

기초컴퓨터프로그래밍

제 5장 수식과 연산자

양우석 yws87874912@gmail.com



이번 장에서 학습할 내용



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

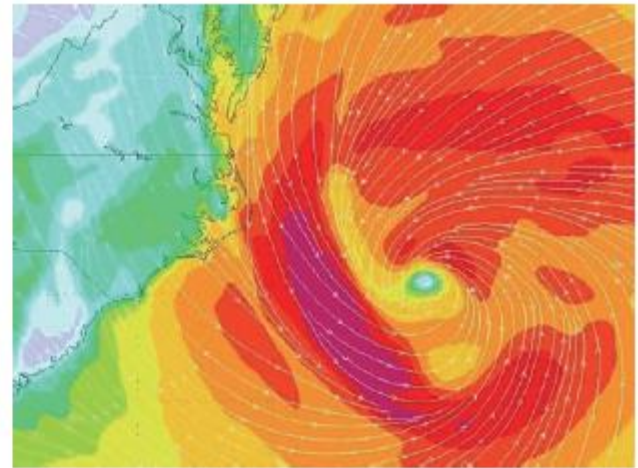
이번 장에서는 수식과
연산자를 살펴봅니다.





컴퓨터는 근본적으로 계산하는 기계

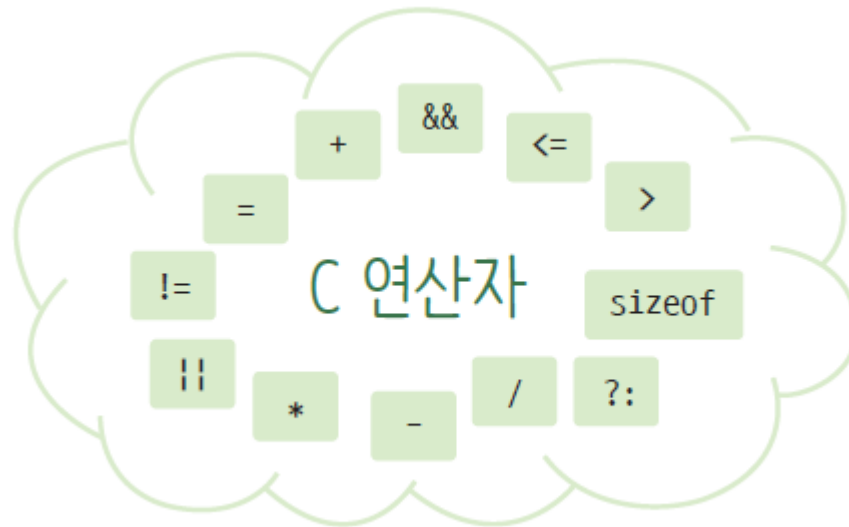
기상청에서는 슈퍼 컴퓨터를 이용하여 날씨를 계산한다.





C언어의 연산자

- 사실상의 업계 표준
- 자바, C++, 파이썬, 자바 스크립트 등의 최신 언어들이 C언어의 연산자를 거의 그대로 사용한다.





수식의 예



```
int x, y;
```

```
x = 3;
```

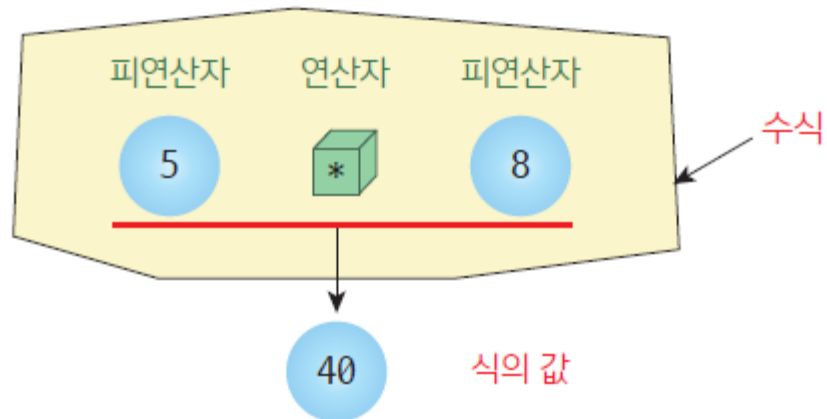
```
y = x*x - 5*x + 6;
```

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



수식

- 수식(expression)
 - 상수, 변수, 연산자의 조합
 - 연산자와 피연산자로 나뉘어진다.





기능에 따른 연산자의 분류

연산자의 분류	연산자	의미
대입	=	오른쪽을 왼쪽에 대입
산술	+ - * / %	사칙연산과 나머지 연산
부호	+ -	양수와 음수 표시
증감	++ --	증가, 감소 연산
관계	> < == != >= <=	오른쪽과 왼쪽을 비교
논리	&& !	논리적인 AND, OR
조건	?	조건에 따라 선택
콤마	,	피연산자들을 순차적으로 실행
비트 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



중간 점검

1. 수식(expression)이란 어떻게 정의되는가?
2. 상수 10도 수식이라고 할 수 있는가?
3. 아래의 수식에서 피연산자와 연산자를 구분하여 보라.
$$y = 10 + 20;$$
4. 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	사용예	결과값
덧셈	+	$7 + 4$	11
뺄셈	-	$7 - 4$	3
곱셈	*	$7 * 4$	28
나눗셈	/	$7 / 4$	1
나머지	%	$7 \% 4$	3



산술 연산자의 예

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

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

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



(참고) 거듭 제곱 연산자는?

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



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
```

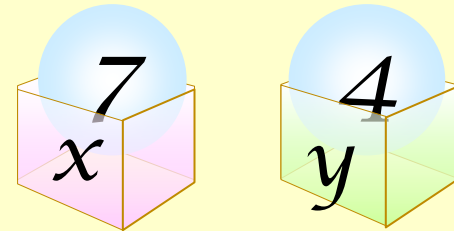
```
    printf("%d / %d = %d", x, y, result);
```

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

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

```
    return 0;
```

```
}
```

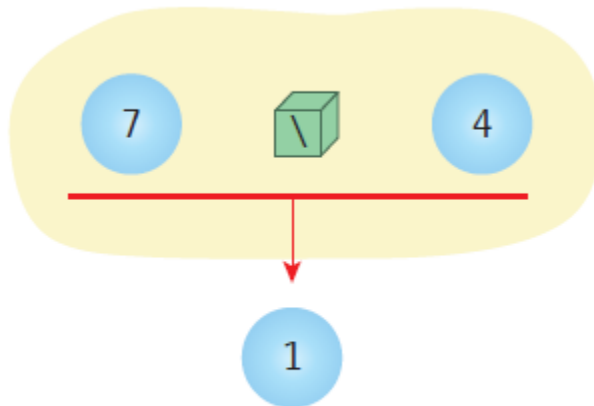


```
두개의 정수를 입력하시오: 7 4
7 + 4 = 11
7 - 4 = 3
7 * 4 = 28
7 / 4 = 1
7 % 4 = 3
```

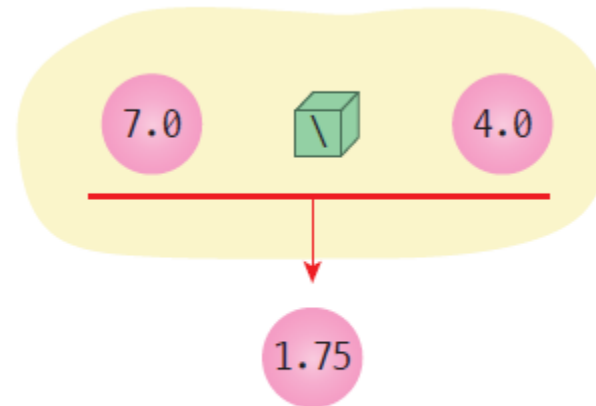


나눗셈 연산자

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



정수와 정수끼리의 나눗셈



실수와 실수끼리의 나눗셈



실수 사칙 연산

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





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
```

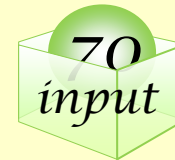
```
    second = input % SEC_PER_MINUTE; // 몇 초
```

```
    printf("%d초는 %d분 %d초입니다. \n",
```

```
           input, minute, second);
```

```
    return 0;
```

```
}
```



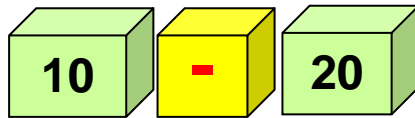
초를 입력하시요: 1000
1000초는 16분 40초 입니다.



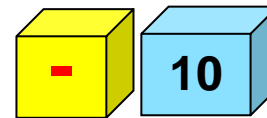
부호 연산자

- 변수나 상수의 부호를 변경

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



이항연산자



단항연산자

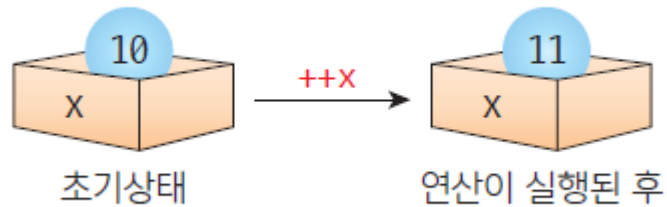
-는 이항
연산자이기도
하고 단항
연산자이기도
하죠





증감 연산자

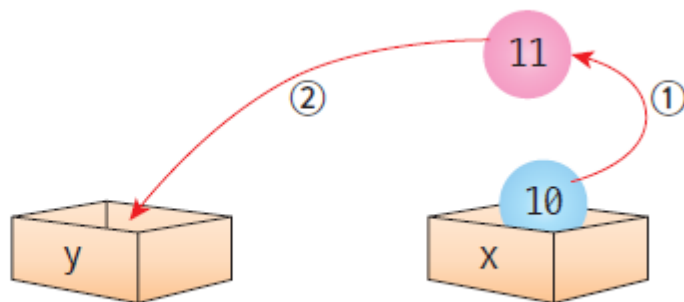
- 증감 연산자: $++$, $--$
- 변수의 값을 하나 증가시키거나 감소시키는 연산자
- (예) $++x$, $--x$;





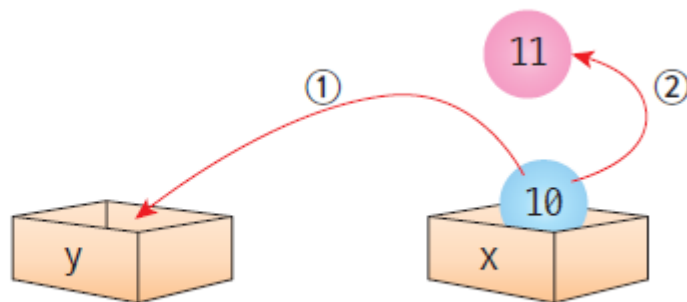
++x와 x++의 차이

`y=++x;`



증가된 x의 값이 y에 대입된다.

`y=x++;`



먼저 대입하고 나중에 증가한다.



증감 연산자 정리

증감 연산자	차이점
$++x$	수식의 값은 증가된 x 값이다.
$x++$	수식의 값은 증가되지 않은 원래의 x 값이다.
$--x$	수식의 값은 감소된 x 값이다.
$x--$	수식의 값은 감소되지 않은 원래의 x 값이다.



증감 연산자의 예

$y = (1 + x++) + 10;$ // 괄호가 있어도 x 값의 증가는 맨 나중에 실행된다.

$x = 10++;$ // 상수에 적용할 수 없다.
 $y = (x+1)++;$ // 수식에 적용할 수 없다.



예제: 증감 연산자

```
#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;
```

```
}
```

먼저 증가하고 증가된 값이
수식에 사용된다.

현재 값을 먼저 수식에 사용
하고 나중에 증가된다.

```
x=10  
++x의 값=11  
x=11
```

```
y=10  
y++의 값=10  
y=11
```



증감 연산자

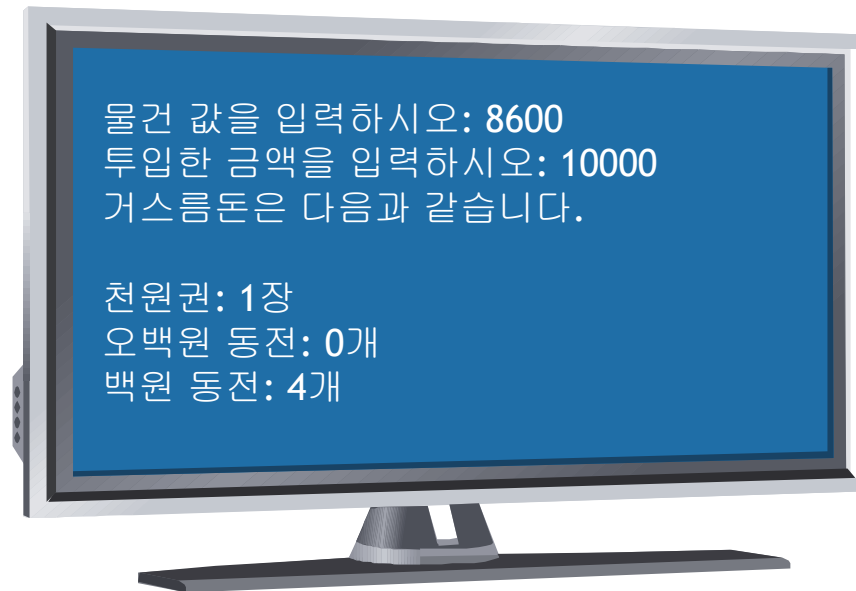
1. 증감 연산자 `x++`를 풀어쓰면 어떻게 되는가?
참고) `temp = x; x = x+1; temp;`
참고) 그러면 `++x`는? $\rightarrow x = x+1; x;$
2. `x++`와 `++x`의 차이점은 무엇인가?
3. `int x=10; printf("%d \n", (1+x++)+2);`의 출력은?





Lab: 거스름돈 계산하기

- 편의점에서 물건을 구입하고 만 원을 냈을 때, 거스름돈의 액수와 점원이 지급해야 할 거스름돈을 화폐와 동전수를 계산하는 프로그램을 작성해보자.





예제

```
#include <stdio.h>

int main(void)
{
    int money, change;
    int price, c5000, c1000, c500, c100;

    printf("물건 값을 입력하십시오: ");
    scanf("%d", &price); // 물건 값을 입력받는다.

    printf("투입한 금액을 입력하십시오: ");
    scanf("%d", &money); // 물건 값을 입력받는다.

    change = money - price; // 거스름돈을 change에 저장

    c1000 = change / 1000; // 남은 잔돈에서 1000원권의 개수를 계산한다.
    change = change % 1000; // 나머지 연산자를 사용하여 남은 잔돈을 계산한다.

    c500 = change / 500; // 남은 잔돈에서 500원 동전의 개수를 계산한다.
    change = change % 500; // 나머지 연산자를 사용하여 남은 잔돈을 계산한다.
}
```




```
printf("\n천원권: %d장\n", c1000);  
printf("오백원 동전: %d개\n", c500);  
printf("백원 동전: %d개\n", c100);  
return 0;  
}
```



대입(배정, 할당) 연산자

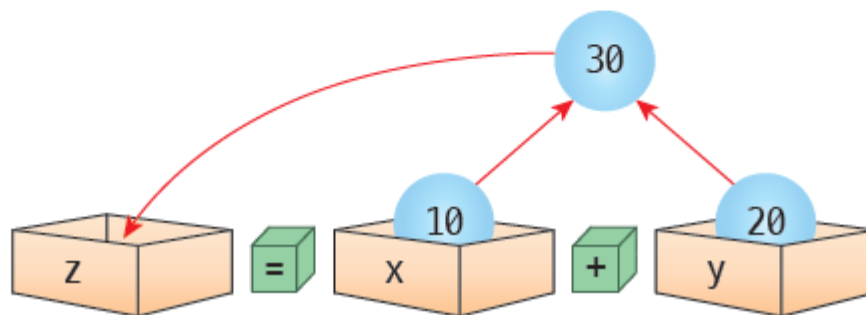
Syntax

대입 연산자

예

변수 수식

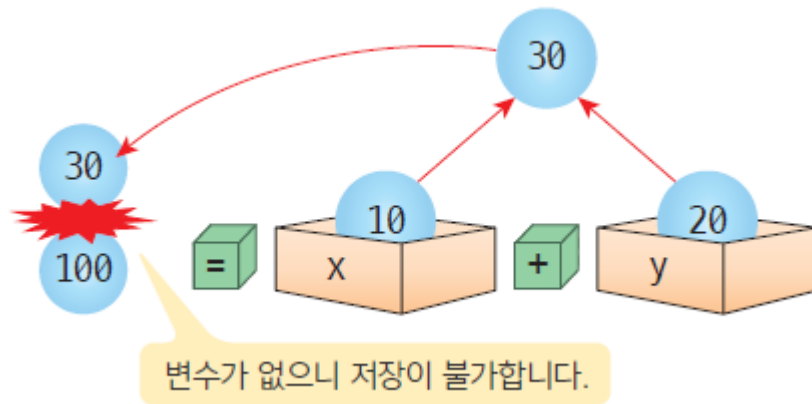
$$z = x + y$$





대입 연산자 주의점

- $100 = x + y;$ // 컴파일 오류!

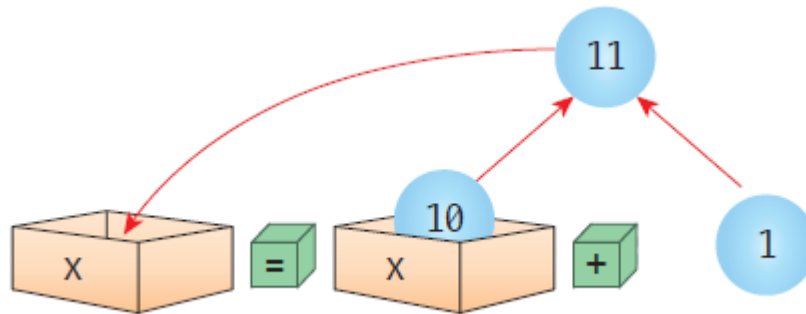




대입 연산자 주의점

수학적으로는 올바르지 않지만
C에서는 올바른 문장임

`x = x + 1;`





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, resulting in 9. This is labeled "덧셈연산의 결과값은 9" (Result of addition operation is 9).
- The assignment operation $x = 9$ is then performed. The result of this operation is 9. This is labeled "대입연산의 결과값은 9" (Result of assignment operation is 9).
- The addition operation $10 + 9$ is then performed, resulting in 19. This is labeled "덧셈연산의 결과값은 19" (Result of addition operation is 19).
- The final result of the entire expression is 19. This is labeled "대입연산의 결과값은 19(현재는 사용되지 않음)" (Result of assignment operation is 19 (not used currently)).

모든 연산에는
결과값이 있고
대입 연산도
결과값이 있습니다.





다음과 같은 문장도 가능하다.

$y = x = 3;$

여러 변수에다가 같은 값을 대입하는 문장을 다음과 같이 작성할 수 있다. 여기서는 먼저 $x = 3$ 이 수행되고 그 결과값인 3이 다시 y 에 대입된다





예제

```
/* 대입 연산자 프로그램 */
```

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

```
int main(void)
```

```
{
```

```
    int x, y;
```

```
    x = 1;
```

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

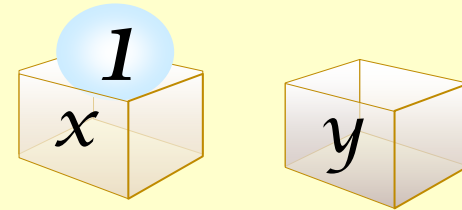
```
    printf("수식 y=x+1의 값은 %d\n", y=x+1);
```

```
    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

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

$x = x + y$ 와 의미가 같음!

$x += y$



복합 대입 연산자

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

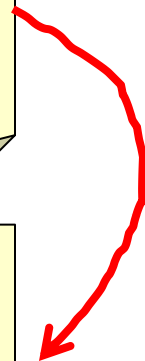


Quiz

- 다음 수식을 풀어서 다시 작성하면?

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

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





복합 대입 연산자

// 복합 대입 연산자 프로그램

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

```
int main(void)
```

```
{
```

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

```
    x += 1;
```

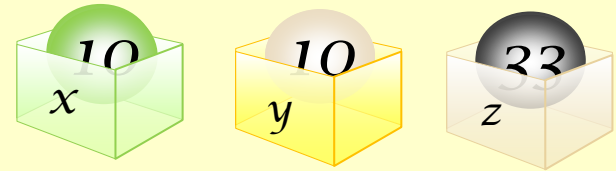
```
    y *= 2;
```

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

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

```
    return 0;
```

```
}
```



x = 11 y = 20 z = 3



오류 주의

오류 주의

다음과 같은 수식은 오류이다. 왜 그럴까?

`++x = 10;` // 등호의 왼쪽은 항상 변수이어야 한다.

`x + 1 = 20;` // 등호의 왼쪽은 항상 변수이어야 한다.

`x =* y;` // `=*` 이 아니라 `*=` 이다.



중간 점검

1. 대입 연산자에서 lvalue가 될 수 있는 것은 무엇인가?
2. 등호(=)가 수학에서의 의미와 다른 점은 무엇인가?
3. 복합 대입 연산자 $x *= y$ 의 의미를 설명하라.
4. $10\%6$ 이 계산되는 과정을 설명하라.
5. 나눗셈 연산인 $10/6$ 의 값은 얼마인가?
6. $x += y$ 와 $x =+ y$ 의 차이점은 무엇인가?



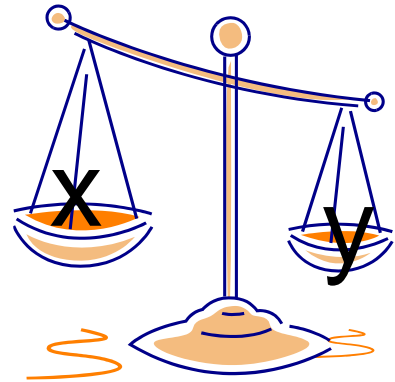


관계 연산자

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

$x == y$

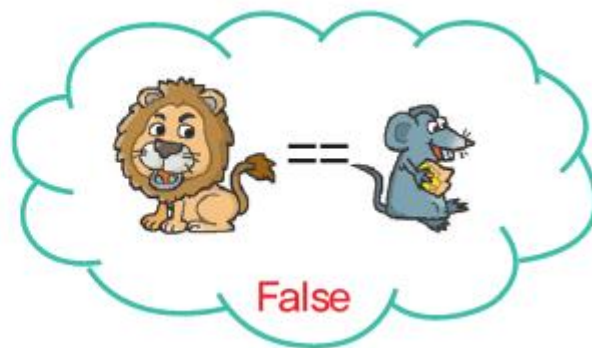
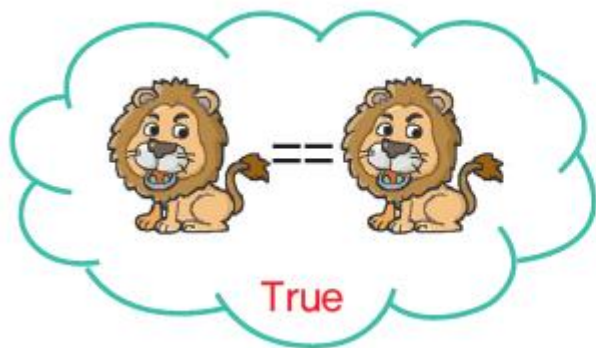
x 와 y의
값이 같은지
비교한다.





관계 연산자

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





관계 연산자의 예

```
1 == 1          // 참(1)
1 != 2          // 참(1)
2 > 1           // 참(1)
x >= y          // x가 y보다 크거나 같으면 참(1) 그렇지 않으면 거짓(0)
```



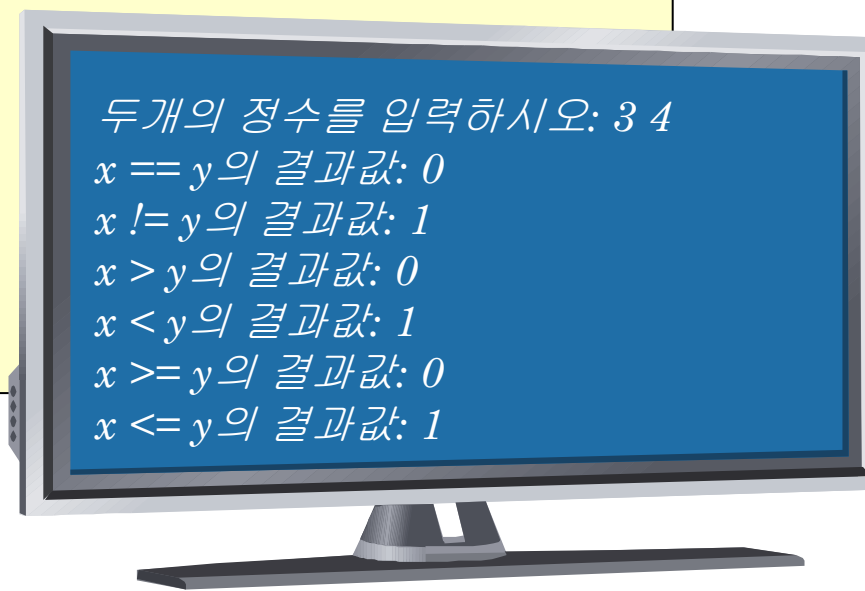

예제

```
#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;
}
```



두개의 정수를 입력하시오: 3 4
x == y의 결과값: 0
x != y의 결과값: 1
x > y의 결과값: 0
x < y의 결과값: 1
x >= y의 결과값: 0
x <= y의 결과값: 1



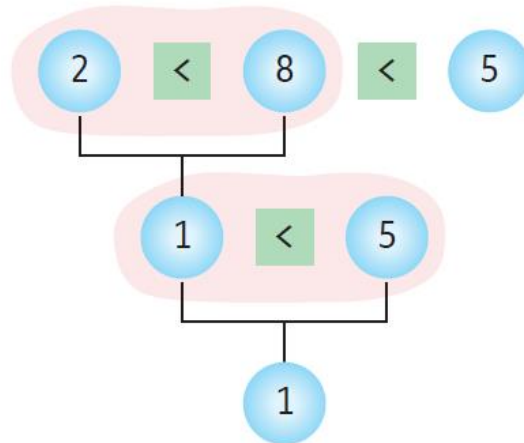
주의할 점!

- $(x = y)$
 - y 의 값을 x 에 대입한다. 이 수식의 값은 x 의 값이다.
- $(x == y)$
 - x 와 y 가 같으면 1, 다르면 0이 수식의 값이 된다.
 - $(x == y)$ 를 $(x = y)$ 로 잘못 쓰지 않도록 주의!



관계 연산자 사용시 주의점

- 수학에서처럼 $2 < x < 5$ 와 같이 작성하면 잘못된 결과가 나온다.



- 올바른 방법: $(2 < x) \ \&\& \ (x < 5)$



실수를 비교하는 경우

- $(1e32 + 0.01) > 1e32$
 - -> 양쪽의 값이 같은 것으로 간주되어서 거짓

- $(\text{fabs}(x-y)) < 0.0001$
 - 올바른 수식

실수는 약간의 오차가
있을 수 있죠!





예제

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

int main(void)
{
    double a, b;
    a = (0.3 * 3) + 0.1;
    b = 1;
    printf("a==b의 결과: %d \n", a == b);

    printf("fabs(a-b)<0.00001의 결과: %d \n", fabs(a - b) < 0.0001);
    return 0;
}
```

a==b의 결과: 0

fabs(a-b)<0.00001의 결과: 1



정가 점검

1. 관계 수식의 결과로 생성될 수 있는 값은 무엇인가?
2. $(3 \geq 2) + 5$ 의 값은?
3. 부동소수점수 2개를 비교할 때는 어떻게 하는 것이 좋은가?



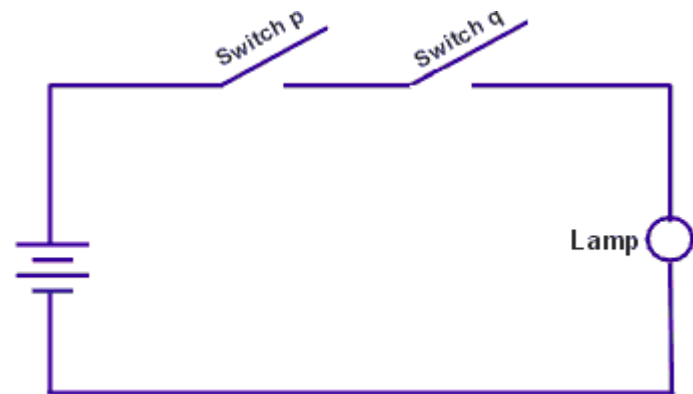


논리 연산자

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

$x \ \&\& \ y$

x와 y가 모두 참인 경우에만 참이 된다.





논리 연산자

연산	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





AND 연산자

- 어떤 회사에서 신입 사원을 채용하는데 나이가 30살 이하이고 토익 성적이 700점 이상 이라는 조건을 걸었다고 가정하자.

²⁷
(age <= 30) && (⁸⁰⁰toeic >= 700)

참(1) 참(1)

참(1)



OR 연산자

- 신입 사원을 채용하는 조건이 변경되어서 나이가 30살 이하이거나 토익 성적이 700점 이상이면 된다고 하자.

$$\begin{array}{c} 27 \qquad \qquad \qquad 699 \\ \textcircled{\text{age}} \leq 30 \quad || \quad \textcircled{\text{toeic}} \geq 700 \\ \underbrace{\hspace{10em}} \\ \text{참(1)} \qquad \qquad \text{거짓(0)} \\ \underbrace{\hspace{10em}} \\ \text{참(1)} \end{array}$$



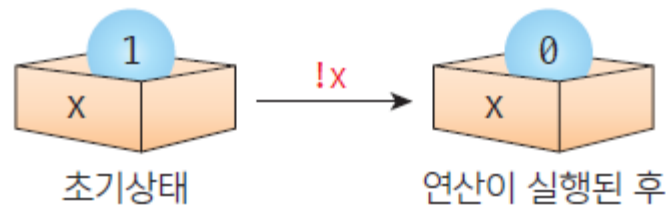
논리 연산자의 예

- “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.



NOT 연산자

- 피연산자의 값이 참이면 연산의 결과값을 거짓으로 만들고, 피연산자의 값이 거짓이면 연산의 결과값을 참으로 만든다.



```
result = !1;           // result에는 0가 대입된다.  
result = !(2==3);      // result에는 1이 대입된다.
```



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



예제

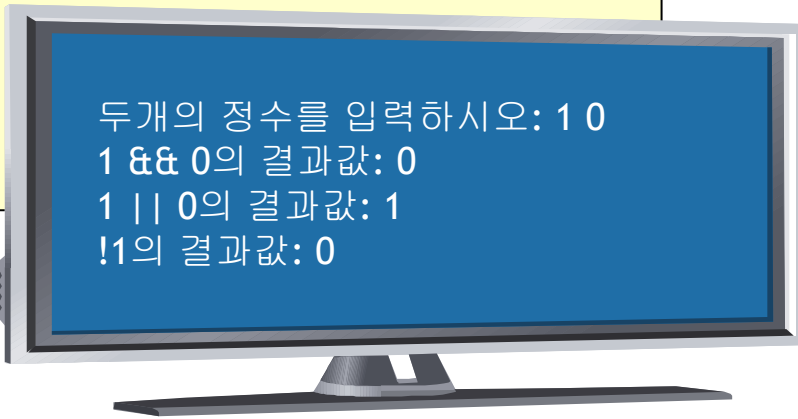
```
#include <stdio.h>

int main(void)
{
    int x, y;

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

    printf("%d && %d의 결과값: %d", x, y, x && y);
    printf("%d || %d의 결과값: %d", x, y, x || y);
    printf("!d의 결과값: %d", x, !x);

    return 0;
}
```



두개의 정수를 입력하시오: 1 0
1 && 0의 결과값: 0
1 || 0의 결과값: 1
!1의 결과값: 0



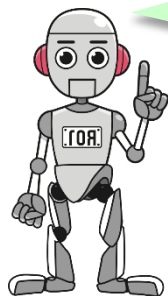
단축 계산

- && 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)`



첫번째 연산자가
거짓이면 다른
연산자는 계산할
필요가 없게!!

++나 --는
실행이 안될 수도
있으니
주의하세요.





장간 점검

1. 다음의 조건에 해당하는 논리 연산식을 만들어 보시오. 변수는 적절하게 선언되어 있다고 가정한다.

“무주택 기간 3년 이상, 가구주의 연령이 40세 이상, 가족의 수가 3명 이상”

2. 상수 10은 참인가 거짓인가?

3. 수식 !3의 값은?

4. 단축 계산의 예를 들어보라.





Lab: 윤년

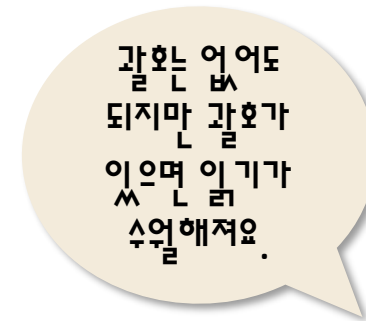
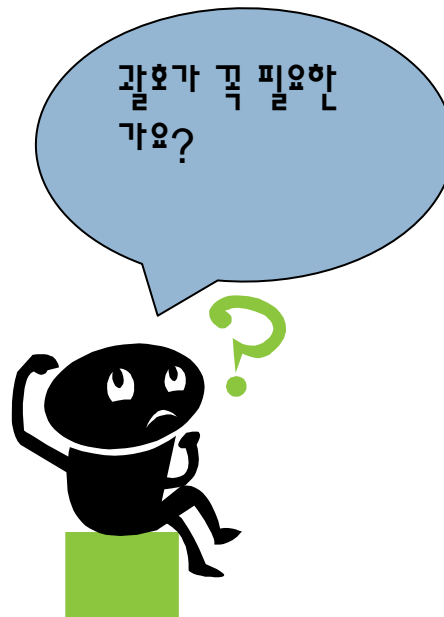
- 윤년의 조건
 - 연도가 4로 나누어 떨어진다.
 - 100으로 나누어 떨어지는 연도는 제외한다.
 - 400으로 나누어 떨어지는 연도는 윤년이다.





Lab: 윤년

- 윤년의 조건을 수식으로 표현
 - $((year \% 4 == 0) \ \&\& \ (year \% 100 != 0)) \ || \ (year \% 400 == 0)$






Lab: 윤년

```
#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);

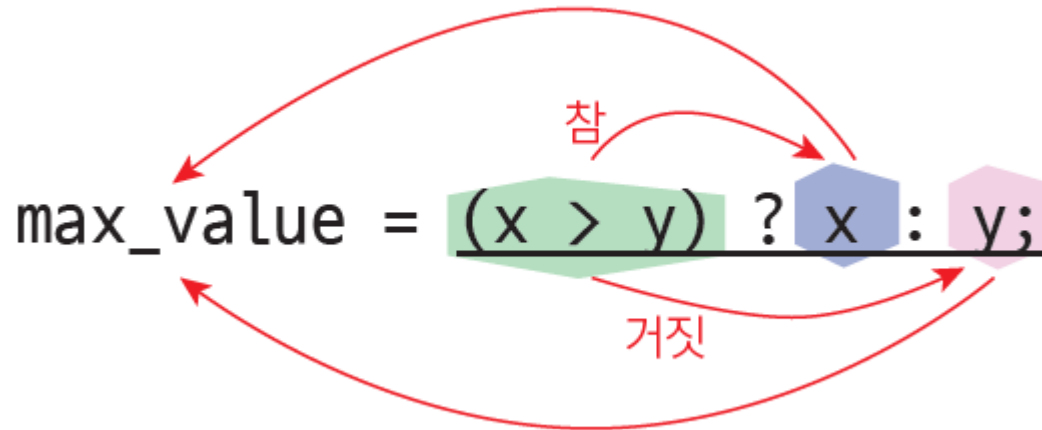
    return 0;
}
```



연도를 입력하시오: 2012
result=1



조건 연산자



```
absolute_value = (x > 0) ? x : -x;    // 절대값 계산  
max_value = (x > y) ? x : y;    // 최대값 계산  
min_value = (x < y) ? x : y;    // 최소값 계산  
(age > 20) ? printf("성인\n") : printf("청소년\n");
```



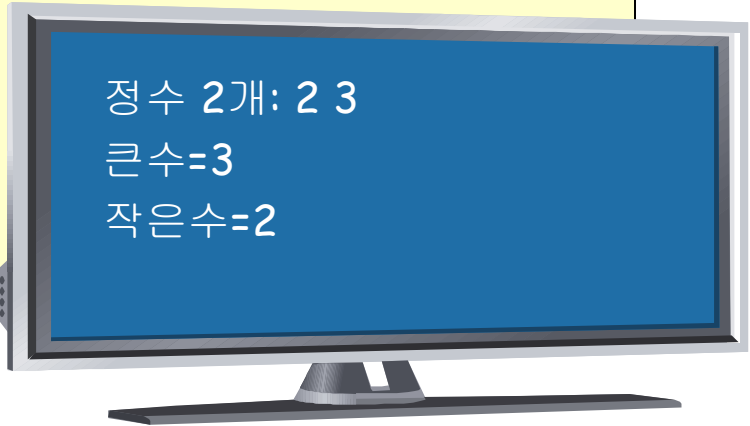
예제

```
// 조건 연산자 프로그램
#include <stdio.h>

int main(void)
{
    int x,y;

    printf("정수 2개: ");
    scanf("%d %d", &x, &y);

    printf("큰 수=%d \n", (x > y) ? x : y);
    printf("작은 수=%d \n", (x < y) ? x : y);
}
```

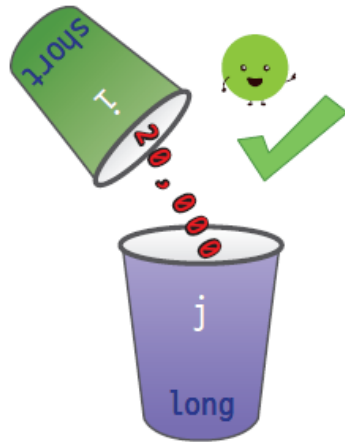


정수 2개: 2 3
큰수=3
작은수=2



형변환

- 형변환(type conversion)이란 실행 중에 데이터의 타입을 변경하는 것이다



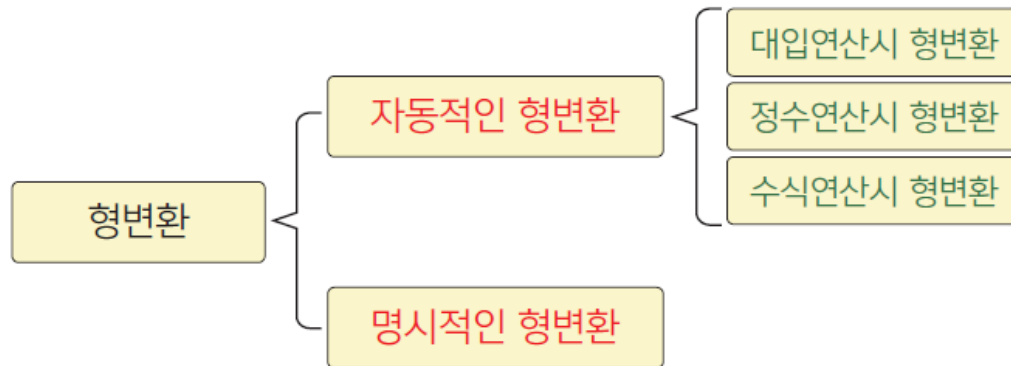
형변환을
잘못하면
데이터의 일부가
사라질 수도
있기 때문에
주의하여야
한다.





형변화

- 연산시에 데이터의 유형이 변환되는 것



변수의 타입이
변경되는 것이
아니고 변수에
저장되는
데이터의 타입이
변경됩니다.

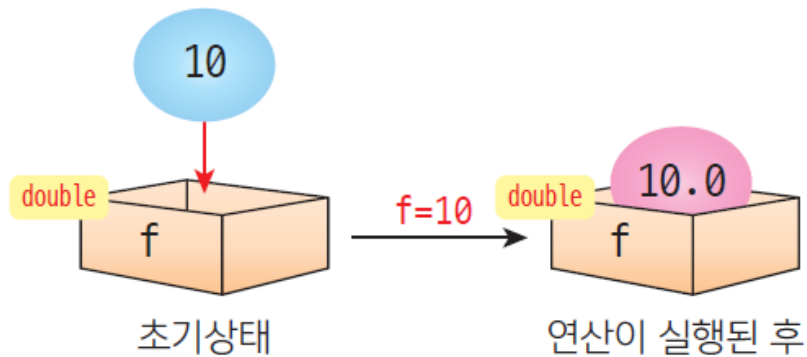




대입 연산자의 자동적인 형변환

- 올림 변환

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

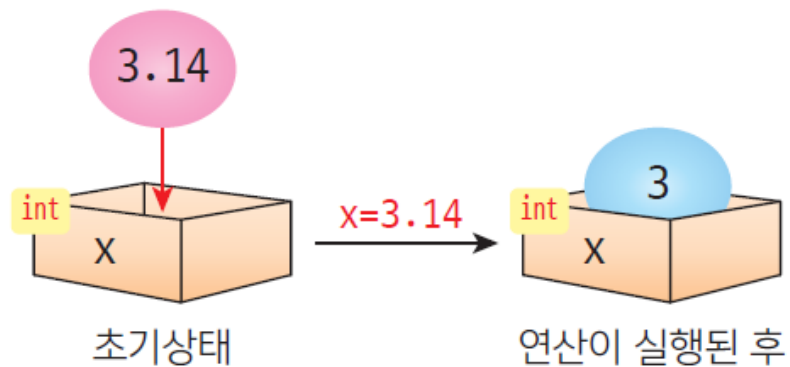




대입 연산시의 자동적인 형변환

- 내림변환

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





정수형끼리 형변환

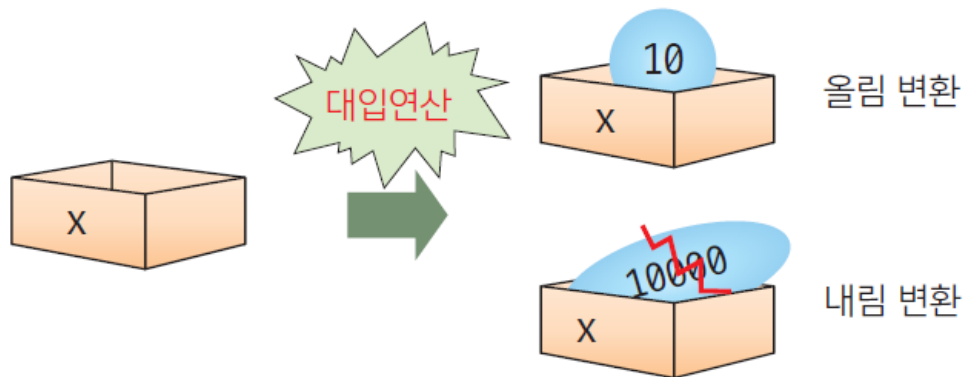
```
char x;
```

```
x = 10;
```

```
// OK
```

```
x = 10000;
```

```
// 상위 바이트는 없어진다.
```





올림 변환과 내림 변환

```
#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;
}
```

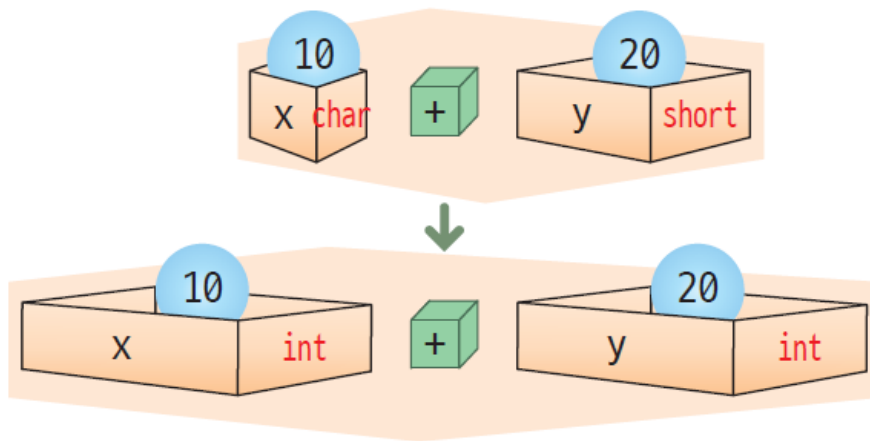
```
c:\...\convert1.c(10) : warning C4305: '=' : 'int'에서 'char'(-)로 잘립니다
.
c:\...\convert1.c(11) : warning C4244: '=' : 'double'에서 'int'(-)로 변환하면서
데이터가 손실될 수 있습니다.

c=16, i=11, f=30.000000
```



정수 연산시의 자동적인 형변환

- 정수 연산시 **char**형이나 **short**형의 경우, 자동적으로 **int**형으로 변환하여 계산한다.



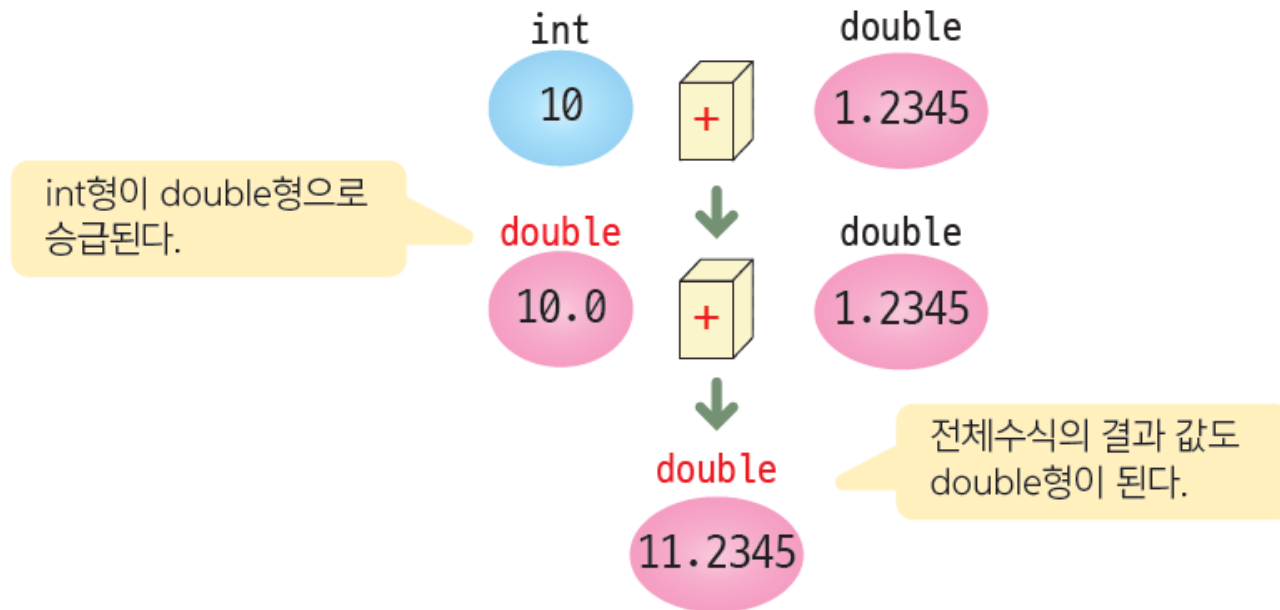
char나 short형은 int형으로
통일하여서 처리합니다.





수식에서의 자동적인 형변화

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





명시적인 형변환

Syntax

형변환

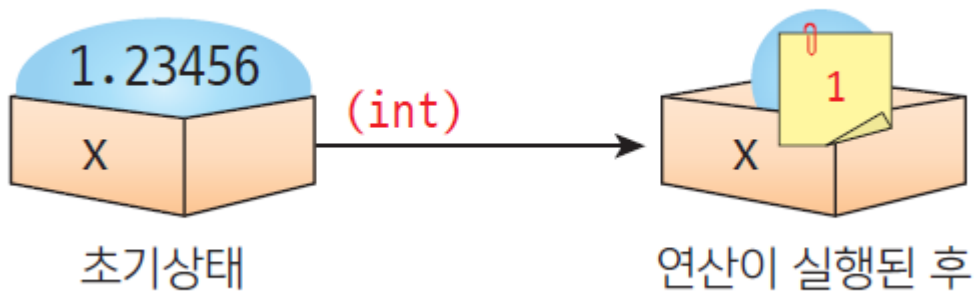
예

자료형 $(int)1.23456$ 수식
 $(double) x$
 $(long) (x+y)$

// int형으로 변환

// double형으로 변환

// long형으로 변환





예제

```
#define _CRT_SECURE_NO_WARNINGS  
#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);
```

5/4는 1이 되고
이것이 1.0이
된다.

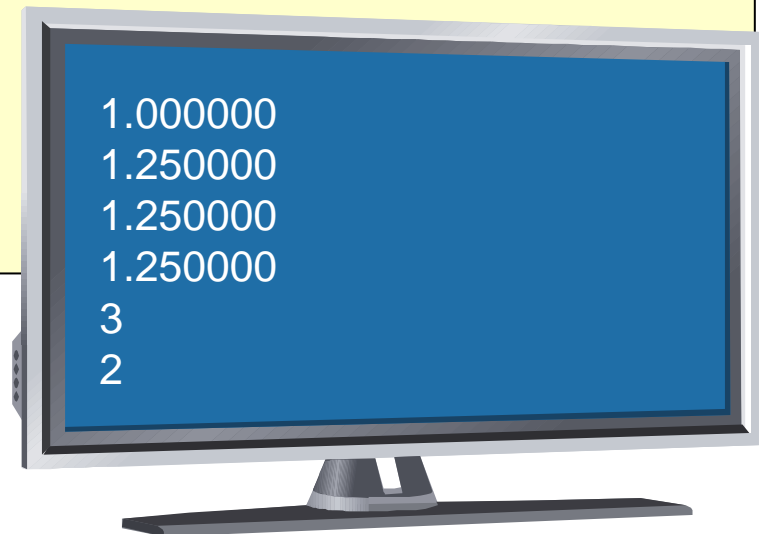
5가 5.0으로 되어서
전체 결과가
1.25가 된다.



예제

```
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;  
}
```

1.3이 1이 되고
1.8도 1이 되어서
최종 결과는 2가
된다.





장간 점검

1. 내림 변환과 올림 변환을 설명하라.
2. `int`형 변수 `x`를 `double`형으로 형변환하는 문장을 써보라.
3. 하나의 수식에 정수와 부동소수점수가 섞여 있으면 어떻게 되는가?





Q & A

