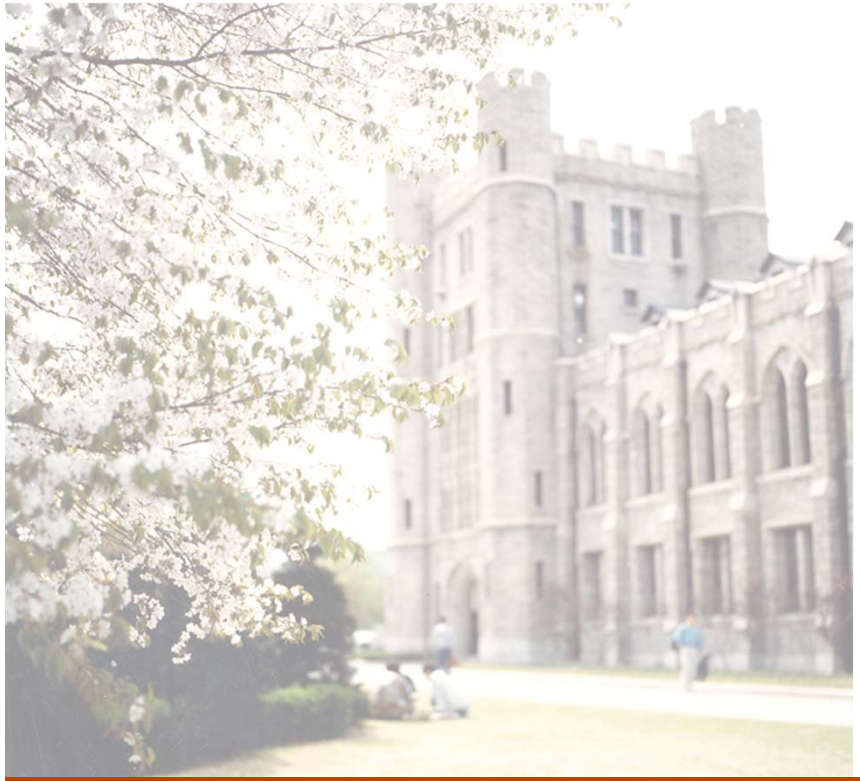
DAEGU UNIVERSITY

42경산 라피신(La piscine) 대비 사전 SW교육과정 - C



Heungwoo Nam

Computer Engineering
Daegu University
2023. 7. 11

Objective & Contents

❑ 수업목표

- 수식과 연산자의 이해

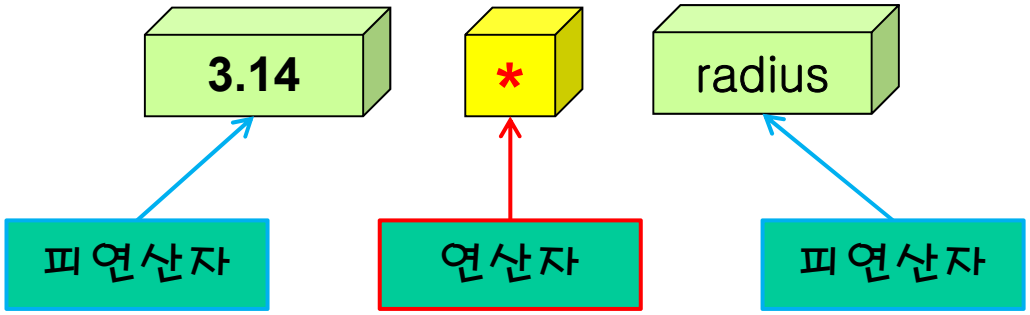
❑ Contents

- 5.1 수식과 연산자
- 5.2 산술 연산자
- 5.3 대입 연산자
- 5.4 관계 연산자
- 5.5 논리 연산자
- 5.6 조건 연산자
- 5.7 콤마 연산자
- 5.9 형변환
- 5.10 연산자의 우선 순위의 결합 규칙

5.1 수식과 연산자

❑ 수식 (expression)

- 피연산자들과 연산자의 조합
  - 연산자(operator): 어떤 연산을 나타내는 기호
  - 피연산자(operand): 연산의 대상이 되는 것
- 상수, 변수, 연산자의 조합



5.1 수식과 연산자

❑ 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
콤마	,	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

5.1 수식과 연산자

❑ 피연산자 수에 따른 연산자 분류

- 단항 연산자: 피연산자의 수가 1개

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

- 이항 연산자: 피연산자의 수가 2개

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

- 삼항 연산자: 피연산자의 수가 3개

```
x ? y : z
```

5.2 산술 연산자

❑ 산술 연산자

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

연산자	기호	사용예	결과값
덧셈	+	7 + 4	11
뺄셈	–	7 – 4	3
곱셈	*	7 * 4	28
나눗셈	/	7 / 4	1
나머지	%	7 % 4	3

5.2 산술 연산자

❑ 산술 연산자의 예

$y = mx + b$ → `y = m*x + b`

$y = ax^2 + bx + c$ → `y = a*x*x + b*x +c`

$m = \frac{x + y + x}{3}$ → `m = (x + y + z) / 3`

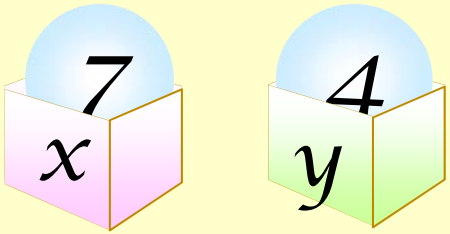
(참고) 거듭 제곱 연산자는?

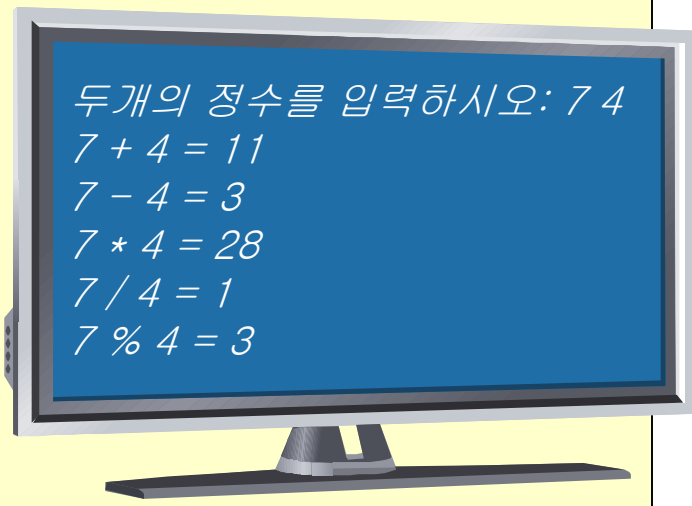
C에는 거듭 제곱을 나타내는 연산자는 없다.
x * x와 같이 단순히 변수를 두 번 곱한다.

5.2 산술 연산자

❑ 정수 사칙 연산

```
#include <stdio.h>

int main(void)
{
        int x, y, result;
        printf("두개의 정수를 입력하시오: ");
        scanf("%d %d", &x, &y);

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

        result = x - y;                 // 뺄셈
        printf("%d - %d = %d", x, y, result);
        result = x * y;                 // 곱셈
        printf("%d + %d = %d", x, y, result);
        result = x / y;                 // 나눗셈
        printf("%d / %d = %d", x, y, result);
        result = x % y;                 // 나머지
        printf("%d %% %d = %d", x, y, result);
        return 0;
}
```

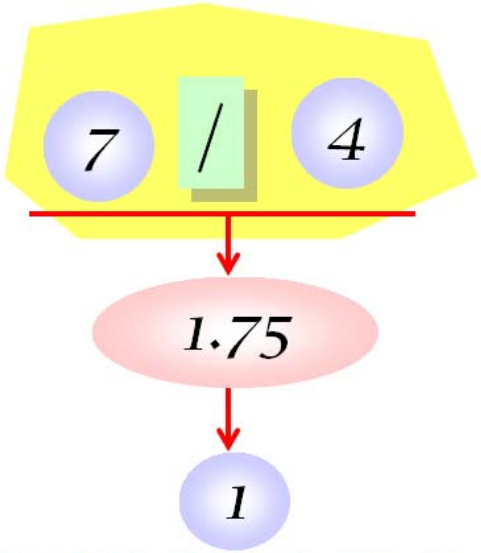




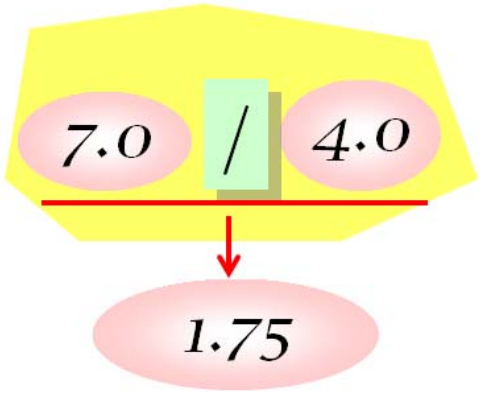
5.2 산술 연산자

❑ 나눗셈 연산자

- 정수형끼리의 나눗셈에서는 결과가 정수형으로 생성하고 부동소수점형끼리는 부동소수점 값을 생성함
- 정수형끼리의 나눗셈에서는 소수점 이하는 버려짐



정수와 정수 끼리의 나눗셈.



실수와 실수 끼리의 나눗셈.

5.2 산술 연산자

❑ 나머지 연산자

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

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

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

5.2 산술 연산자

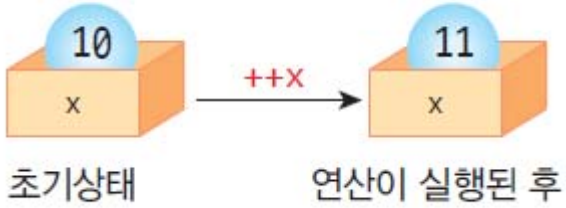
❑ 나머지 연산자

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

int main(void)
{
        int input, minute, second;

        printf( " 초를 입력하시요: ");
        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;
}
```

5.2 산술 연산자

❑ 증감 연산자

- 증감 연산자: ++, --
- 변수의 값을 하나 증가시키거나 감소시키는 연산자
- ++x, --x;



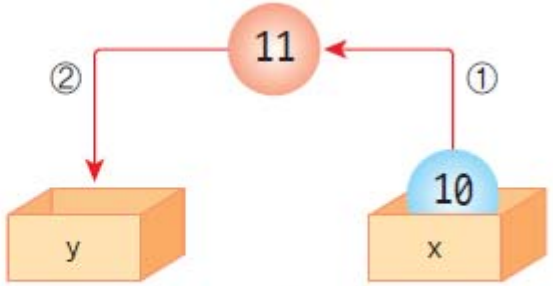
증감 연산자	의미
++x	수식의 값은 증가된 x값이다.
x++	수식의 값은 증가되지 않은 원래의 x값이다.
--x	수식의 값은 감소된 x값이다.
x--	수식의 값은 감소되지 않은 원래의 x값이다.

5.2 산술 연산자

❑ 증감 연산자

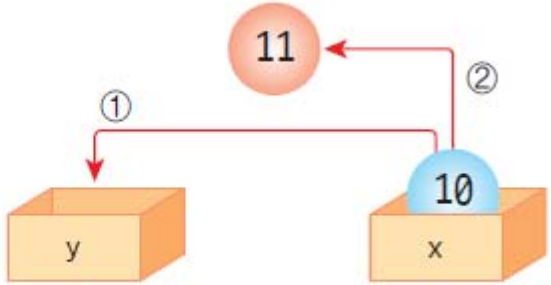
- ++x와 x++의 차이

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



증가된 x의 값이 y에 대입된다.

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



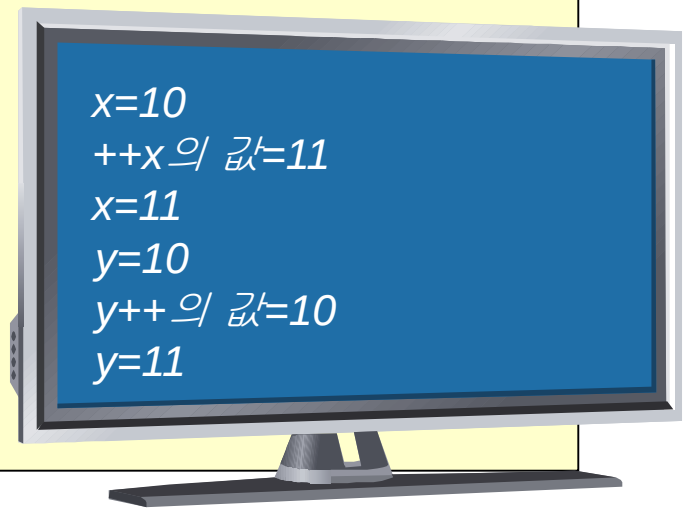
먼저 대입하고 나중에 증가한다.

5.2 산술 연산자

❑ 증감 연산자

- 예제

```
#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;
}
```

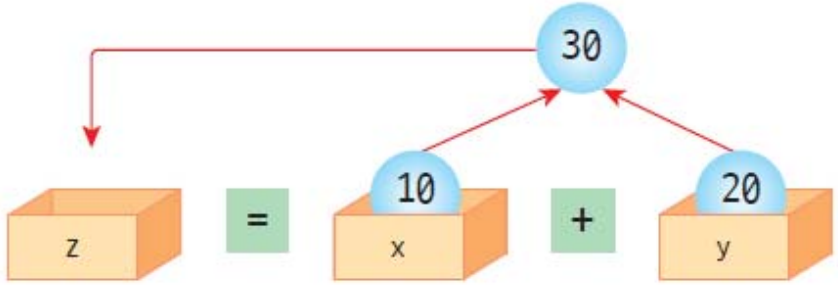


5.3 대입(배정, 할당) 연산자

❑ 대입 연산자 (assignment operator)

- 변수에 수식의 값을 계산하여 저장하는 연산자임
- 등호의 왼쪽은 반드시 변수이어야 하고, 등호의 오른쪽은 어떠한 수식이라도 가능함.

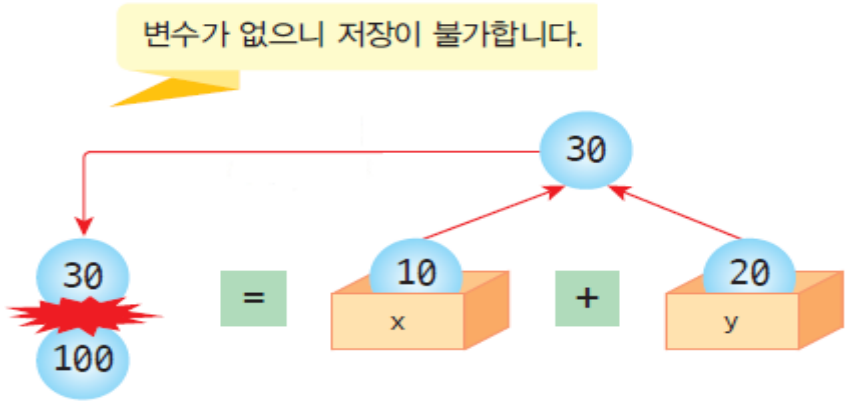
Syntax: 대입 연산자



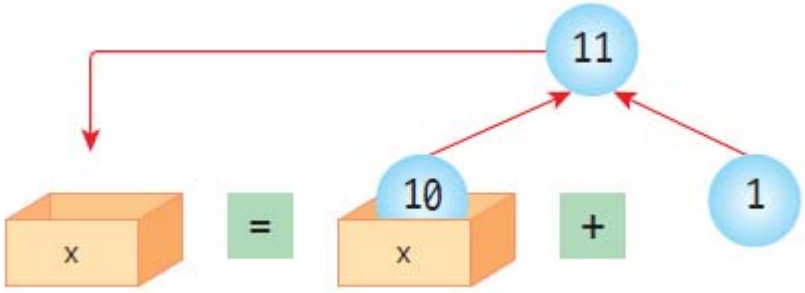
5.3 대입(배정, 할당) 연산자

❑ 대입 연산자 주의점

- 100 = x + y; // 컴파일 오류!



- x+y=100; //등호의 왼편이 변수가 아님 → 잘못된 서식
- x=x+1;



5.3 대입(배정, 할당) 연산자

❑ 대입 연산의 결과값

예제1)

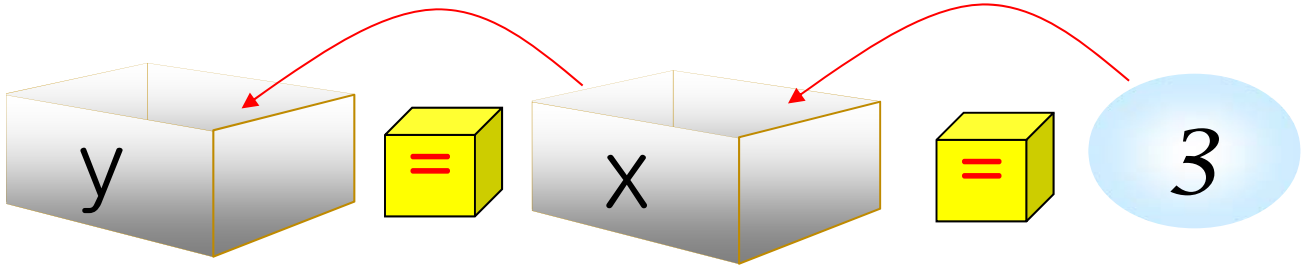
$$y = 10 + (\underbrace{x = \overbrace{2 + 7}^{\text{덧셈연산의 결과값은 9}}}_{\text{대입연산의 결과값은 9}});$$

덧셈연산의 결과값은 19

대입연산의 결과값은 19

예제2)

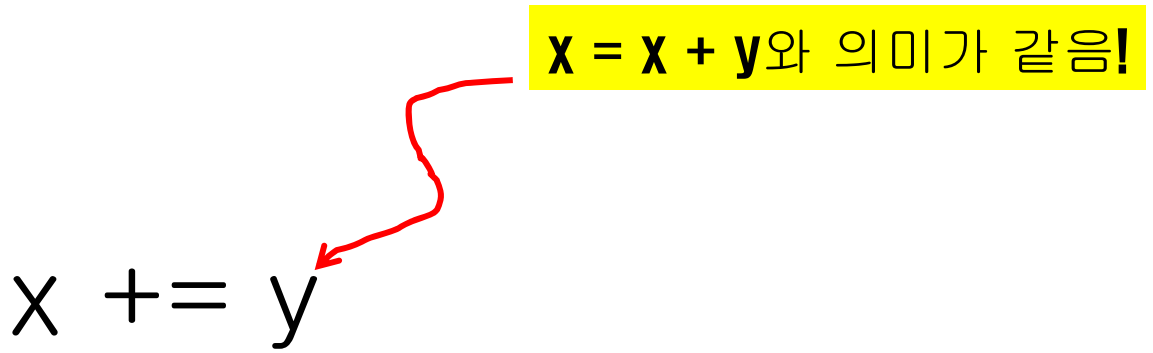
```
y = x = 3;
```



5.3 대입(배정, 할당) 연산자

❑ 복합 대입 연산자

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



5.3 대입(배정, 할당) 연산자

❑ 복합 대입 연산자

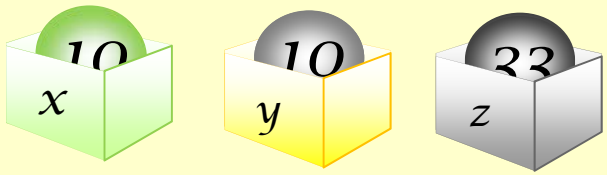
복합 대입 연산자	의미
x += y	x = x + y
x -= y	x = x - y
x *= y	x = x * y
x /= y	x = x / y
x %= y	x = x % y
x &= y	x = x & y
x |= y	x = x | y
x ^= y	x = x ^ y
x >>= y	x = x >> y
x <<= y	x = x << y

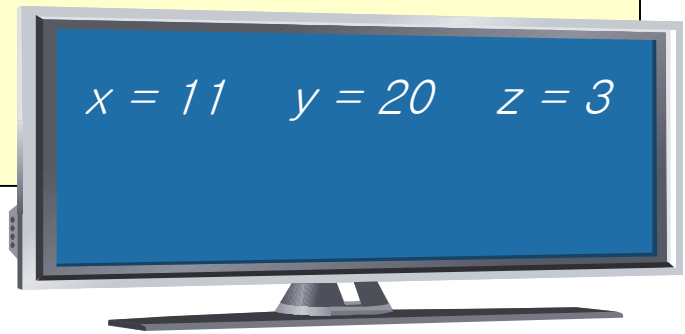
5.3 대입(배정, 할당) 연산자

❑ 복합 대입 연산자

- 예제

```
// 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;
}
```





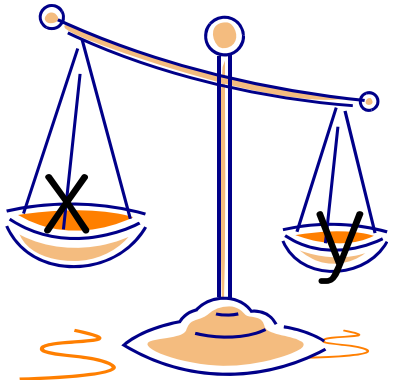
5.4 관계 연산자

❑ 관계 연산자 (relational operator)

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

x == y

x 와 y의 값이 같은지 비교한다.



5.4 관계 연산자

❑ 관계 연산자 (relational operator)

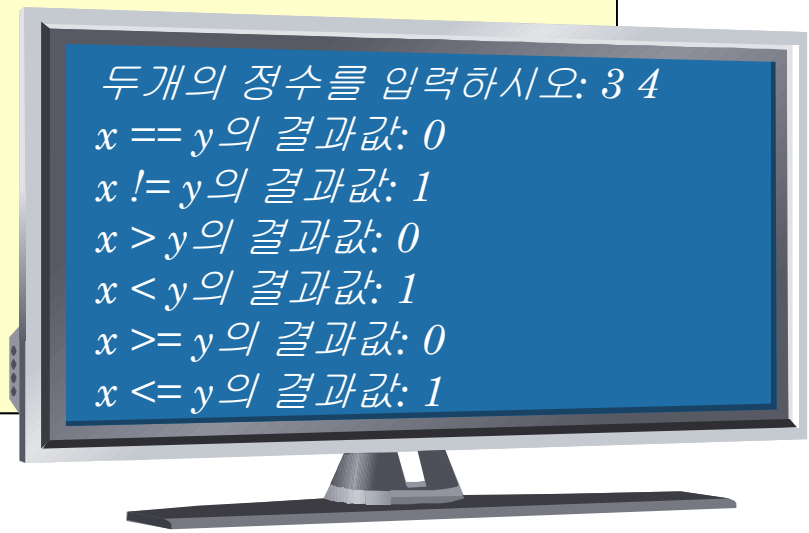
연산자	의미
x == y	x와 y가 같은가?
x != y	x와 y가 다른가?
x > y	x가 y보다 큰가?
x < y	x가 y보다 작은가?
x >= y	x가 y보다 크거나 같은가?
x <= y	x가 y보다 작거나 같은가?

5.4 관계 연산자

❑ 관계 연산자 (relational operator)

- 예제

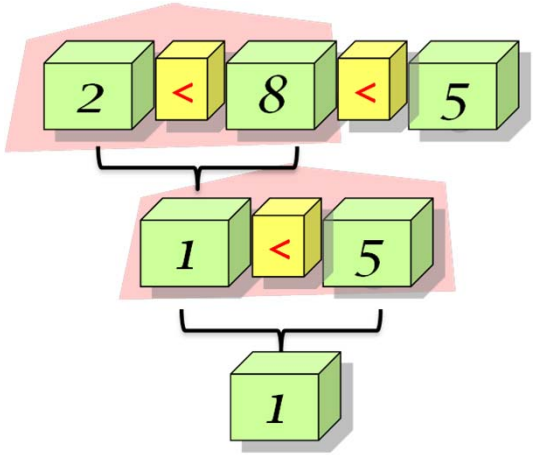
```
#include <stdio.h>
int main(void)
{
        int x, y;

        printf("두개의 정수를 입력하시오: ");
        scanf("%d%d", &x, &y);

        printf("x == y의 결과값: %d", x == y);
        printf("x != y의 결과값: %d", x != y);
        printf("x > y의 결과값: %d", x > y);
        printf("x < y의 결과값: %d", x < y);
        printf("x >= y의 결과값: %d", x >= y);
        printf("x <= y의 결과값: %d", x <= y);

        return 0;
}
```



5.4 관계 연산자

❑ 관계 연산자 (relational operator)

- 주의할 점
  - (x = y)
    - y의 값을 x에 대입한다. 이 수식의 값은 x의 값임
  - (x == y)
    - x와 y가 같으면 1, 다르면 0이 수식의 값이 됨
    - (x == y)를 (x = y)로 잘못 쓰지 않도록 주의!
  - 수학에서처럼 2 < x < 5와 같이 작성하면 잘못된 결과가 나옴



→올바른 방법: (2 < x) && (x < 5)

5.5 논리 연산자

❑ 논리 연산자 (logical operator)

- 여러 개의 조건을 조합하여 참인지 거짓인지를 따질 때 사용
- 결과 값은 참(1) 아니면 거짓(0)
- Ex) “비가 오지 않고 휴일이면 테니스를 친다”
  - “비가 오지 않는다”라는 조건과 “휴일이다”라는 조건이 동시에 만족이 되면 테니스를 친다는 의미가 포함됨

연산자	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

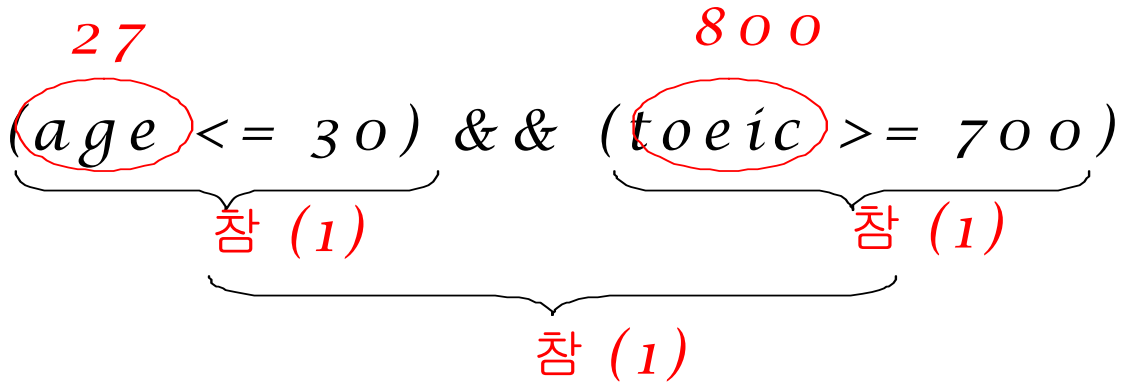
5.5 논리 연산자

❑ 논리 연산자 (logical operator)

- AND와 OR 연산자

x	y	X&&y	X||y
참	참	참	참
참	거짓	거짓	참
거짓	참	거짓	참
거짓	거짓	거짓	거짓

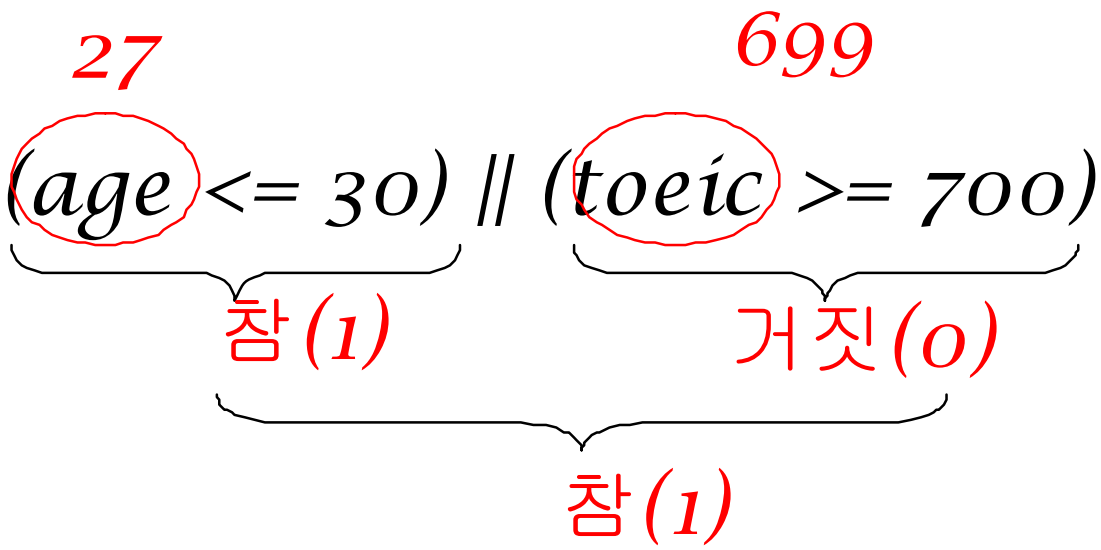
– AND 연산자



5.5 논리 연산자

❑ 논리 연산자 (logical operator)

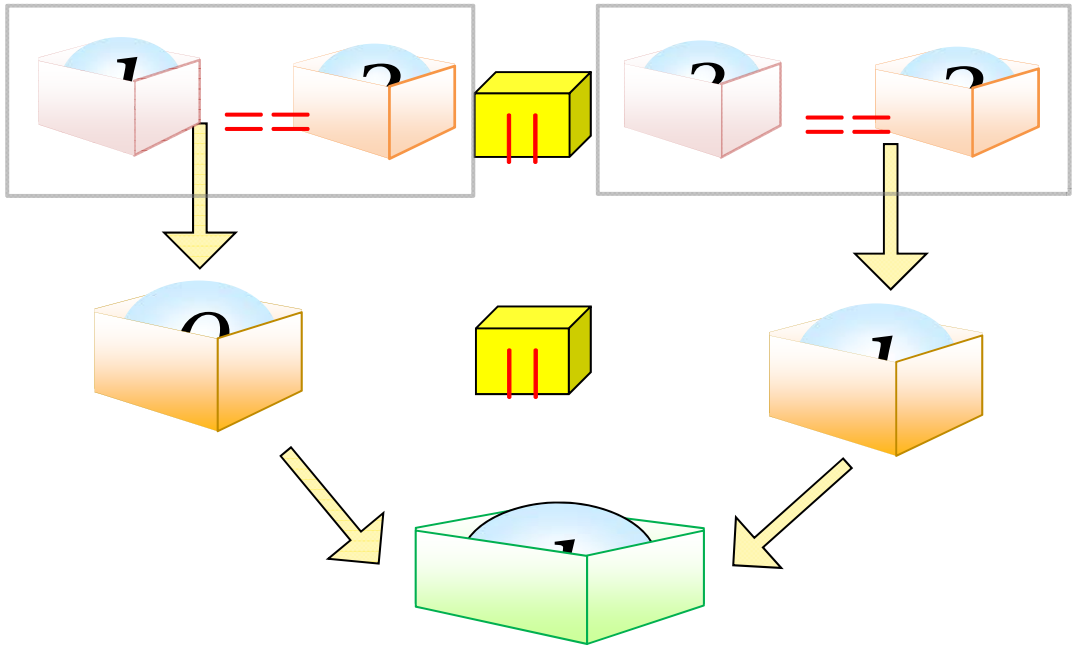
- OR 연산자



5.5 논리 연산자

❑ 논리 연산자의 계산 과정

- 논리 연산의 결과값은 항상 1 또는 0이다.
- (예) (1 == 2) || (2 == 2)



5.5 논리 연산자

❑ 참과 거짓의 표현 방법

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

```
!0              // 식의 값은 1
!3              // 식의 값은 0
!-3             // 식의 값은 0
```

5.5 논리 연산자

❑ 논리 연산자의 예

- "x는 1, 2, 3중의 하나인가"
  - (x == 1) || (x == 2) || (x == 3)

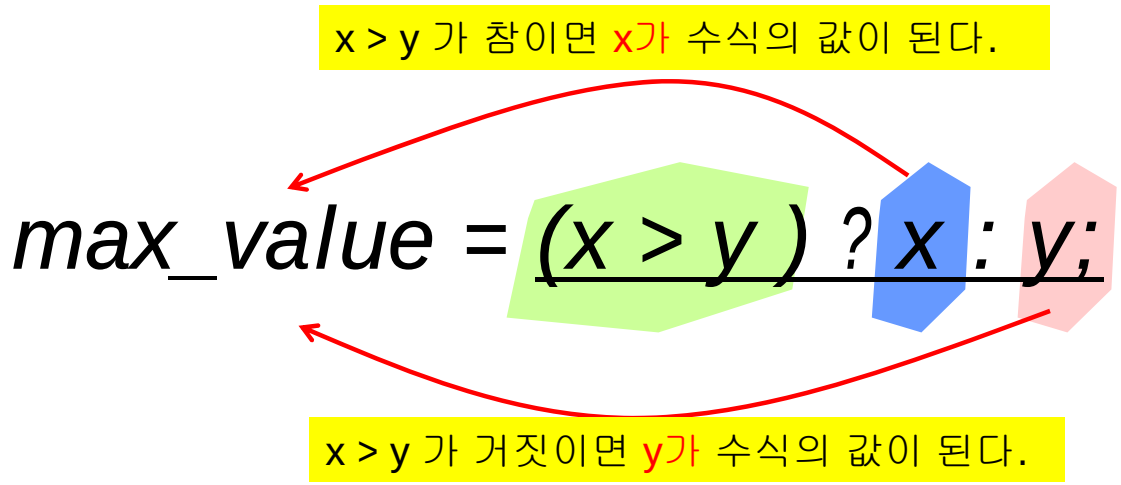
- "x가 60이상 100미만이다."
  - (x >= 60) && (x < 100)

- "x가 0도 아니고 1도 아니다."
  - (x != 0) && (x != 1) // x≠0 이고 x≠1이다.

5.6 조건 연산자

❑ 조건 연산자

- 조건 연산자는 유일하게 3개의 피연산자를 가지는 삼항 연산자임.
- 예제)



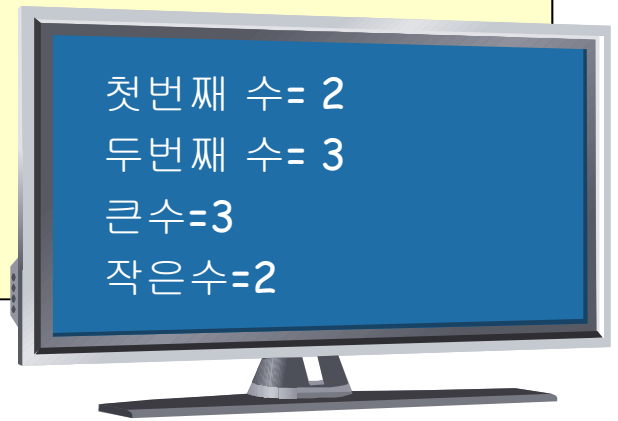

```
1) absolute_value = (x > 0) ? x: -x;          // 절대값 계산
2) max_value = (x > y) ? x: y;                // 최대값 계산
3) min_value = (x < y) ? x: y;                // 최소값 계산

4) (age > 20) ? printf("성인\n"): printf("청소년\n");
```

5.6 조건 연산자

❑ 조건 연산자

- 예제)

```
#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);

        return 0;
}
```

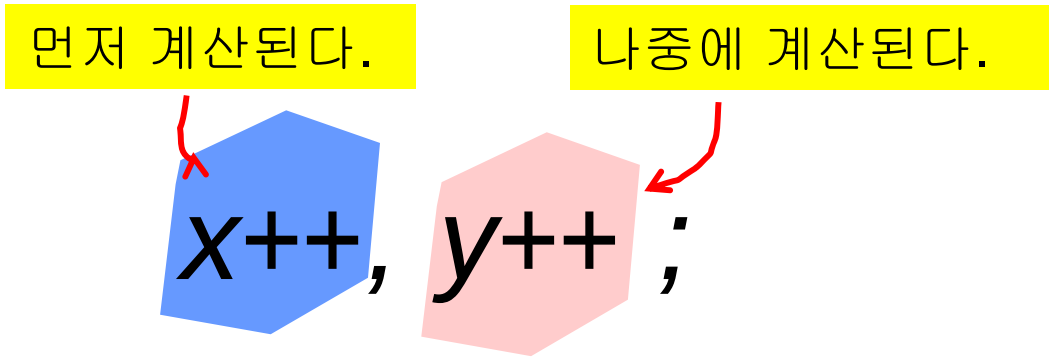


5.7 콤마 연산자

❑ 콤마 연산자

- 콤마로 연결된 수식은 순차적으로 계산됨.
  - 각각의 수식은 왼쪽부터 오른쪽으로 계산됨.

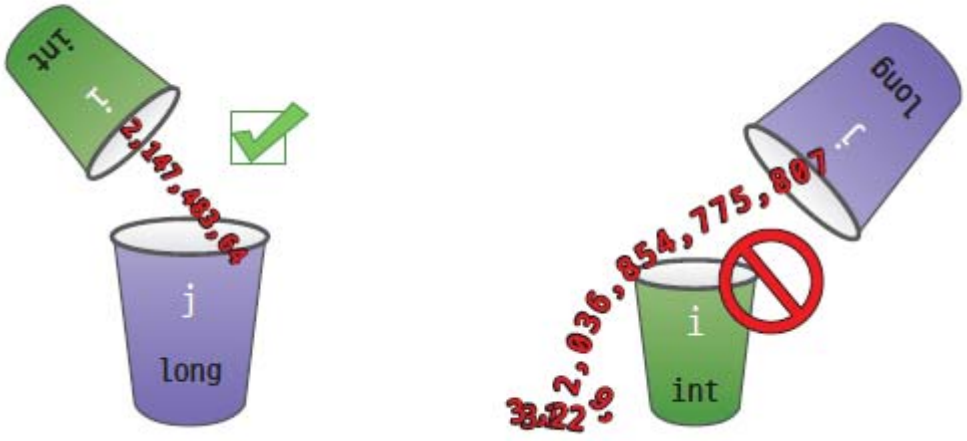
– (x++, y++)의 경우



5.9 형변환

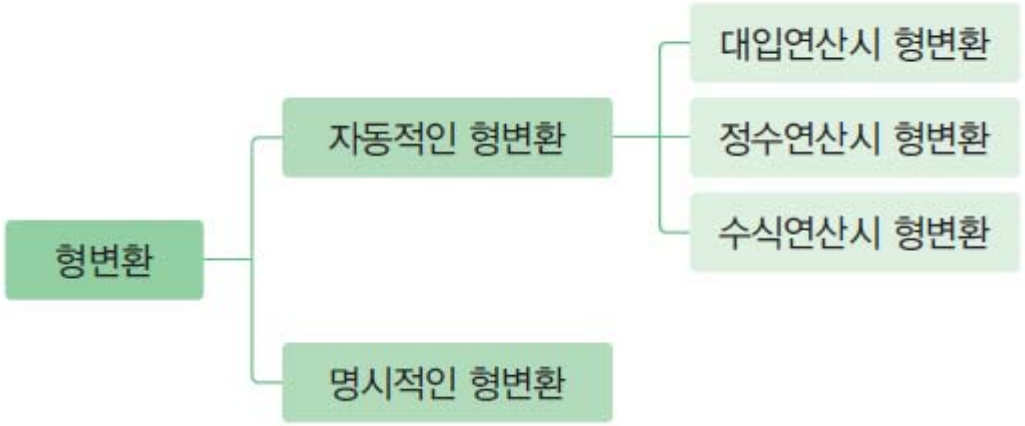
❑ 형변환 (type conversion)

- 실행 중에 데이터의 타입을 변경하는 것임
- 형변환을 잘못하면 데이터의 일부가 사라질 수도 있기 때문에 주의하여야 함



5.9 형변환

❑ 형변환의 분류



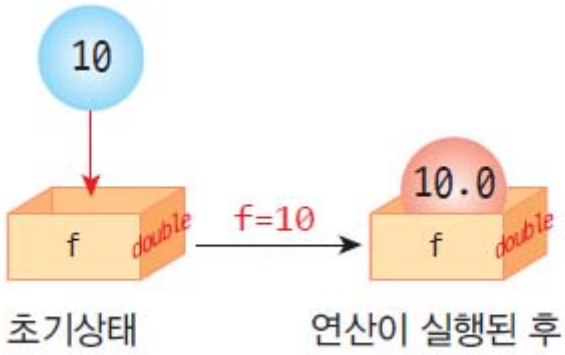
- 자동적인 형변환: 컴파일러에 의하여 자동으로 수행됨.
- 명시적인 형변환: 프로그래머가 명시적으로 데이터의 형을 변환함.

5.9 형변환

❑ 대입 연산시의 자동적인 형변환

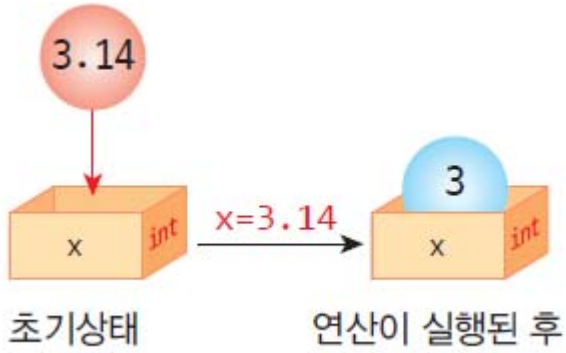
- 올림 변환 (promotion)

```
double f;
f = 10 ;     // f에는 10.0이 저장된다.
```



- 내림 변환 (demotion)

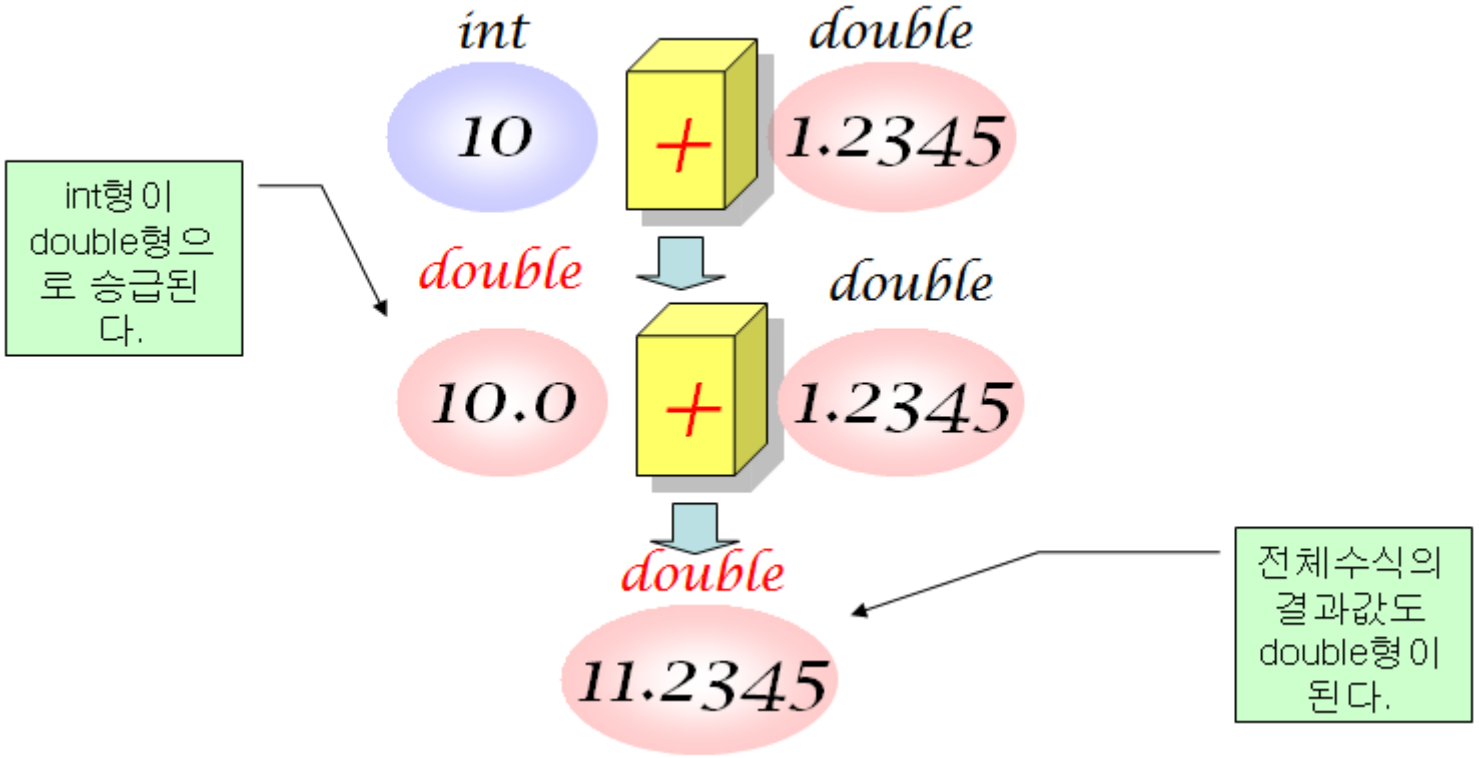
```
int i;
i = 3.141592;   // i에는 3이 저장된다.
```



5.9 형변환

❑ 수식에서의 자동적인 형변환

- 서로 다른 자료형이 혼합하여 사용되는 경우, 더 큰 자료형으로 통일됨



5.9 형변환

❑ 명시적인 형변환

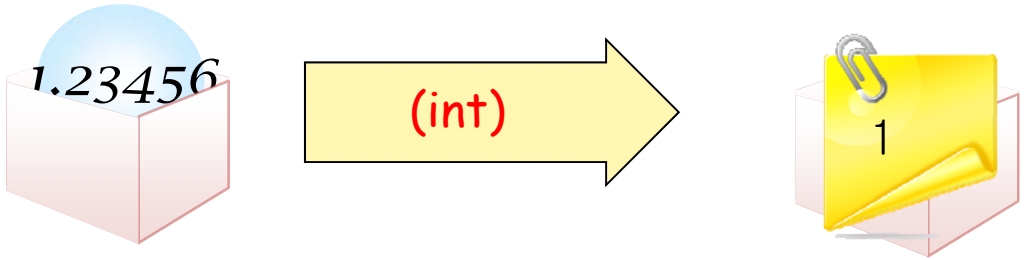
- 형변환(type cast) 연산자: 명시적으로 형을 변환하는 경우 사용하는 연산자로서, 캐스트 연산자라고 함

Syntax: 형변환

자료형 수식

예

```
(int)1.23456      // int형으로 변환
(double) x        // double형으로 변환
(long) (x+y)      // long형으로 변환
```



5.9 형변환

❑ 명시적인 형변환

- 예제

```
#include <stdio.h>

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

        f = 5 / 4;

        printf("%f\n", f);

        f = (double)5/ 4;
        printf("%f\n", f);

        f = 5.0 / 4;
        printf("%f\n", f);

        f = (double)5/ (double)4;
        printf("%f\n", f);

        i = 1.3 + 1.8;
        printf("%d\n", i);

        i = (int)1.3+ (int)1.8;

        printf("%d\n", i);
        return 0;
}
```

5.10 연산자의 우선순위와 결합 규칙

❑ 연산자의 우선순위 (precedence)

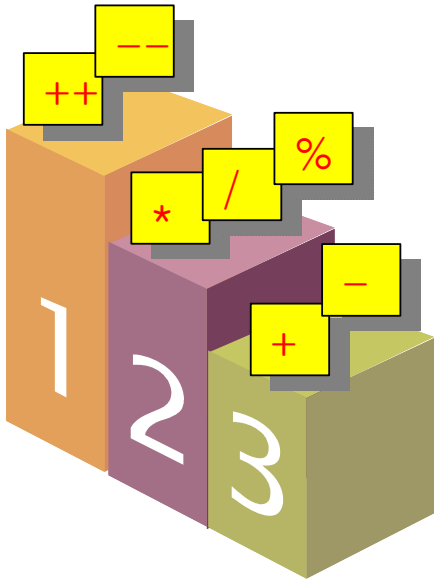
- 많은 연산들 중에서 어떤 연산을 먼저 수행할지를 결정하는 규칙

$x + y * z$

① ②

$(x + y) * z$

① ②



5.10 연산자의 우선순위와 결합 규칙

❑ 연산자의 우선순위 (precedence)

우선순위	연산자	설명	결합성
1	++ --	후위 증감 연산자	→ (좌에서 우)
	()	함수 호출	
	[]	배열 인덱스 연산자	
	.	구조체 멤버 접근	
	->	구조체 포인터 접근	
	(type){list}	복합 리터럴(C99 규격)	
2	++ --	전위 증감 연산자	← (우에서 좌)
	+ -	양수, 음수 부호	
	! ~	논리적인 부정, 비트 NOT	
	(type)	형변환	
	*	간접 참조 연산자	
	&	주소 추출 연산자	
	sizeof	크기 계산 연산자	
	_Alignof	정렬 요구 연산자 (C11 규격)	

5.10 연산자의 우선순위와 결합 규칙

❑ 연산자의 우선순위 (precedence)

3	* / %	곱셈, 나눗셈, 나머지	
4	+ -	덧셈, 뺄셈	
5	<< >>	비트 이동 연산자	
6	< <=	관계 연산자	
	> >=	관계 연산자	
7	== !=	관계 연산자	→ (좌에서 우)
8	&	비트 AND	
9	^	비트 XOR	
10	|	비트 OR	
11	&&	논리 AND 연산자	
12	||	논리 OR 연산자	
13	?:	삼항 조건 연산자	

5.10 연산자의 우선순위와 결합 규칙

❑ 연산자의 우선순위 (precedence)

14	=	대입 연산자	← (우에서 좌)
	+= -=	복합 대입 연산자	
	*= /= %=	복합 대입 연산자	
	<<= >>=	복합 대입 연산자	
	&= ^= |=	복합 대입 연산자	
15	,	콤마 연산자	→ (좌에서 우)

5.10 연산자의 우선순위와 결합 규칙

❑ 우선 순위의 일반적인 지침

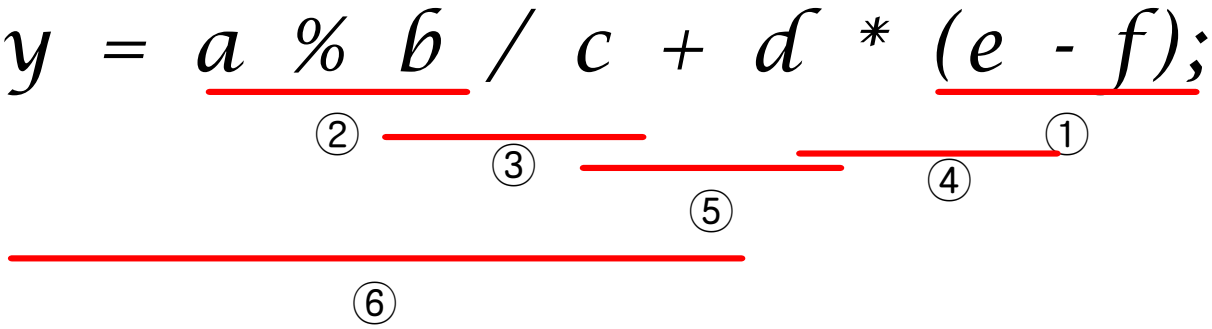
- 콤마 < 대입 < 논리 < 관계 < 산술 < 단항
- 괄호 연산자는 가장 우선순위가 높다.
- 모든 단항 연산자들은 이항 연산자들보다 우선순위가 높다.
- 콤마 연산자를 제외하고는 대입 연산자가 가장 우선순위가 낮다.
- 연산자들의 우선 순위가 생각나지 않으면 괄호를 이용
  - (x <= 10) && (y >= 20)
- 관계 연산자나 논리 연산자는 산술 연산자보다 우선순위가 낮다.
  - x + 2 == y + 3 → (x + 2) == (y + 3)
- 관계 연산자는 논리 연산자보다 우선 순위가 높다.
  - x > y && z > y → (x > y) && (z > y)
- 논리 연산자 중에서 &&연산자가 ||연산자보다 우선 순위가 높다
  - x < 5 || x > 10 && x > 0 → x < 5 || (x > 10 && x > 0)

5.10 연산자의 우선순위와 결합 규칙

❑ 연산자의 결합 규칙 (association)

- 만약 같은 우선순위를 가지는 연산자들이 여러 개가 있으면 어떤 것을 먼저 수행하여야 하는가의 규칙
- 즉, 동일한 우선 순위의 연산이 있는 경우에 무엇을 먼저 수행하느냐에 대한 규칙

- 결합규칙의 예



5.10 연산자의 우선순위와 결합 규칙

❑ 연산자의 결합 규칙 (association)

- 예제

```
#include <stdio.h>
int main(void)
{
        int x=0, y=0;
        int result;

        result = 2 > 3 || 6 > 7;
        printf("%d", result);

        result = 2 || 3 && 3 > 2;
        printf("%d", result);

        result = x = y = 1;
        printf("%d", result);

        result = - ++x + y--;  //수식 계산 전, x=1, y=1
        printf("%d", result);

        return 0;
}
```