

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

5장. 수식과 연산자

성결대학교 컴퓨터공학부
임 상 순

강의 목표 및 내용

▶ 강의 목표

- 수식과 연산자의 개념을 이해한다.
- 대입, 산술, 증감, 관계, 논리 연산자를 사용할 수 있고 결과값을 이해할 수 있다.
- 연산자의 우선 순위와 결합 규칙을 이해한다.

▶ 내용

- 수식과 연산자의 개념
- 산술, 대입 연산자
- 형변환
- 관계, 논리 연산자
- 조건, 콤마, 비트 단위 연산자
- 연산자의 우선 순위와 결합 규칙

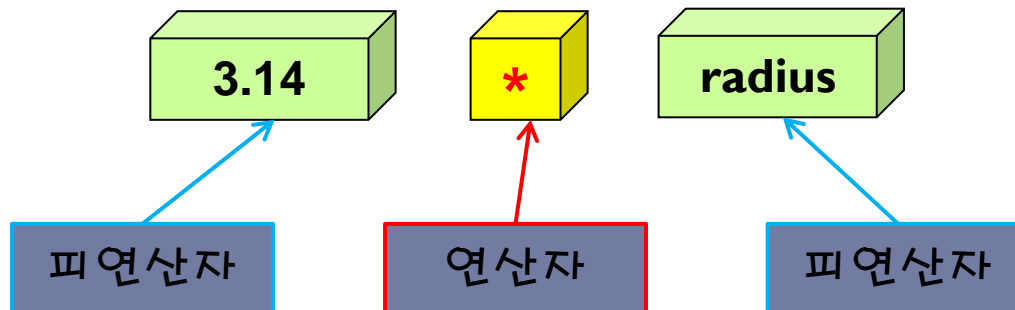
수식

- ▶ 수식(expression)
 - 상수, 변수, 연산자의 조합
 - 연산자와 피연산자로 나누어짐

$x + y$

$x * x + 5 * x + 6$

$(\text{principal} * \text{interest_rate} * \text{period}) / 12.0$



기능에 따른 연산자의 분류

연산자의 분류	연산자	의미
대입	=	오른쪽을 왼쪽에 대입
산술	+ - * / %	사칙연산과 나머지 연산
부호	+ -	
증감	++ --	증가, 감소 연산
관계	> < == != >= <=	오른쪽과 왼쪽을 비교
논리	&& !	논리적인 AND, OR, NOT
조건	?	조건에 따라 선택
coma	,	피연산자들을 순차적으로 실행
비트 단위 연산자	& ^ ~ << >>	비트별 AND, OR, XOR, 반전, 이동
sizeof 연산자	sizeof	자료형이나 변수의 크기를 바이트 단위로 반환
형변환	(type)	변수나 상수의 자료형을 변환
포인터 연산자	* & []	주소계산, 포인터가 가리키는 곳의 내용 추출
구조체 연산자	. ->	구조체의 멤버 참조

피연산자수에 따른 연산자 분류

- ▶ 단항 연산자: 피연산자의 수가 1개

```
++x;  
--y;
```

- ▶ 이항 연산자: 피연산자의 수가 2개

```
x + y  
x - y
```

- ▶ 삼항 연산자: 연산자의 수가 3개

```
x ? y : z
```

중간 점검

- ▶ 수식(expression)이란 어떻게 정의되는가?
- ▶ 상수 10도 수식이라고 할 수 있는가?
- ▶ 아래의 수식에서 피연산자와 연산자를 구분하여 보라.
 - $y = 10 + 20;$
- ▶ 연산자를 단항 연산자, 이항 연산자, 삼항 연산자로 나누는 기준은 무엇인가?

산술 연산자

▶ 산술 연산

- 컴퓨터의 가장 기본적인 연산
- 덧셈, 뺄셈, 곱셈, 나눗셈 등의 사칙 연산을 수행하는 연산자

연산자	기호	의미
덧셈	$x+y$	x와 y를 더한다
뺄셈	$x-y$	x에서 y를 뺀다.
곱셈	$x*y$	x와 y를 곱한다.
나눗셈	x/y	x를 y로 나눈다.
나머지	$x\%y$	x를 y로 나눌 때의 나머지값

산술 연산자의 예

$$y = mx + b \quad \rightarrow \quad y = m * x + b$$

$$y = ax^2 + bx + c \quad \rightarrow \quad y = a * x * x + b * x + c$$

$$m = \frac{x + y + x}{3} \quad \rightarrow \quad m = (x + y + z) / 3$$



(참고) 거듭 제곱 연산자는?

C에는 거듭 제곱을 나타내는 연산자는 없다.
 $x * x$ 와 같이 단순히 변수를 두 번 곱한다.

예제 : 산술 연산자(arithmetic.c)

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

```
int main()
```

```
{
```

```
    int x, y, result;
```

```
    printf("두개의 정수를 입력하시오: ");
```

```
    scanf("%d %d", &x, &y);
```

```
    result = x + y;
```

```
    printf("%d + %d = %d\\n", x, y, result);
```

```
    result = x - y;           // 뺄셈
```

```
    printf("%d - %d = %d\\n", x, y, result);
```

```
    result = x * y;           // 곱셈
```

```
    printf("%d * %d = %d\\n", x, y, result);
```

```
    result = x / y;           // 나눗셈
```

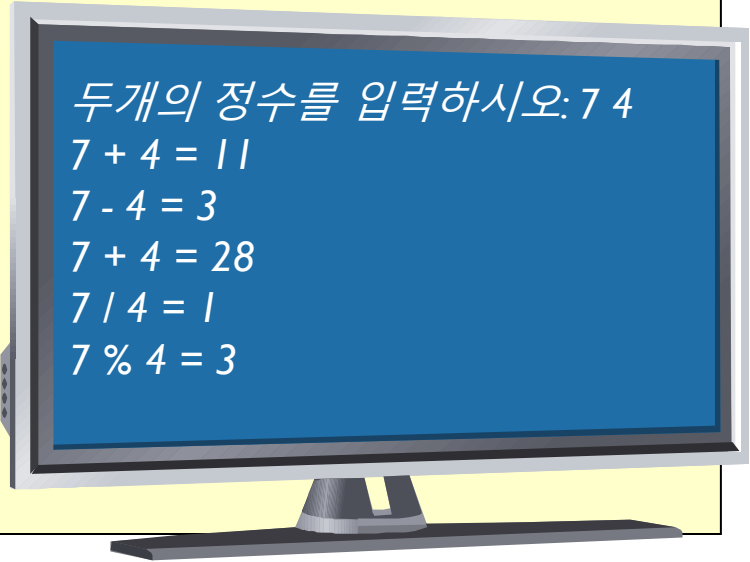
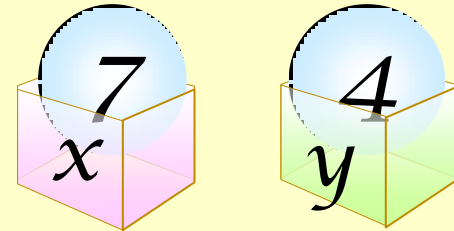
```
    printf("%d / %d = %d\\n", x, y, result);
```

```
    result = x % y;           // 나머지
```

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

```
    return 0;
```

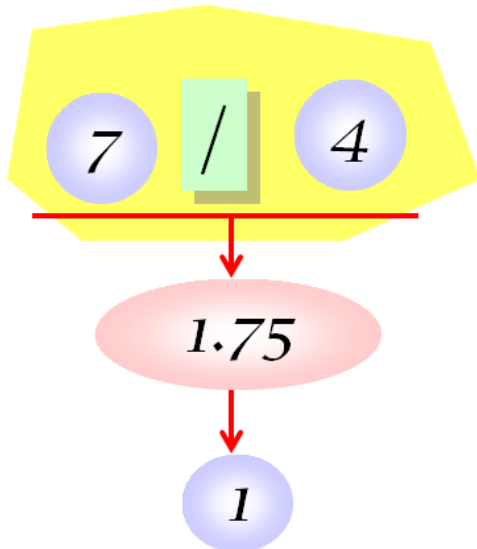
```
}
```



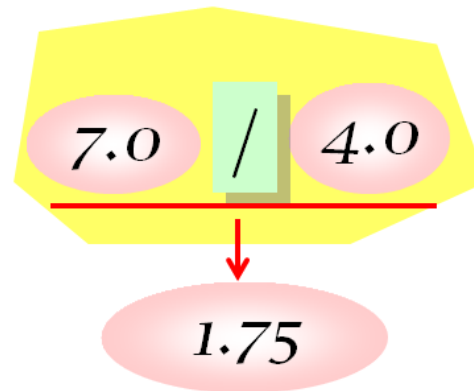
```
두개의 정수를 입력하시오: 7 4  
7 + 4 = 11  
7 - 4 = 3  
7 * 4 = 28  
7 / 4 = 1  
7 % 4 = 3
```

나눗셈 연산자

- ▶ 정수형끼리의 나눗셈
 - 결과가 정수형으로 생성
 - 소수점 이하는 버려짐
- ▶ 부동 소수점형끼리의 나눗셈
 - 부동 소수점형끼리는 부동 소수점 값 생성



정수와 정수 끼리의 나눗셈.



실수와 실수 끼리의 나눗셈.

형변환에서
자세히 학
습합니다.



예제 : 나눗셈 연산자(arithmetic1.c)

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

```
int main()
```

```
{
```

```
    double x, y, result;
```

```
    printf("두개의 실수를 입력하시오: ");
```

```
    scanf("%lf %lf", &x, &y);
```

```
    result = x + y;           // 덧셈 연산을 하여서 결과를 result에 대입
```

```
    printf("%f / %f = %f", x, y, result);
```


```
    ...
```

```
    result = x / y;
```

```
    printf("%f / %f = %f", x, y, result);
```

```
    return 0;
```

```
}
```



두개의 실수를 입력하시오: 7 4
7.000000 + 4.000000 = 11.000000
7.000000 - 4.000000 = 3.000000
7.000000 * 4.000000 = 28.000000
7.000000 / 4.000000 = 1.750000

나머지 연산자

- ▶ 나머지 연산자(modulus operator)
 - 첫 번째 피연산자를 두 번째 피연산자로 나누었을 경우의 나머지를 계산
 - ▶ $10 \% 2$ 는 0이다.
 - ▶ $5 \% 7$ 는 5이다.
 - ▶ $30 \% 9$ 는 3이다.
- ▶ 나머지 연산자를 이용한 짝수와 홀수를 구분
 - $x \% 2$ 가 0이면 짝수
- ▶ 나머지 연산자를 이용한 5의 배수 판단
 - $x \% 5$ 가 0이면 5의 배수

아주
유용한 연
산자 입니
다.



예제 : 나머지 연산자(modulo.c)

```
// 나머지 연산자 프로그램
```

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

```
#define SEC_PER_MINUTE 60 // 1분은 60초
```

```
int main(void)
```

```
{
```

```
    int input, minute, second;
```

```
    printf("초단위의 시간을 입력하시요:(32억초이하) ");
```

```
    scanf("%d", &input);        // 초단위의 시간을 읽는다.
```

```
    minute = input / SEC_PER_MINUTE; // 몇 분
```

```
    second = input % SEC_PER_MINUTE; // 몇 초
```

```
    printf("%d초는 %d분 %d초입니다. \n", input, minute, second);
```

```
    return 0;
```

```
}
```

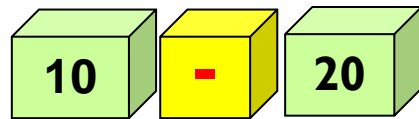


초단위의 시간을 입력하시요:(32억초이하) 70
70초는 1분 10초 입니다.

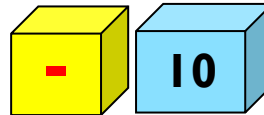
부호 연산자

▶ 변수나 상수의 부호를 변경

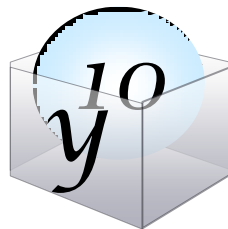
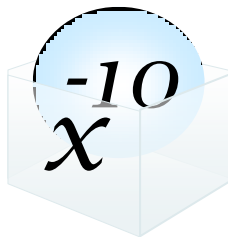
```
x = -10;  
y = -x; // 변수 y의 값은 10이 된다.
```



이항연산자



단항연산자



-는 이항 연산
자이기도 하고
단항 연산자이
기도 하죠

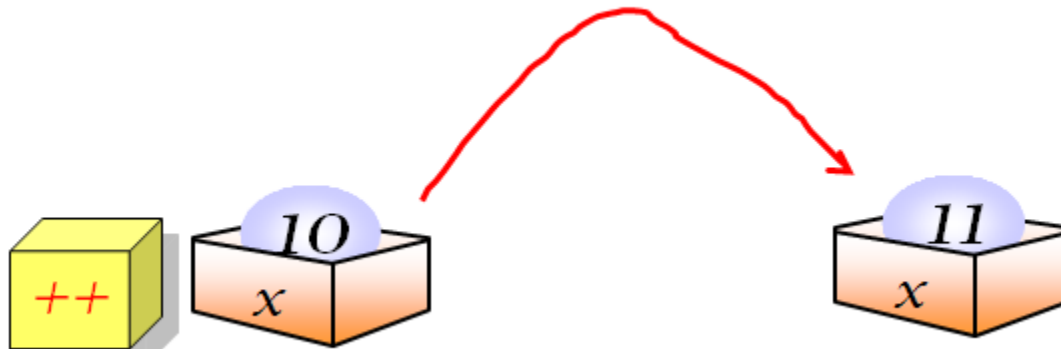


증감 연산자

▶ 증감 연산자

- ++, --
- 변수의 값을 하나 증가시키거나 감소시키는 연산자

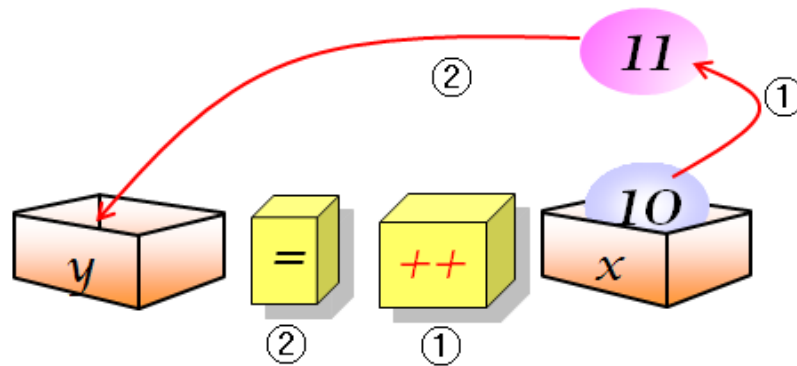
++x는 변수 x의 값을 하나 증가시킨다.



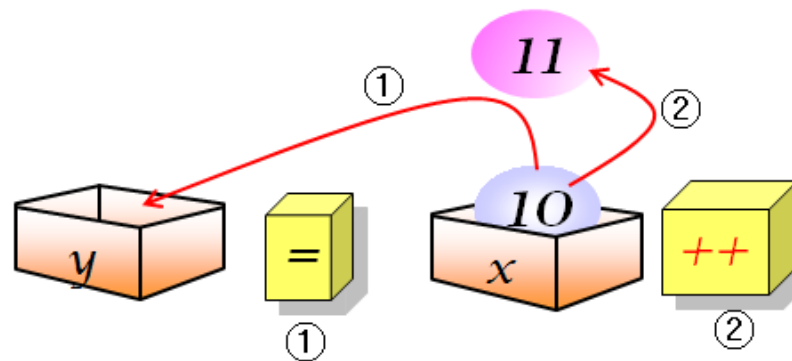
MiniScience.com

++x와 x++의 차이

▶ ++x와 x++는 어떤 차이가 있을까?



증가된 x의 값이 y에 대입된다.



증가되지 않은 x의 값이 y에 대입된다.

증감 연산자 정리

증감 연산자	의미
<code>++x</code>	수식의 값은 증가된 x값이다.
<code>x++</code>	수식의 값은 증가되지 않은 원래의 x값이다.
<code>--x</code>	수식의 값은 감소된 x값이다.
<code>x--</code>	수식의 값은 감소되지 않은 원래의 x값이다.

Quiz

▶ nextx와 nexty의 값은?

```
x = 1;  
y = 1;  
  
nextx = ++x;  
nexty = y++;
```



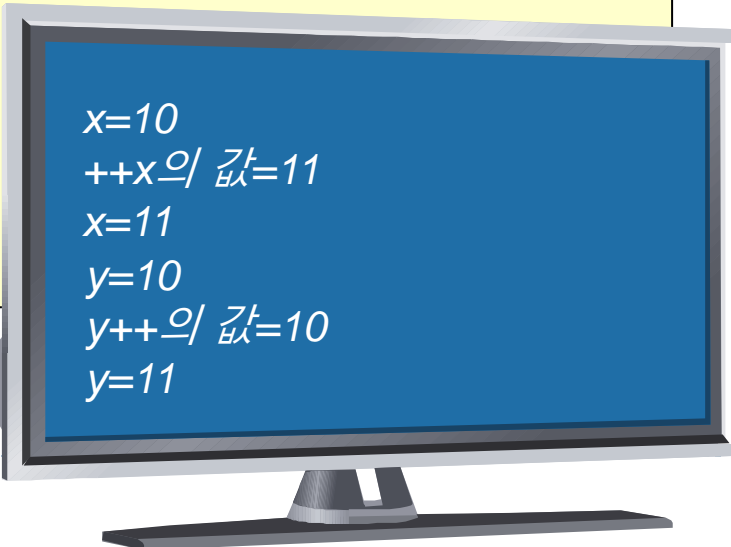
예제 : 증감 연산자(incdec.c)

```
#include <stdio.h>
int main(void)
{
    int x=10, y=10;

    printf("x=%d\n", x);
    printf("++x의 값=%d\n", ++x);
    printf("x=%d\n\n", x);

    printf("y=%d\n", y);
    printf("y++의 값=%d\n", y++);
    printf("y=%d\n", y);

    return 0;
}
```



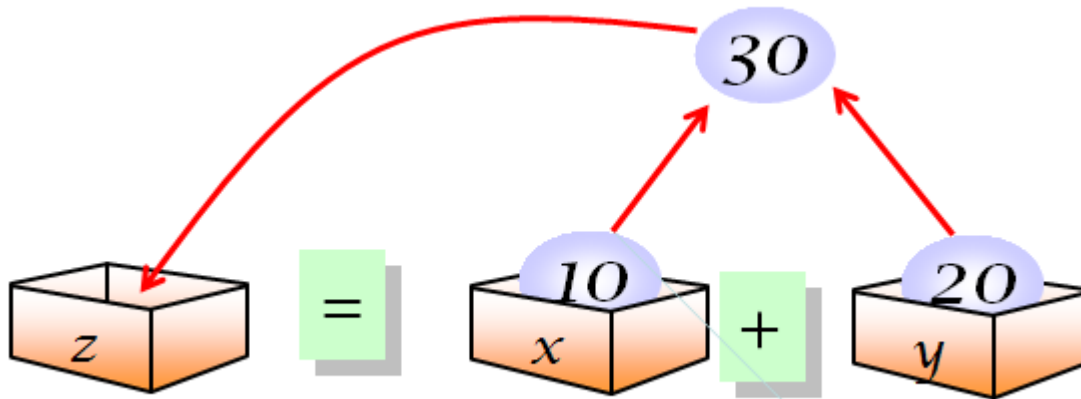
x=10
++x의 값=11
x=11
y=10
y++의 값=10
y=11

대입(배정, 할당) 연산자

- ▶ 왼쪽에 있는 변수에 오른쪽의 수식의 값을 계산하여 대입

변수(variable) = 수식(expression);

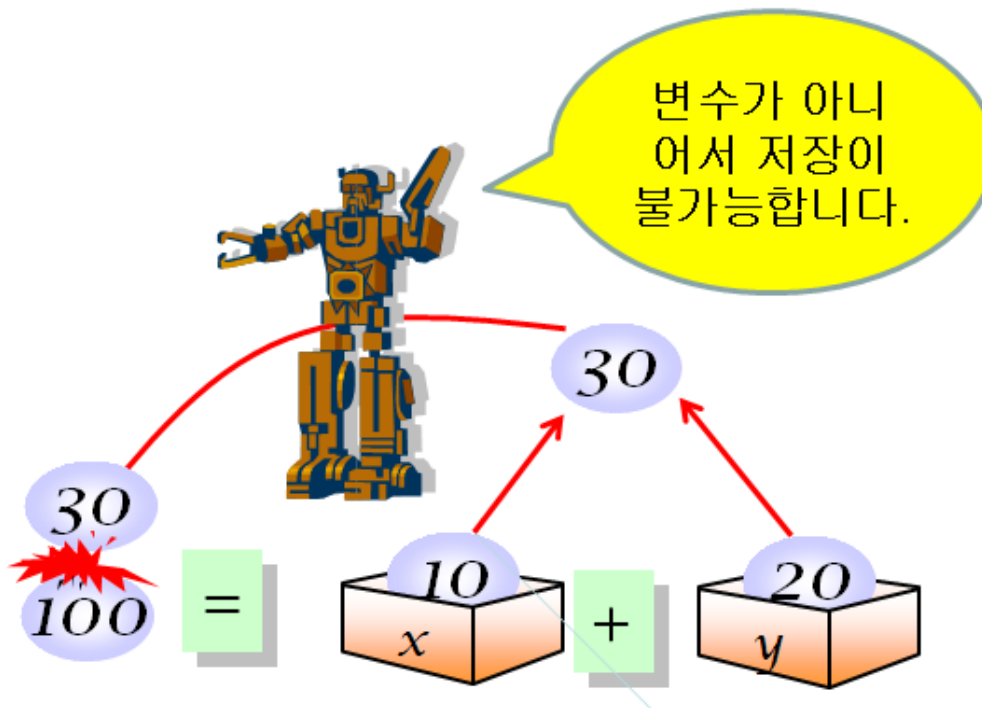
- ▶ (예) $z = x + y;$



대입 연산자 주의점 [1/2]

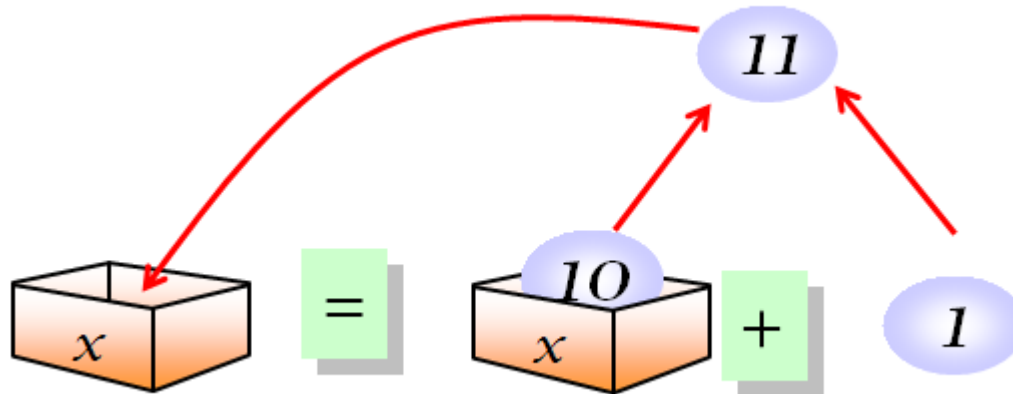
▶ 등호의 왼편에는 변수가 와야함

- $100 = x + y;$ // 컴파일 오류!



대입 연산자 주의점 [2/2]

- ▶ $x = x + 1;$
 - 변수의 값을 1 만큼 증가



대입 연산의 결과값

$$y = 10 + (x = 2 + 7);$$

Diagram illustrating the evaluation of the expression $y = 10 + (x = 2 + 7);$ using operator precedence (parentheses first, then assignment, then addition).

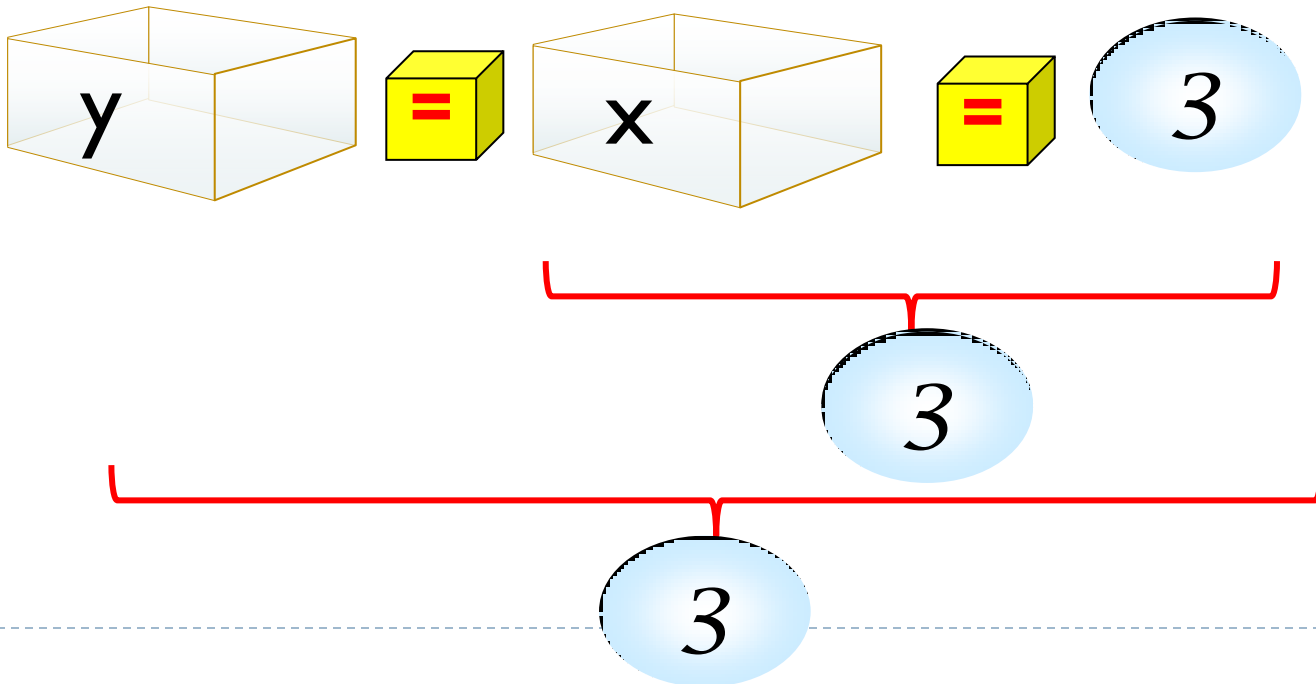
- The innermost expression $2 + 7$ is evaluated first (Addition), resulting in 9. (Label: 덧셈연산의 결과값은 9)
- The assignment operation $x = 9$ is then performed (Assignment), resulting in 9. (Label: 대입연산의 결과값은 9)
- The final addition $10 + 9$ is performed (Addition), resulting in 19. (Label: 덧셈연산의 결과값은 19)
- The overall expression $y = 19$ is the final result of the assignment operation. (Label: 대입연산의 결과값은 19)

모든 연산에는
결과값이 있고
대입 연산도
결과값이 있습니다.



여러 변수에 같은 값을 대입

$y = x = 3;$



예제 : 대입 연산자(assignment.c)

```
/* 대입 연산자 프로그램 */
```

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

```
int main(void)
```

```
{
```

```
    int x, y;
```

```
    x = 1;
```

```
    printf("수식 x+1의 값은 %d\n", x+1);
```

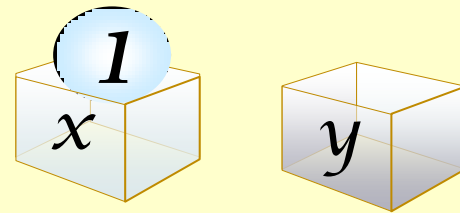
```
    printf("수식 y=x+1의 값은 %d\n", y=x+1);
```

```
    printf("수식 y=10+(x=2+7)의 값은 %d\n", y=10+(x=2+7));
```

```
    printf("수식 y=x=3의 값은 %d\n", y=x=3);
```

```
    return 0;
```

```
}
```



수식 x+1의 값은 2

수식 y=x+1의 값은 2

수식 y=10+(x=2+7)의 값은 19

수식 y=x=3의 값은 3

복합 대입 연산자[1/2]

▶ 복합 대입 연산자

- +=처럼 대입연산자 =와 산술연산자를 합쳐 놓은 연산자
- 소스를 간결하게 만들 수 있음

$x += y$

$x = x + y$ 와 의미가 같음!



복합 대입 연산자[2/2]

복합 대입 연산자	의미
$x += y$	$x = x + y$
$x -= y$	$x = x - y$
$x *= y$	$x = x * y$
$x /= y$	$x = x / y$
$x \% = y$	$x = x \% y$
$x \& = y$	$x = x \& y$
$x = y$	$x = x y$
$x \wedge = y$	$x = x \wedge y$
$x >> = y$	$x = x >> y$
$x << = y$	$x = x << y$

Quiz

- ▶ 다음 수식을 풀어서 다시 작성하면?

$x *= y + 1$
 $x \% = x + y$

$x = x * (y + 1)$
 $x = x \% (x + y)$



예제 : 복합 대입 연산자(abbr.c)

```
// 복합 대입 연산자 프로그램
```

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

```
int main(void)
```

```
{
```

```
    int x = 10, y = 10, z = 33;
```

```
    x += 1;
```

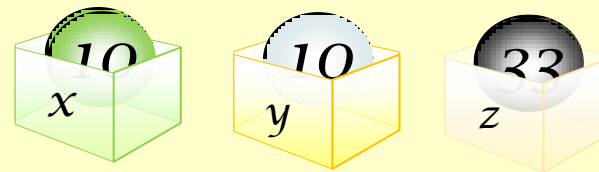
```
    y *= 2;
```

```
    z %= 10 + 20;
```

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

```
    return 0;
```

```
}
```



x = 11 y = 20 z = 3