2008 Spring

Computer Engineering Programming 1

Lesson 4
- 제5장 수식과 연산자

Lecturer: JUNBEOM YOO
jbyoo@konkuk.ac.kr

※ 본 강의자료는 생능출판사의 “PPT 강의자료”를 기반으로 제작되었습니다.

이번 장에서 학습할 내용

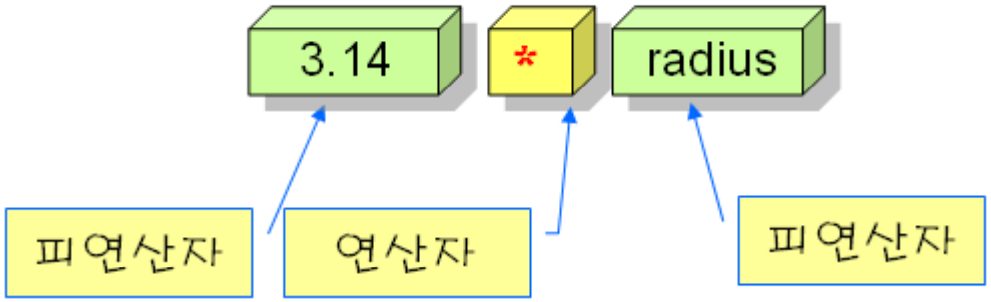
* 수식과 연산자란?
* 대입 연산
* 산술 연산
* 논리 연산
* 관계 연산
* 우선 순위와 결합 법칙

수식

- 수식(expression)

```
x + y
x*x + 5*x + 6
(principal * interest_rate * period) / 12.0
```

  - 상수, 변수, 연산자의 조합
  - 연산자와 피연산자로 나누어진다.



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
조건	?	조건에 따라 선택
콤마	,	피연산자들을 순차적으로 실행
비트 단위 연산자	& | ^ ~ << >>	비트별 AND, OR, XOR, 이동, 반전
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

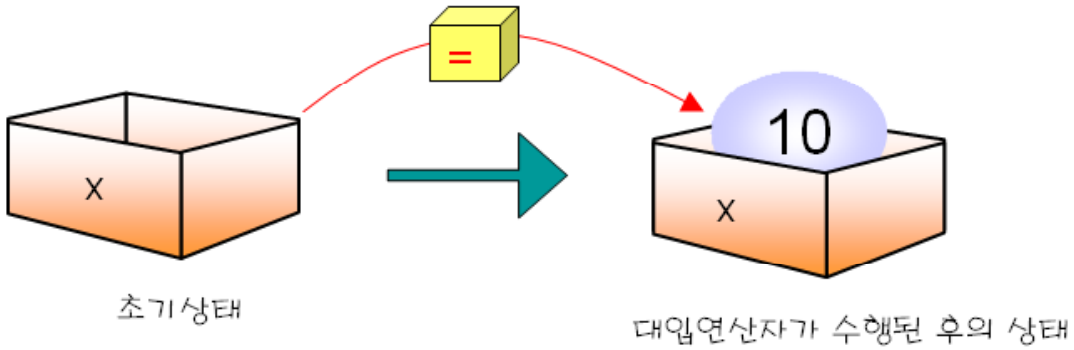
```
x ? y : z
```

대입(배정, 할당) 연산자

- 왼쪽에 있는 변수에 오른쪽의 수식의 값을 계산하여 대입

```
변수(variable)    =    수식(expression);
```

```
x = 10;          // 상수 10을 변수 x에 대입한다.
y = x;           // 변수 x의 값을 변수 y에 대입한다.
z = 2 * x + y;   // 수식 2 * x + y를 계산하여 변수 z에 대입한다.
```



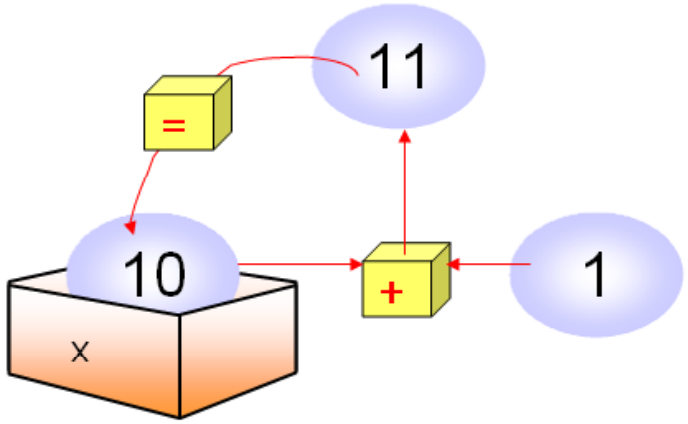
대입 연산자 주의점

- 왼쪽에는 항상 변수가 와야 한다.

```
x + 2 = 0;          // 왼편이 변수이름이 아니기 때문에 잘못된 수식!!
2 = x;              // 왼편이 변수이름이 아니기 때문에 잘못된 수식!!
```

- 다음의 문장은 수학적으로는 올바르지 않지만 C에서는 가능

```
x = x + 1;          // x의 값이 하나 증가 된다.
```



대입 연산의 결과값

덧셈연산의 결과값은 9

```
x = 2 + 7;
```

대입연산의 결과값은 9 (현재는 사용되지 않음)

덧셈연산의 결과값은 9

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

대입연산의 결과값은 9

덧셈연산의 결과값은 19

대입연산의 결과값은 19 (현재는 사용되지 않음)

예제

```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
```

산술 연산자

- 덧셈, 뺄셈, 곱셈, 나눗셈 등의 사칙 연산을 수행하는 연산자

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

$y = mx + b$

$y = ax^2 + bx + c$

$m = \frac{x + y + x}{3}$

```
y = m*x + b
y = a*x*x + b*x +c
m = (x + y + z) / 3
```

(참고) 거듭 제곱 연산자는?

C에는 거듭 제곱을 나타내는 연산자는 없다.

x * x와 같이 단순히 변수를 두번 곱한다.

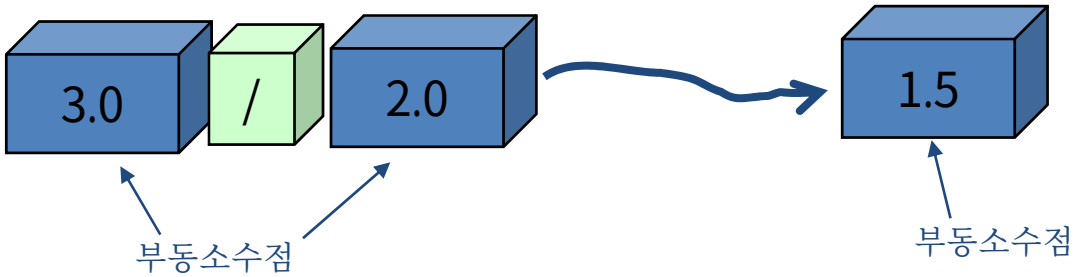
예제

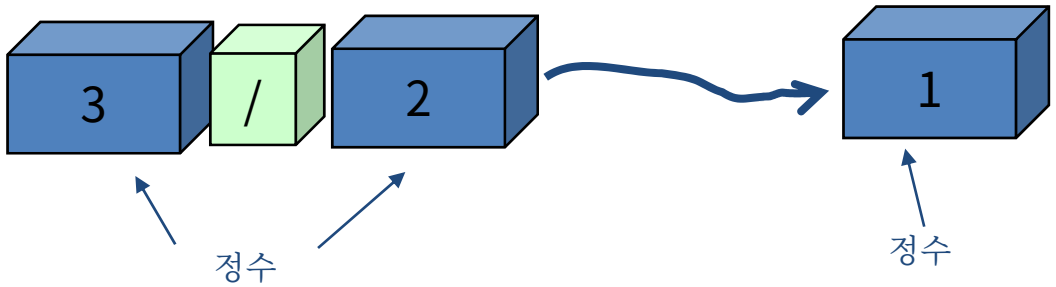
```c
// 산술 연산자를 이용한 프로그램
#include <stdio.h>
int main(void)
{
	int a,b;

	printf("첫번째 연산자=");
	scanf("%d", &a);
	printf("두번째 연산자=");
	scanf("%d", &b);

	printf("%d + %d 은 %d\n", a, b, a+b);
	printf("%d - %d 은 %d\n", a, b, a-b);
	printf("%d * %d 은 %d\n", a, b, a*b);
	printf("%d / %d 은 %d\n", a, b, a/b);
	printf("%d %% %d 은 %d\n", a, b, a%b);
}
```

```
첫번째 연산자=2
두번째 연산자=3
2 + 3 은 5
2 - 3 은 -1
2 * 3 은 6
2 / 3 은 0
2 % 3 은 2
```

나눗셈 연산자

- 정수형끼리의 나눗셈에서는 결과가 정수형으로 생성하고
- 부동소수점형끼리는 부동소수점 값을 생성된다.
- 정수형끼리의 나눗셈에서는 소수점 이하는 버려진다.





나눗셈 연산자

```c
// 나눗셈연산자프로그램
#include <stdio.h>

int main(void)
{
	printf("3/2 = %d \n", 3/2);			// 정수 / 정수
	printf("4/2 = %d \n", 4/2);
	printf("5/2 = %d \n", 5/2);
	printf("3.0/2.0 = %f \n", 3.0/2.0);		// 부동 소수점 / 부동 소수점
	printf("4.0/2.0 = %f \n", 4.0/2.0);
	printf("5.0/2.0 = %f \n", 5.0/2.0);
	printf("3.0/2 = %f \n", 3.0/2);		// 부동 소수점 / 정수
	return 0;
}
```

```
3/2 = 1
4/2 = 2
5/2 = 2
3.0/2.0 = 1.500000
4.0/2.0 = 2.000000
5.0/2.0 = 2.500000
3.0/2 = 1.500000
```

나머지 연산자

- 나머지 연산자(modulus operator)는 첫 번째 피연산자를 두 번째 피연산자로 나누었을 경우의 나머지를 계산
  - 10 % 2는 0이다.
  - 5 % 7는 5이다.
  - 30 % 9는 3이다.

- 나머지 연산자를 이용한 짝수와 홀수를 구분
  - x % 2가 0이면 짝수

- 나머지 연산자를 이용한 5의 배수 판단
  - x % 5가 0이면 5의 배수

나머지 연산자

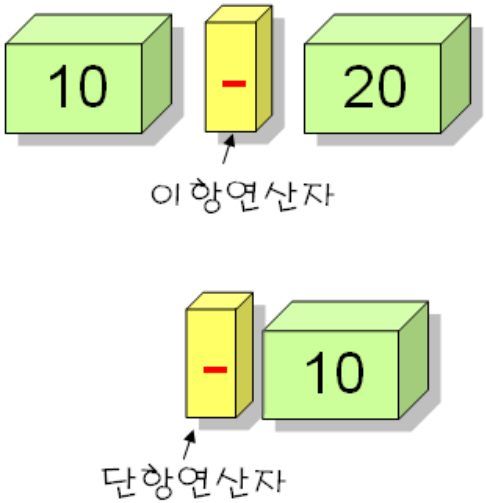
```c
// 나머지 연산자 프로그램
#include <stdio.h>
#define SEC_PER_MINUTE 60  // 1분은 60초

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 10초입니다.
```

부호 연산자

- 변수나 상수의 부호를 변경

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



복합 대입 연산자

- 복합 대입 연산자란 +=처럼 대입연산자 =와 산술연산자를 합쳐 놓은 연산자
- 소스를 간결한게 만들 수 있음

```
x += 1          // x = x + 1
x *= 5          // x = x * 5
x -= y + 1      // x = x - (y + 1)
x *= y + 1      // x = x * (y + 1)
x += y / z      // x = x + y / z
x %= x + y      // x = x % (x + y)
```

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 ^= y	x = x ^ y
x >>= y	x = x >> y
x <<= y	x = x << y

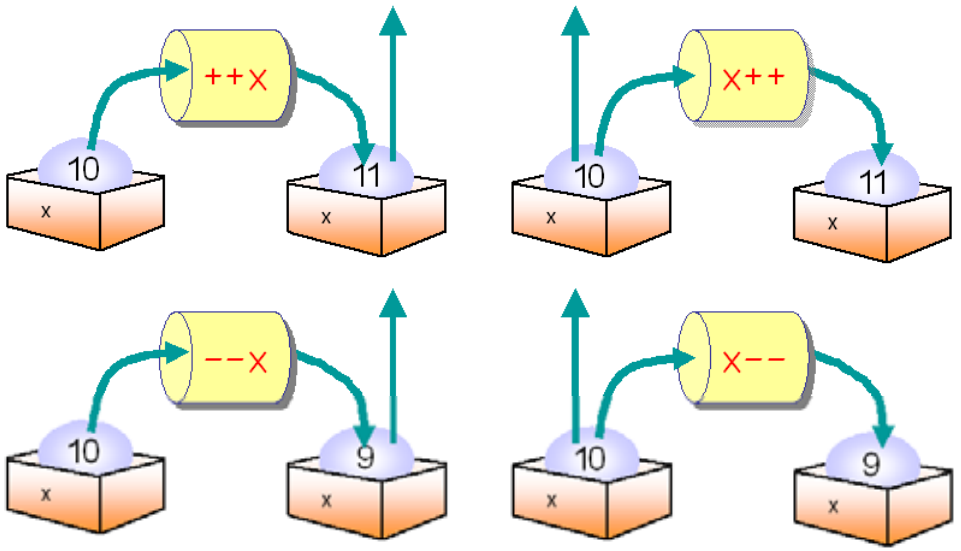
복합 대입 연산자

```c
// 복합 대입 연산자 프로그램
#include <stdio.h>

int main(void)
{
	int x = 10, y = 10, z = 33;

	x += 1;		// x = x + 1;
	y *= 2;		// y = y * 2;
	z %= x + y;	// z = z % (x + y );  주의!!

	printf("x = %d	y = %d	z = %d \n", x, y, z);
	return 0;
}
```

```
x = 11	y = 20	z = 2
```

증감 연산자

증감 연산자	의미
++x	x값을 먼저 증가한 후에 다른 연산에 사용한다. 이 수식의 값은 증가된 x값이다.
x++	x값을 먼저 사용한 후에, 증가한다. 이 수식의 값은 증가되지 않은 원래의 x값이다.
--x	x값을 먼저 감소한 후에 다른 연산에 사용한다. 이 수식의 값은 감소된 x값이다.
x--	x값을 먼저 사용한 후에, 감소한다. 이 수식의 값은 감소되지 않은 원래의 x값이다.



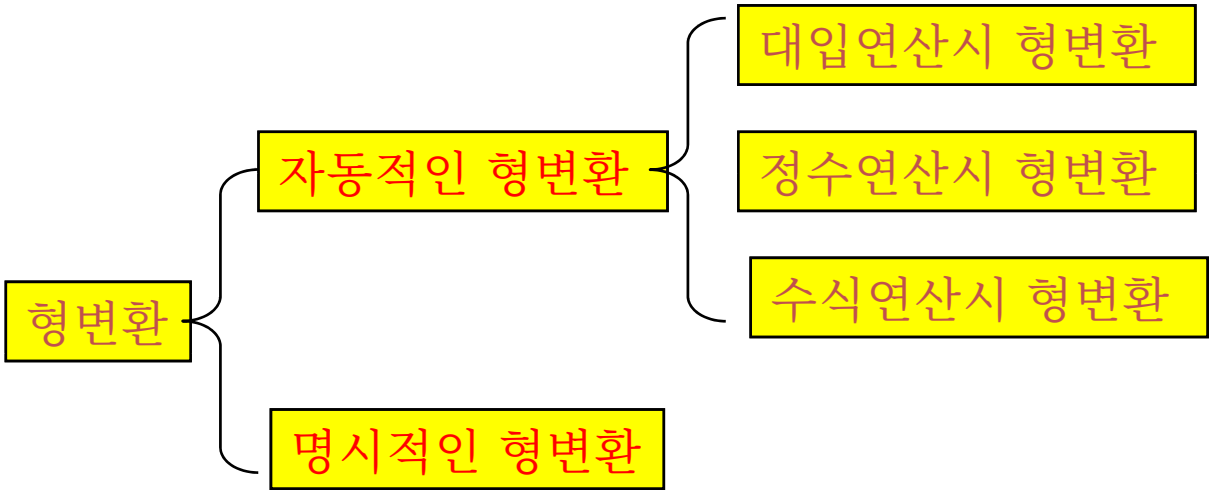
증감 연산자

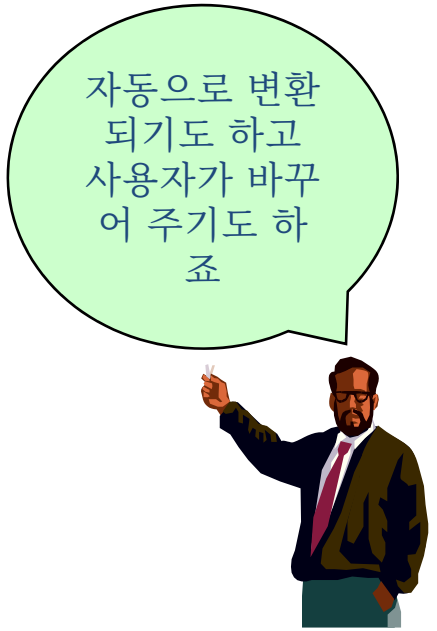
```c
// 증감연산자를 이용한 프로그램
#include <stdio.h>

int main(void)
{
	int x = 10;

	printf("수식 x++ 의 값: %d \n", x++);
	printf("현재 x의 값: %d \n", x);
	printf("수식 ++x 의 값: %d \n", ++x);
	printf("현재 x의 값: %d \n", x);

	printf("수식 x-- 의 값: %d \n", x--);
	printf("현재 x의 값: %d \n", x);
	printf("수식 --x 의 값: %d \n", --x);
	printf("현재 x의 값: %d \n", x);
}
```

```
수식 x++ 의 값: 10
현재 x의 값: 11
수식 ++x 의 값: 12
현재 x의 값: 12
수식 x-- 의 값: 12
현재 x의 값: 11
수식 --x 의 값: 10
현재 x의 값: 10
```

형변환

- 연산시에 데이터의 유형이 변환되는 것





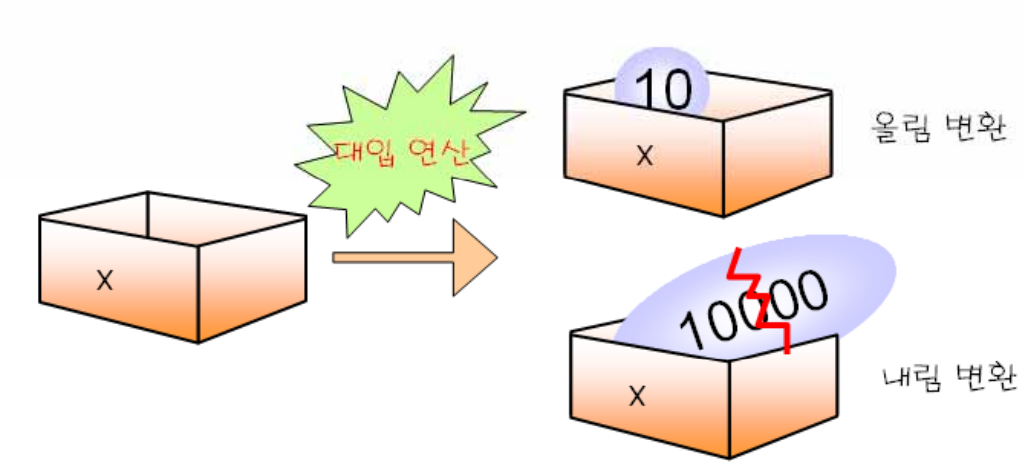
대입 연산시의 자동적인 형변환

- 올림 변환

```
double f;
f = 10 + 20;            // f에는 30.0이 저장된다.
```

- 내림 변환

```
int i;
i = 3.141592;           // i에는 3이 저장된다.
```



올림 변환과 내림 변환

```c
#include <stdio.h>
int main(void)
{
	char c;
	int i;
	float f;

	c = 10000;		// 내림 변환
	i = 1.23456 + 10;	// 내림 변환
	f = 10 + 20;		// 올림 변환
	printf("c = %d, i = %d, f = %f \n", c, i, f);
	return 0;
}
```

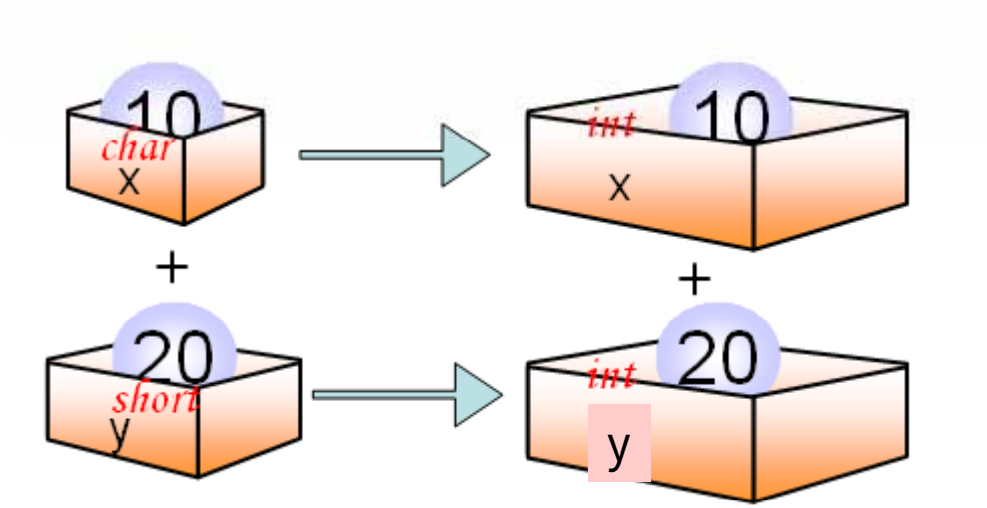
C: | CPROGRAM | convert1 | convert1.c(10) : warning C4305: '=' : truncation from 'const int ' to 'char '
C: | CPROGRAM | convert1 | convert1.c(11) : warning C4244: '=' : conversion from 'const double ' to 'int ', possible loss of data

c = 16, i = 11, f = 30.000000

정수 연산시의 자동적인 형변환

- 정수 연산시 char형이나 short형의 경우, 자동적으로 int형으로 변환하여 계산한다.

```
char x = 10;
short y = 20;
z = x + y;
```



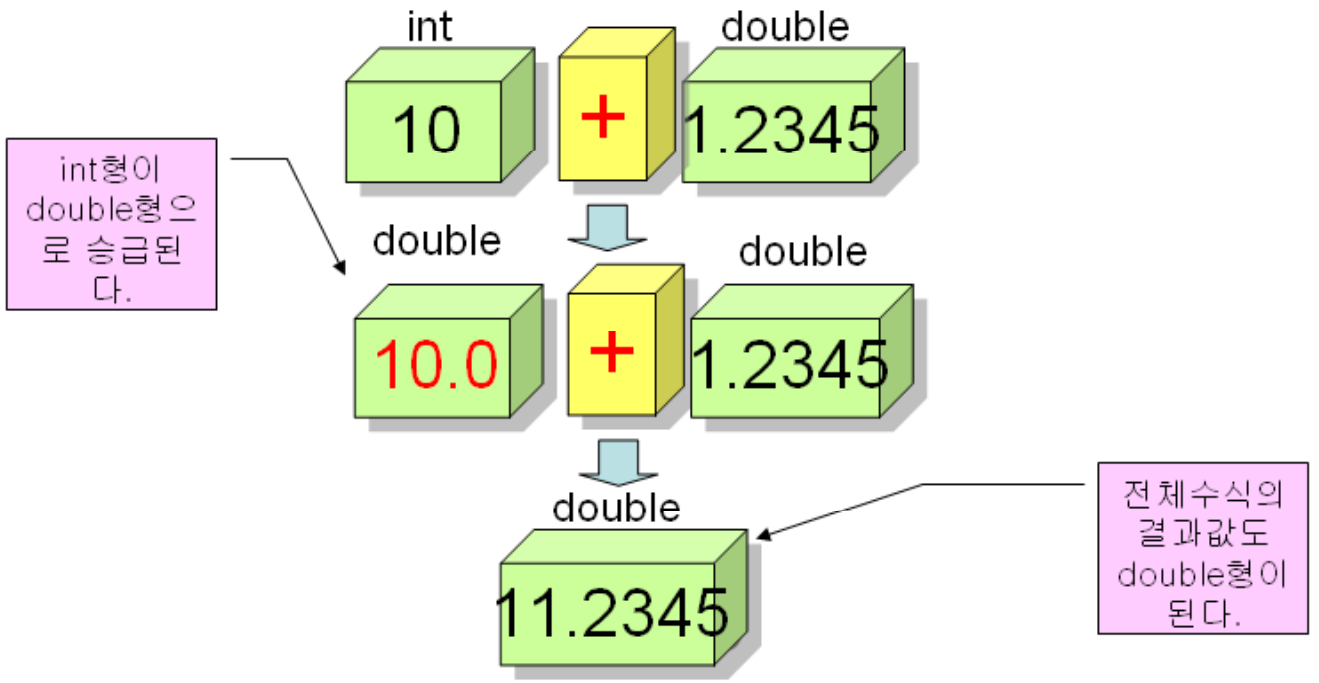
예제

```c
// 정수 연산시의 자동 형변환 프로그램
#include <stdio.h>

int main(void)
{
	char x = 100;
	char y = 100;
	char z;

	z = x + y;
	printf("z = %d \n", z);
	printf("x + y = %d \n", x + y);
	return 0;
}
```

```
z = -56
x + y = 200
```

수식에서의 자동적인 형변환

- 서로 다른 자료형이 혼합하여 사용되는 경우, 더 큰 자료형으로 통일된다.

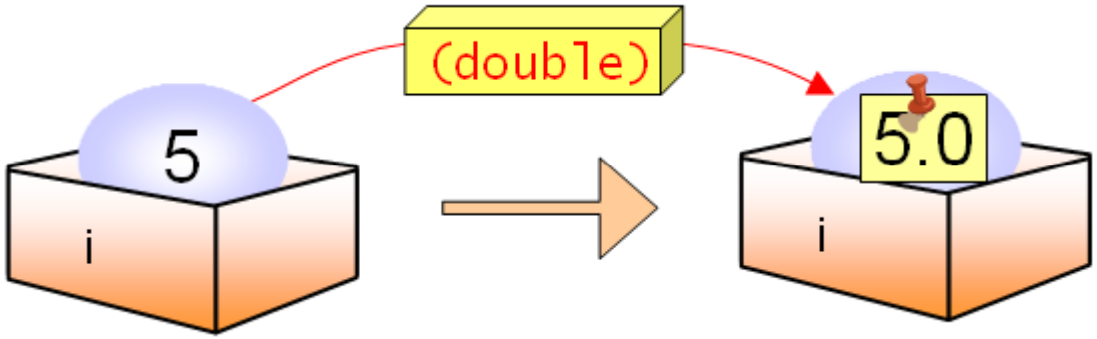


명시적인 형변환

- 형변환(type cast): 사용자가 데이터의 타입을 변경하는 것

```
(자료형) 상수 또는 변수
```

```
(double) (5)
f = (double)i + (double)j;
f = (double)((int)y + 3);
f = (float)(x = 5);        // 수식 x = 5의 결과값인 5가 float형으로 변환
```



예제

```
1.  int i;
2.  double f;

3.  f = 5 / 4;
4.  f = (double)5 / 4;
5.  f = 5 / (double)4;
6.  f = (double)5 / (double)4;

7.  i = 1.3 + 1.8;
8.  i = (int)1.3 + (int)1.8;
```

1. 정수형 변수 i 선언
2. 부동 소수점형 변수 f 선언

3. (정수/ 정수)는 정수
4. 5를 부동소수점으로 변환하여 계산, 전체는 부동소수점형이 됨
5. 4를 부동소수점으로 변환하여 계산, 전체는 부동소수점형이 됨
6. 5와 4를 모두 부동소수점으로 변환하여 계산

7. 1.3+1.8은 3.1로 계산되고 정수형 변수에 대입되므로 i는 3
8. (int)1.3 + (int)1.8은 1+1로 되어서 i는 2

3. f는 1
4. f는 1.25
5. f는 1.25
6. f는 1.25

7. i는 3
8. i는 2

관계 연산자

- 두개의 피연산자를 비교하는 연산자
- 결과값은 참(1) 아니면 거짓(0)

연산자 기호	의미	사용예
==	x와 y가 같은가?	x == y
!=	x와 y가 다른가?	x != y
>	x가 y보다 큰가?	x > y
<	x가 y보다 작은가?	x < y
>=	x가 y보다 크거나 같은가?	x >= y
<=	x가 y보다 작거나 같은가?	x <= y

사용예

```
1 == 2              // 1과 2가 다르므로 거짓
1 != 2              // 1와 2가 다르므로 참
1 <= 2              // 1이 2보다 작으므로 참
1 <  2              // 1이 2보다 작으므로 참
(1+2) == (1*2)      // (1+2)가 (1*2)와 같지 않으므로 거짓
x >= y              // x가 y보다 크거나 같으면 참
i == 10             // i가 10과 같으면 참
k > 3               // k가 3보다 크면 참
m != 6              // m과 6이 같지 않으면 참
```

```
int bool;
bool = (3 == 5);                // bool에는 0이 대입된다.
bool = (3 == 3);                // bool에는 1이 대입된다.
bool = (5 == 5) + (6 != 1);     // bool에는 1+1=2가 대입된다.
```

예제

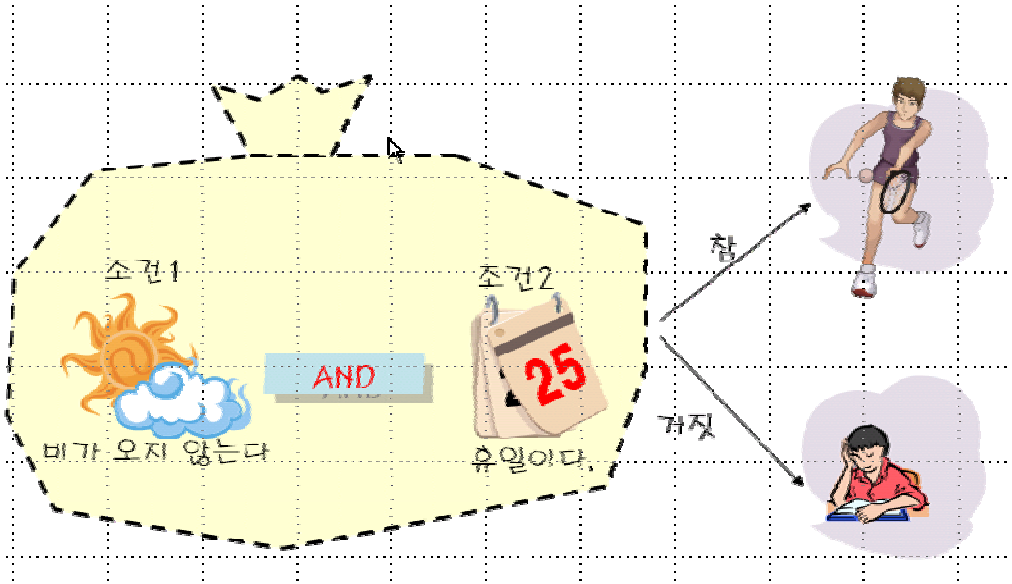
```c
#include <stdio.h>
int main(void)
{
        int x=10, y=20;
        int r1, r2, r3, r4;

        r1 = (x == y);                  // 같으면 1
        r2 = (x != y);                  // 다르면 1
        r3 = (x >= y);                  // 크거나 같으면 1
        r4 = (x <= y);                  // 작거나 같으면 1

        printf("r1=%d \n", r1);
        printf("r2=%d \n", r2);
        printf("r3=%d \n", r3);
        printf("r4=%d \n", r4);
        return 0;
}
```

```
r1=0
r2=1
r3=0
r4=1
```

논리 연산자

- 여러 개의 조건을 조합하여 참과 거짓을 따지는 연산자
- 결과값은 참(1) 아니면 거짓(0)

연산자 기호	사용예	의미
&&	x && y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
||	x || y	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓
!	!x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참



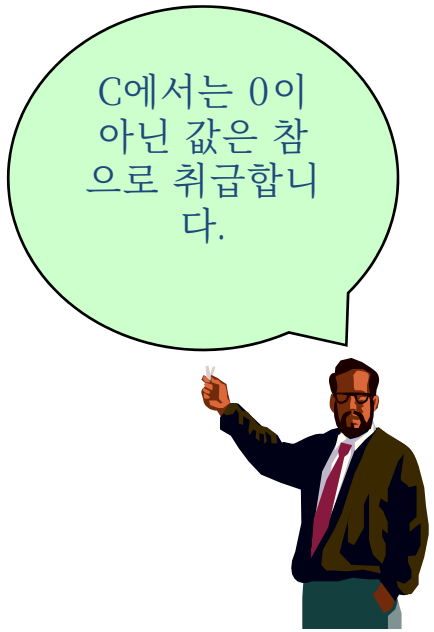
논리 연산의 결과값

- 수학적인 논리 연산

x	y	x AND y	x OR y	NOT x
F	F	F	F	T
F	T	F	T	T
T	F	F	T	F
T	T	T	T	F

- C에서의 논리 연산

x	y	x && y	x || y	!x
0	0	0	0	1
0	0이 아닌값	0	1	1
0이 아닌값	0	0	1	0
0이 아닌값	0이 아닌값	1	1	0

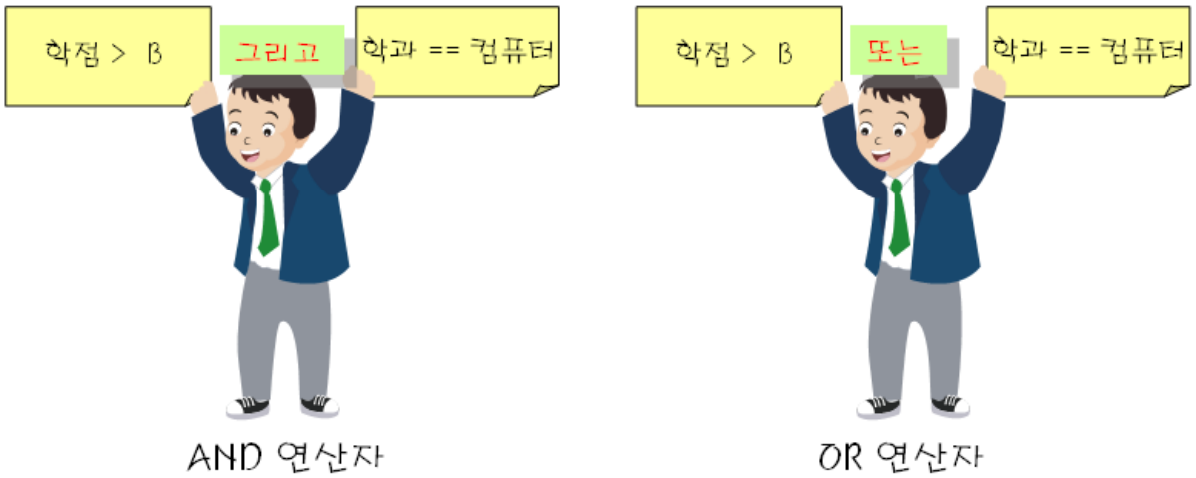


참과 거짓의 표현 방법

- 관계 수식이나 논리 수식이 만약 참이면 1이 생성되고 거짓이면 0이 생성된다.
- 피연산자의 참, 거짓을 가릴 때에는 0이 아니면 참이고 0이면 거짓으로 판단한다.
- 음수는 거짓으로 판단한다.
- (예) NOT 연산자를 적용하는 경우

```
!0                  // 0이면 1로 만든다.
!3                  // 0이 아니면 0으로 만든다.
!100                // 0이 아니면 0으로 만든다.
!-3                 // 0이 아니면 0으로 만든다.
!x                  // 변수 x의 값이 0이면 1, 0이 아니면 0
!(x + 1)            // 수식 (x+1)의 값이 0이면 1, 0이 아니면 0
!(x > y && x < z)   // 관계수식 x > y && x < z의 값이 0이면 1, 0이 아니면 0
```

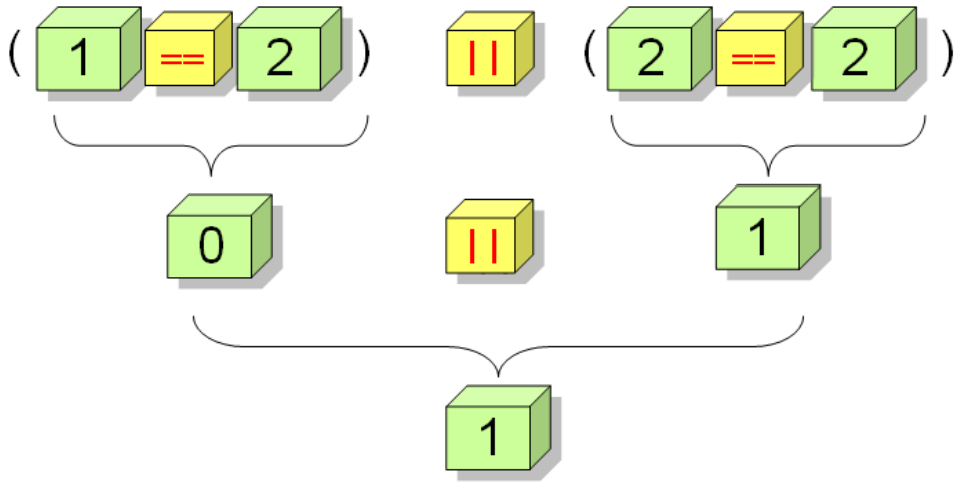
AND와 OR 연산자




```
1 && 2              // 피연산자 모두 0이 아니므로 전체 수식은 참
(1==2) && (2==2)    // 하나의 피연산자만 참이므로 전체 수식은 거짓
(1==2) || (2==2)    // 하나의 피연산자가 참이므로 전체 수식은 참
(x>10) && (x<20)    // x가 10보다 크고 20보다 작으면 참이다.
(x>10) || (x<20)    // x가 10보다 크거나 20보다 작으면 참이다(항상 참).
```

논리 연산자의 계산 과정

- 논리 연산의 결과값은 항상 1 또는 0이다.
- (예)



예제

```c
// 논리 연산자 프로그램
#include <stdio.h>

int main(void)
{
	int x=10, y=20;
	int r1, r2, r3, r4;

	r1 = (x == 10 && y == 20);
	r2 = (x == 10 && y == 30);
	r3 = (x >= 10 || y >= 30);
	r4 = !(x == 5);

	printf("r1=%d \n", r1);
	printf("r2=%d \n", r2);
	printf("r3=%d \n", r3);
	printf("r4=%d \n", r4);

	return 0;
}
```

```
r1=1
r2=0
r3=1
r4=1
```

논리 연산자의 우선 순위

- !연산자의 우선 순위는 증가 연산자 ++나 감소 연산자 --와 동일
- &&와 || 연산자의 우선 순위는 모든 산술 연산자나 관계 연산자보다 낮다.
- &&가 || 연산자보다는 우선 순위가 높다.

```
x < 0 || x > 10
x>5 || x<10 && x>0              // x>5 || (x<10 && x>0) 와 동일
(x>5 || x<10) && x>0
```

예제

- 윤년을 판단하는 문제
  ① 연도가 4로 나누어 떨어진다.
  ② 100으로 나누어 떨어지는 연도는 제외한다.
  ③ 400으로 나누어 떨어지는 연도는 윤년이다.

```
// 윤년 프로그램
#include <stdio.h>

int main(void)
{
	int year, result;

	printf("연도를 입력하시오: ");
	scanf("%d", &year);

	result = (year%4 == 0 && year%100 != 0) || year%400 == 0;
	printf("result=%d \n", result);
	return 0;
}
```

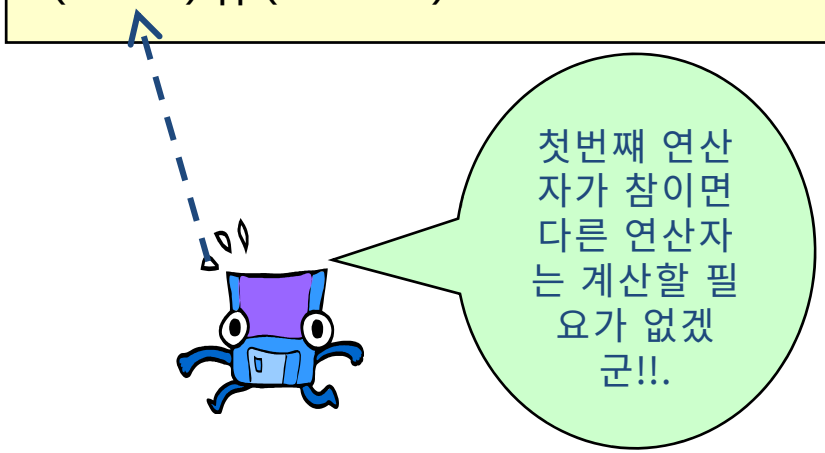
단축 계산

- && 연산자의 경우, 첫번째 피연산자가 거짓이면 다른 피연산자들을 계산하지 않는다.

```
( 2 > 3 ) && ( ++x < 5 )
```

- || 연산자의 경우, 첫번째 피연산자가 참이면 다른 피연산자들을 계산하지 않는다.

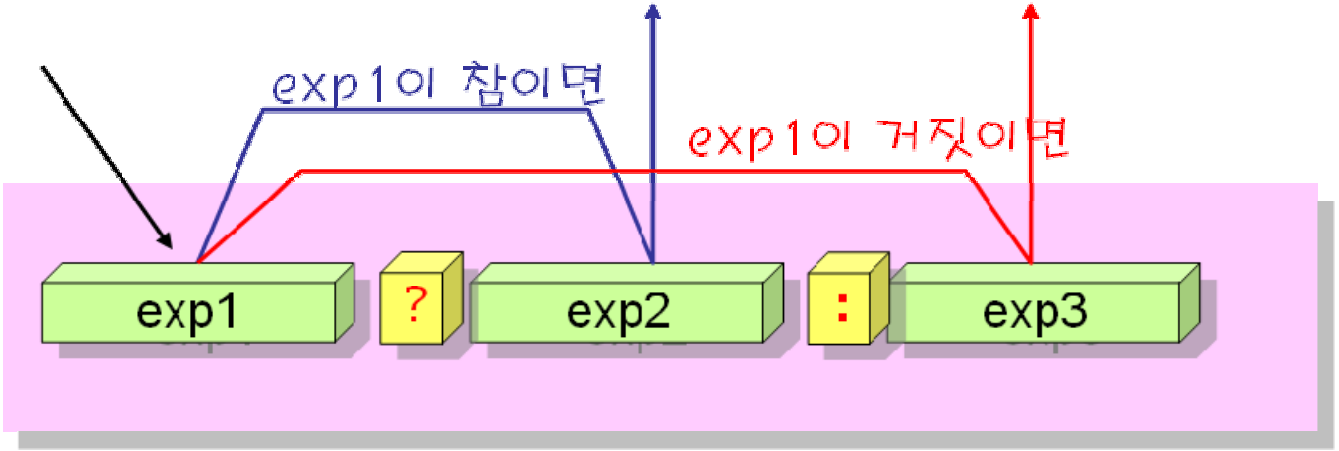
```
( 3 > 2 ) || ( --x < 5 )
```



조건 연산자

- exp1가 참이면 exp2를 반환, 그렇지 않으면 exp3를 반환

```
exp1 ? exp2 : exp3
```




```
(5 > 2) ? 5 : 2          // 5가 2보다 크므로  5
(1.2 > 1.1) ? 1 : 0      // 1.2가 1.1보다 크므로 1
(x == 0) ? 100: 200      // x가 0과 같으면 100 그렇지 않으면 200
```

예제

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

        printf("첫번째 수=");
        scanf("%d", &x);
        printf("두번째 수=");
        scanf("%d", &y);


        printf("큰수=%d \n", (x > y) ? x : y);
        printf("작은수=%d \n", (x < y) ? x : y);
}
```

```
첫번째 수=2
두번째 수=3
큰수=3
작은수=2
```

콤마 연산자

- 콤마로 연결된 수식은 순차적으로 계산된다.

x=1, y=2;	x=1; y=2;와 동일
x = (2+5, 5-3);	x=2가 된다
x = 2+3, 5-3;	x=5가 된다
x++, y++;	x와 y는 1 증가된다.
printf("Thank"), printf(" you!\n");	Thank you!

sizeof 연산자

- 피연산자들의 크기를 바이트 단위로 반환


```
size_t  n = sizeof( int );
```

예제

```c
#include <stdio.h>

int main(void)
{
        int i;
        double f;
        size_t n;

        n = sizeof(int);
        printf("int형의 크기=%u \n", n);

        n = sizeof(i);
        printf("변수 i의 크기=%u \n", n);

        n = sizeof f;
        printf("변수 f의 크기=%u \n", n);
}
```

```
int형의 크기=4
변수 i의 크기=4
변수 f의 크기=8
```

우선 순위

- 수식에서 어떤 연산자를 먼저 계산할 것인지의 문제

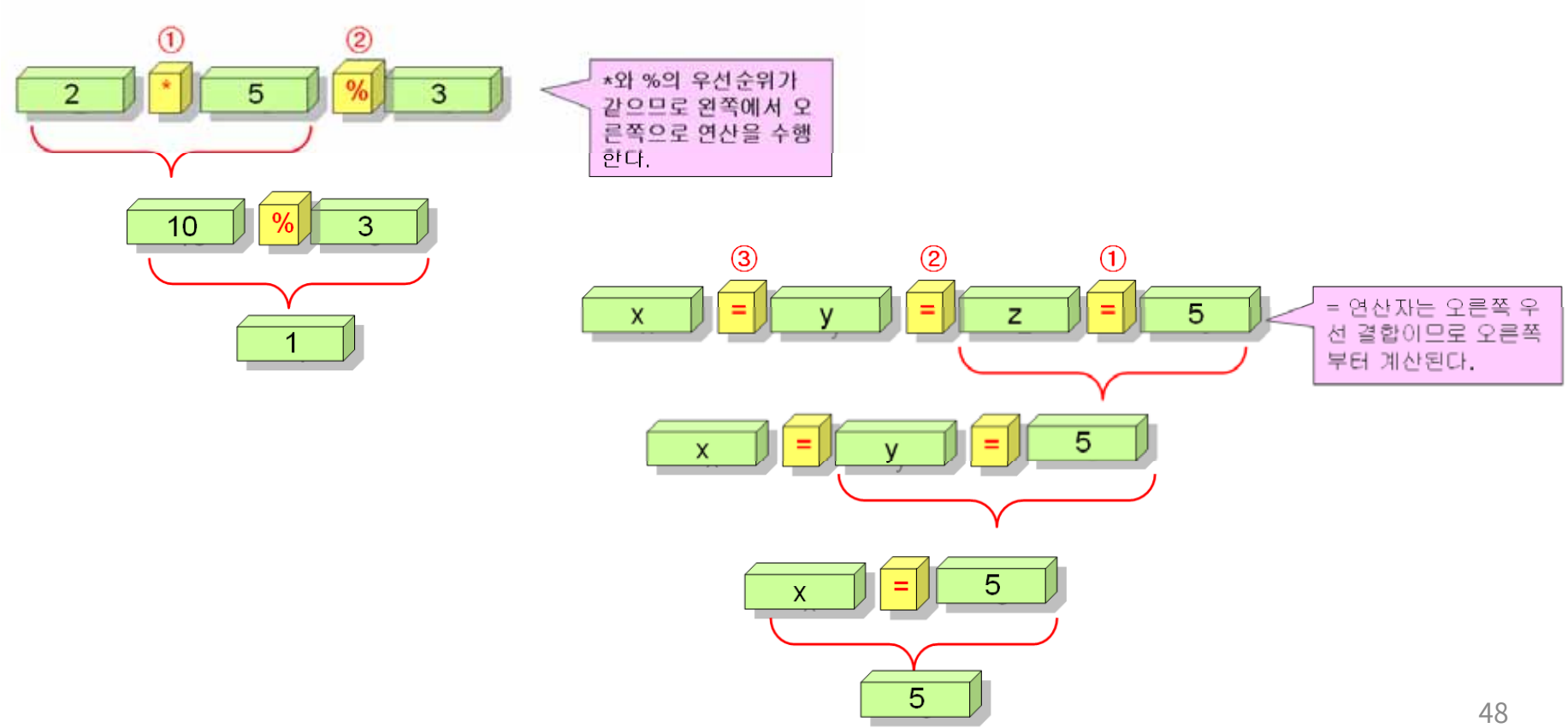
우선 순위	연산자	결합 규칙
1	() [] > . ++(후위) (후위)	>(좌에서 우)
2	sizeof &(주소) ++(전위) --(전위) ~ ! *(역참조) +(부호) -(부호), 형변환	<-(우에서 좌)
3	*(곱셈) / %	->(좌에서 우)
4	+(덧셈) -(뺄셈)	->(좌에서 우)
5	<< >>	->(좌에서 우)
6	< <= >= >	->(좌에서 우)
7	== !=	->(좌에서 우)
8	&(비트연산)	->(좌에서 우)
9	^	->(좌에서 우)
10	|	->(좌에서 우)
11	&&	->(좌에서 우)
12	||	->(좌에서 우)
13	?(삼항)	->(우에서 좌)
14	= += *= /= %= &= ^= |= <<= >>=	->(우에서 좌)
15	,(콤마)	->(좌에서 우)

우선 순위의 일반적인 지침

- 단항 > 산술 > 관계 > 논리 > 대입 > 콤마
- 괄호 연산자는 가장 우선순위가 높다.
- 모든 단항 연산자들은 이항 연산자들보다 우선순위가 높다.
- 콤마 연산자를 제외하고는 대입 연산자가 가장 우선순위가 낮다.
- 연산자들의 우선 순위가 생각나지 않으면 괄호를 이용
  - (x <= 10) && (y >= 20)
- 관계 연산자나 논리 연산자는 산술 연산자보다 우선순위가 낮다.
  - x + 2 == y + 3

결합 규칙

- 만약 같은 우선순위를 가지는 연산자들이 여러 개가 있으면 어떤 것을 먼저 수행하여야 하는가의 규칙



예제

```c
#include <stdio.h>
int main(void)
{
        int x = 2, y = 3, z = 4;


        printf("%d \n", 2 + 3 >= 3 + !2);
        printf("%d \n", 2 > 3 || 6 > 7);
        printf("%d \n", 2 || 3 && 3 > 2);
        printf("%d \n", - ++x + y--);
        printf("%d \n", x = y = z = 6 );
        printf("%d \n", (x = 2 + 3, 2 - 3));
        printf("%d \n", x /= x = x * y );
}
```

```
1
0
1
0
6
-1
1
```

Q & A