

CHAPTER

05

수식과 연산자

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

Contents

3

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

5.8

비트 연산자

5.9

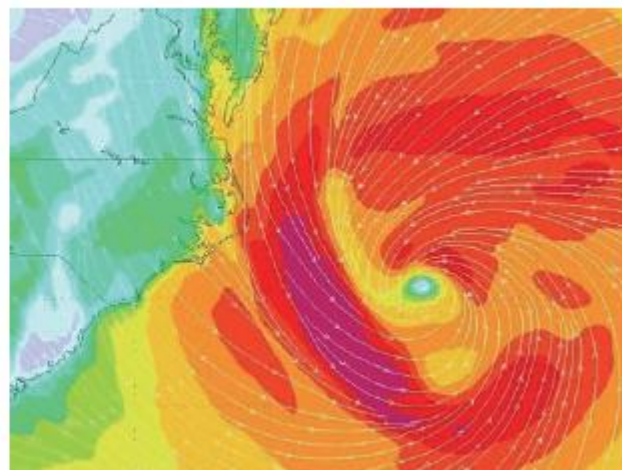
형변환

5.10

연산자의 우선 순위와 결합 규칙

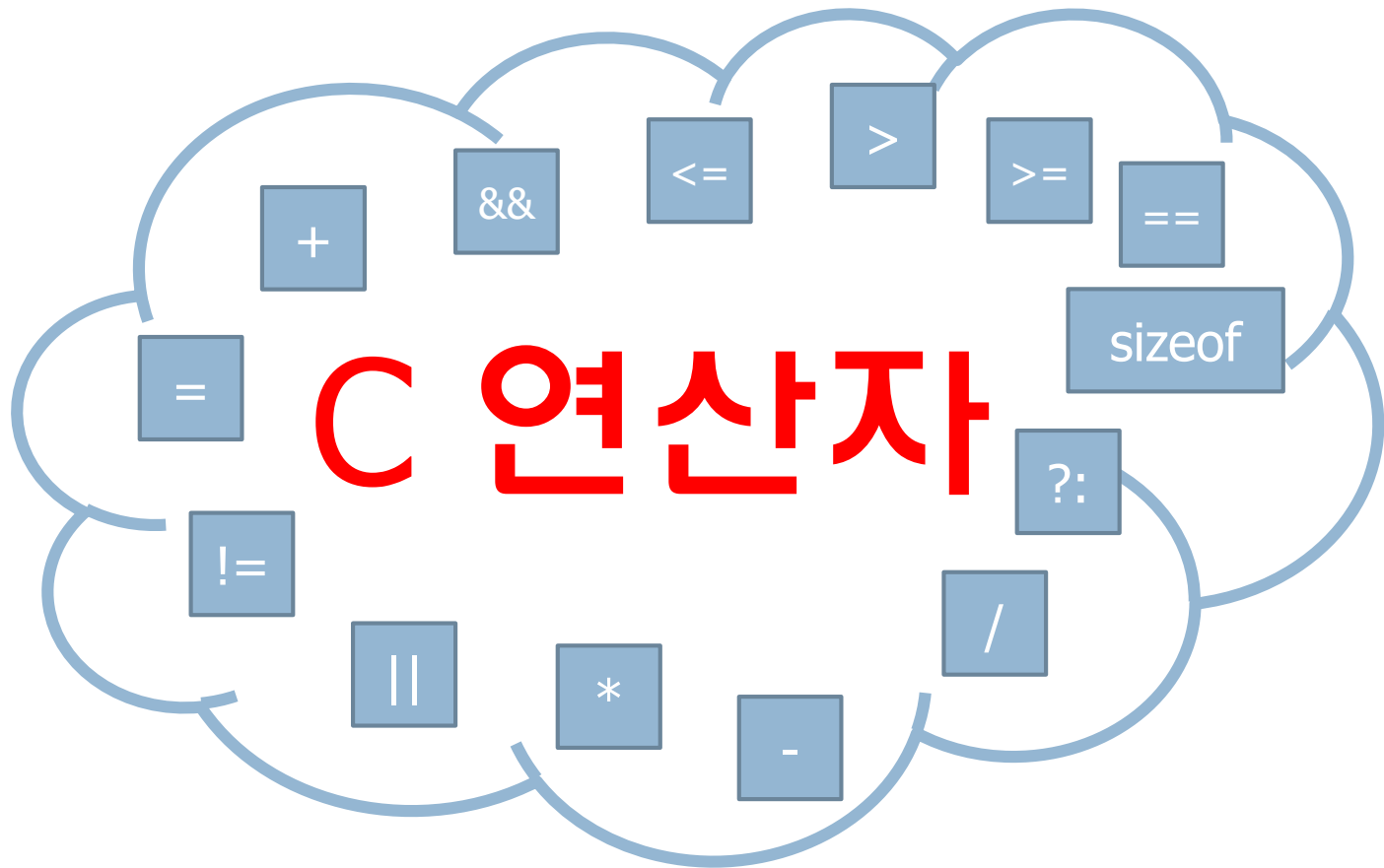
컴퓨터는 근본적으로 계산하는 기계

4



C 연산자

5



수식의 예

6



```
int x, y;
```

```
x = 3;
```

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

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

수식

7

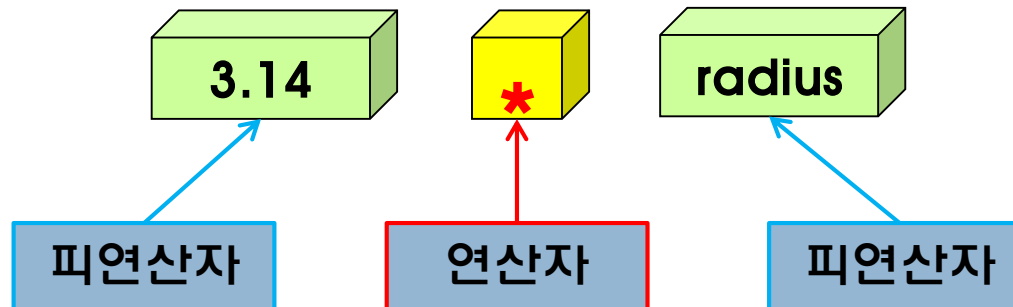
□ 수식(expression)

$x + y$

$x * x + 5 * x + 6$

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

- 상수, 변수, 연산자의 조합
- 연산자와 피연산자로 나누어진다.



기능에 따른 연산자의 분류

8

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
coma	,	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

9

- 단항 연산자: 피연산자의 수가 1개

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

- 이항 연산자: 피연산자의 수가 2개

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

- 삼항 연산자: 연산자의 수가 3개

```
x ? y : z
```

Contents

10

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

5.8

비트 연산자

5.9

형변환

5.10

연산자의 우선 순위와 결합 규칙

산술 연산자

11

- 산술 연산: 컴퓨터의 가장 기본적인 연산
- 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	$x\%y$

산술 연산자의 예

12

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

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

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



(참고) 거듭 제곱 연산자는?

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

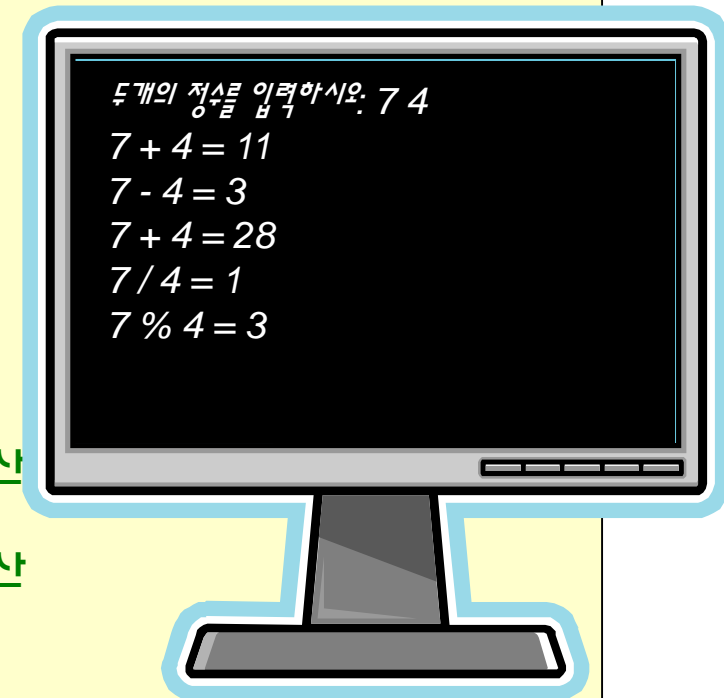
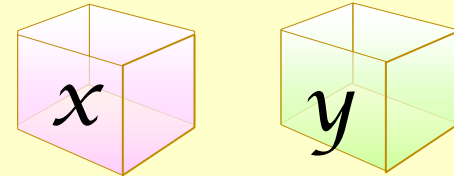
예제

13

```
#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", x, y, result);

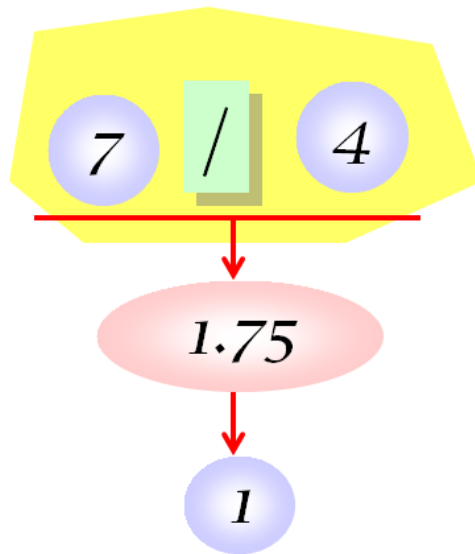
    result = x - y;
    printf("%d - %d = %d", x, y, result);
    result = x * y;
    printf("%d * %d = %d", x, y, result);
    result = x / y;
    printf("%d / %d = %d", x, y, result);
    result = x % y;
    printf("%d %% %d = %d", x, y, result);
    return 0;
}
```



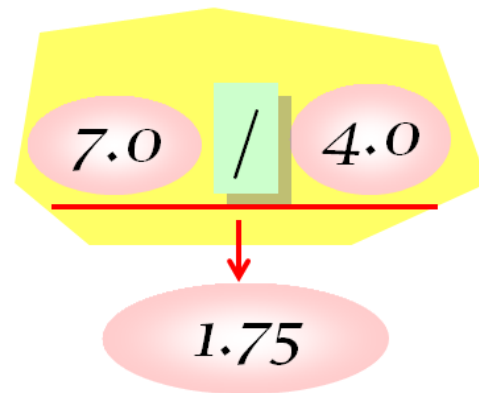
나눗셈 연산자

14

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



정수와 정수 끼리의 나눗셈.



실수와 실수 끼리의 나눗셈.

실수 사칙 연산

15

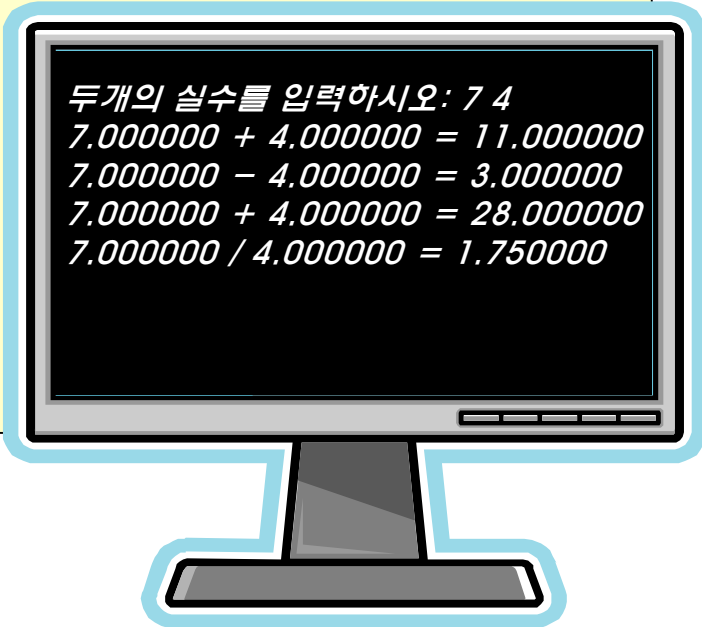
```
int main()
{
    double x, y, result;

    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

16

- 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.
- 나머지 연산자를 이용한 짝수와 홀수를 구분
 - ▣ $x \% 2$ 가 0이면 짝수
- 나머지 연산자를 이용한 5의 배수 판단
 - ▣ $x \% 5$ 가 0이면 5의 배수

나머지 연산자

17

// 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

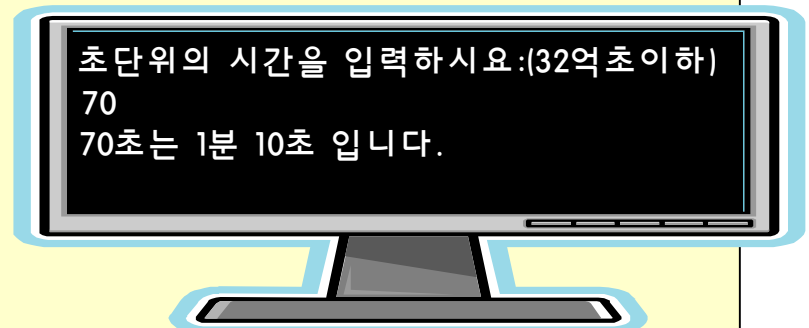
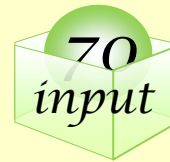
second = input % SEC_PER_MINUTE; // 몇 초

printf("%d초는 %d분 %d초입니다. □n",

input, minute, second);

return 0;

}

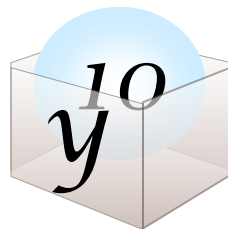
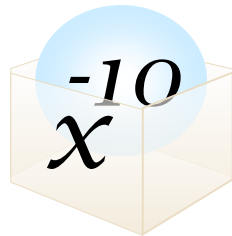
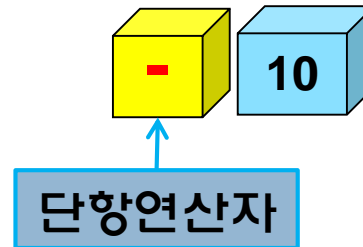
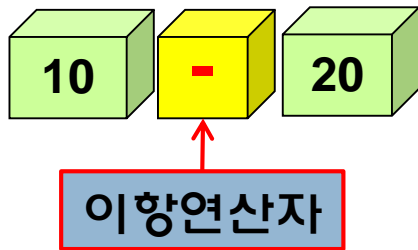


부호 연산자

18

□ 변수나 상수의 부호를 변경

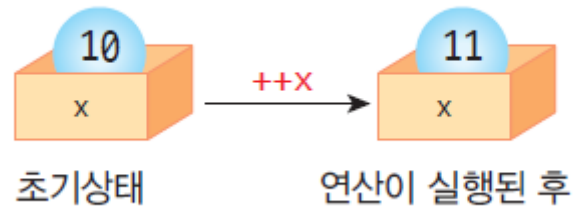
```
x = -10;  
y = -x; // 변수 y의 값은 10이 된다.
```



증감 연산자

19

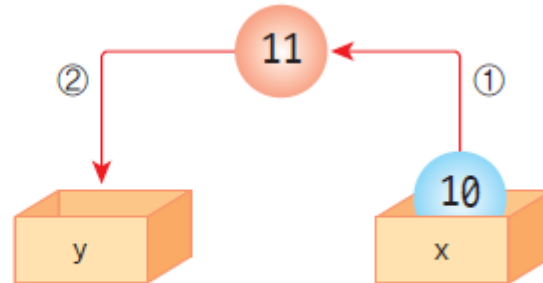
- 증감 연산자: ++, --
- 변수의 값을 하나 증가시키거나 감소시키는 연산자
- ++X, --X;



++x와 x++의 차이

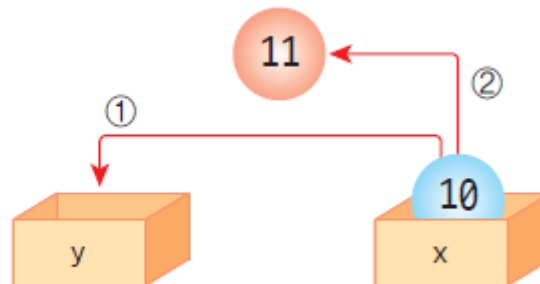
20

`y=++x;`



증가된 x의 값이 y에 대입된다.


`y=x++;`

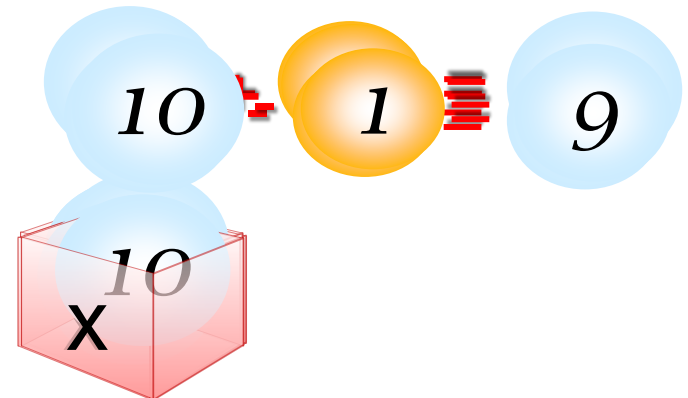


먼저 대입하고 나중에 증가한다.

증감 연산자

21

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.



Quiz

22

□ nextx와 nexty의 값은?

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



예제: 중감 연산자

23

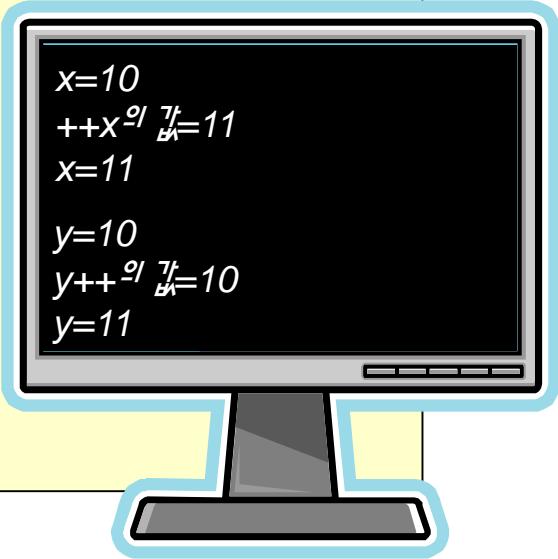
```
#include <stdio.h>
int main(void)
{
    int x=10, y=10;

    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

예제: 증감 연산자

24

// 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);

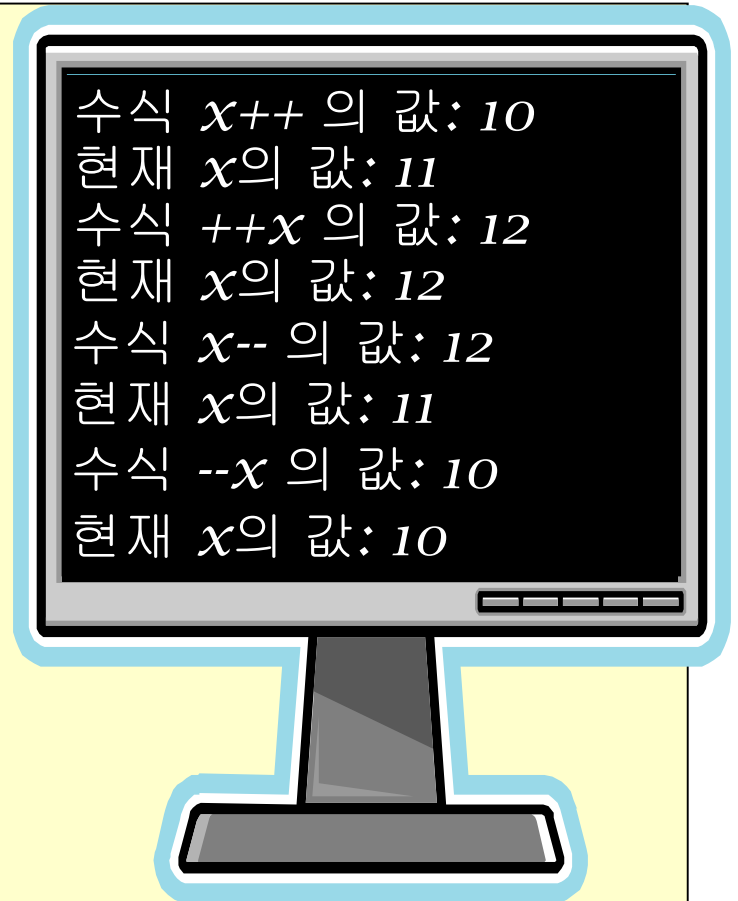
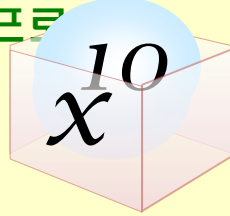
printf("수식 x-- 의 값: %d □n", x--);

printf("현재 x의 값: %d □n", x);

printf("수식 --x 의 값: %d □n", --x);

printf("현재 x의 값: %d □n", x);

}



Lab: 거스름돈 계산하기

25

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



```
#include <stdio.h>
int main(void)
{
    int user, change = 0;
    int price, c5000, c1000, c500, c100;

    printf("물건 값을 입력하시오: ");
    scanf("%d", &price); // 물건 값을 입력받는다.
    printf("사용자가 낸 돈: ");
    scanf("%d", &user);
    change = user - price;           // 거스름돈을 change에 저장
```

c5000 = change / 5000; // 몫 연산자를 사용하여 5000원권의 개수를 계산한다.
change = change % 5000; // 나머지 연산자를 사용하여 남은 잔돈을 계산한다.

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.

c100 = change / 100; // 남은 잔돈에서 100원 동전의 개수를 계산한다.
change = change % 100; //나머지 연산자를 사용하여 남은 잔돈을 계산한다.

printf("오천원권: %d장\n", c5000);
printf("천원권: %d장\n", c1000);
printf("오백원 동전: %d개\n", c500);
printf("백원 동전: %d개\n", c100);
return 0;

}

Contents

28

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

5.8

비트 연산자

5.9

형변환

5.10

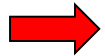
연산자의 우선 순위와 결합 규칙

대입(배정, 할당) 연산자

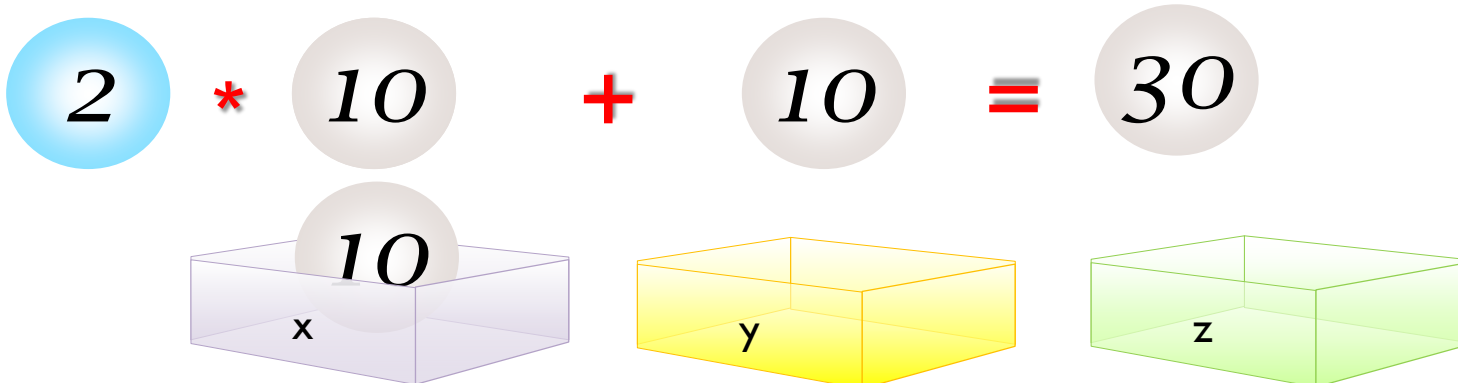
29

- 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

30

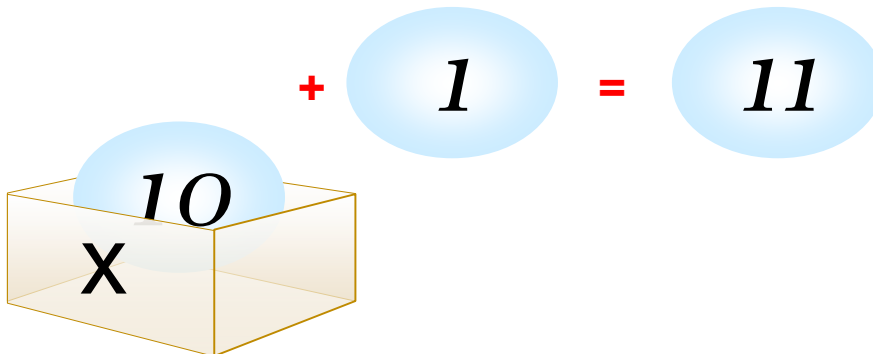
- 왼쪽에는 항상 변수가 와야 한다.

```
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

31

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

Diagram illustrating the evaluation of the expression $y = 10 + (x = 2 + 7);$ using curly braces to group sub-expressions and red arrows to show the flow of evaluation.

- The innermost expression $x = 2 + 7$ is evaluated first, resulting in 9. This is labeled "덧셈연산의 결과값은 9" (Result of addition operation is 9).
- The assignment operation $x = 2 + 7$ is then evaluated, resulting in 9. This is labeled "대입연산의 결과값은 9" (Result of assignment operation is 9).
- The addition operation $10 + (x = 2 + 7)$ is evaluated next, resulting in 19. This is labeled "덧셈연산의 결과값은 19" (Result of addition operation is 19).
- The final assignment operation $y = 10 + (x = 2 + 7);$ is evaluated, resulting in 19. This is labeled "대입연산의 결과값은 19" (Result of assignment operation is 19).

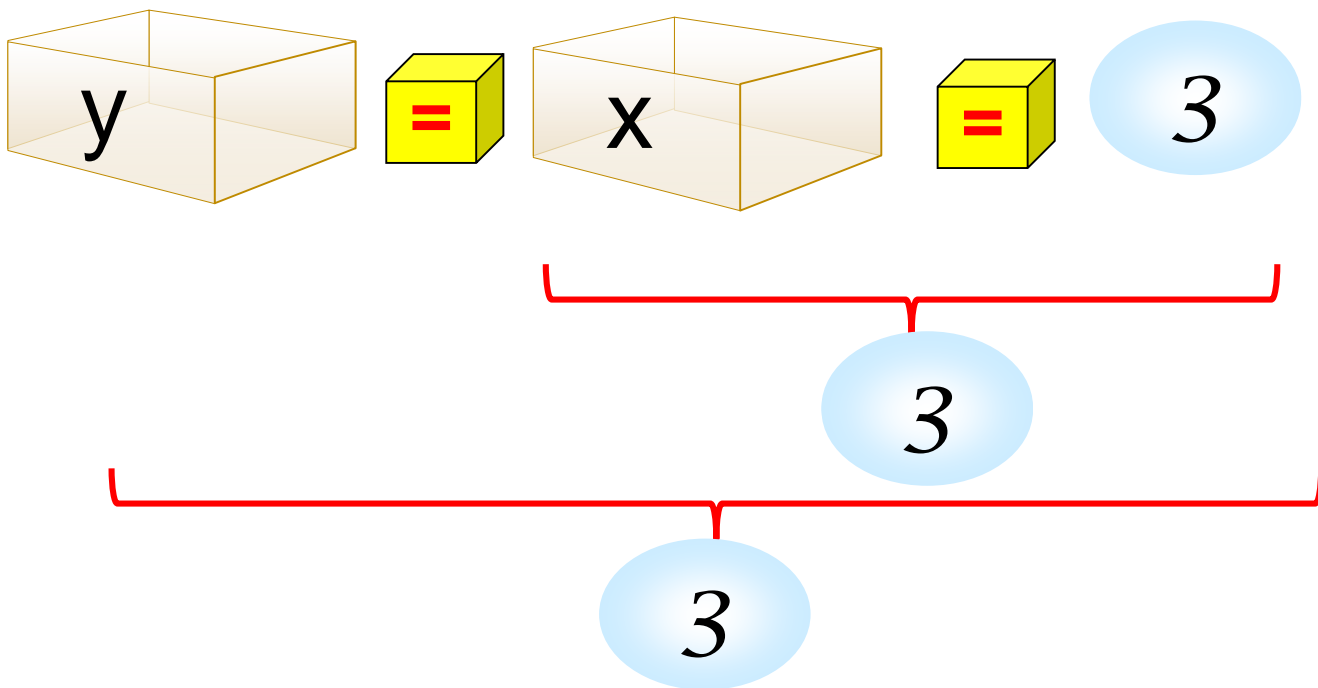
모든 연산에는
결과값이 있고
대입 연산도
결과값이 있습니다.



대입 연산의 결과값

32

$y = x = 3;$



예제

33

```
/* 대입 연산자 프로그램 */
```

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

```
int main(void)
```

```
{
```

```
→ int x, y;
```

```
    x = 1;
```

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

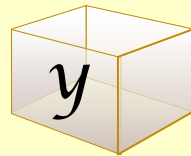
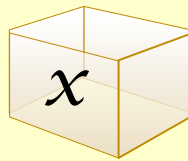
```
    printf("수식  $y=x+1$ 의 값은 %d\n", y=x+1);
```

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

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

```
    return 0;
```

```
}
```



수식 $x+1$ 의 값은 2

수식 $y=x+1$ 의 값은 2

수식 $y=10+(x=2+7)$ 의 값은 19

수식 $y=x=3$ 의 값은 3

복합 대입 연산자

34

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

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

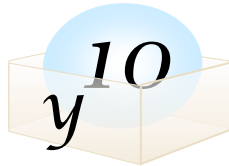
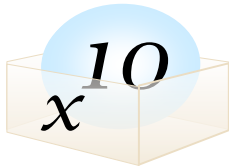
```
x += 1      // x = x + 1
x *= y + 1   // x = x * (y + 1)
x %= x + y   // x = x % (x + y)
```

복합 대입 연산자

35



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



11

55

44

복합 대입 연산자

36

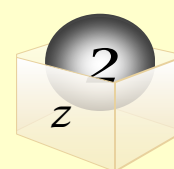
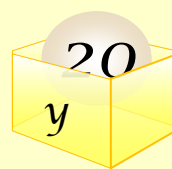
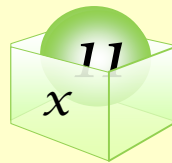
// 복합 대입 연산자 프로그램

#include <stdio.h>

int main(void)

{

→ int x = 10, y = 10, z = 33;



x += 1; // x = x + 1;

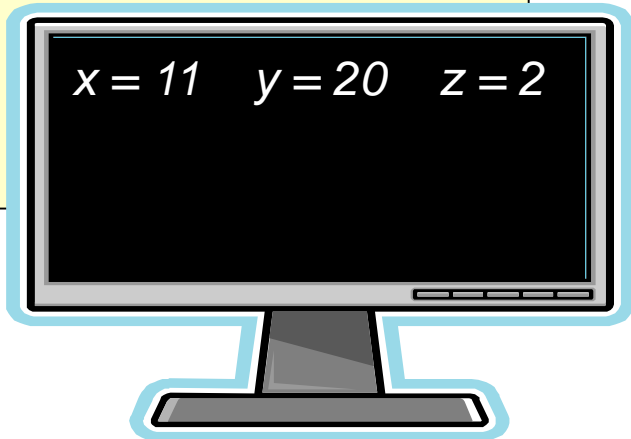
y *= 2; // y = y * 2;

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

Contents

37

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

5.8

비트 연산자

5.9

형변환

5.10

연산자의 우선 순위와 결합 규칙

관계 연산자

38

- 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

관계 연산자 사용예

39

- $1 == 1$ // 참(1)
- $1 != 2$ // 참(1)
- $2 > 1$ // 참(1)
- $x \geq y$ // x가 y보다 크거나 같으면 참(1)

예제

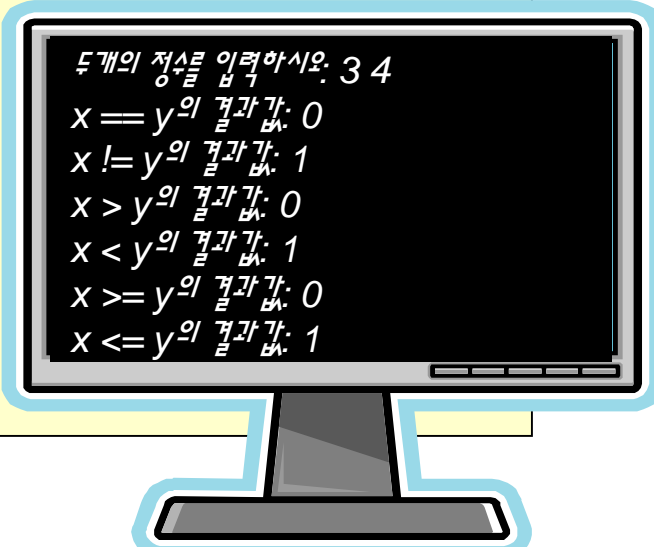
40

```
#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!

41

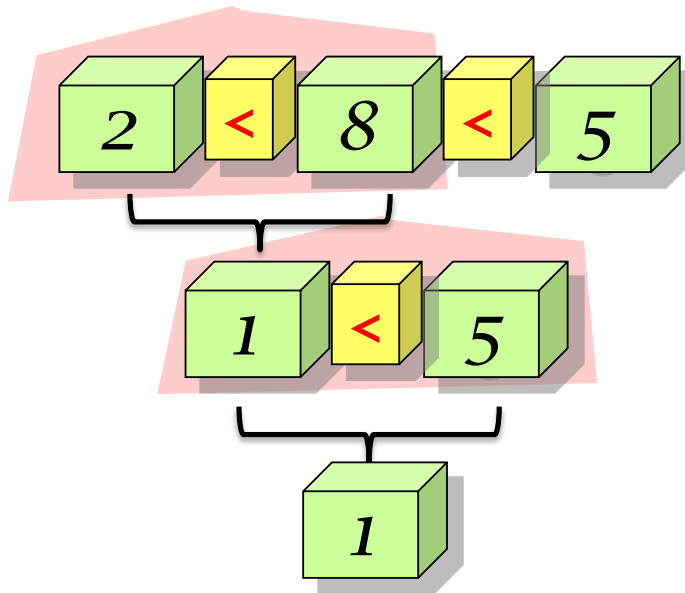
- $(x = y)$
 - ▣ x 의 값을 y 에 대입한다. 이 수식의 값은 x 의 값이다.

- $(x == y)$
 - ▣ 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

42

- 잘못된 방법: $2 < x < 5$



- 올바른 방법: $(2 < x) \&\& (x < 5)$

실수를 비교하는 경우

43

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



Contents

44

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

5.8

비트 연산자

5.9

형변환

5.10

연산자의 우선 순위와 결합 규칙

논리 연산자

45

- 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

46

x	y	x && y	x y	!x
참	참	참	참	거짓
참	거짓	거짓	참	거짓
거짓	참	거짓	참	참
거짓	거짓	거짓	거짓	참

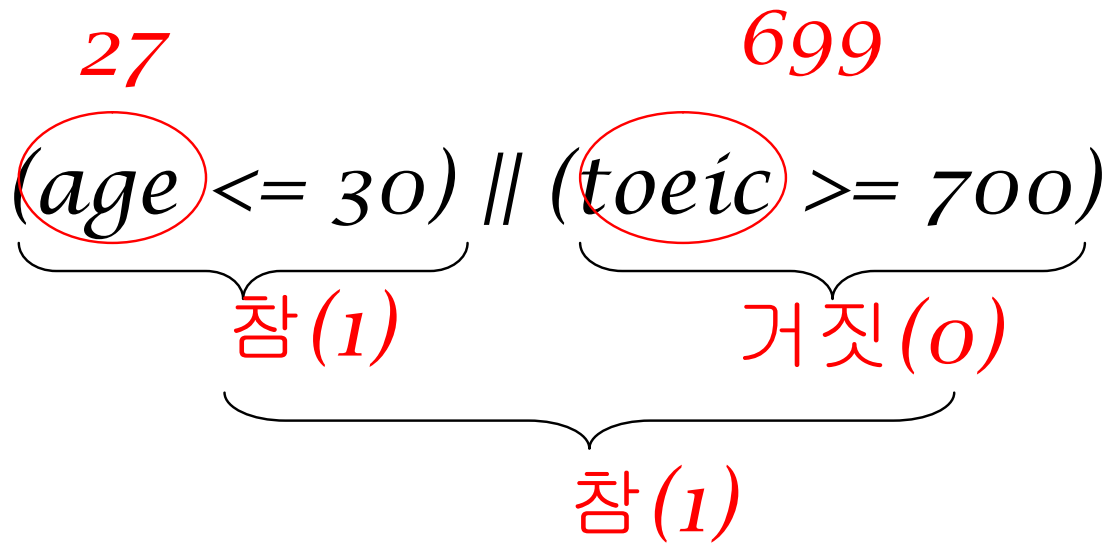
AND 연산자

47

27
800
(age <= 30) && (toeic >= 700)
참 (1) 참 (1)
참 (1)

OR 연산자

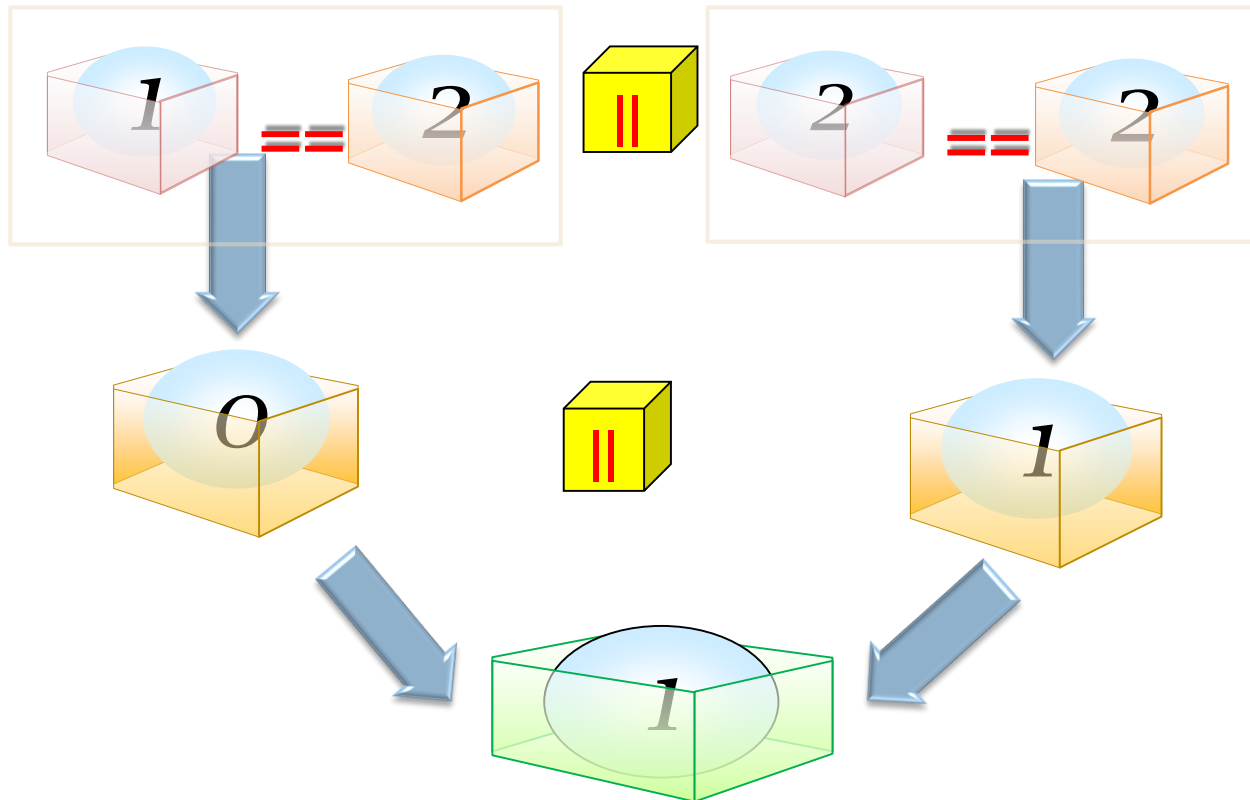
48



논리 연산자의 계산 과정

49

- 논리 연산의 결과값은 항상 1 또는 0이다.
- (예) $(1 == 2) \parallel (2 == 2)$



논리 연산자의 예

50

- “x는 1, 2, 3중의 하나인가 “
 - ▣ `(x == 1) || (x == 2) || (x == 3)`

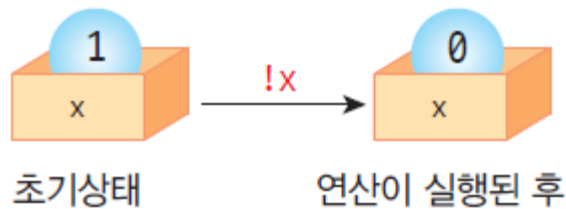
- “x가 60이상 100미만이다.”
 - ▣ `(x >= 60) && (x < 100)`

- “x가 0도 아니고 1도 아니다.”
 - ▣ `(x != 0) && (x != 1)` `// x≠0 이고 x≠1이다.`

NOT 연산자

51

- 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

52

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

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

예제

53

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

```
int main(void)
```

```
{
```

```
    int x, y;
```

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

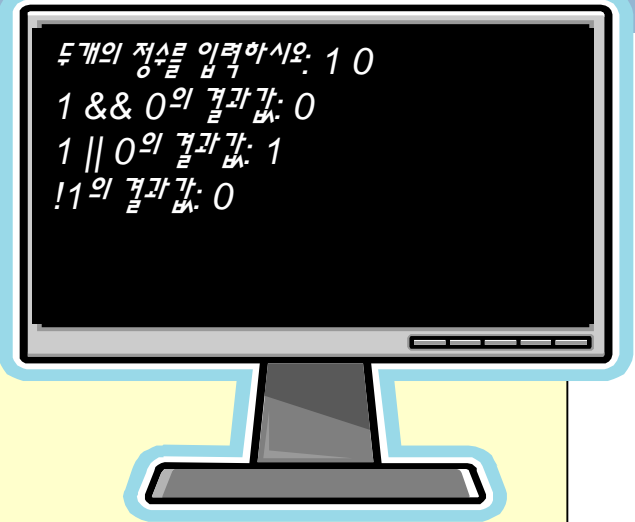
```
    printf("%d && %d의 결과값: %d", x, y, x && y);
```

```
    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

54

- && 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 )
```

Lab: 윤년

55

February 29



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



실습: 윤년

56

□ 윤년의 조건을 수식으로 표현

▣ $((\text{year} \% 4 == 0) \ \&\& \ (\text{year} \% 100 \neq 0)) \ || \ (\text{year} \% 400 == 0)$

실습: 윤년

57

```
// 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 ", result);
```

```
    return 0;
```

```
}
```



Contents

58

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

5.8

비트 연산자

5.9

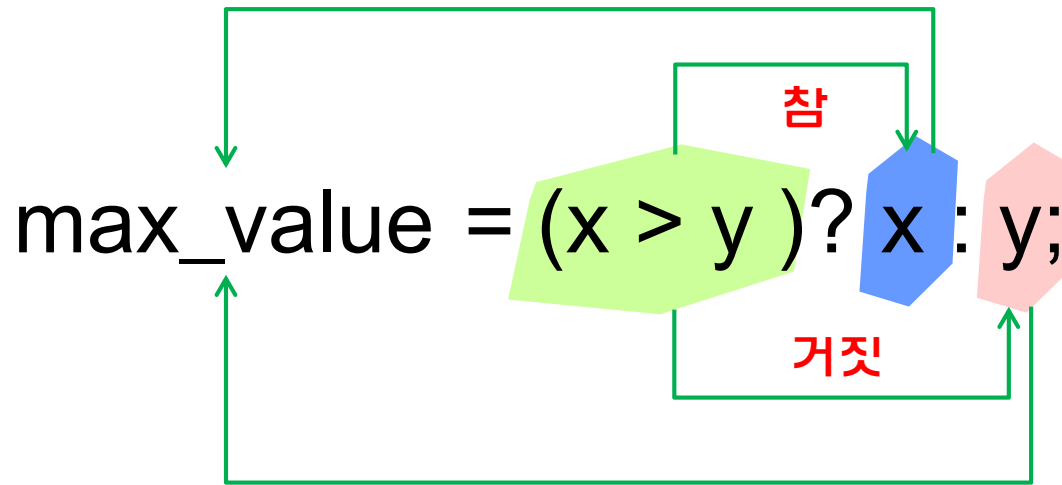
형변환

5.10

연산자의 우선 순위와 결합 규칙

조건 연산자

59



```
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");
```

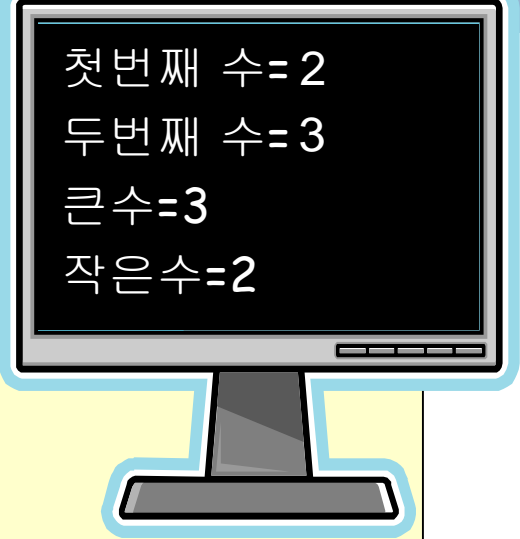
예제

60

```
#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

Contents

61

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

5.8

비트 연산자

5.9

형변환

5.10

연산자의 우선 순위와 결합 규칙

콤마 연산자

62

- 콤마로 연결된 수식은 순차적으로 계산된다.

먼저 계산된다.

나중에 계산된다.

$x++$, $y++$;

Contents

63

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

5.8

비트 연산자

5.9

형변환

5.10

연산자의 우선 순위와 결합 규칙

비트 연산자

64

연산자	연산자의 의미	설명
&	비트 AND	두개의 피연산자의 해당 비트가 모두 1이면 1, 아니면 0
	비트 OR	두개의 피연산자의 해당 비트중 하나만 1이면 1, 아니면 0
^	비트 XOR	두개의 피연산자의 해당 비트의 값이 같으면 0, 아니면 1
<<	왼쪽으로 이동	지정된 개수만큼 모든 비트를 왼쪽으로 이동한다.
>>	오른쪽으로 이동	지정된 개수만큼 모든 비트를 오른쪽으로 이동한다.
~	비트 NOT	0은 1로 만들고 1은 0로 만든다.

비트 AND 연산자(&)

65

□ 각 비트 단위로 AND 연산을 수행

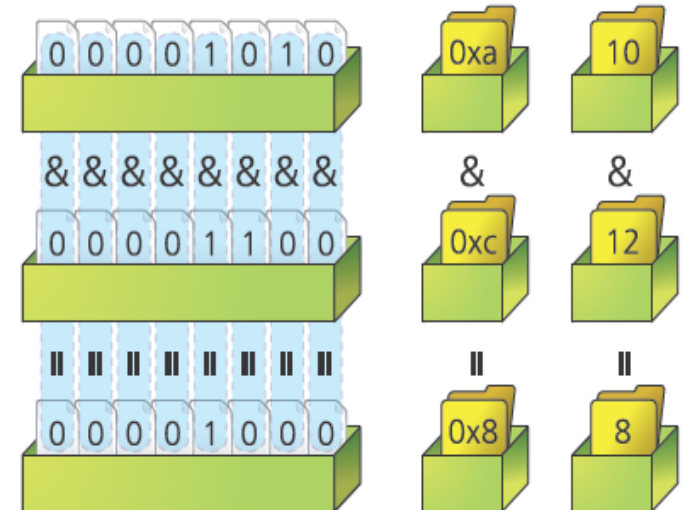
a의 비트	b의 비트	a의 비트 & b의 비트
0	0	0
0	1	0
1	0	0
1	1	1

```
int a = 10;  
int b = 12;  
int c = a & b;
```

2진수

16진수

10진수



비트 OR 연산자

66

- 피연산자의 같은 위치에 있는 비트에 대해 비트 OR 연산을 수행

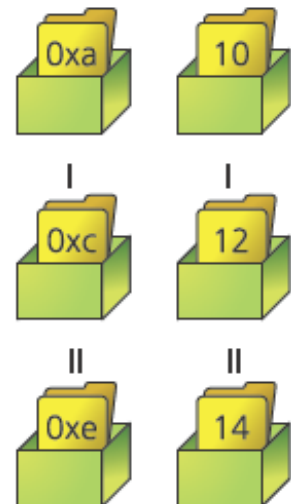
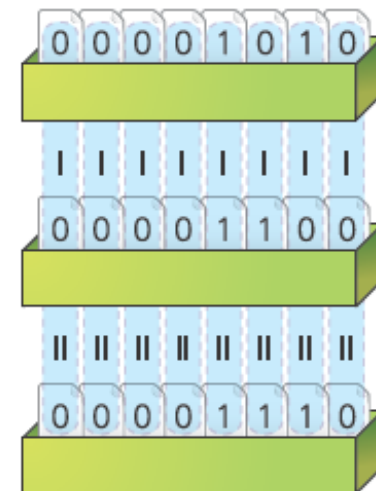
a의 비트	b의 비트	a의 비트 b의 비트
0	0	0
0	1	1
1	0	1
1	1	1

2진수

16진수

10진수

```
int a = 10;  
int b = 12;  
int c = a | b;
```



비트 XOR 연산자

67

- 피연산자의 같은 위치에 있는 비트에 대해 비트 XOR 연산을 수행

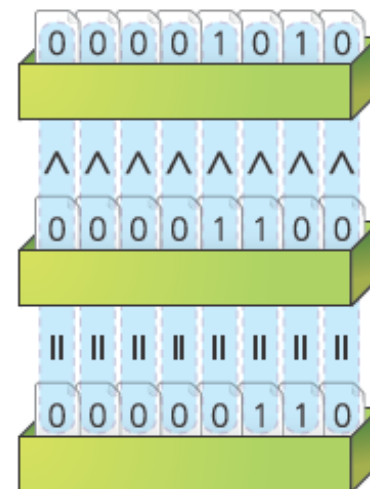
a의 비트	b의 비트	a의 비트 ^ b의 비트
0	0	0
0	1	1
1	0	1
1	1	0

```
int a = 10;  
int b = 12;  
int c = a ^ b;
```

2진수

16진수

10진수



비트 NOT 연산자

68

- 피연산자의 각 비트를 반전시킨다. 즉 0은 1로, 1은 0으로 만든다

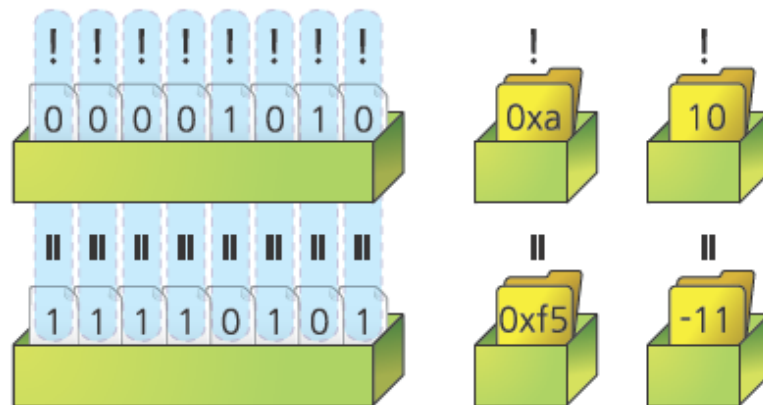
a의 비트	$\sim a$ 의 비트
0	1
1	0

```
int a = 10;  
int c =  $\sim a$ ;
```

2진수

16진수

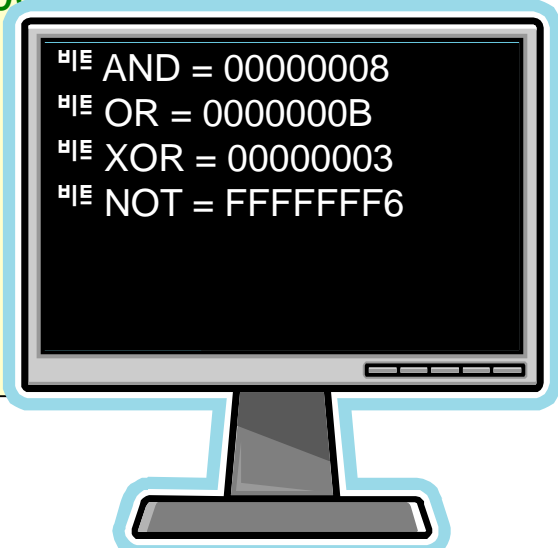
10진수



예제: 비트 연산자

69

```
#include <stdio.h>
int main(void)
{
    int x = 9;           // 1001
    int y = 10;          // 1010
    printf("비트 AND = %08X", x & y); // 00001000
    printf("비트 OR = %08X", x | y);  // 00001011
    printf("비트 XOR = %08X", x ^ y); // 00000011
    printf("비트 NOT = %08X", ~x );
    return 0;
}
```



비트 AND = 00000008
비트 OR = 0000000B
비트 XOR = 00000003
비트 NOT = FFFFFFFF6

비트 이동 연산자

70

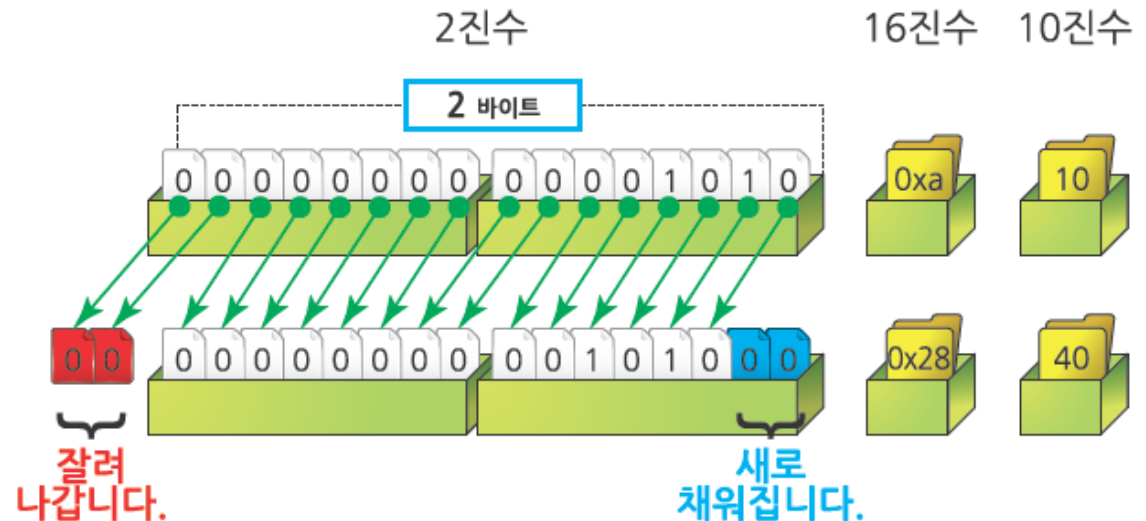
연산자	기호	설명
왼쪽 비트 이동	\ll	$x \ll y$ x 의 비트들을 y 칸만큼 왼쪽으로 이동
오른쪽 비트 이동	\gg	$x \gg y$ x 의 비트들을 y 칸만큼 오른쪽으로 이동

<< 연산자

71

- 비트를 왼쪽으로 이동
- 왼쪽으로 밀려난 비트가 사라져버리고, 오른쪽 빈자리에는 0이 채워진다.
- 비트 왼쪽 이동은 2^n 을 곱하는 것이다.

$10 \ll 2$

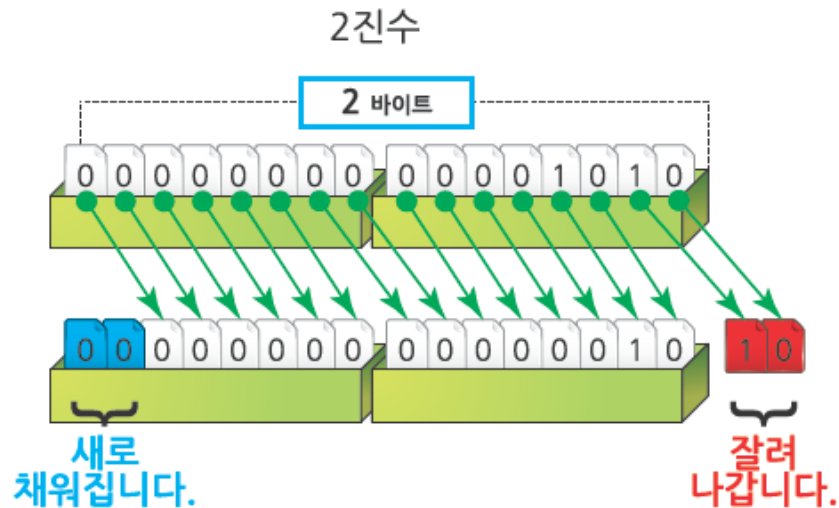


>> 연산자

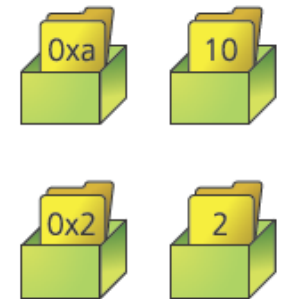
72

- 비트를 오른쪽으로 이동
- 오른쪽으로 밀려난 비트가 사라져버리고, 왼쪽 빈자리에는 부호 비트가 채워진다.
- 비트 오른쪽 이동은 2^n 을 나누는 것이다.

10 >> 2



16진수 10진수



예제: 비트 이동 연산자

73

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

```
int main(void)
```

```
{
```

```
    int x = 4;
```

```
// 0100
```


```
    printf("비트 << = %#08x", x << 1);
```

```
// 1000
```

```
    printf("비트 >> = %#08x", x >> 1);
```

```
    return 0;
```

```
}
```

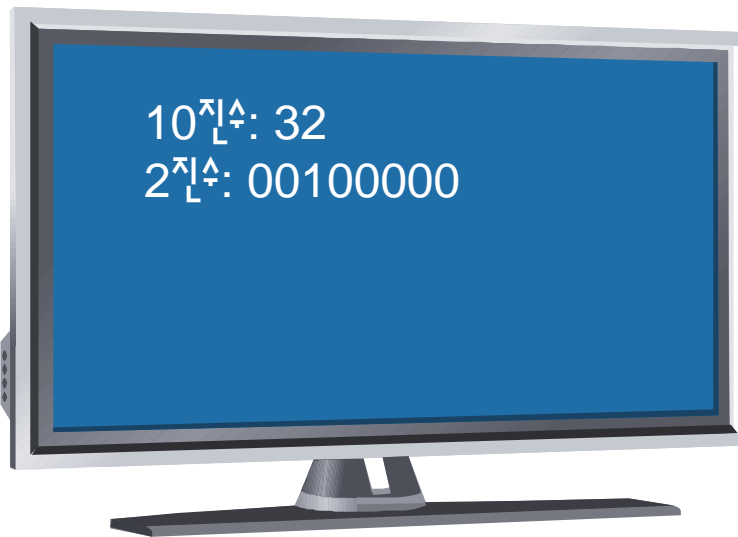


```
비트 << = 0x000008  
비트 >> = 0x000002
```

Lab: 10진수를 2진수로 출력하기

74

- 비트 연산자를 이용하여 128보다 작은 10진수를 2진수 형식으로 화면에 출력해보자.



Lab: 10진수를 2진수로 출력하기

75

```
#include<stdio.h>
int main(void)
{
    unsigned int num;
    printf("십진수: ");
    scanf("%u", &num);

    unsigned int mask = 1 << 7; // mask = 10000000
    printf("이진수: ");

    ((num & mask) == 0) ? printf("0") : printf("1");
    mask = mask >> 1;           // 오른쪽으로 1비트 이동한다.
    ((num & mask) == 0) ? printf("0") : printf("1");
    mask = mask >> 1;           // 오른쪽으로 1비트 이동한다.
    ((num & mask) == 0) ? printf("0") : printf("1");
    mask = mask >> 1;           // 오른쪽으로 1비트 이동한다.
```

Lab: 10진수를 2진수로 출력하기

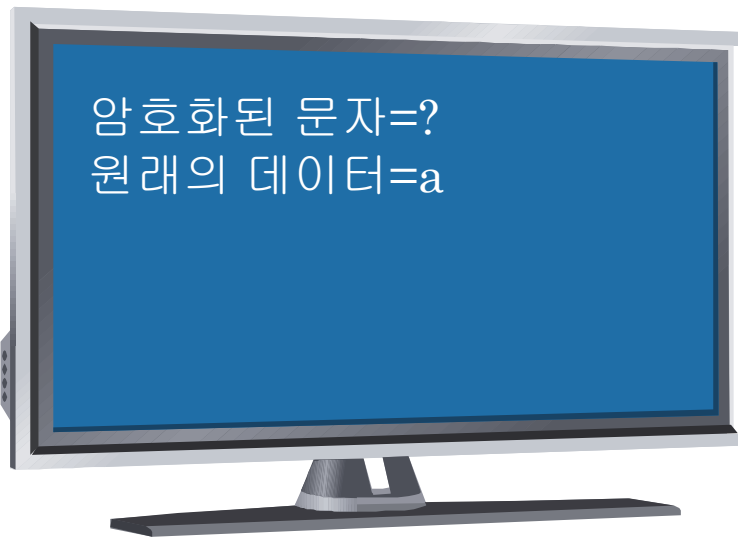
76

```
((num & mask) == 0) ? printf("0") : printf("1");  
mask = mask >> 1;  
((num & mask) == 0) ? printf("0") : printf("1");  
mask = mask >> 1;  
((num & mask) == 0) ? printf("0") : printf("1");  
mask = mask >> 1;  
((num & mask) == 0) ? printf("0") : printf("1");  
mask = mask >> 1;  
((num & mask) == 0) ? printf("0") : printf("1");  
printf("\n");  
  
return 0;  
  
}
```

Lab: XOR를 이용한 암호화

77

- 하나의 문자를 암호화하기 위해서는 $x = x \oplus \text{key}$; 하면 된다.
복호화도 $x = x \oplus \text{key}$; 하면 된다.



Lab: XOR를 이용한 암호화

78

```
#include <stdio.h>
int main(void)
{
    char data = 'a';
    char key = 0xff;

    char encrpted_data;
    encrpted_data = data ^ key;

    printf("암호화된 문자=%c \n", encrpted_data);

    char orig_data;
    orig_data = encrpted_data ^ key;
    printf("원래의 데이터=%c\n", orig_data);

    return 0;
}
```

Contents

79

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

5.8

비트 연산자

5.9

형변환

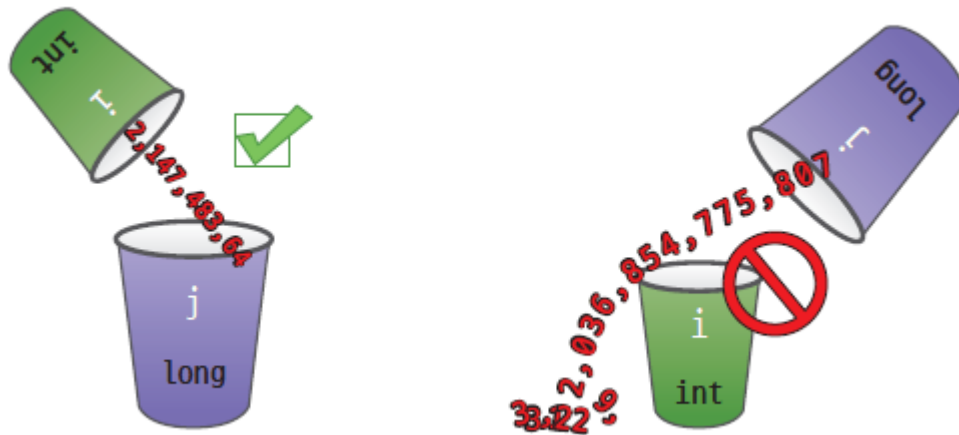
5.10

연산자의 우선 순위와 결합 규칙

형변환

80

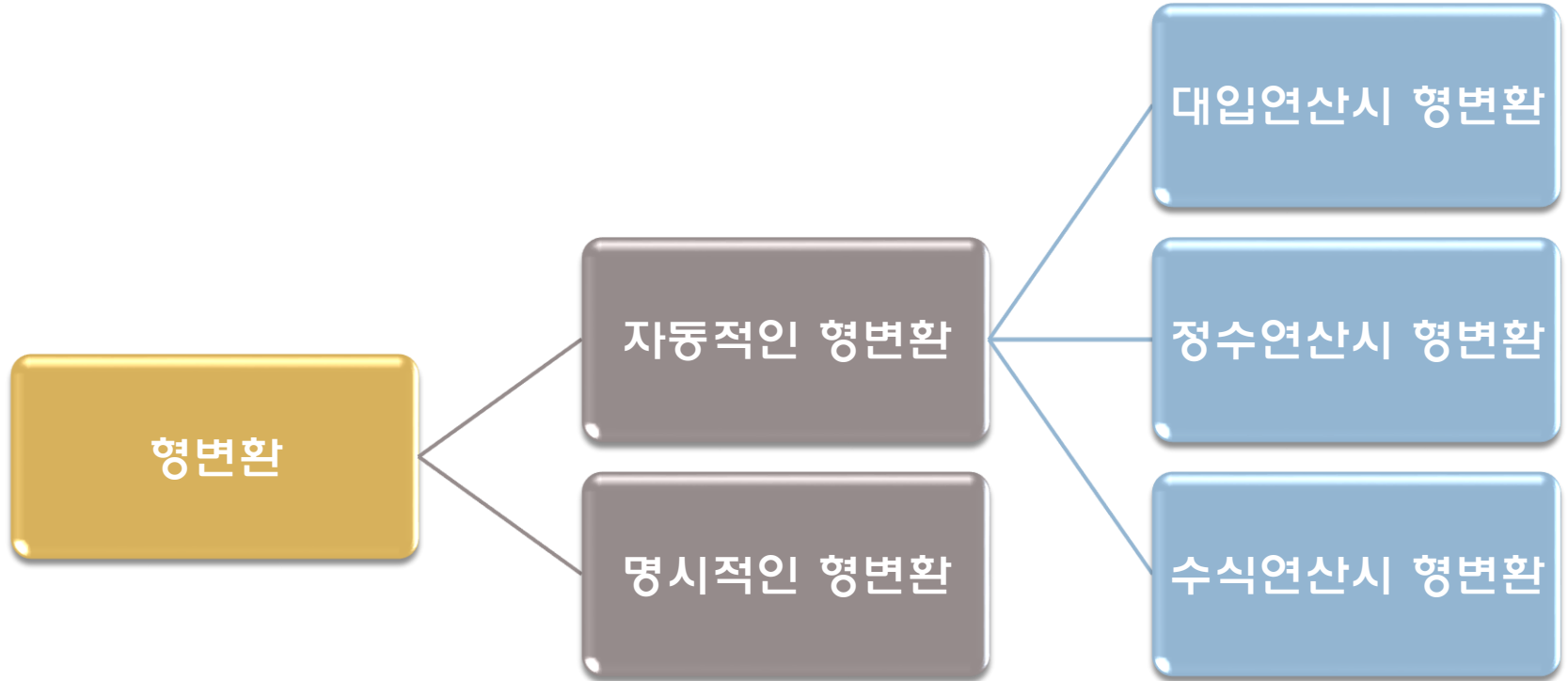
- 형변환(type conversion)이란 실행 중에 데이터의 타입을 변경하는 것이다



형변환

81

□ 연산시에 데이터의 유형이 변환되는 것



대입 연산시의 자동적인 형변환

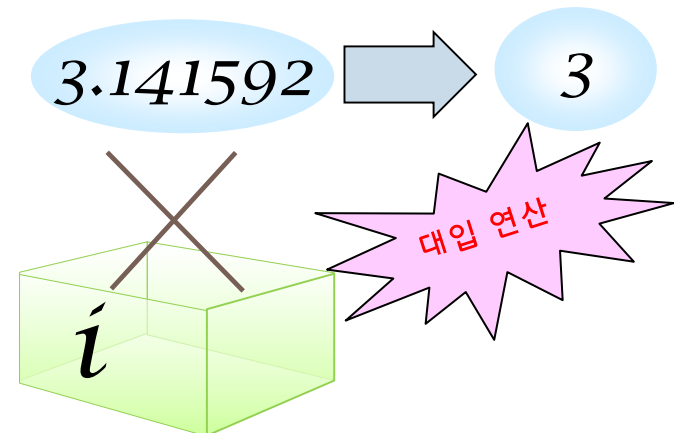
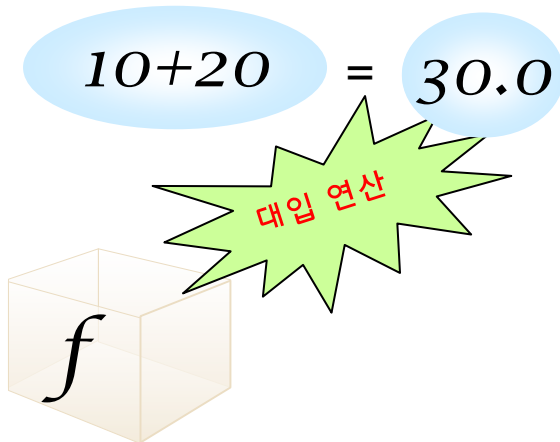
82

□ 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

83

16

11

11

30.000000

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

```
int main(void)
```

```
{
```

```
    char c;
```

```
    int i;
```

```
    float f;
```

```
    c = 10000;
```

```
    // 내림 변환
```

```
    i = 1.23456 + 10;
```

```
    // 내림 변환
```

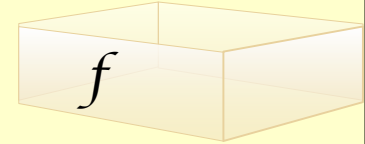
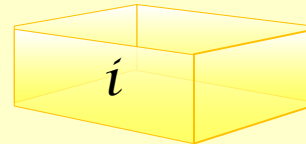
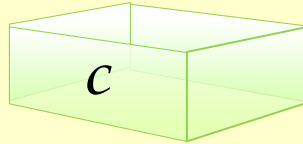
```
    f = 10 + 20;
```

```
    // 올림 변환
```

```
    printf("c = %d, i = %d, f = %f □n", c, i, f);
```

```
    return 0;
```

```
}
```



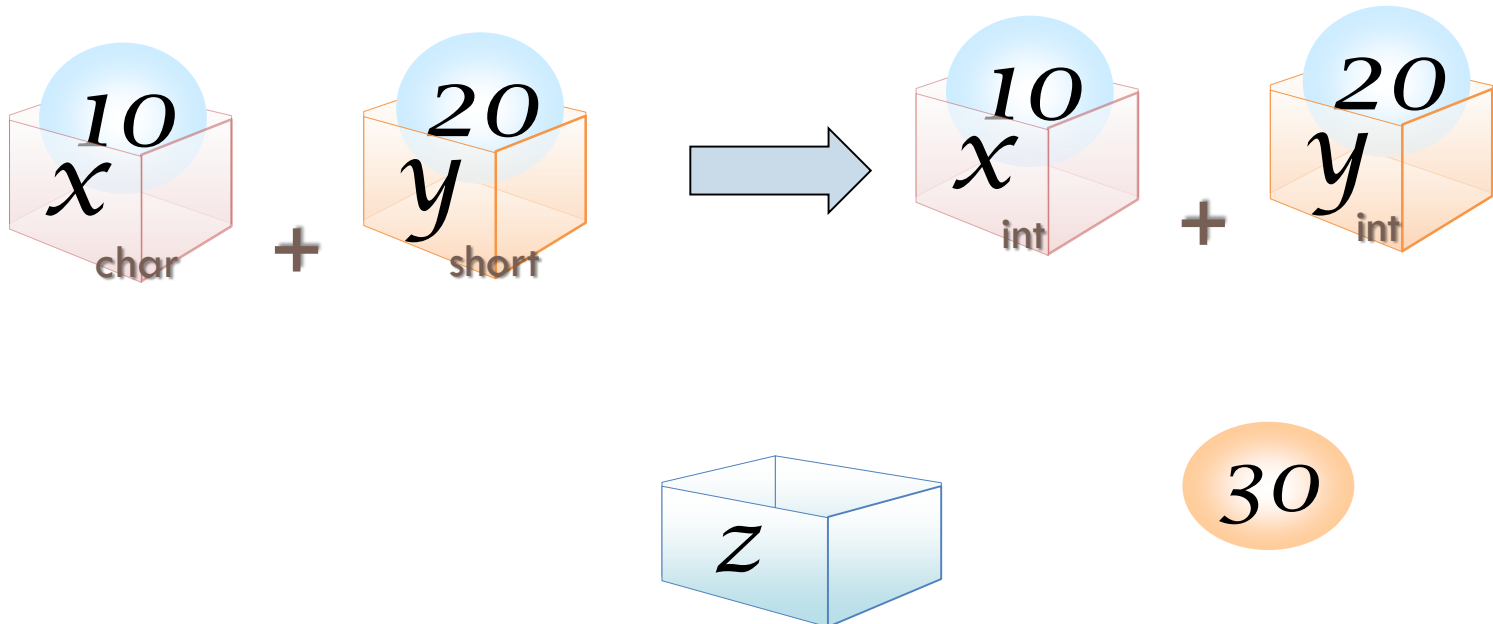
```
c:\...\convert1.c(10) : warning C4305: '=' :  
'int'에서 'char'으로 잘립니다.  
c:\...\convert1.c(11) : warning C4244: '=' :  
'double'에서 'int'로 변환하면서 데이터가 손실될 수 있습니다.  
c=16, i=11,  
f=30.000000
```

정수 연산시의 자동적인 형변환

84

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

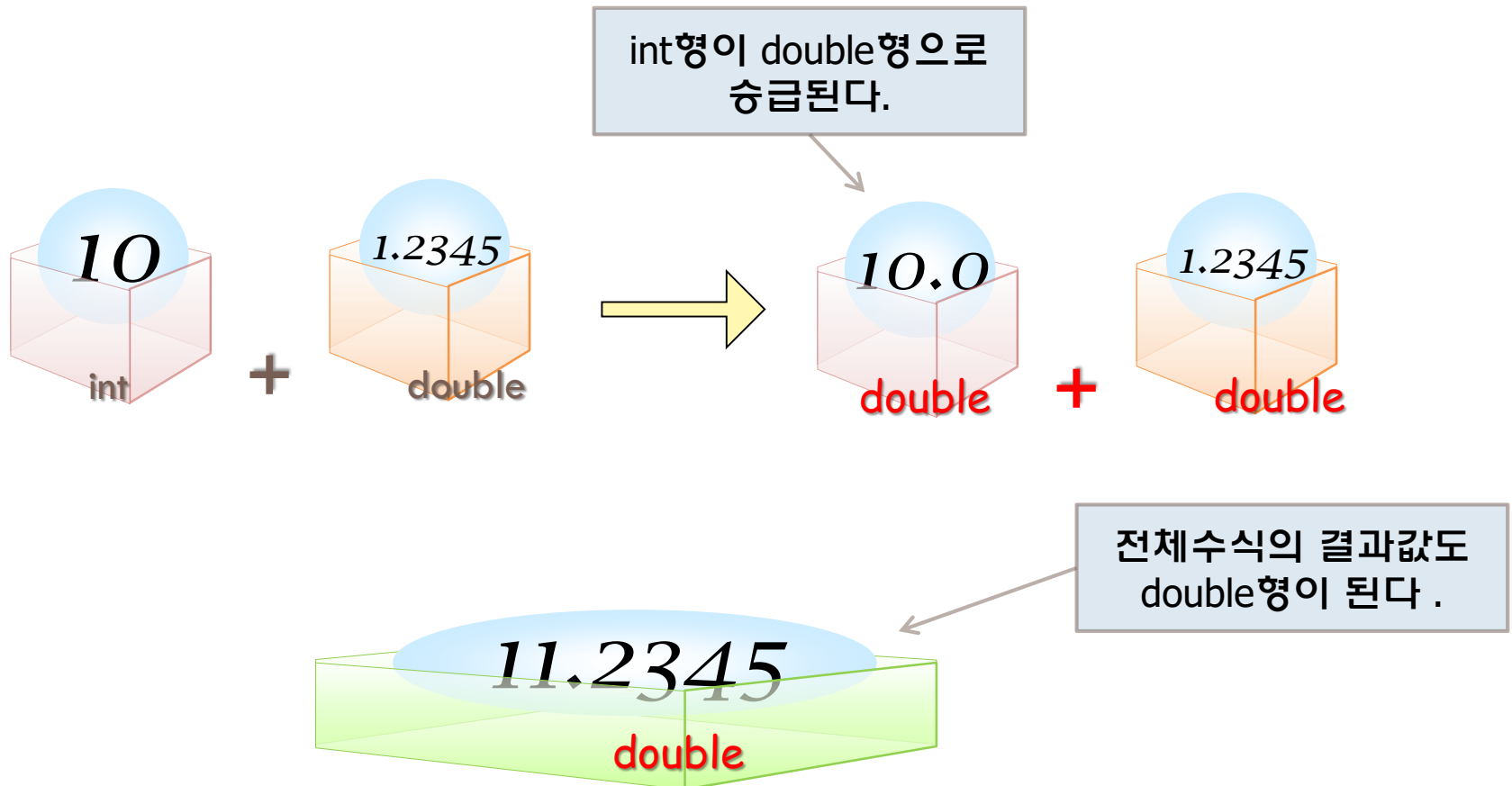
```
char x = 10;  
short y = 20;  
z = x + y;
```



수식에서의 자동적인 형변환

85

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



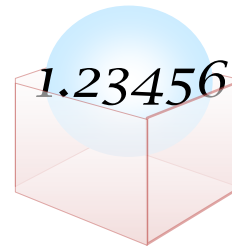
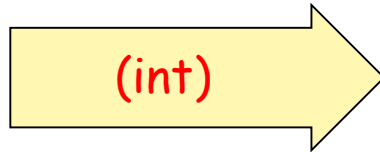
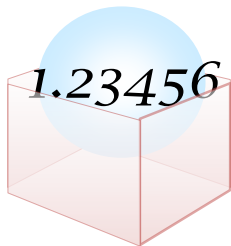
명시적인 형변환

86

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

(자료형) 상수 또는 변수

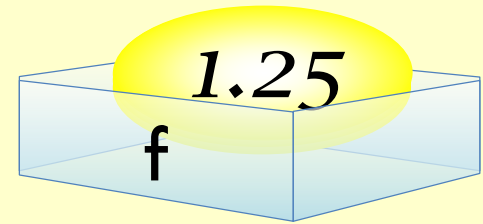
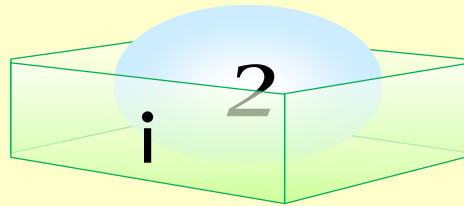
- (int) 1.23456
- (double) x // double형으로 변환
- (long) (x+y) // long형으로 변환



예제

87

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 정수지만 f에 저장되면서 1.0으로 변환된다.
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

Contents

88

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

5.8

비트 연산자

5.9

형변환

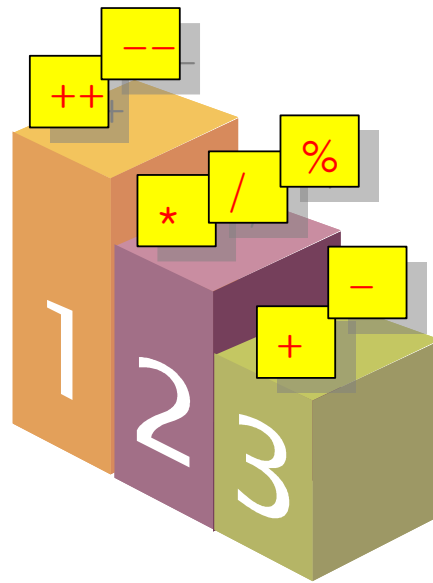
5.10

연산자의 우선 순위와 결합 규칙

우선 순위

89

- 어떤 연산자를 먼저 계산할 것인지에 대한 규칙



$$x + y * z$$

Diagram illustrating operator precedence for the expression $x + y * z$. A bracket labeled ① groups $y * z$, and a larger bracket labeled ② groups the entire expression $x + y * z$.

$$(x + y) * z$$

Diagram illustrating operator precedence for the expression $(x + y) * z$. A bracket labeled ① groups $x + y$, and a larger bracket labeled ② groups the entire expression $(x + y) * z$.

우선 순위

90

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

91

- 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
 - $(x \leq 10) \ \&\& \ (y \geq 20)$
- 관계 연산자나 논리 연산자는 산술 연산자보다 우선순위가 낮다.
 - $x + 2 == y + 3$

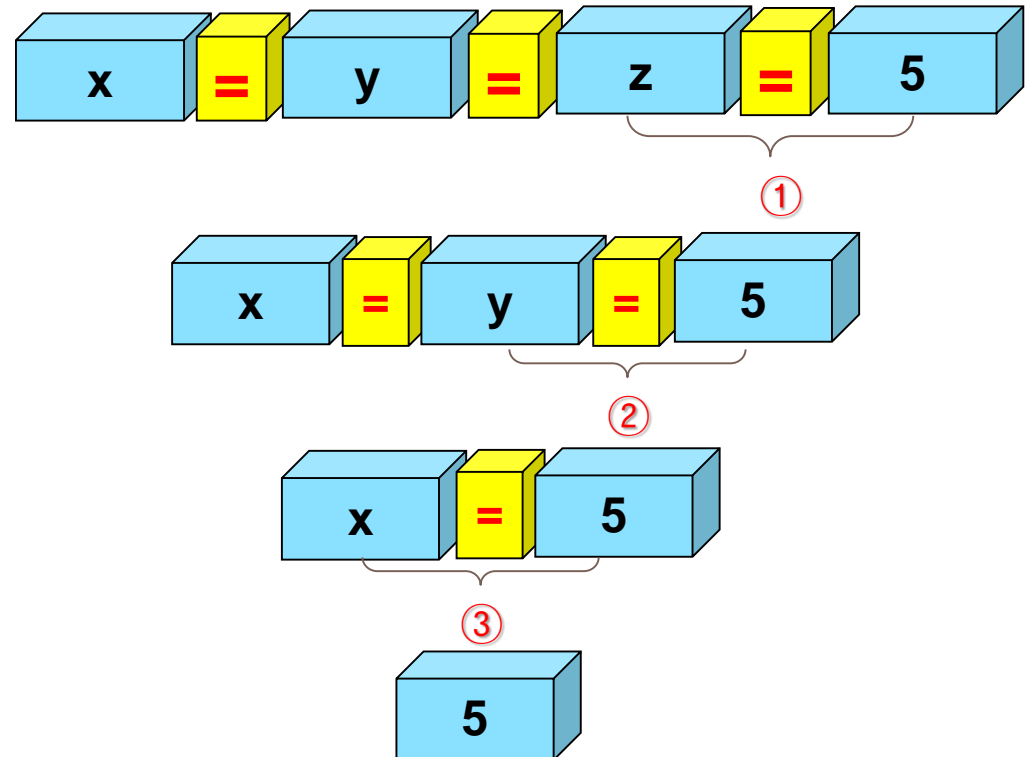
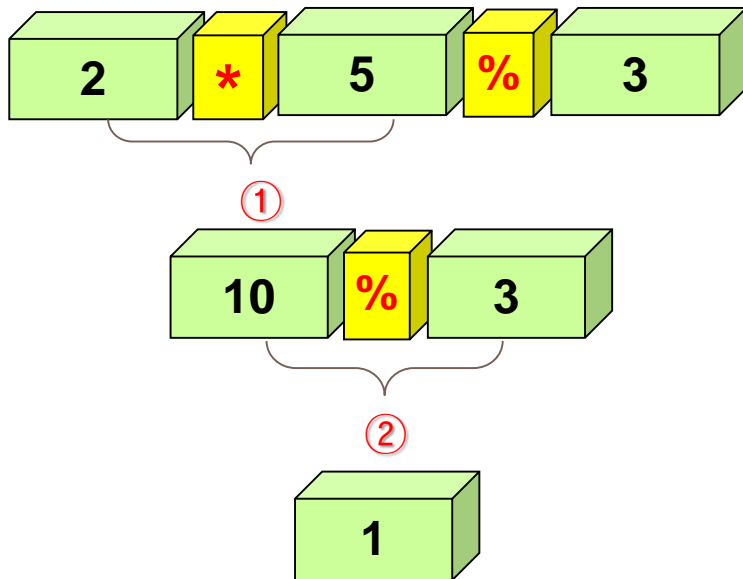
결합 규칙

92

- 만약 같은 우선순위를 가지는 연산자들이 여러 개가 있으면 어떤 것을 먼저 수행하여

*와 %의 우선순위가 같으므로 왼쪽에서 오른쪽으로 연산을 수행한다.

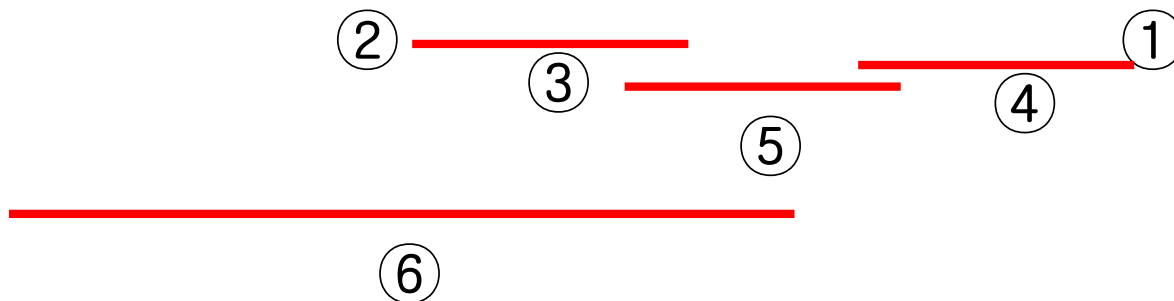
= 연산자는 오른쪽 우선 결합이므로 오른쪽부터 계산된다.



결합규칙의 예

93

$$y = \underline{a \% b} / c + d * \underline{(e - f)};$$



예제

94

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

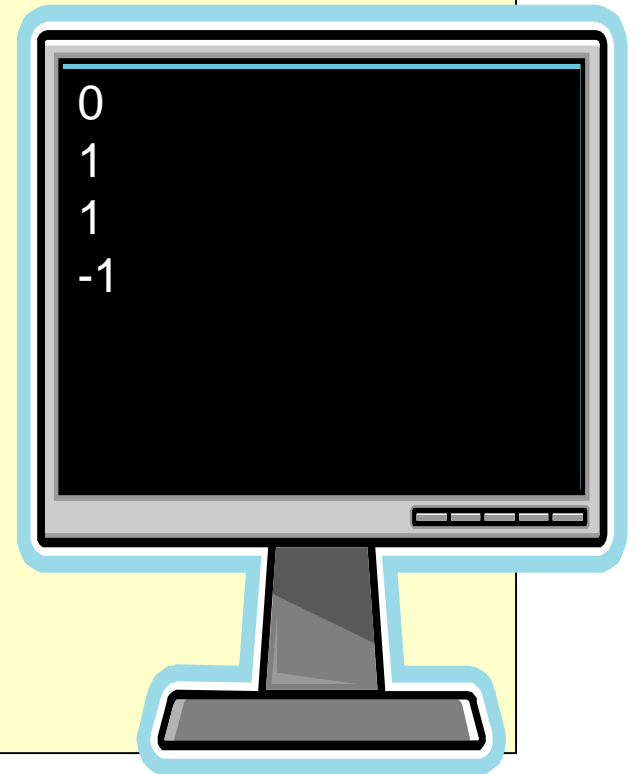
    result = 2 > 3 || 6 > 7;
    printf("%d", result);

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

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

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

    return 0;
}
```

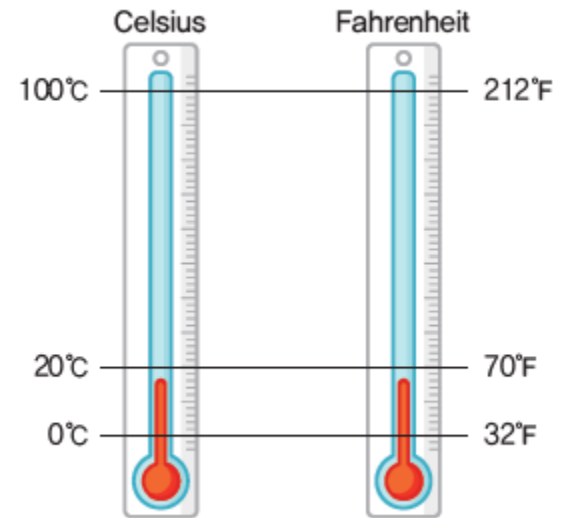


mini project: 화씨 온도를 섭씨로 바꾸기

95

- 화씨 온도를 섭씨 온도로 바꾸는 프로그램을 작성하여 보자.

$$\text{섭씨온도} = \frac{5}{9}(\text{화씨온도} - 32)$$



잘못된 부분은 어디에?

96

```
#include <stdio.h>
int main(void)
{
    double f_temp;
    double c_temp;

    printf("화씨온도를 입력하시오");
    scanf("%lf", &f_temp);
    c_temp = 5 / 9 * (f_temp - 32);
    printf("섭씨온도는 %f입니다, c_temp);

    return 0;
}
```

$c_temp = 5.0 / 9.0 * (f_temp - 32);$

