSFUL SEARCH". *  
* Otherwise, a message "UNSUCCESSFUL SEARCH" is displayed. */
```

```
#include <stdio.h>  
void main()  
{  
...  
}
```


들여쓰기

▶ 들여쓰기(indentation)

- 같은 수준에 있는 문장들을 왼쪽 끝에서 몇 자 안으로 들여쓰는 것
- 소스 코드의 가독성을 높이는 역할
- 일반적으로 탭(Tap)키를 활용
- IDE에 자동 들여쓰기가 지원되는 경우가 있음

The diagram shows a C code snippet with annotations explaining the purpose of indentation:

```
#include <stdio.h>

int main(void)
{
    int x;
    int y;
    int sum;
    ...
    return 0;
}
```

Annotations:

- 빈줄을 넣어서 의미별로 구별을 한다.** (A red box highlights the blank line between `#include` and `int main`.)
- 같은 내용의 처리이면 들여쓰기를 한다.** (A red box highlights the indentation of `int x;`, `int y;`, `int sum;`, and `return 0;`.)
- 프로그램의 의도를 주석으로 설명한다.** (A red box highlights the comments: `// 첫번째 정수를 저장할 변수`, `// 두번째 정수를 저장할 변수`, and `// 두 정수의 합을 저장하는 변수`.)

주석과 들여 쓰기가 없다면..

- ▶ 들여쓰기를 하지 않으면 읽기가 매우 어려워짐
- ▶ 주석과 들여쓰기도 프로그래밍 실력에 포함

```
#include <stdio.h>
int main(void) { int x; int y; int sum;
x = 100; y = 200; sum = x + y;
printf("두수의 합: %d", sum); return 0; }
```

실행은 되지만 무슨
처리를 하고 있는
프로그램인지 알기
가 힘들고 또한 들
여쓰기가 안 되어
있어서 같은 수준에
있는 문장들을 구분
하기 힘듭니다.





중간 점검

- ▶ 주석은 /* /* */ */와 같이 중첩할 수 있을까?
- ▶ 주석은 한 줄 이상이 될 수 있는가?
- ▶ 주석에는 어떤 내용을 쓰면 좋은가?
- ▶ 주석은 프로그램의 동작에 어떤 영향을 끼치는가?

전처리기[1/2]

- ▶ `#include <stdio.h>`
 - 전처리기 지시어
 - 헤더 파일 `stdio.h`를 소스 코드 안에 포함
 - ▶ 헤더 파일 : 코드의 일부분이 들어있는 텍스트 파일(.h)
- ▶ 모니터로 출력하는 프로그램을 변환할 때 컴파일러는 `printf()` 함수에 대한 정의가 있어야 올바르게 컴파일 가능
 - `printf()` 함수의 정의는 `stdio.h`에 존재함

• `stdio.h`는 표준 입출력에 대한 라이브러리 함수의 정의가 들어 있다.

`#include <stdio.h>`

- 외부 파일을 포함시키라는 의미의 전처리기
- `#`기호로 시작

전처리기[2/2]

```
/* 첫번째 프로그램 */
```

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

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

hello.c

```
// stdio.h
```

```
...
```

```
int printf(char *,...);
```

```
...
```

stdio.h

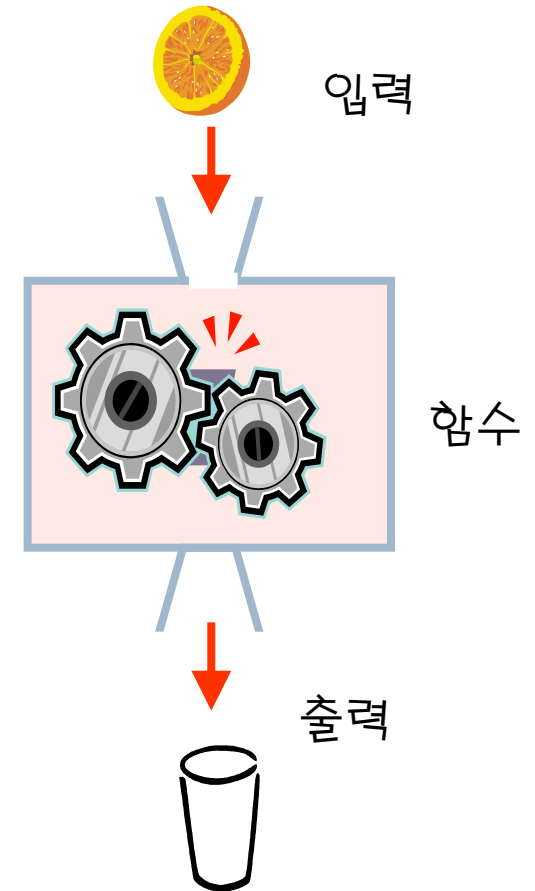


중간 점검

- ▶ printf()를 사용하기 위하여 포함시켜야 하는 헤더 파일은 무엇인가?
- ▶ 전처리기 #include의 의미는 무엇인가?

함수의 기본 개념

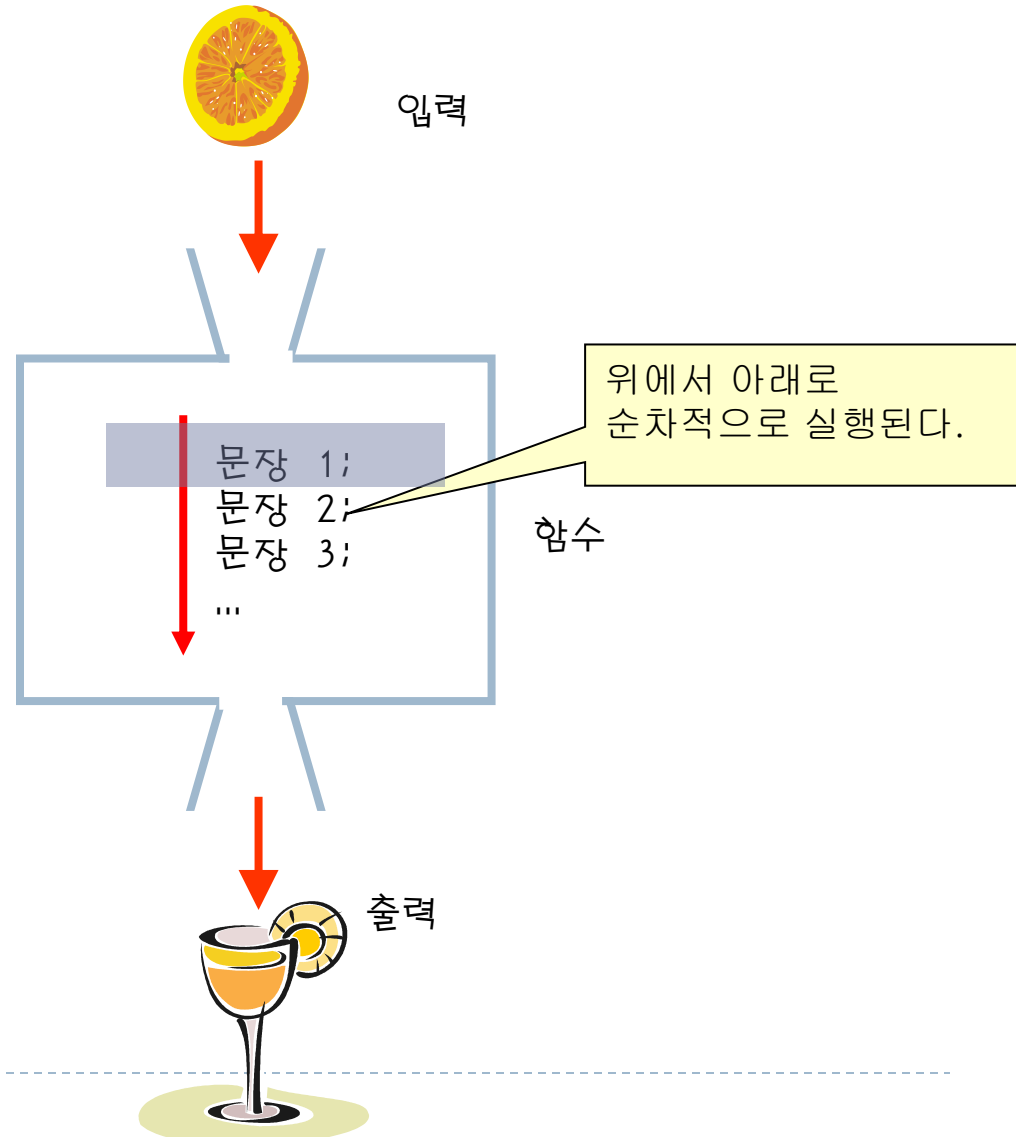
- ▶ 함수(function)
 - 특정 기능을 수행하는 처리 단계들을 괄호로 묶어서 이름을 붙인 것
 - 입력을 받아 지시대로 처리하고 출력을 생성
- ▶ 함수는 프로그램을 구성하는 기본적인 단위(부품)
- ▶ 강의 전반부에는 함수를 하나만 사용하지만 일반적인 C 프로그램은 많은 함수로 이루어짐



함수 안에 들어 있는 것[1/2]


Q) 그렇다면 함수 안에 들어 있는 것은 무엇인가?

A) 함수 안에는 함수가 처리하는 처리 단계(문장)들이 중괄호 안에 나열



함수 안에 들어 있는 것[2/2]

- ▶ 작업을 수행하는 문장은 함수 안에 들어가야 함

```
int main(void)
{
    
}
```

- 여기에 작업을 넣을 것.

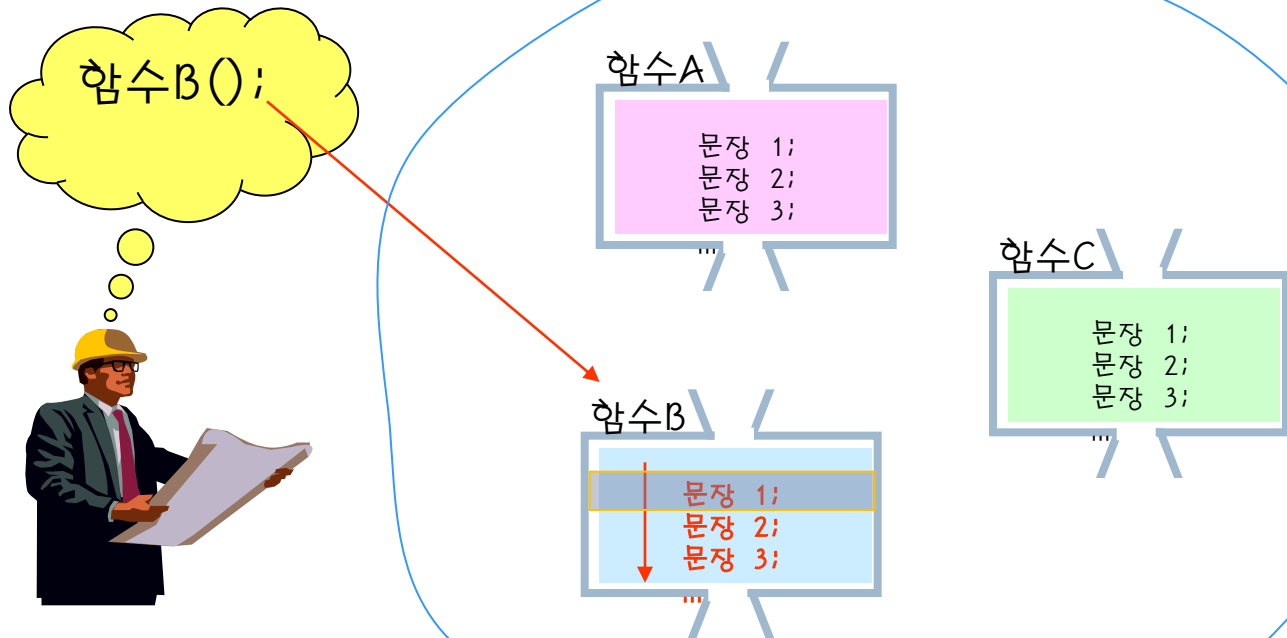
함수 호출[1/2]

Q) 함수 안에 있는 문장들은 언제 실행되는가?

A) 함수가 호출되면 실행된다.

Q) 함수 호출은 어떻게 하는가?

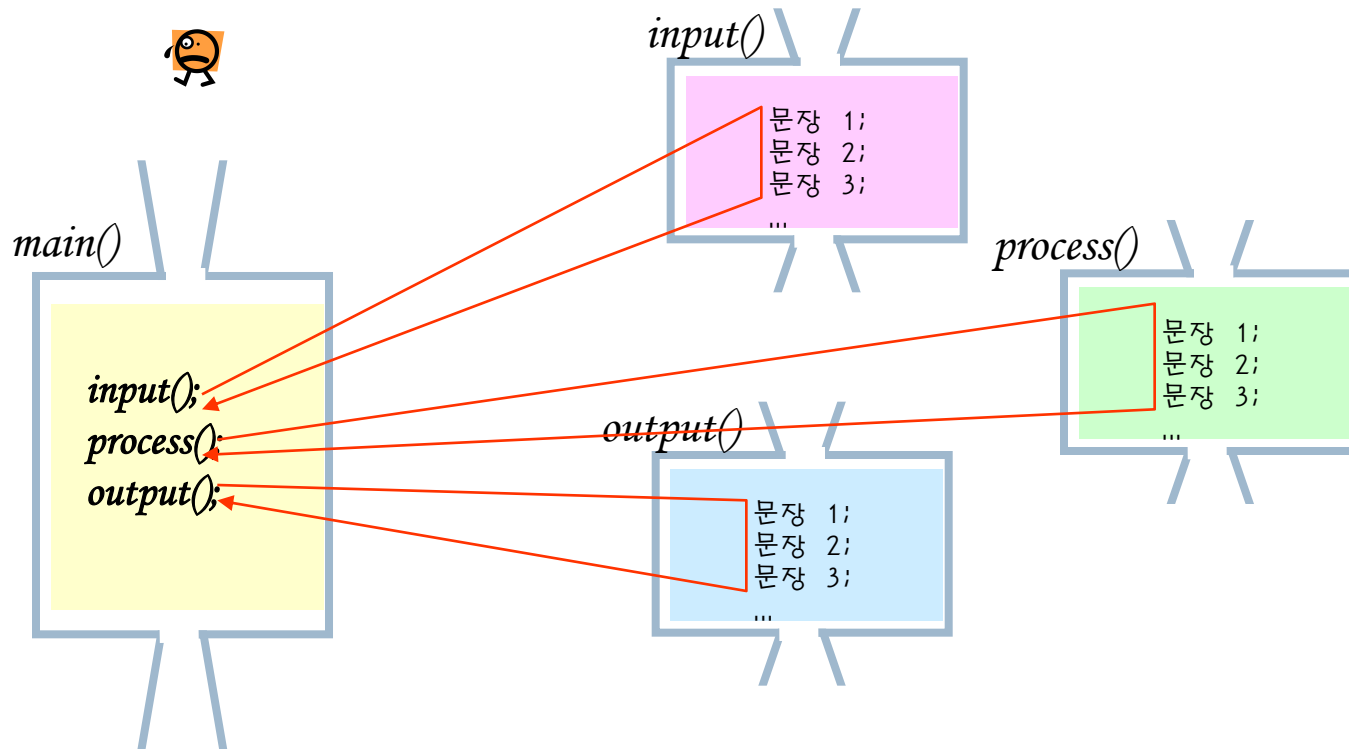
A) 함수의 이름을 적어주면 된다.



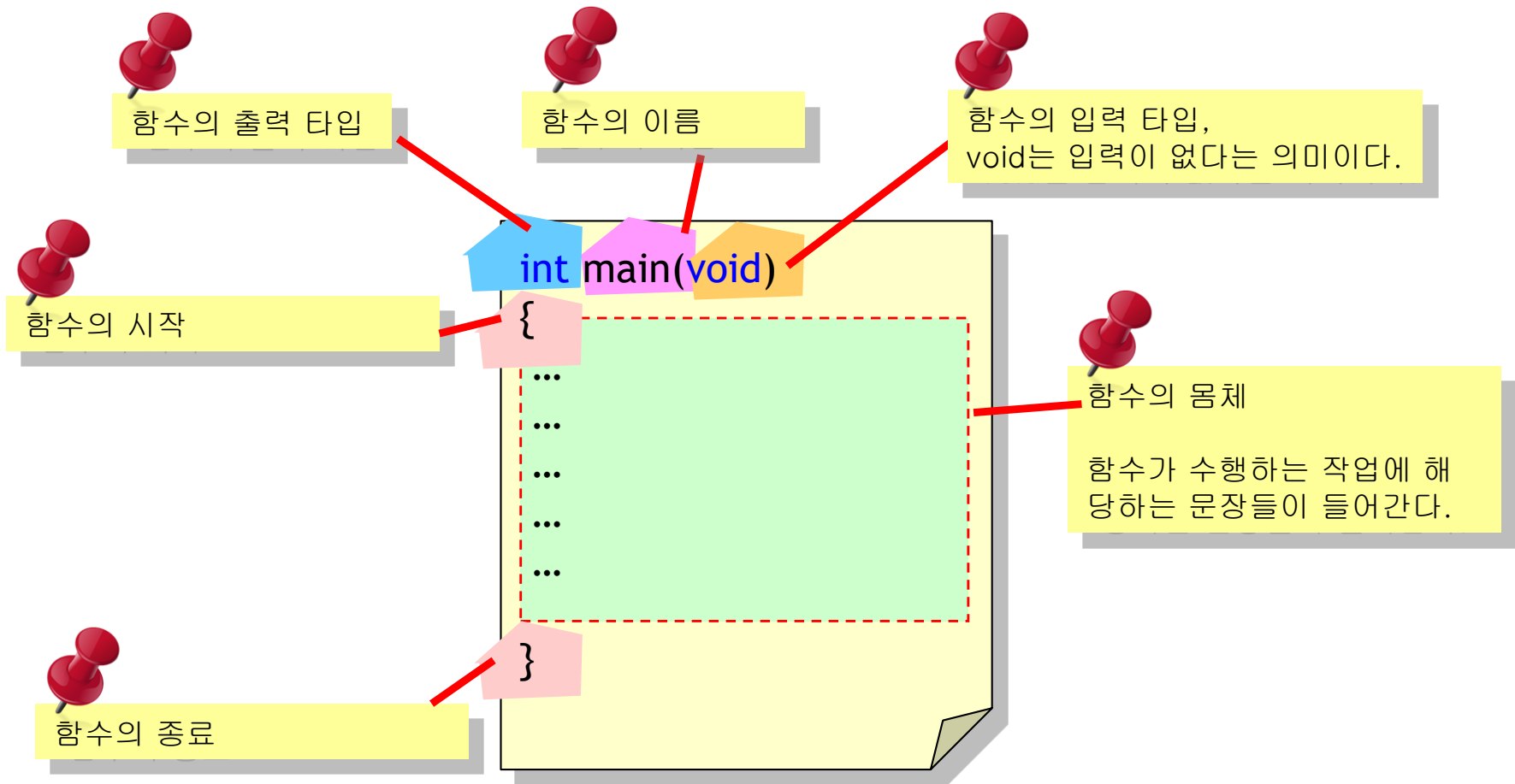
함수 호출 [2/2]

Q) 많은 함수 중에서 가장 먼저 실행되는 것은?

A) `main()` 함수이다. 다른 함수들은 `main()`으로부터 직간접적으로 호출된다.

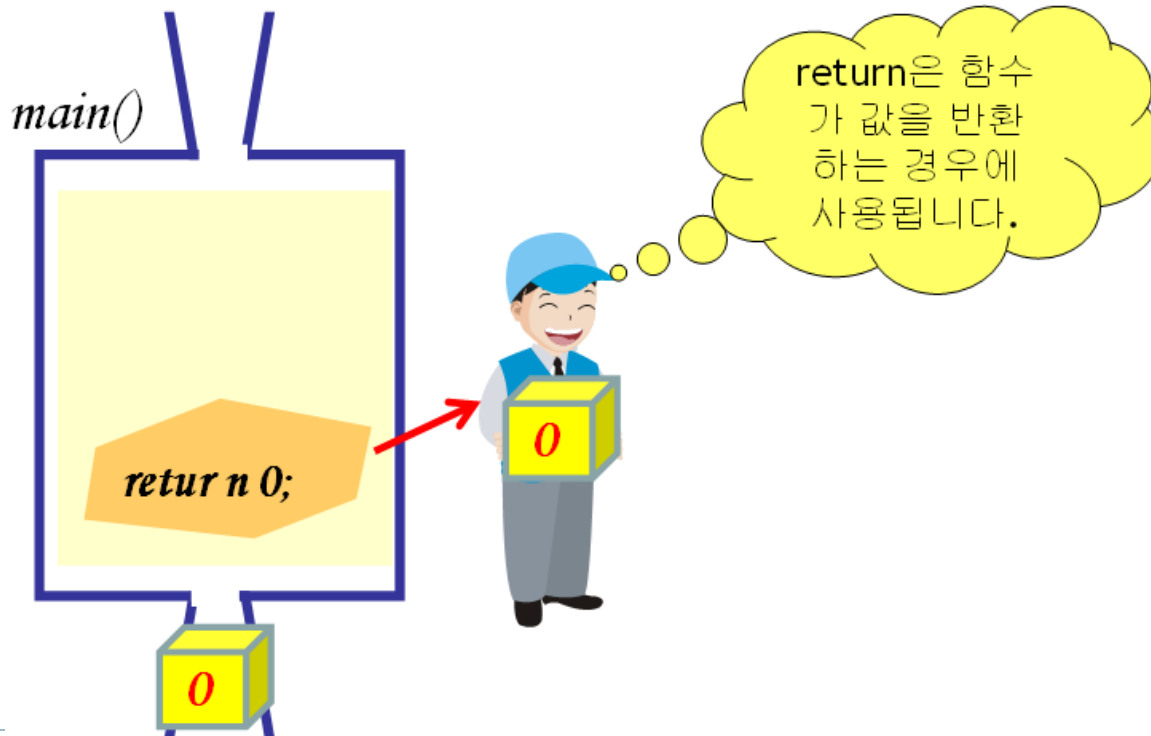


함수의 구조



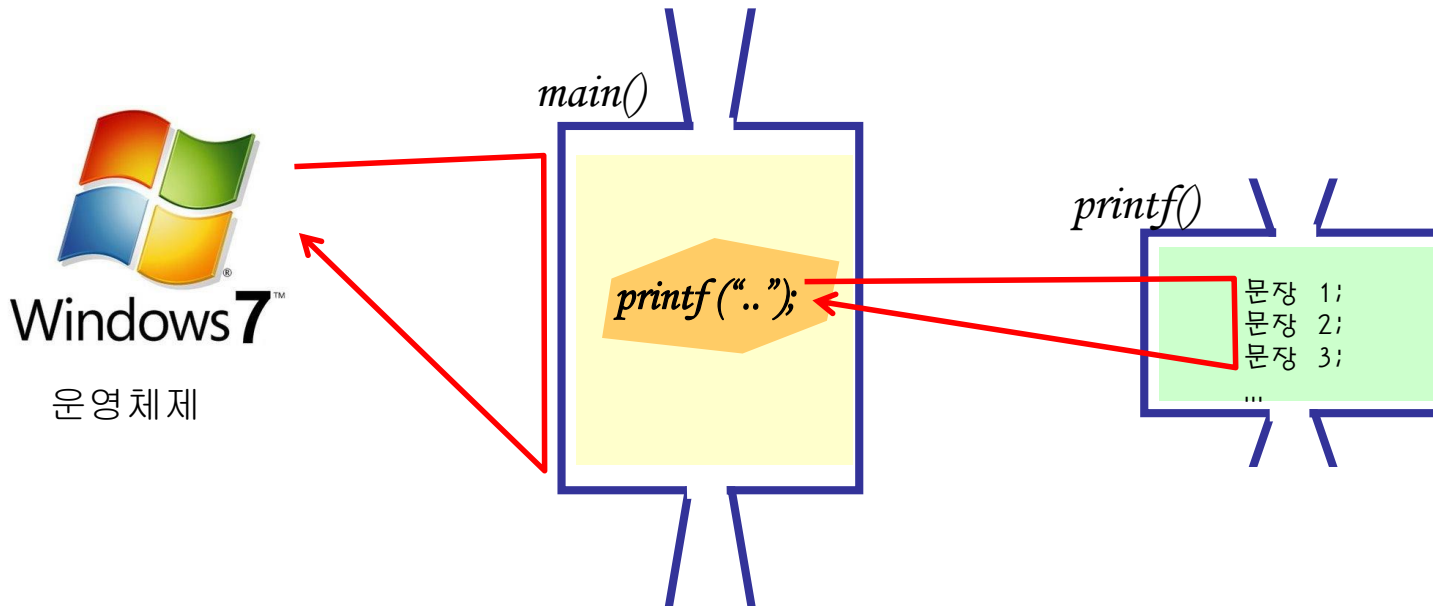
return 문장

- ▶ return은 함수를 종료시키면서 값을 반환하는 키워드
 - int main(void)처럼 int를 반환하는 것으로 정의하고 값을 반환하지 않는다면 컴파일러가 경고 메시지 발생



main()은 누가 호출할까?

- ▶ main()은 일반적으로 운영체제가 호출
 - 보통 성공 시 0 반환



함수를 실행하
려면 함수를
호출하면 됩니
다.





중간 점검

- ▶ 모든 C 프로그램에 반드시 있어야 되는 함수는 무엇인가?
- ▶ 함수의 시작과 끝을 나타내는 기호는 무엇인가?
- ▶ 모든 문장은 어떤 기호로 끝나는가?

변수의 정의

```
int x;    // 첫번째 정수를 저장하는 변수  
int y;    // 두번째 정수를 저장하는 변수  
int sum;  // 두 정수의 합을 저장하는 변수
```

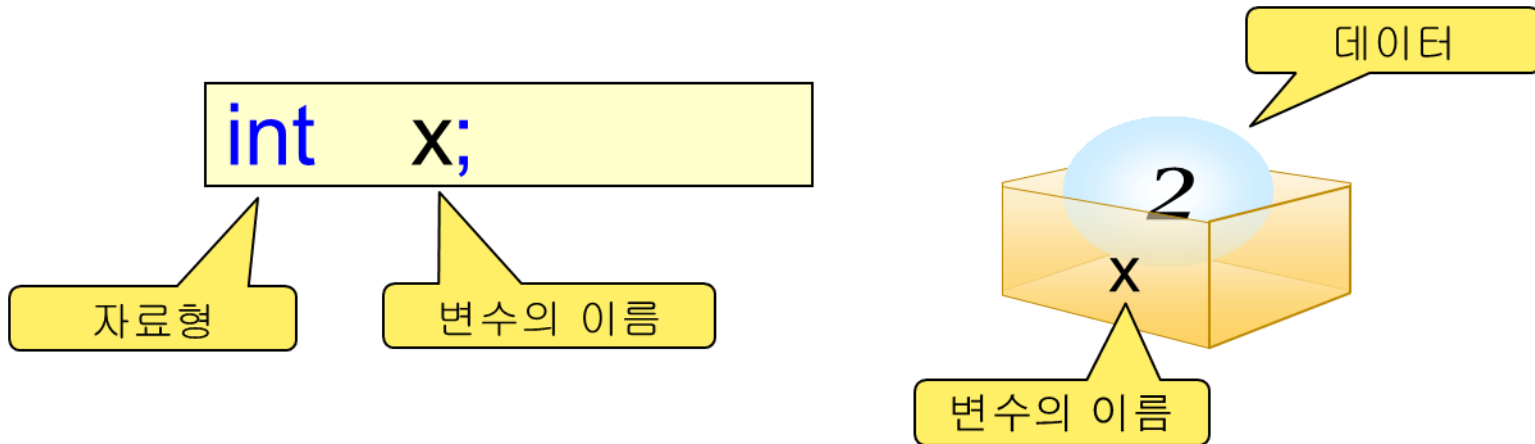
Q) 변수란 무엇인가?

A) 프로그램이 사용하는 데이터를 일시적으로 저장할 목적으로 사용하는 메모리 공간



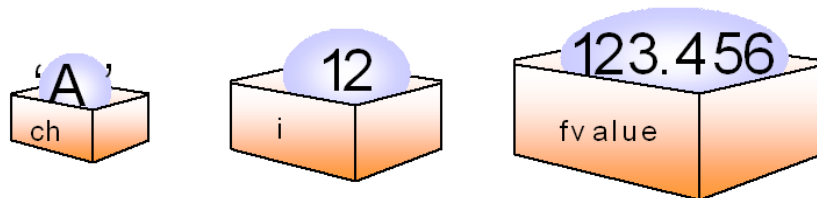
변수의 종류[1/2]

- ▶ 변수는 데이터를 담는 상자
 - 이 상자는 컴퓨터 안의 메인 메모리 안에 생성됨



변수의 종류[2/2]

- ▶ 변수에는 저장하는 데이터의 종류나 범위에 따라 여러 가지 유형(type)이 존재
 - 정수를 저장할 수 있는 변수
 - 실수를 저장할 수 있는 변수
 - 문자를 저장할 수 있는 변수



Note: Not to Scale

변수의 이름

▶ 식별자(identifier)

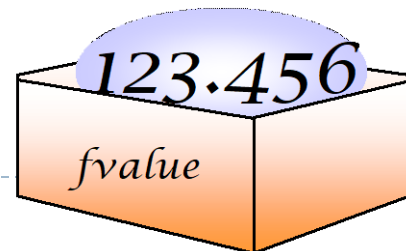
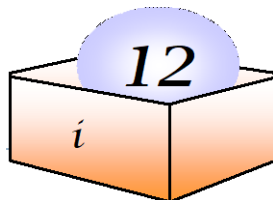
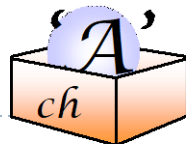
- 변수나 함수의 이름

▶ 식별자를 만드는 규칙

- 식별자는 영어의 대소문자, 숫자, 밑줄 문자 _로 이루어짐
- 식별자는 숫자로 시작할 수 없음
- 대문자와 소문자를 구별하며 C 언어의 키워드와 똑같은 이름은 허용되지 않음

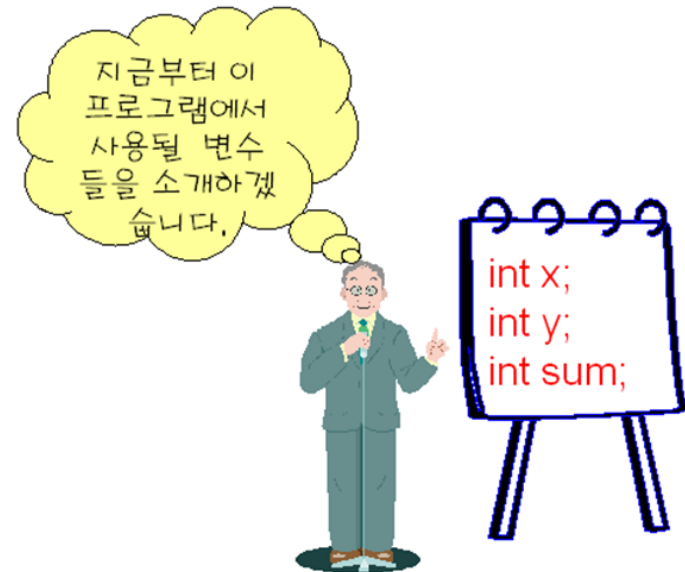
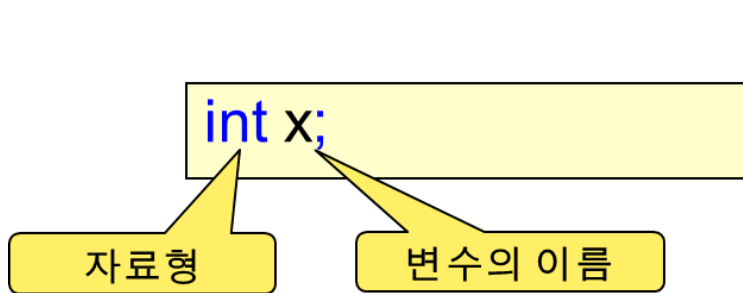
▶ 식별자의 예:

- s, s1, student_number: 올바른 식별자
- \$s, 2nd_student , int: 잘못된 식별자



변수 선언[1/3]

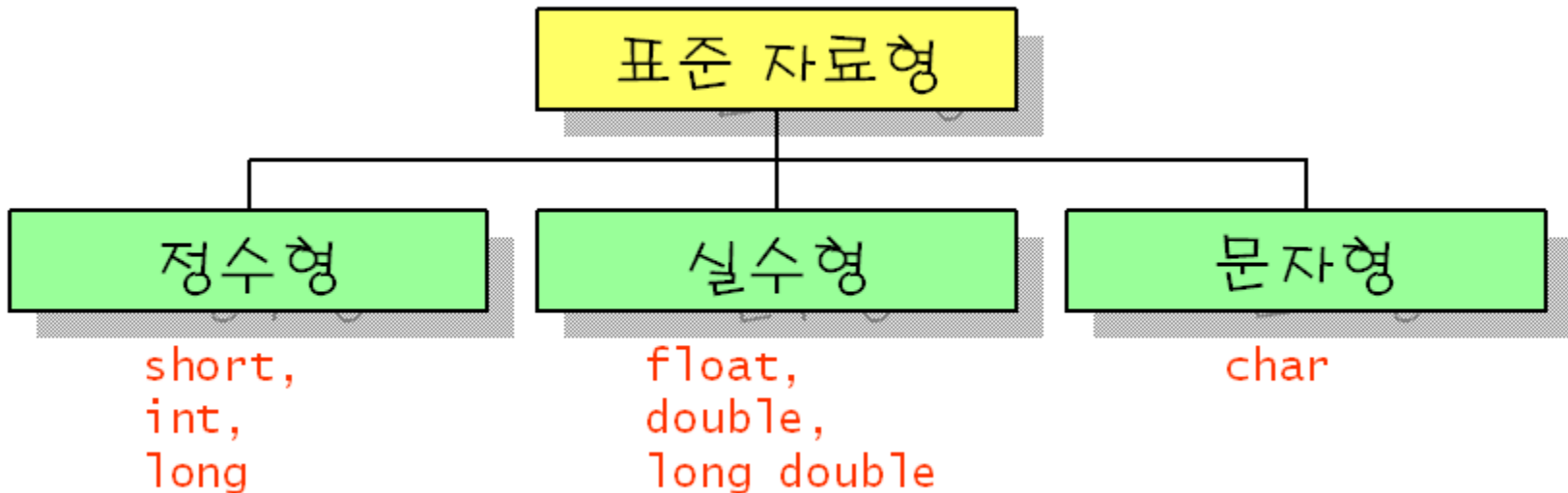
- ▶ C언어에서 변수를 사용하려면 변수 선언 필요
 - 컴파일러에게 어떤 타입의 변수가 사용되는지를 미리 알리는 것
 - 자료형과 변수의 이름을 적어주는 것
 - 함수의 첫 부분에서 주로 변수 선언



변수 선언[2/3]

▶ 자료형(data type)

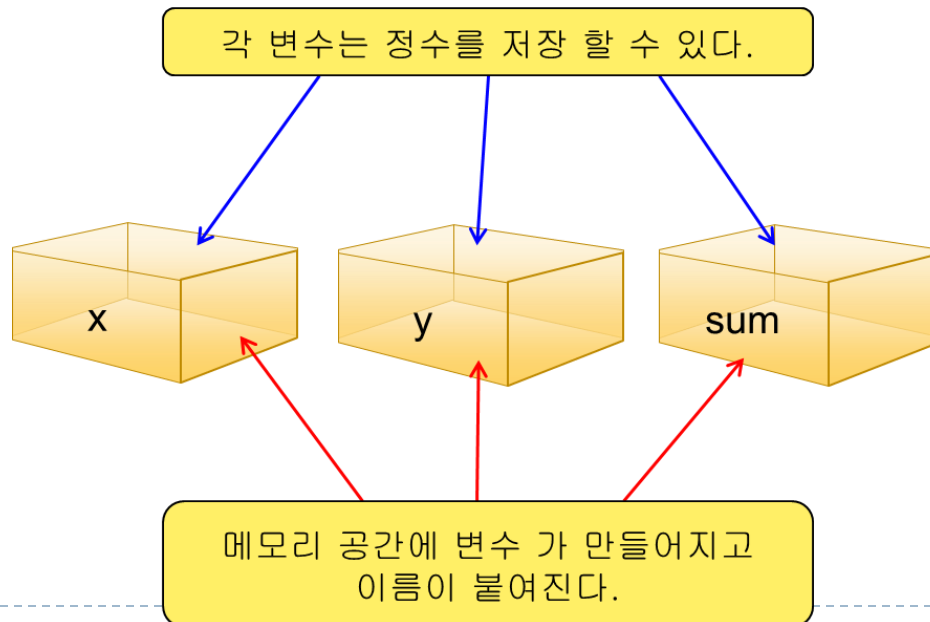
- 변수가 저장할 데이터가 정수인지 실수인지, 아니면 또 다른 어떤 데이터인지를 지정하는 것



변수 선언[3/3]

- ▶ 같은 자료형의 변수를 여러 개 선언 가능
 - 예) `int x, y, sum;`

```
int x;    // 첫번째 정수를 저장하는 변수  
int y;    // 두번째 정수를 저장하는 변수  
int sum;  // 두 정수의 합을 저장하는 변수
```

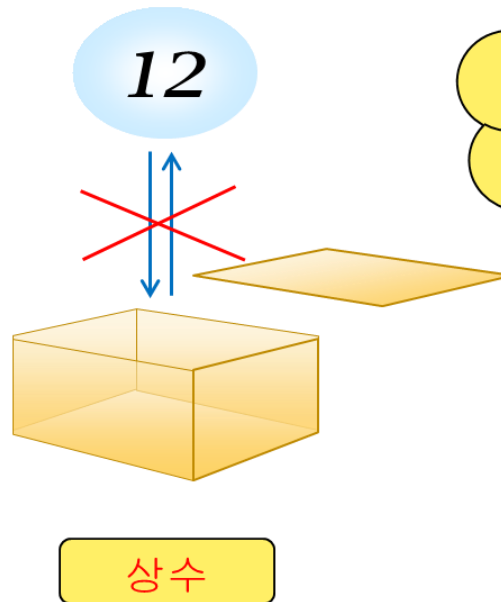
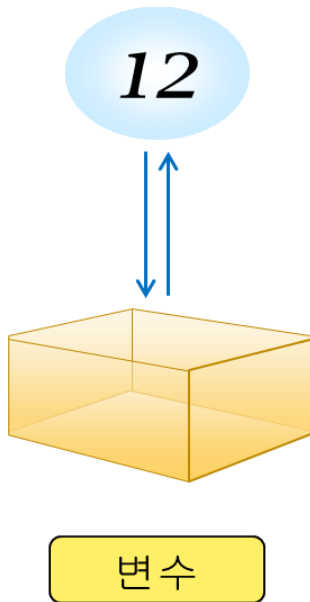


상수

▶ 변하지 않는 값

```
x = 100;  
y = 200;
```

상수



변수는 실행도중에
값을 변경할 수
있으나 상수는
한번 값이
정해지면 변경이
불가능합니다.





중간 점검

- ▶ int형 변수 i를 선언하는 문장을 작성하여 보자.
- ▶ double형 변수 f를 선언하는 문장을 작성하여 보자.
- ▶ 변수 선언은 함수의 어떤 위치에서 하여야 하는가?

수식

▶ 수식(expression)

- 피연산자와 연산자로 구성된 식
- 수식은 결과값을 가짐

```
sum = x + y;
```

x 가 3일 때 수식
 $x^2 - 5x + 6$ 의 값을
계산하라.



```
int x, y;
```

```
x = 3;
```

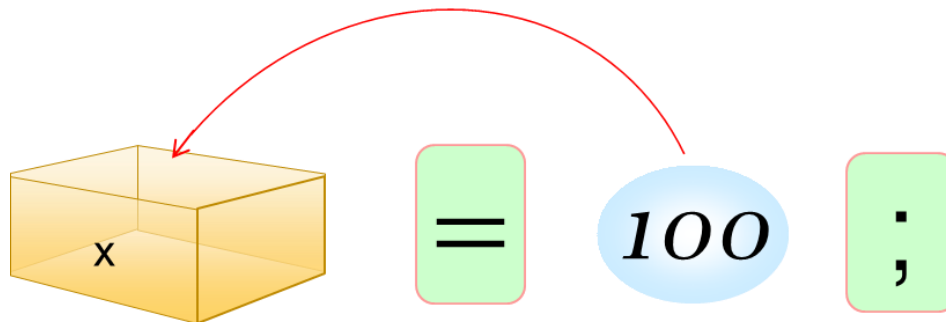
```
y = x * x - 5 * x + 6;
```

```
printf("%d\n", y);
```

대입 연산

- ▶ 대입 연산(assignment operation)
 - 변수에 값을 저장하는 연산
 - 대입 연산 = 배정 연산 = 할당 연산
 - =의 좌변에는 항상 변수가 위치하고 우변에는 값이 위치

```
x = 100;
```



= 연산자는
변수에 값을
저장합니다.



산술 연산[1/2]

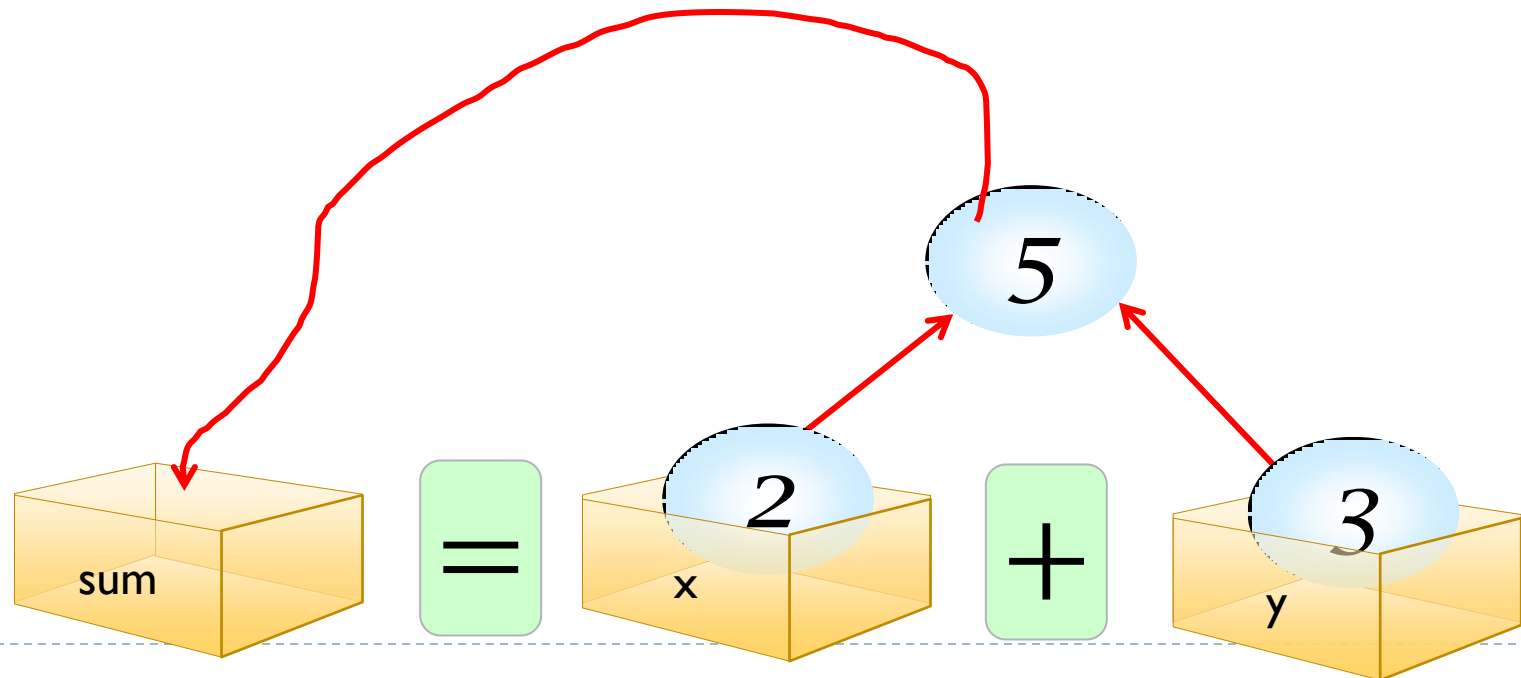
▶ 산술 연산을 수행하는 연산자

연산	연산자	C 수식	수학에서의 기호
덧셈	+	$x + y$	$x + y$
뺄셈	-	$x - y$	$x - y$
곱셈	*	$x * y$	xy
나눗셈	/	x / y	x / y
나머지	%	$x \% y$	$x \bmod y$

산술 연산[2/2]

- ▶ 변수 x에 들어있는 정수와 변수 y에 들어있는 정수를 더해서 변수 sum에 대입

```
sum = x + y;
```



정리

프로그램

컴퓨터 내부

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

```
int main(void)
```

```
{
```

```
int x;
```

```
int y;
```

```
int sum;
```

```
x = 100;
```

```
y = 200;
```

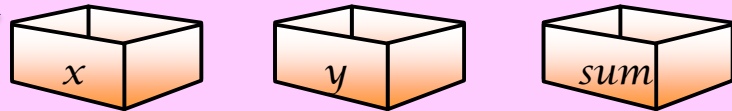
```
sum = x + y;
```

```
printf("두수의 합: %d", sum);
```

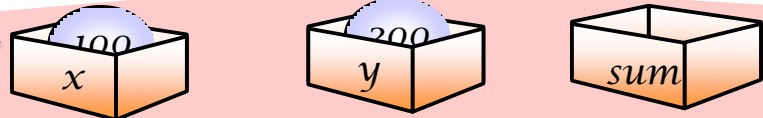
```
return 0;
```

```
}
```

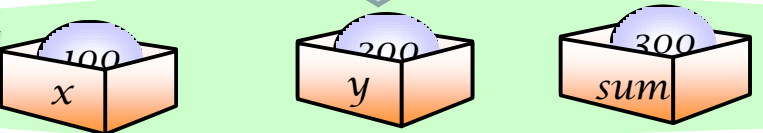
변수가 생성된다.



변수에 값이 대입된다.



덧셈 연산이 수행된다.



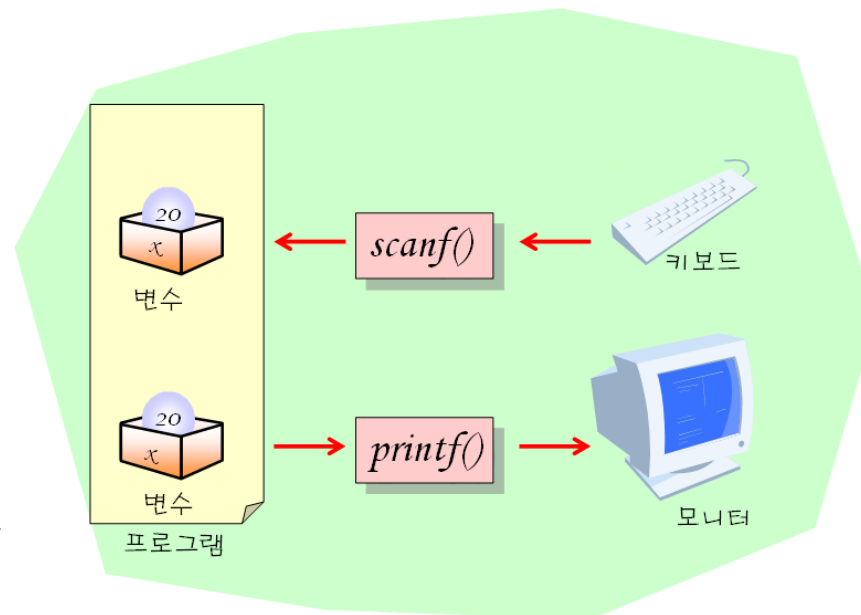


중간 점검

- ▶ 변수 a 와 변수 b 의 곱을 변수 `product`에 저장하는 문장을 작성하여 보자.
- ▶ 변수 a 를 변수 b 로 나눈 값을 변수 `quotient`에 저장하는 문장을 작성하여 보자.

printf()

- ▶ C언어는 입력과 출력을 위하여 라이브러리 함수 제공
 - 라이브러리 함수란 컴파일러가 프로그래머가 사용할 수 있도록 제공하는 함수들
- ▶ printf()
 - 모니터에 출력을 하기 위한 표준 출력 라이브러리 함수
- ▶ scanf()
 - 키보드에서의 입력을 위한 표준 입력 라이브러리 함수



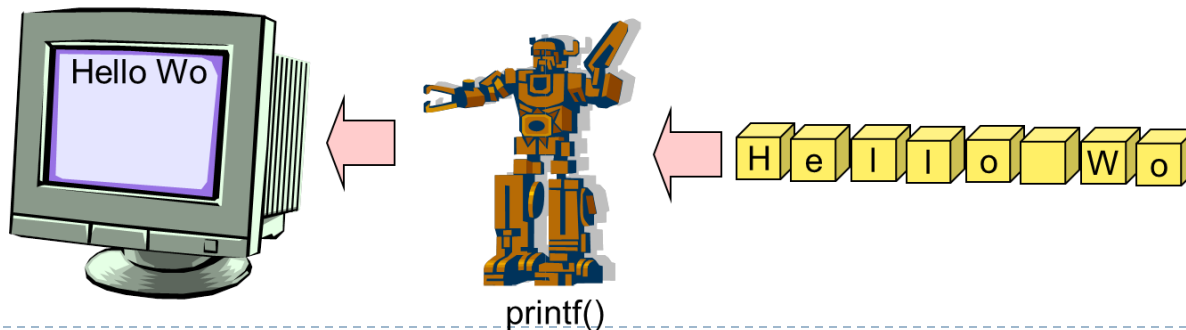
문자열 출력

▶ printf() 함수 사용 방법

- printf()를 적어주고 필요한 데이터를 () 안에 삽입
- 문자열 출력을 위해서는 " " 안에 원하는 문자열을 적고 () 안에 삽입
- 함수에게 전달하는 데이터를 인수(Argument)라고 함
 - ▶ 아래의 예에서는 "Hello World!"가 printf() 함수의 인수

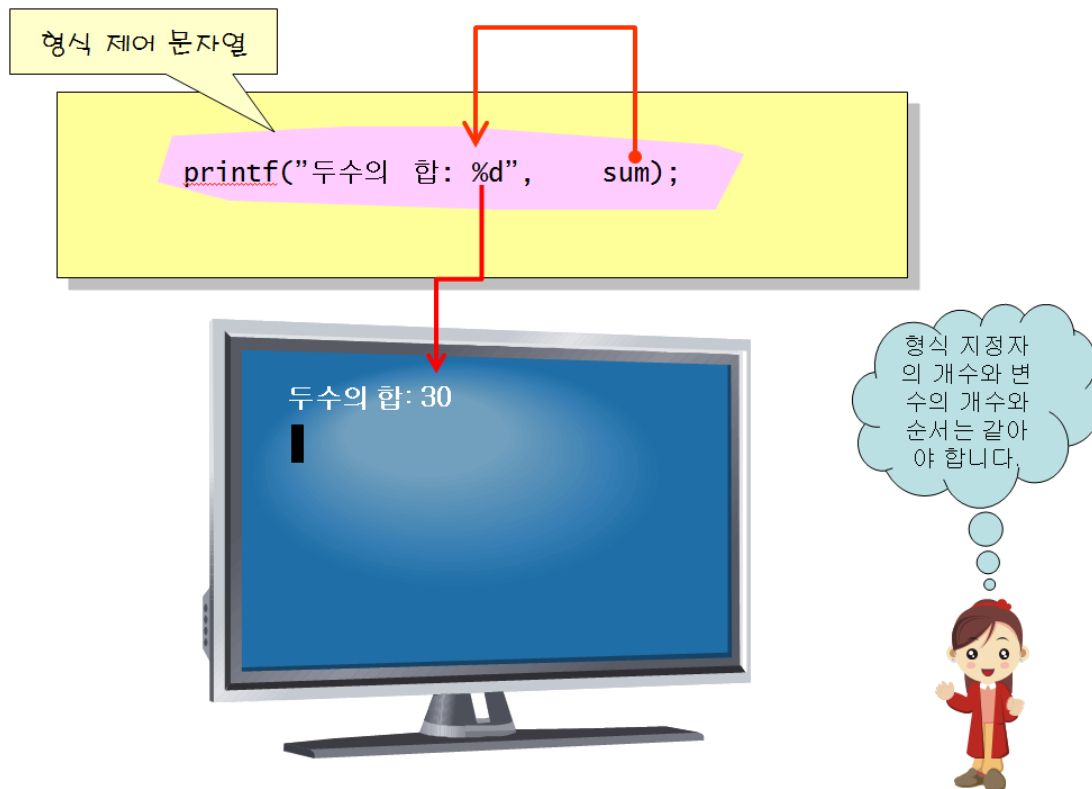
```
printf("Hello World!\n");
```

- 문자열(string): "Hello World!\n"와 같이 문자들을 여러 개 나열한 것



변수값 출력[1/4]

- ▶ printf()에는 형식을 지정하여 변수의 값을 출력하는 기능도 존재
 - %d는 %d를 출력하라는 의미가 아니고 출력 형식을 지정
 - ▶ %d는 정수의 형태로 출력하라는 형식 제어 문자열
 - 실제로는 ,(coma) 이후에 나오는 변수의 값이 출력됨



변수값 출력 [2/4]

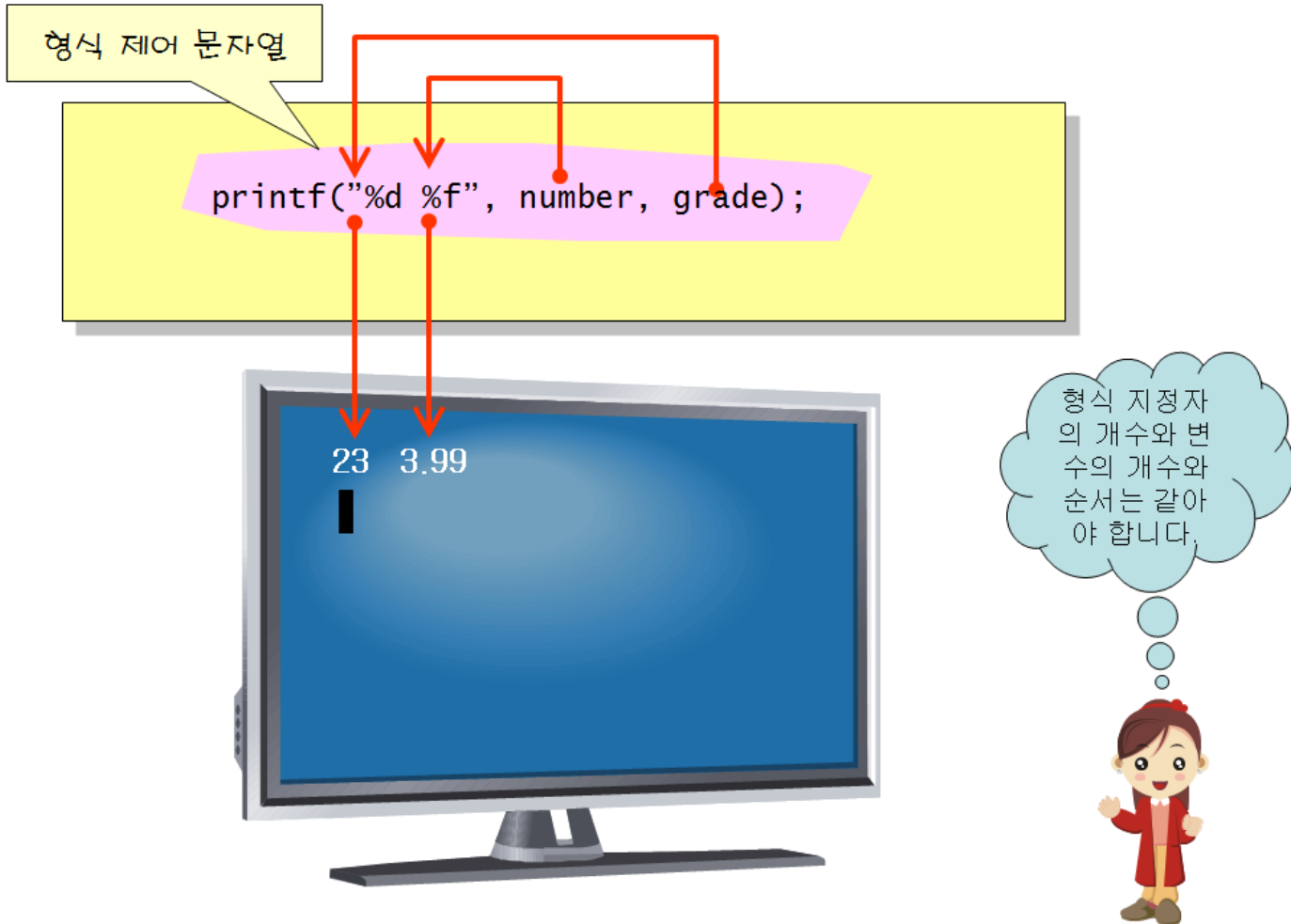
▶ 형식 지정자

- printf()에서 값을 출력하는 형식을 지정
- 형식 지정자와 변수들은 1개 이상일 수 있음
- 중간에 문자열이 있을 수 있음
- 형식 지정자의 자리에 변수의 값이 대치되어 출력

형식 지정자	의미	예	실행 결과
%d	10진 정수로 출력	printf("%d \n", 10);	10
%f	실수로 출력	printf("%f \n", 3.14);	3.14
%c	문자로 출력	printf("%c \n", 'a');	a
%s	문자열로 출력	printf("%s \n", "Hello");	Hello

변수값 출력[3/4]

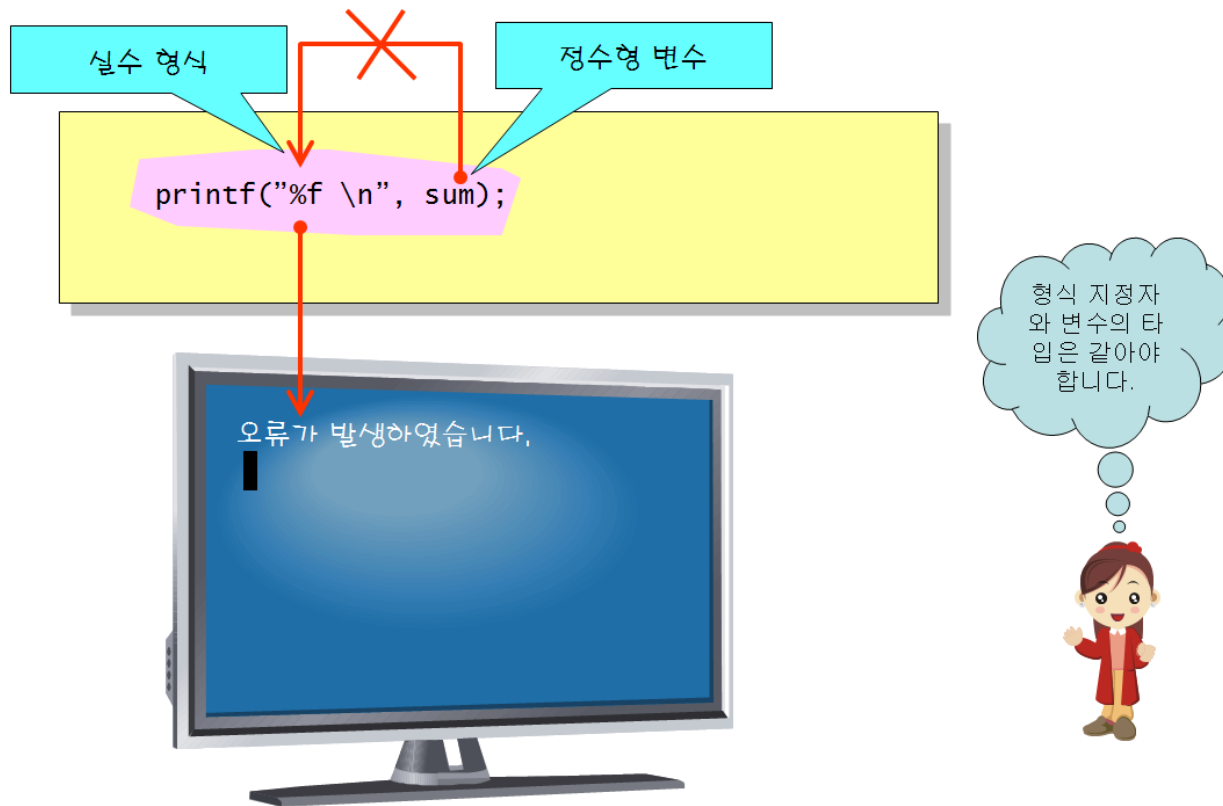
- ▶ 여러 개의 형식 지정자와 변수들 출력



변수값 출력 [4/4]

▶ 주의!

- 형식 지정자와 변수의 자료형은 반드시 일치해야 함



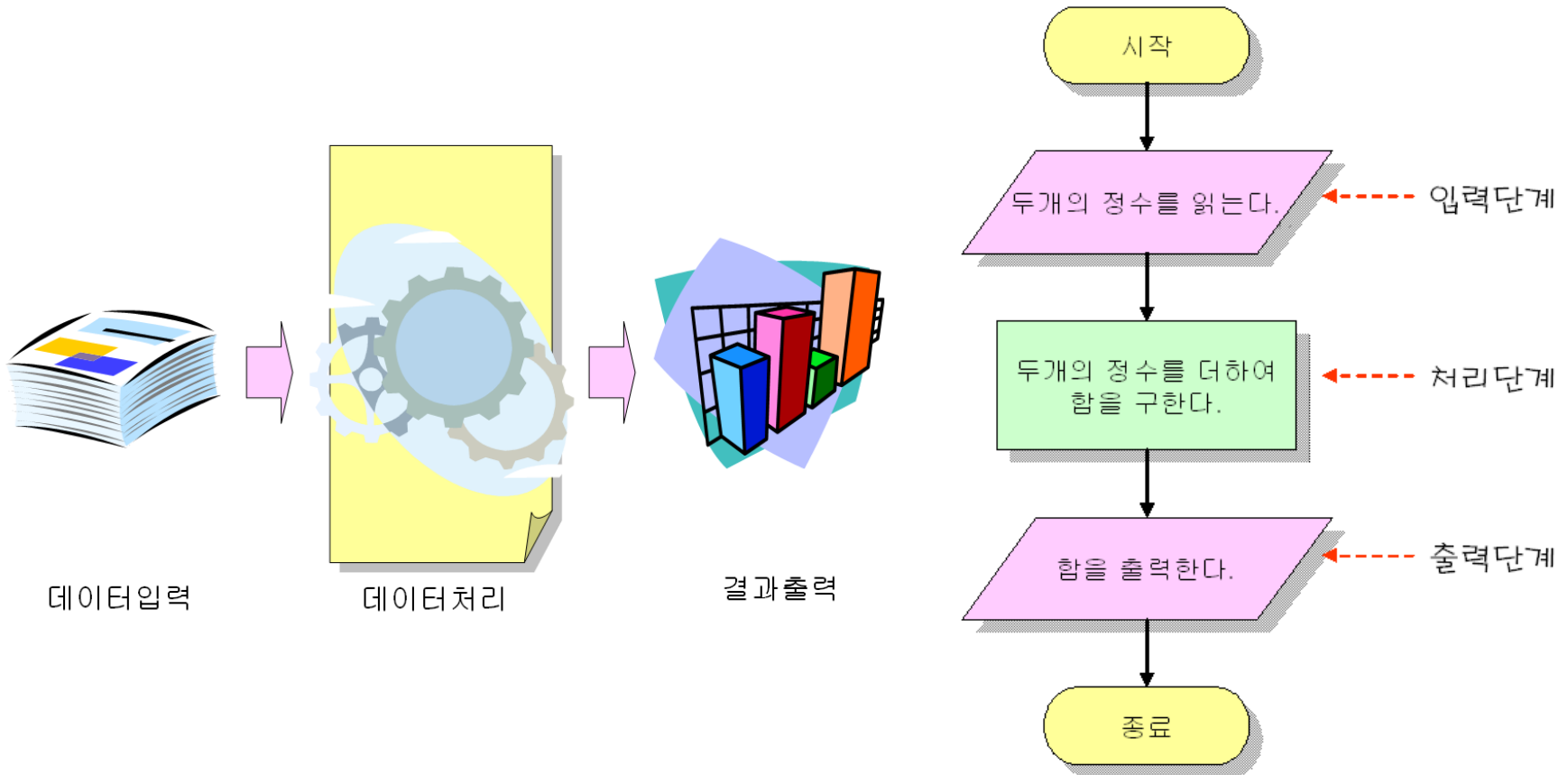


중간 점검

- ▶ printf()에서 변수의 값을 실수 형태로 출력할 때 사용하는 형식 지정자는 무엇인가?
- ▶ printf()를 사용하여 정수형 변수 k의 값을 출력하는 문장을 작성하여 보자.

덧셈 프로그램 #2

- ▶ 사용자로부터 입력을 받고 계산하는 프로그램 작성
- ▶ 소스 파일 이름 : add2.c



덧셈 프로그램 #2



```
// 사용자로부터 입력받은 2개의 정수의 합을 계산하여 출력
#include <stdio.h>

int main(void)
{
    int x;                // 첫번째 정수를 저장할 변수
    int y;                // 두번째 정수를 저장할 변수
    int sum;              // 2개의 정수의 합을 저장할 변수

    printf("첫번째 숫자를 입력하시오:");
    scanf("%d", &x);      // 입력 안내 메시지 출력
                        // 하나의 정수를 받아서 x에 저장

    printf("두번째 숫자를 입력하시오:");
    scanf("%d", &y);      // 입력 안내 메시지 출력
                        // 하나의 정수를 받아서 x에 저장

    sum = x + y;          // 변수 2개를 더한다.
    printf("두수의 합: %d", sum); // sum의 값을 10진수 형태로 출력

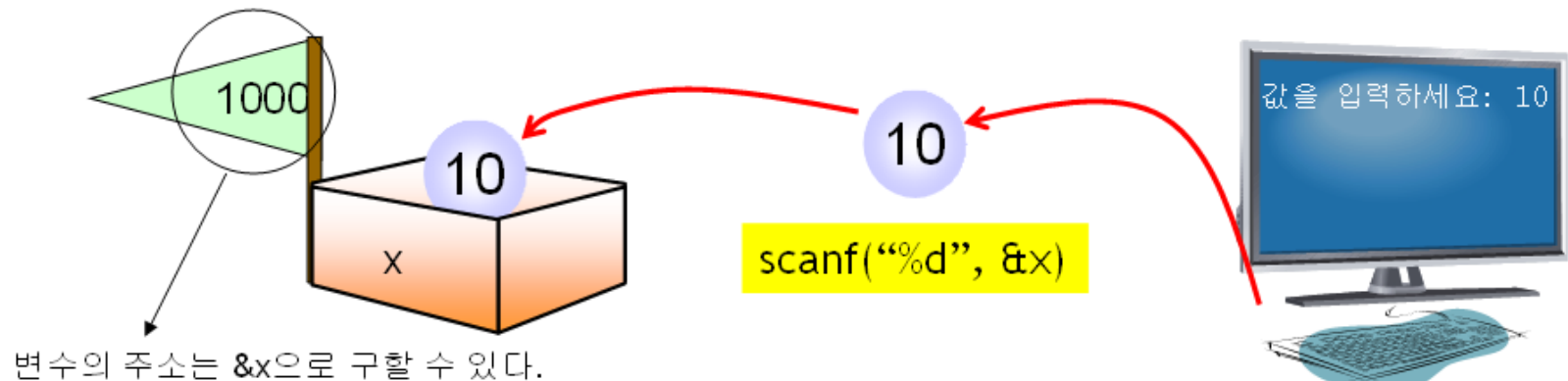
    return 0;            // 0을 외부로 반환
}
```

```
첫번째 숫자를 입력하시오:10
두번째 숫자를 입력하시오:20
두수의 합: 30
```



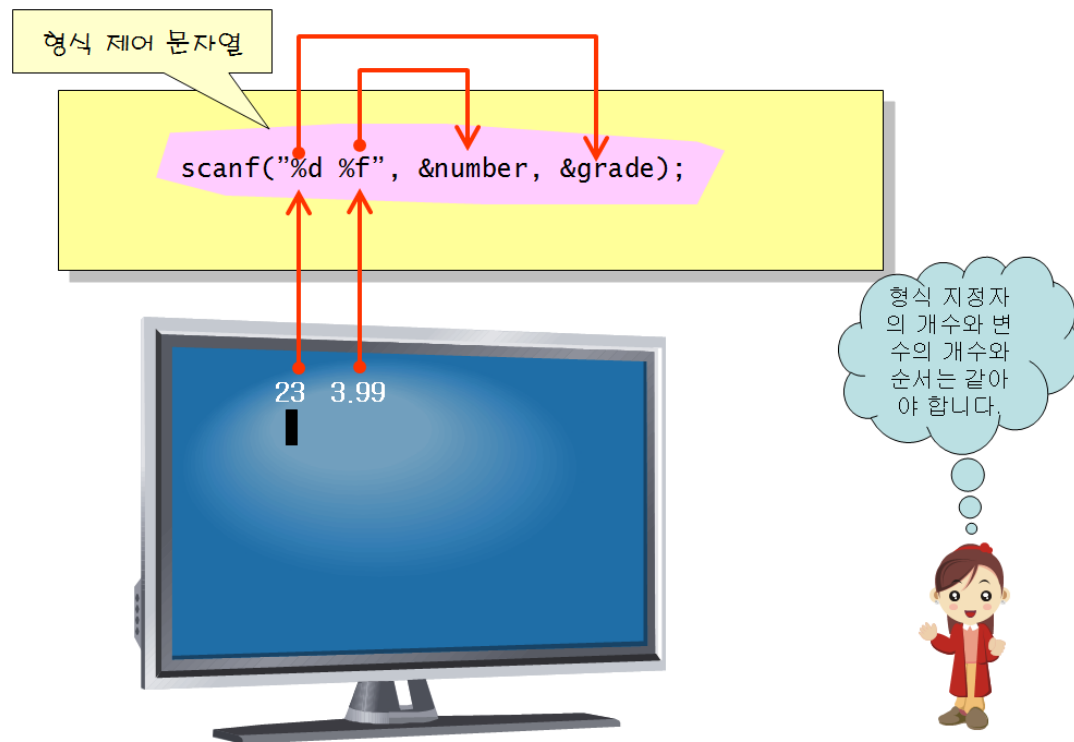
scanf() 함수[1/4]

- ▶ 키보드로부터 값을 받아서 변수에 저장
- ▶ 변수의 주소를 필요로 함
 - 변수의 주소는 변수 이름 앞에 &(앰퍼샌드) 기호를 붙여서 나타냄
- ▶ scanf()의 인수
 - "%d" : 형식 지정자로 정수형 데이터를 받을 것임을 지정
 - &x : 입력받은 정수형 데이터를 저장할 변수 x의 주소



scanf() 함수 [2/4]

- ▶ scanf()가 호출되면 컴퓨터는 사용자가 숫자를 입력할 때까지 대기
 - 숫자를 입력하고 엔터키를 누르면 정수가 변수에 저장되고 scanf() 종료
- ▶ 한번에 여러 변수에 값을 입력 받을 수 있음
 - 형식 지정자의 개수와 변수의 개수는 같아야 함



scanf() 함수 [3/4]

▶ 형식 지정자

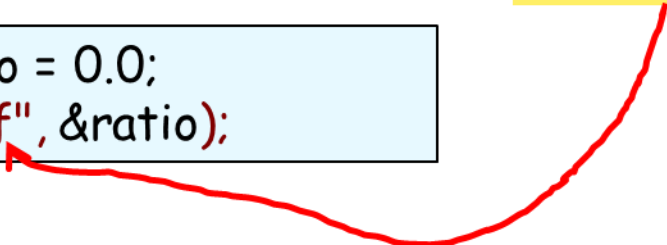
- scanf()의 형식 지정자는 대부분 printf()의 형식 지정자와 동일

형식 지정자	의미	예
%d	정수를 10진수로 입력한다	scanf("%d", &i);
%f	float 형의 실수로 입력한다.	scanf("%f", &f);
%lf	double 형의 실수로 입력한다.	scanf("%lf", &d);
%c	문자 형태로 입력한다.	scanf("%c", &ch);
%s	문자열 형태로 입력한다.	char s[10]; scanf("%s", &s);

scanf() 함수 [4/4]

- ▶ 실수형 입력 받을 때 float와 double 구분 필요

```
float ratio = 0.0;  
scanf("%f", &ratio);
```

- float 형은 %f 사용
- 

```
double scale = 0.0;  
scanf("%lf", &scale);
```

- double 형은 %lf 사용
- 



중간 점검

- ▶ scanf()를 사용하여 사용자로부터 실수값을 받아서 double형의 변수 value에 저장하는 문장을 작성하여 보자.

연봉 계산 프로그램(deposit.c)



```
/* 저축액을 계산하는 프로그램 */
```

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

```
int main(void)
```

```
{
```

```
    int salary; // 월급
```

```
    int deposit; // 저축액
```

```
    printf("월급을 입력하시오: ");  
    scanf("%d", &salary);
```

```
    deposit = 10 * 12 * salary;
```

```
    printf("10년 동안의 저축액: %d\n", deposit);
```

```
    return 0;
```

```
}
```

사용자로부터 월급을
입력받는다.


월급에 10*12를 곱하
여 10년동안의 저축액
을 계산한다.

결과를 출력한다.



```
월급을 입력하시오: 200  
10년 동안의 저축액: 24000
```

원의 면적 프로그램(circle.c)



```
/* 원의 면적을 계산하는 프로그램*/
```

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

```
int main(void)
```

```
{
```

```
    float radius;           // 원의 반지름
```

```
    float area;             // 면적
```

```
    printf("반지름을 입력하시오: ");
```

```
    scanf("%f", &radius);
```


```
    area = 3.14 * radius * radius;
```

```
    printf("원의 면적: %f\n", area);
```

```
    return 0;
```

```
}
```

원의 면적 계산



```
반지름을 입력하시오: 5.0  
원의 면적: 78.500000
```

환율 계산 프로그램(exchange_rate.c)



```
/* 환율을 계산하는 프로그램*/
#include <stdio.h>

int main(void)
{
    float rate;                // 원/달러 환율
    float usd;                 // 달러화
    int krw;                   // 원화

    printf("달러에 대한 원화 환율을 입력하시오: "); // 입력 안내 메시지
    scanf("%f", &rate); // 사용자로부터 환율입력

    printf("원화 금액을 입력하시오: "); // 입력 안내 메시지
    scanf("%d", &krw); // 원화 금액 입력

    usd = krw / rate; // 달러화로 환산

    printf("원화 %d원은 %f달러입니다.\n", krw, usd); // 계산 결과 출력

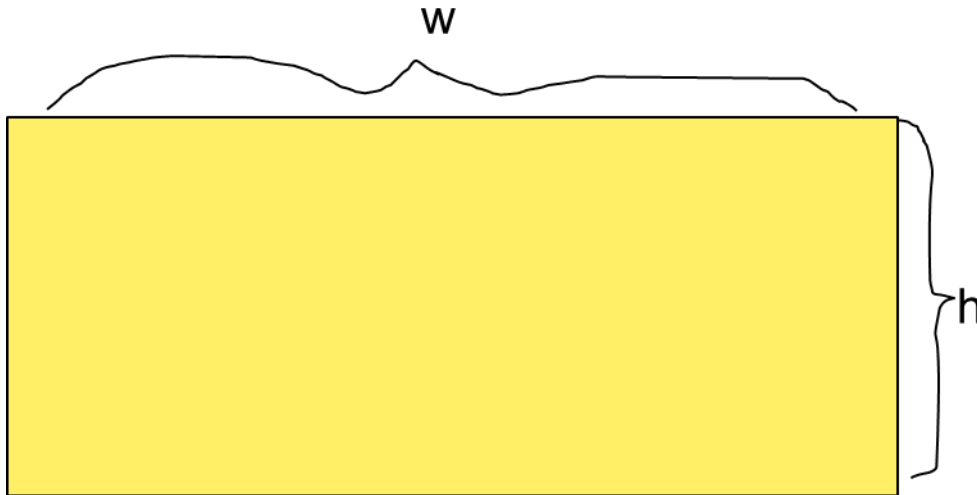
    return 0; // 함수 결과값 반환
}
```

```
달러에 대한 원화 환율을 입력하시오: 928.78
원화 금액을 입력하시오: 1000000
원화 1000000원은 1076.681204달러입니다.
```



실습: 사각형의 둘레와 면적[1/3]

- ▶ 필요한 변수는 w , h , $area$, $perimeter$ 라고 하자.
- ▶ 변수의 자료형은 실수를 저장할 수 있는 `double`형으로 하자.
- ▶ $area = w * h$;
- ▶ $perimeter = 2 * (w + h)$;



실습: 사각형의 둘레와 면적[2/3]

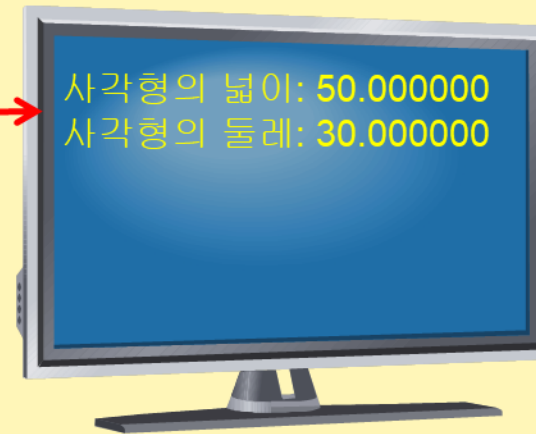
- ▶ Step 1 : 요구사항 분석
 - 사각형의 둘레와 면적을 계산하고 출력
- ▶ Step 2 : 알고리즘 기술
 - 자연어로 요구사항을 처리하기 위한 전체 알고리즘 기술
- ▶ Step 3 : 코딩
 - C언어 문법에 맞도록 프로그램 작성
- ▶ Step 4 : 컴파일과 링크
 - 솔루션 빌드를 통해 소스 파일을 기계어로 번역하고 필요한 라이브러리를 링크하여 실행 파일 생성
- ▶ Step 5 : 실행 및 디버깅
 - 실행 파일 실행 결과 확인 및 오류 수정

실습: 사각형의 둘레와 면적[3/3]

```
#include <stdio.h>
int main(void)
{
    double w;
    double h;
    double area;
    double perimeter;

    w = 10.0;
    h = 5.0;
    area = w*h;
    perimeter = 2*(w+h);

    printf("사각형의 넓이: %lf", area);
    printf("사각형의 둘레: %lf", perimeter);
    return 0;
}
```



현재 작성된 소스에서 논리적 오류는 무엇일까요?



도전문제

- ▶ 한번의 `printf()` 호출로 변수 `perimeter`와 `area`의 값이 동시에 출력되도록 변경하라.
- ▶ 변수들을 한 줄에 모두 선언하여 보자.
- ▶ `w`와 `h`의 값을 사용자로부터 받도록 변경하여 보자. `%lf`를 사용한다.

Q&A

