



학습자가 강의 저작물을 다운로드·캡처 받아 **외부로 유출하는 행위**는 저작권자의
이용허락 없이 저작물을 복제·공중송신 또는 배포 하는 것으로 **저작권 침해 행위**에 해당함.

C 프로그래밍

(001/002)

제 3 강

신 한 대 학 교
소프트웨어융합학과
교수 송 진 희

C 프로그래밍

제 3 강

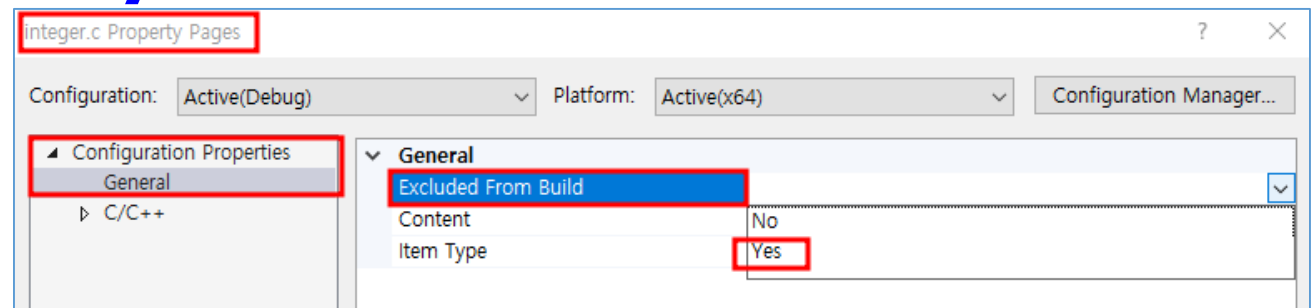
- 연산자와 수식 표현
- 연산 처리
- 자료 형 (Type) 변환

학습 목표

- 연산자의 종류를 분류할 수 있다.
- 산술 연산자의 종류와 우선순위를 이용한 수식 표현을 할 수 있다.
- 관계 연산자의 종류와 우선순위를 이용한 수식 표현을 할 수 있다.
- 논리 연산자의 종류와 우선순위를 이용한 수식 표현을 할 수 있다.
- 연산자 우선순위와 결합 법칙을 이용한 수식 표현을 할 수 있다.

2강 - 정리 요약

- 기존 프로젝트의 열기 / 수정
- 한 개 프로젝트에 여러 개의 소스 파일(.c) 추가하기
- 소스 파일의 빌드 제외 / 빌드 포함



- 자료형
 - 문자(char)
 - 숫자 : 정수(short, int) / 실수(float, double)
- 변수 / 변수의 초기화
- 배열 연산 표현 ('=')

2강 - 정리 요약

○ 산술 연산자

연산	연산자	C 수식
덧셈	+	$x + y$
뺄셈	-	$x - y$
곱셈	*	$x * y$
나눗셈	/	x / y
나머지	%	$x \% y$

○ printf() 함수의 출력 형식 지정

- %d %10d %10.3d %.2d
- %c %s

○ scanf() 함수

- scanf("%d", &sum);
- scanf ("%d %f", &sum, &div);

2강 - 정리 요약

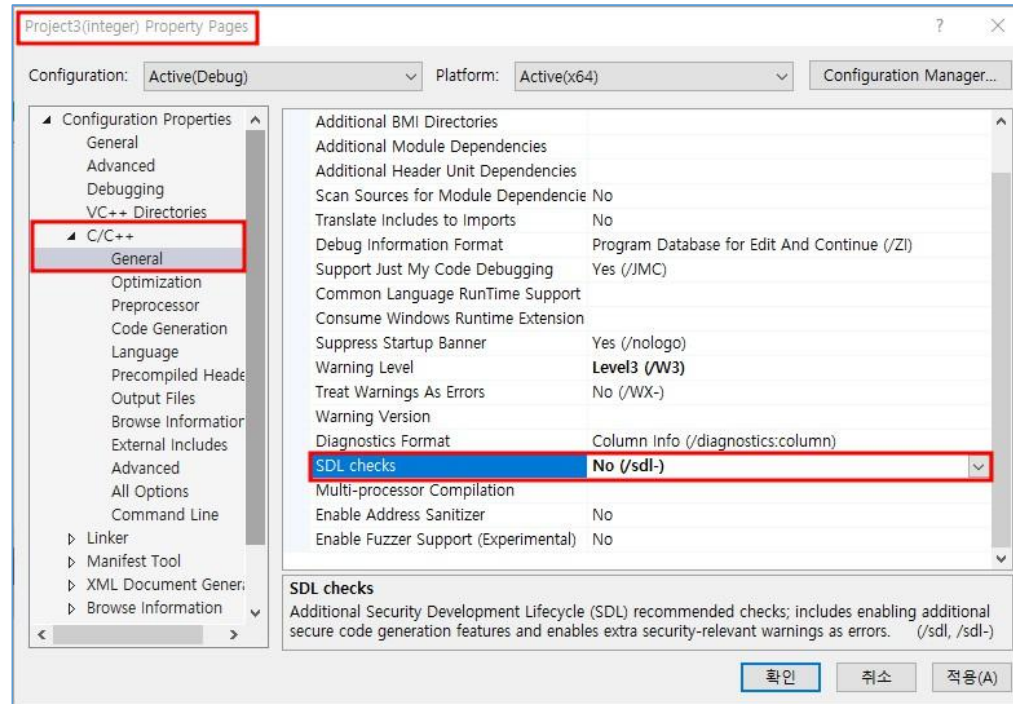
○ 비주얼스튜디오에서 scanf()함수 사용 오류 예방 방법

1) 매크로 선언

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

int main(void)
{
```

2) 프로젝트 설정 변경



2강 실습

• scanf() 함수 사용법

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

```
void main()
{
    int i;
    float f;
    double d;
    char ch;
    char s[10];
```

```
printf("정수 한 개 입력하세요 : \n");
scanf("%d", &i); // 주소 연산자 사용에 주의할 것!
```

```
printf("실수 한 개 입력하세요 : \n");
scanf("%f", &f);
```

```
printf("Double형 실수 한 개 입력하세요 : \n");
scanf("%lf", &d);
```

```
printf("알파벳 한 글자를 입력하세요 : \n");
scanf(" %c", &ch); // %c 앞에 반드시 공백 한 개 부여해서
// printf()의 줄바꿈 문자를 skip 시켜야 함!!!
```

```
//printf("알파벳 한 글자를 입력하세요 : "); //줄바꿈 문자 없음
//scanf("%c", &ch); //공백 불필요
```

```
printf("알파벳 한 글자를 입력하세요 : \n");
scanf(" %s", &s); // 9개 문자만 입력해야 함!!!
}
```


실습

- 사각형의 둘레와 면적을 구하는 프로그램을 작성하기

- 필요한 변수 선언 : w, h, area, perimeter(가로, 세로, 면적, 둘레)
- 변수들의 자료형 : 실수



[실행 화면]

2강 실습

```
#define _CRT_SECURE_NO_WARNINGS
```

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

```
int main(void)
```

```
{
```

```
    double w; double h; double area;  
    double perimeter;
```

```
    w = 10.0;
```

```
    h = 5.0;
```

```
    area = w * h;
```

```
    perimeter = 2 * (w + h);
```

```
    printf("사각형의 넓이: %lf \n", area);
```

```
    printf("사각형의 둘레: %lf \n", perimeter);
```

```
    return 0;
```

```
}
```

사각형의 넓이: 50.000000
사각형의 둘레: 30.000000

사각형의 넓이
사각형의 둘레

(double형)

- 면적 계산 수식 : $area = w * h;$
- 둘레 계산 수식 : $perimeter = 2 * (w + h);$

5장. 수식과 연산자

C언어의 연산자

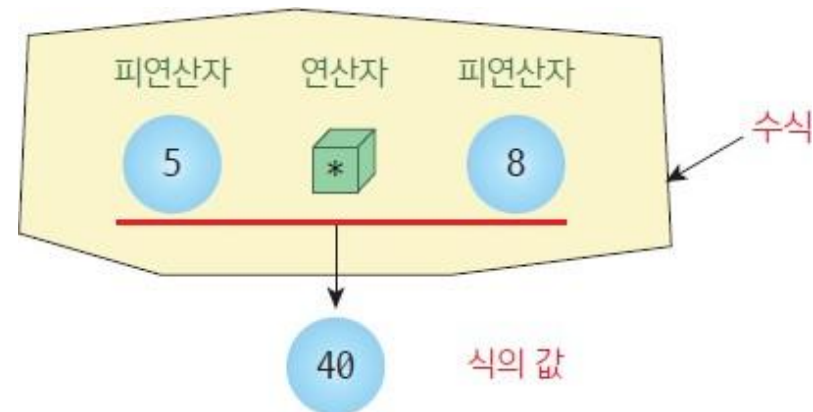
- 자바, C++, 파이썬, 자바 스크립트 등의 최신 언어들이 C언어의 연산자를 거의 그대로 사용한다.



수식

○ 수식(Expression)

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



```
int x, y;  
  
x = 3;  
y = x*x - 5*x + 6;  
printf("%d\n", y);
```

[수식 표현의 예]

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

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

○ a^2 : (참고) 거듭 제곱 연산자? C에는 거듭제곱을 표현하는 연산자가 없다.

→ $a * a$ 와 같이 단순히 변수를 두 번 곱한다.

산술 연산자의 예

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

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

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

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
```

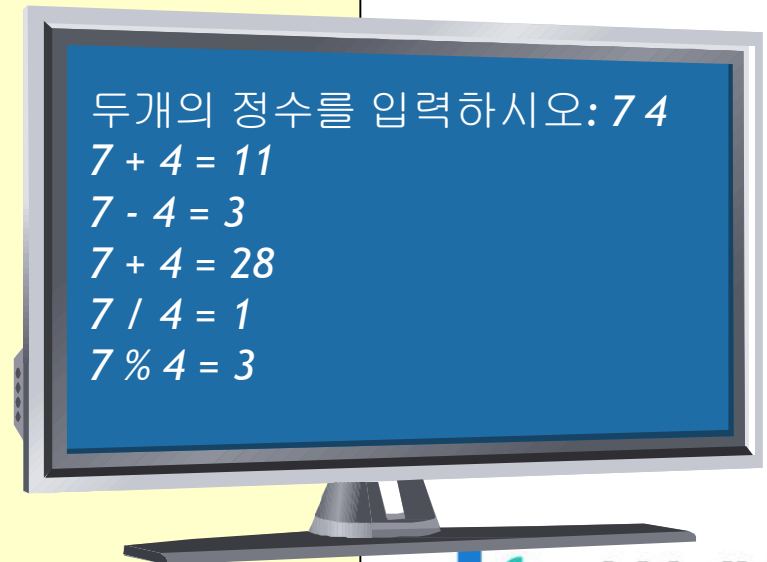
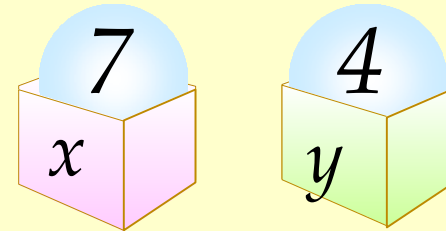
```
    printf("%d / %d = %d", x, y, result);
```

```
    result = x % y; // 나머지
```

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

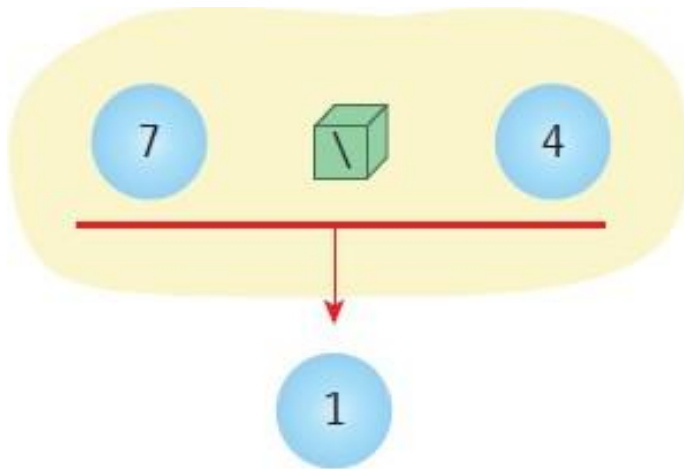
```
    return 0;
```

```
}
```

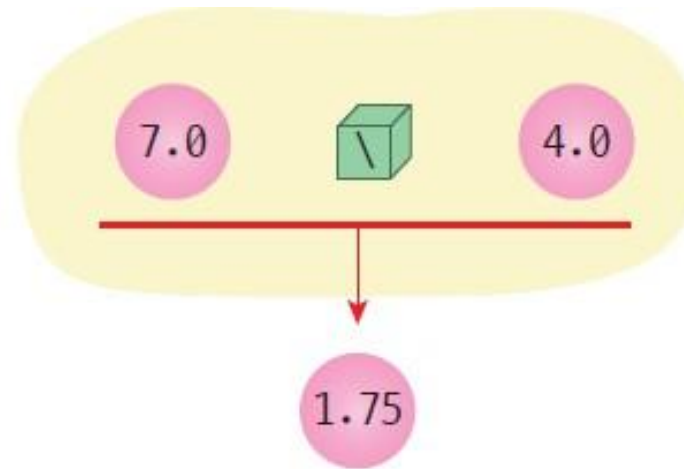


나눗셈 연산자

- 정수와 정수의 나눗셈 결과 : 정수형
- 실수와 실수의 나눗셈 결과 : 실수
- 정수와 정수의 나눗셈에서 소수점 이하는 버려진다.



정수와 정수끼리의 나눗셈



실수와 실수끼리의 나눗셈

실수 사칙 연산

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

```
int main()
```

```
{
```

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

```
    printf("두 개의 실수를 입력하시오: ");
```

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

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

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

```
    :
```

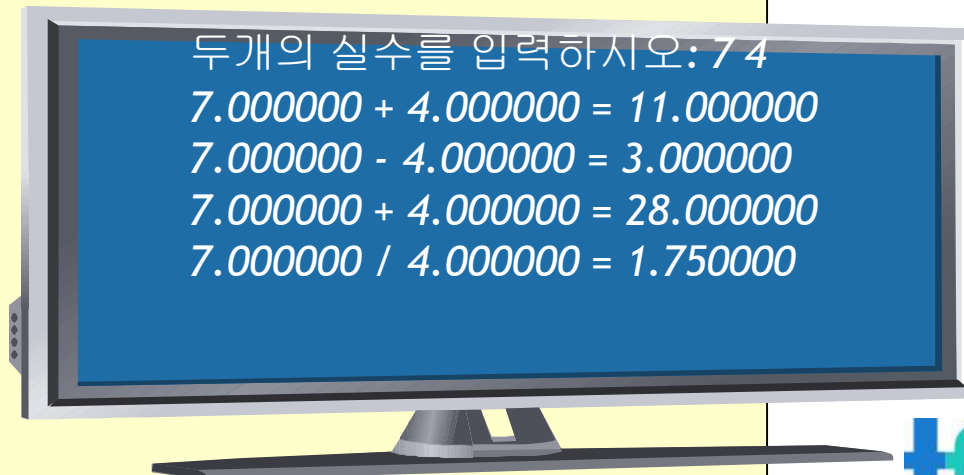
```
    :
```

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

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

```
}
```

```
    return 0;
```



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, 1이면 홀수

○ (예) 나머지 연산자를 이용한 3의 배수 판단

- $x \% 3$ 이 0이면 3의 배수

아주 유용한
연산자입니다



나머지 연산자

// 나머지 연산자 프로그램

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

```
#define SEC_PER_MINUTE 60
```

// 1분은 60초

```
int main(void)
```

```
{
```

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

```
    printf( " 초를 입력하시오: ");
```

```
    scanf("%d", &input);
```

// 초단위의 시간을 읽는다.

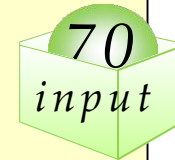
```
    minute = input / SEC_PER_MINUTE; // 몇 분
```

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

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

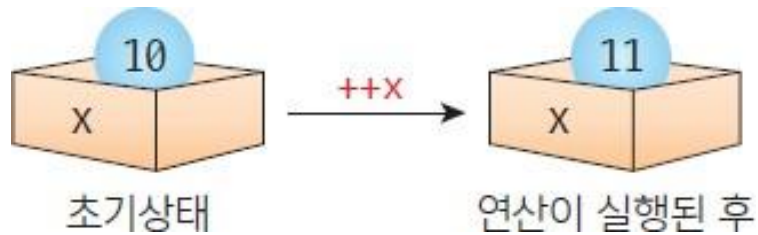
```
    return 0;
```

```
}
```

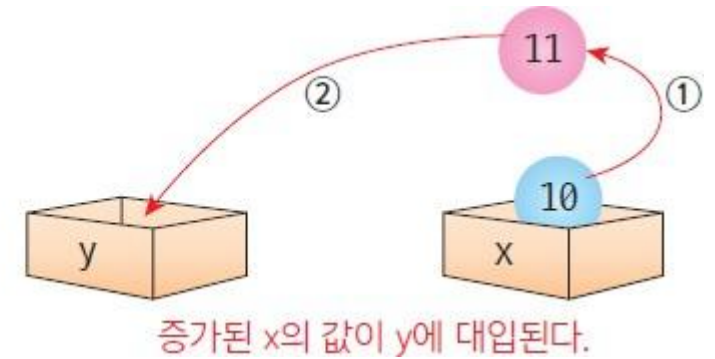


증감 연산자

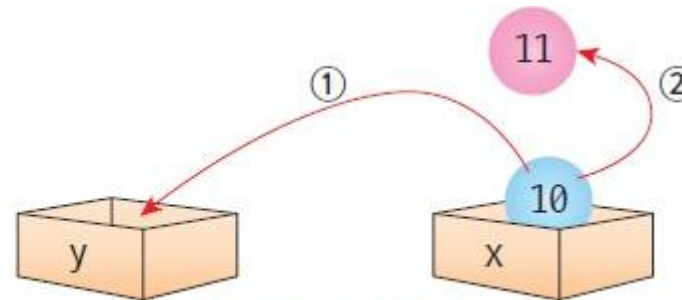
- 증감 연산자: $++$, $--$
- 변수의 값을 하나 증가시키거나 감소시키는 연산자
- (예) $++x$, $--x$;



$y=++x$;



$y=x++$;



★ 먼저 대입하고 나중에 증가한다.

증감 연산자 정리

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

`y = (1 + x++) + 10;` // 괄호가 있어도 x값의 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++);
```

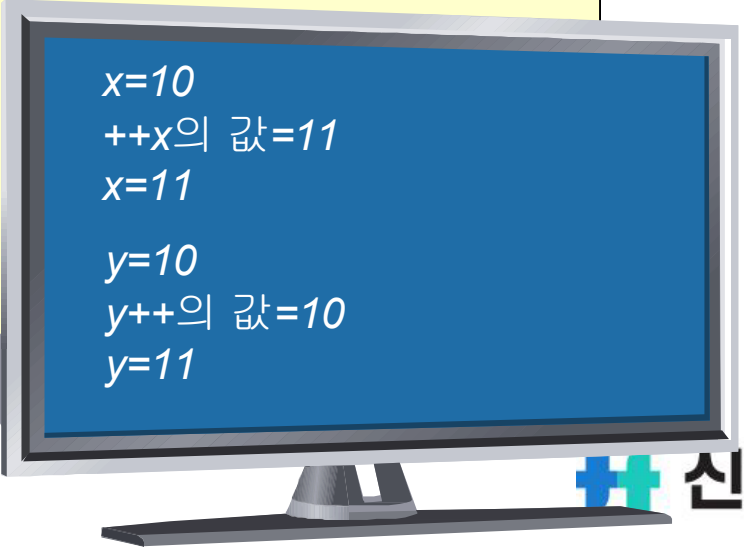
```
    printf("y=%d \n", y);
```

```
    return 0;
```

```
}
```

먼저 증가하고, 증가된 값이
수식 에 사용된다.

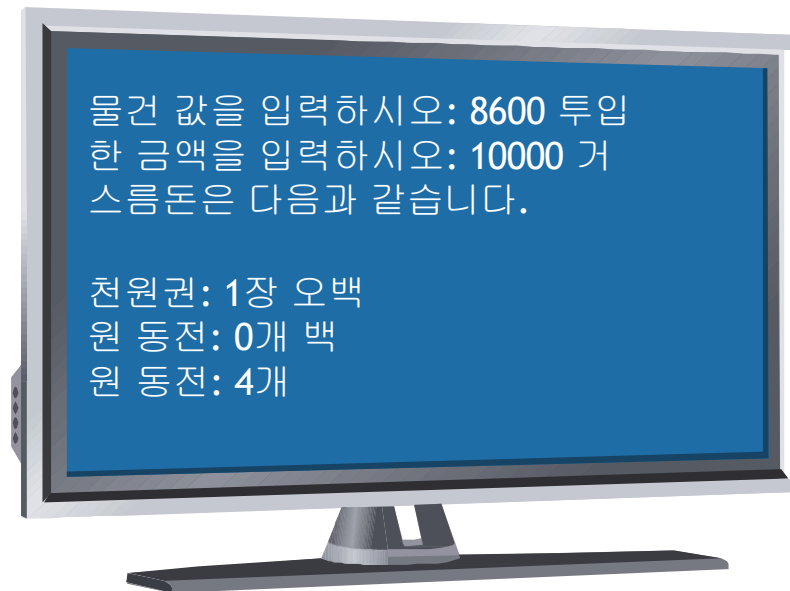
현재 값을 먼저 수식에 사용
하고 나중에 증가된다.



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


실습 : 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, c1000, c500, c100;
```

```
    printf("물건 값을 입력하시오: ");
```

```
    scanf("%d", &price);
```

// 물건 값을 입력 받는다.

```
    printf("투입한 금액을 입력하시오: ");
```

```
    scanf("%d", &money);
```

// 투입 금액을 입력 받는다.

```
    change = money - price;
```

// 거스름돈을 change에 저장

```
    c1000 = change / 1000;
```

```
    change = change % 1000;
```

// 거스름돈에서 1000원권의 개수를 계산

// 나머지 연산자를 사용하여 남은 잔돈을 계산

```
    c500 = change / 500;
```

```
    change = change % 500;
```

// 남은 잔돈에서 500원 동전의 개수를 계산

// 나머지 연산자를 사용하여 남은 잔돈을 계산

```
    c100 = change / 100;
```

```
    change = change % 100;
```

// 남은 잔돈에서 100원 동전의 개수를 계산

// 나머지 연산자를 사용하여 남은 잔돈을 계산

```
    printf("₩n천원권: %d장 ₩n", c1000);
```

```
    printf("오백원 동전: %d개 ₩n", c500);
```

```
    printf("백원 동전: %d개 ₩n", c100);
```

```
    return 0;
```

```
}
```

복합 대입 연산자

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

$y = x = 3;$

먼저 $x = 3$ 이 수행되고, 그 결과값인 3이 다시 y 에 대입된다

$x += y$

$x = x + y$ 와 의미가 같음!

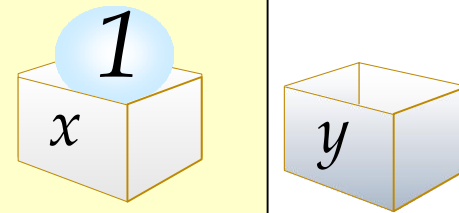
예제

```
/* 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 연산자 프로그램

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

```
int main(void)
```

```
{
```

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

```
    x += 1;
```

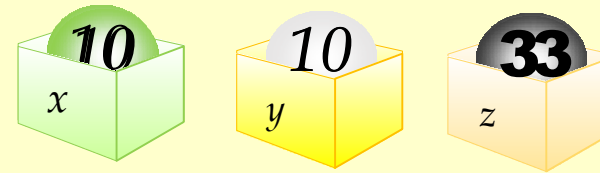
```
    y *= 2;
```

```
    z %= 10 + 20;    // z = z % (10 + 20)
```

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

```
    return 0;
```

```
}
```



x = 11 y = 20 z = 3

관계 연산자

- 두개의 피연산자를 비교하는 연산자
- 참(1) 아니면 거짓(0)

$x == y$

x 와 y의 값이 같은지 **비교**한다.

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?

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

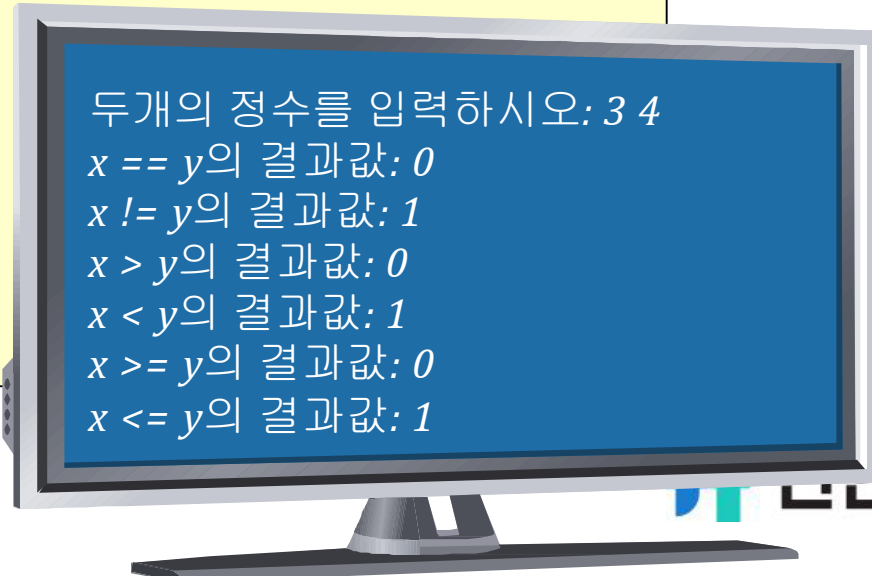
예제

```
#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

관계 연산자 표현

- 수식의 표현 : $2 < x < 5$
- 올바른 표현 방법: $(2 < x) \&\& (x < 5)$

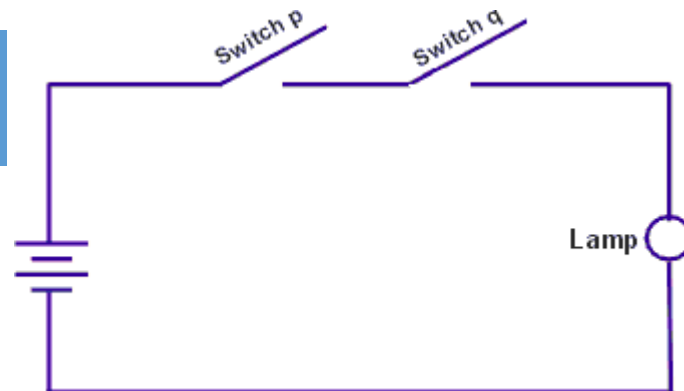
논리 연산자

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

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

$x \ \&\& \ y$

x와 y가 모두 참인 경우에만 참이 된다.



AND / OR 연산자

○ 어떤 회사에서 신입 사원을 채용하는데 나이가 30살 이하이고 (AND) 토익 성적이 700점 이상이라는 조건을 지정

$$\underbrace{(\overset{27}{\text{age}} \leq 30)}_{\text{참(1)}} \quad \boxed{\&\&} \quad \underbrace{(\overset{800}{\text{toeic}} \geq 700)}_{\text{참(1)}}$$

참(1)

○ 신입 사원을 채용하는 조건이 나이가 30살 이하이거나 (OR) 토익 성적이 700점 이상이면 된다.

$$\underbrace{(\overset{27}{\text{age}} \leq 30)}_{\text{참(1)}} \quad \boxed{||} \quad \underbrace{(\overset{699}{\text{toeic}} \geq 700)}_{\text{거짓(0)}}$$

참(1)

논리 연산자의 예

○ "x는 1, 2, 3중의 하나인가"

➤ `(x == 1) || (x == 2) || (x == 3)`

○ "x가 60이상 100미만이다."

➤ `(x >= 60) && (x < 100)`

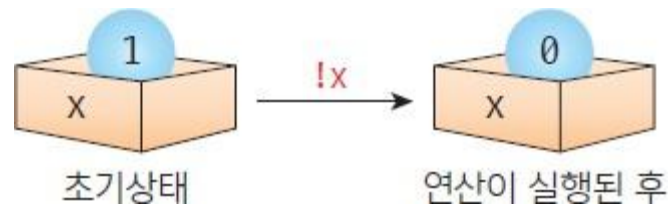
○ "x가 0도 아니고 1도 아니다."

➤ `(x != 0) && (x != 1)` `// x≠0 이고, x≠1이다`

.

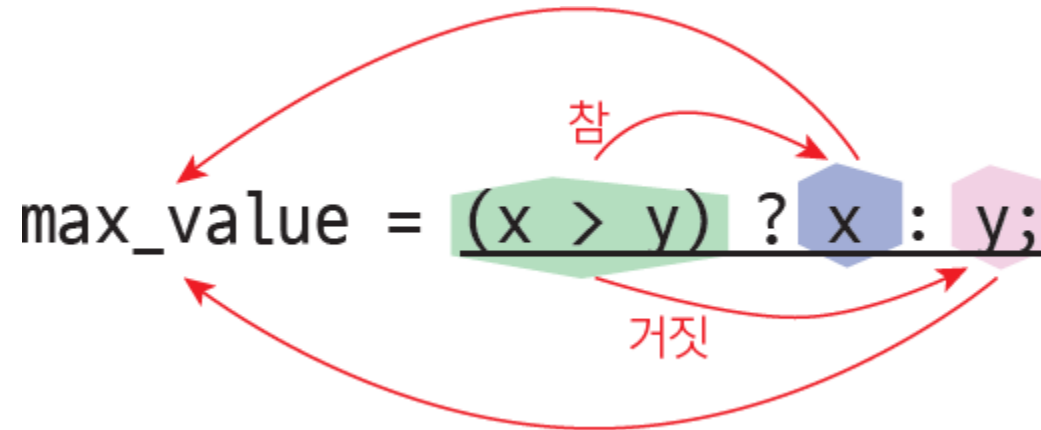
NOT 연산자

- 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.
```

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;
```

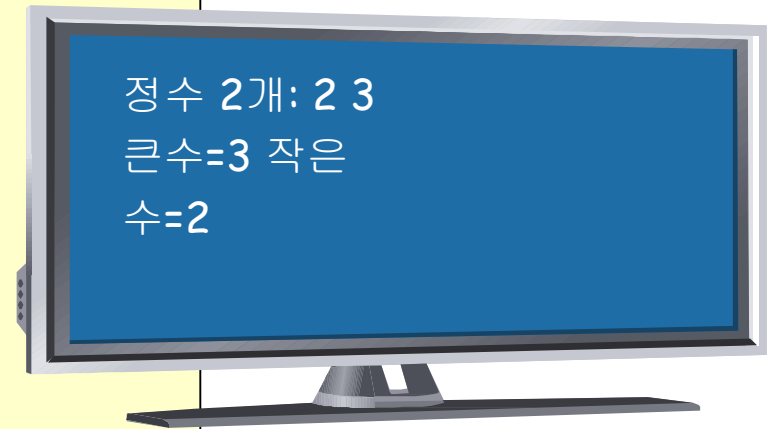
```
    printf("정수 2개: ");
```

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

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

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

```
}
```



비트 연산자

- 모든 데이터는 비트로 표현
- 비트 연산자의 종류

연산자	연산자의 의미	예
&	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

$0 \text{ AND } 0 = 0$
$1 \text{ AND } 0 = 0$
$0 \text{ AND } 1 = 0$
$1 \text{ AND } 1 = 1$

변수1 00000000 00000000 00000000 00001001 (9)
변수2 00000000 00000000 00000000 00001010 (10)

(변수1 AND 변수2) 00000000 00000000 00000000 00001000 (8)

비트 OR 연산자

$0 \text{ OR } 0 = 0$
$1 \text{ OR } 0 = 1$
$0 \text{ OR } 1 = 1$
$1 \text{ OR } 1 = 1$

변수1 00000000 00000000 00000000 00001001 (9)
변수2 00000000 00000000 00000000 00001010 (10)

(변수1 OR 변수2) 00000000 00000000 00000000 00001011 (11)

비트 XOR 연산자

$0 \text{ XOR } 0 = 0$
$1 \text{ XOR } 0 = 1$
$0 \text{ XOR } 1 = 1$
$1 \text{ XOR } 1 = 0$

변수1 00000000 00000000 00000000 00001001 (9)

변수2 00000000 00000000 00000000 00001010 (10)

(변수1 XOR 변수2) 00000000 00000000 00000000 00000011 (3)

비트 NOT 연산자

NOT 0 = 1
NOT 1 = 0

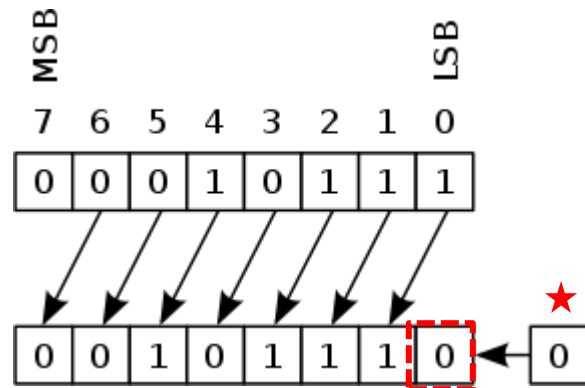
부호비트가 반전되었기 때문에 음수가 된다.

변수1 00000000 00000000 00000000 00001001 (9)

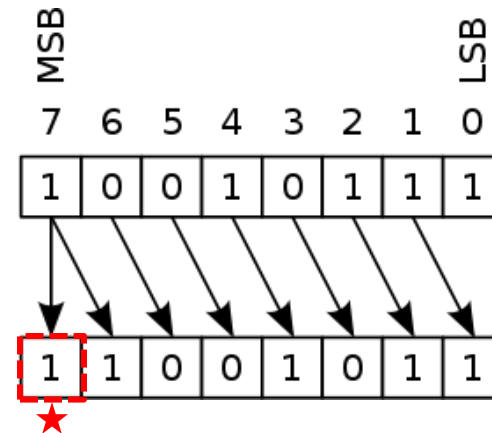
(NOT 변수1) 11111111 11111111 11111111 11110110 (-10)

비트 이동 연산자

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



이동 비트만큼
'0'이 채워짐

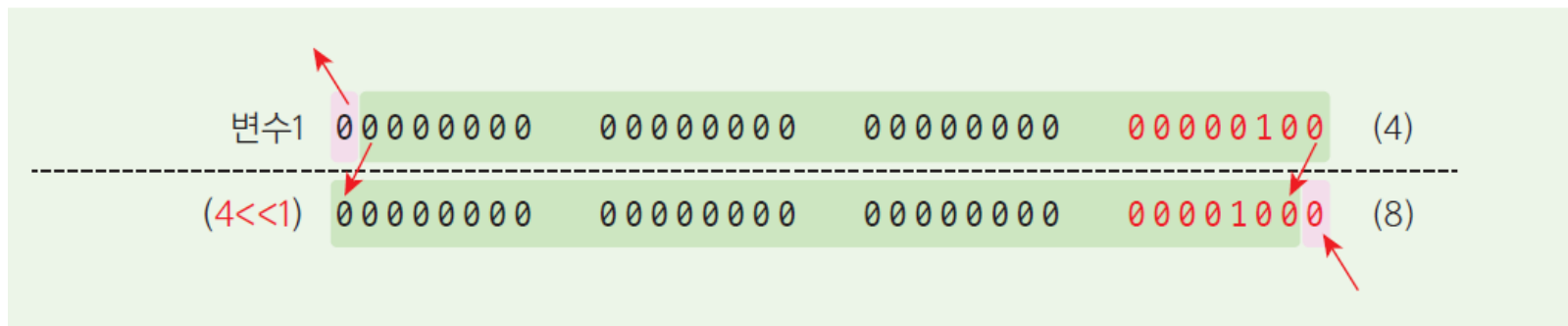


이동 비트만큼 부호
비트가 채워짐

<< 연산자 / >> 연산자

○ << 연산자 : 비트를 왼쪽으로 이동

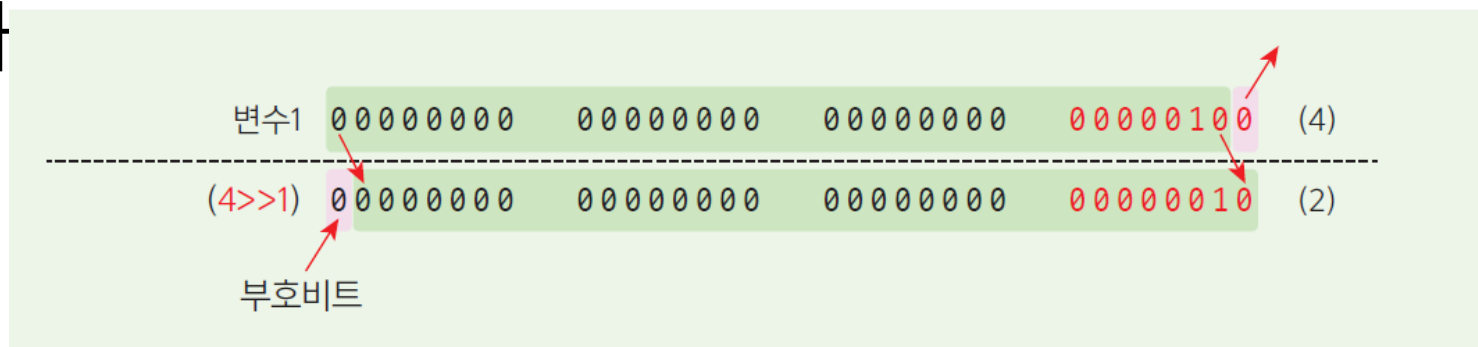
➤ 값은 2배가 된다.



○ >> 연산자 : 비트를 오른쪽으로 이동

➤ 값은 1/2배가 된다

○



예제 : 비트 연산자

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

```
int main(void)
```

```
{
```

```
    printf("AND : %08X \n", 0x9 & 0xA);
```

```
    printf("OR : %08X \n", 0x9 | 0xA);
```

```
    printf("XOR : %08X \n", 0x9 ^ 0xA);
```

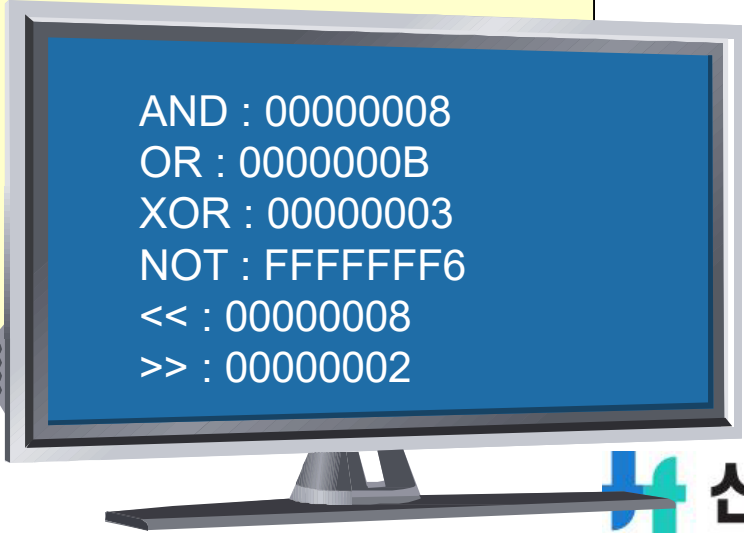
```
    printf("NOT : %08X \n", ~0x9);
```

```
    printf("<< : %08X \n", 0x4 << 1);
```

```
    printf(">> : %08X \n", 0x4 >> 1);
```

```
    return 0;
```

```
}
```

$$\begin{array}{r} 0000\ 1001 \\ \& \)\ 0000\ 1010 \\ \hline 0000\ 1000 \end{array}$$
$$\begin{array}{r} 0000\ 1001 \\ | \)\ 0000\ 1010 \\ \hline 0000\ 1011 \end{array}$$
$$\begin{array}{r} 0000\ 1001 \\ \wedge \)\ 0000\ 1010 \\ \hline 0000\ 0011 \end{array}$$
$$\begin{array}{r} 0000\ 0100 \\ <<1\ 0000\ 100\textcolor{red}{0} \end{array}$$
$$\begin{array}{r} 0000\ 0100 \\ >>1\ \textcolor{red}{0}000\ 0010 \end{array}$$


```
AND : 00000008
OR : 0000000B
XOR : 00000003
NOT : FFFFFFFF6
<< : 00000008
>> : 00000002
```

예제 : 비트 연산자로 2의 보수 만들기

```
#include <stdio.h>

int main(void)
{
    int a = 32;
    a = ~a;           //NOT 연산자로 1의 보수로 만든다.
    a = a + 0x01;      //1을 더한다.
    printf("a= %d \n", a);

    return 0;
}
```

	0010 0000	32
~)	1101 1111	~a
+)	0000 0001	+1
	1110 0000	
1's compl.	0001 1111	
	0000 0001	
2's compl.	0010 0000	-32

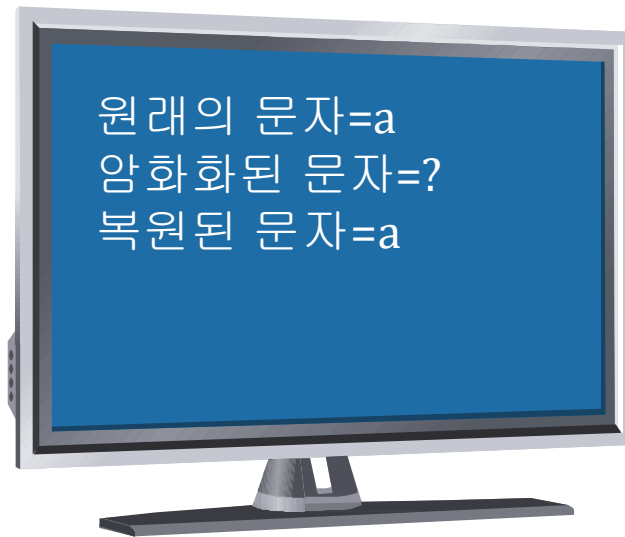
a= -32

실습 : XOR를 이용한 암호화

○ 하나의 문자 x 를 암호화

$x = x \oplus \text{key};$

○ 복호화도 $x = x \oplus \text{key};$



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

    printf("원래의 문자=%c\n", data);

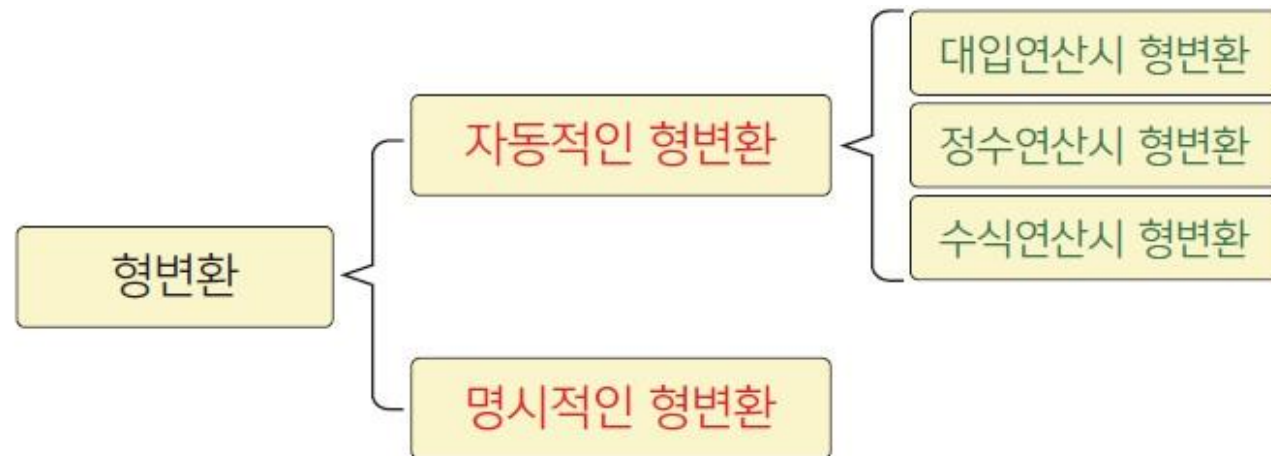
    encrypted_data = data ^ key;
    printf("암호화된 문자=%c\n", encrypted_data);

    orig_data = encrypted_data ^ key;
    printf("복원된 문자=%c\n", orig_data);

    return 0;
}
```


형 (Type) 변환

○ 연산 수행에서 데이터의 유형이 변환되는 것



○ 올림 변환 / 버림 변환

```
double f;  
f = 10 ;    // f에는 10.0이 저장
```

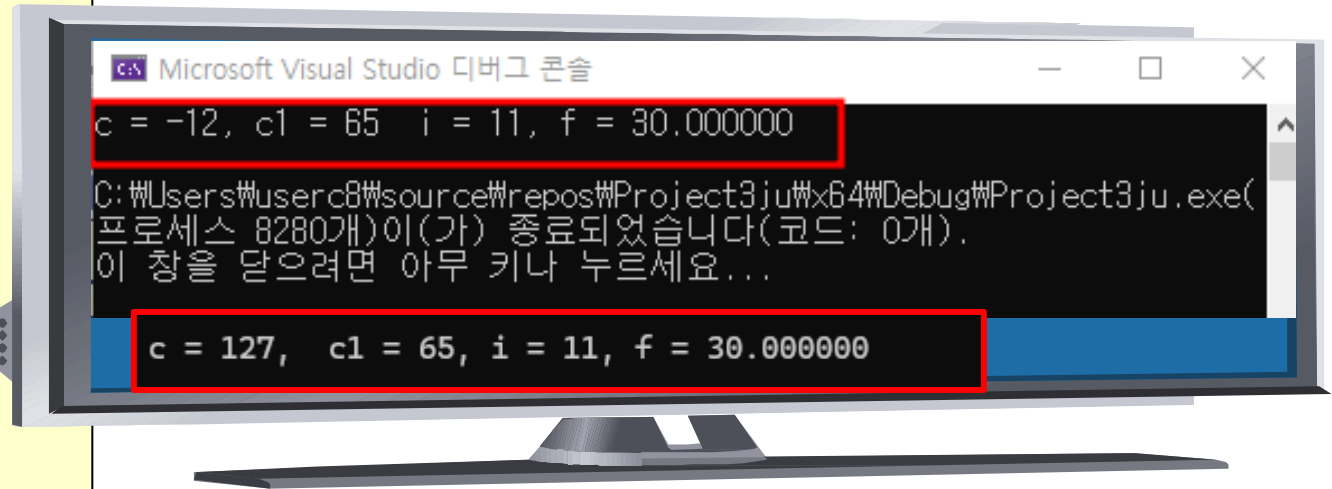
```
int i;  
i = 3.141592;    // i에는 3이 저장
```

자동 형 변환(1)

○ 정수 연산 시 char형, short형은 자동적으로 int형으로 변환

```
#include <stdio.h>
int main(void)
{
    char c, c1;
    int i;
    float f;

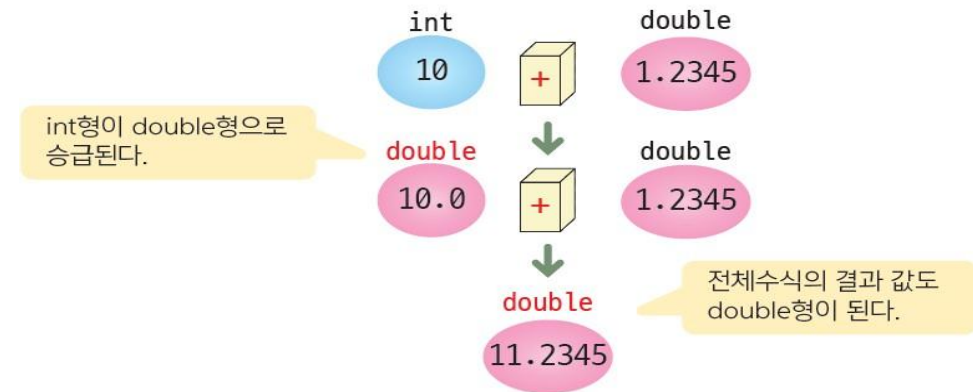
    c = 127;           // 0~127은 그대로 출력( $2^{8-1} - 1$ )
    c1 = 'A';          // 문자 A의 아스키 코드 값 출력
    i = 1.23 + 10;      // 11.23에서 소수점 이하 버림
    f = 10 + 20;        // 합계 30을 실수 형으로 변환
    printf("c = %d, c1 = %d, i = %d, f = %f \n", c, c1, i, f);
    return 0;
}
```



수식의 자동형 변환 / 명시적인 형 변환

○ 수식에 서로 다른 자료형이 혼합 사용되면, 큰 자료형 으
로 변환 처리

○ $\text{double} > \text{float} > \text{int} > \text{short}$



○ 명시적인 형 변환은 변수나 데이터 앞에 ()를 사용해서 형
변환 가능

Syntax

형변환

예

자료형

(int)1.23456
(double) x
(long) (x+y)

수식

// 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;
```

5/4는 1이 되고 이것이 1.00이 된다.

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

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

5가 5.0으로 되어서 전체 결과가 1.25가 된다.

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

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

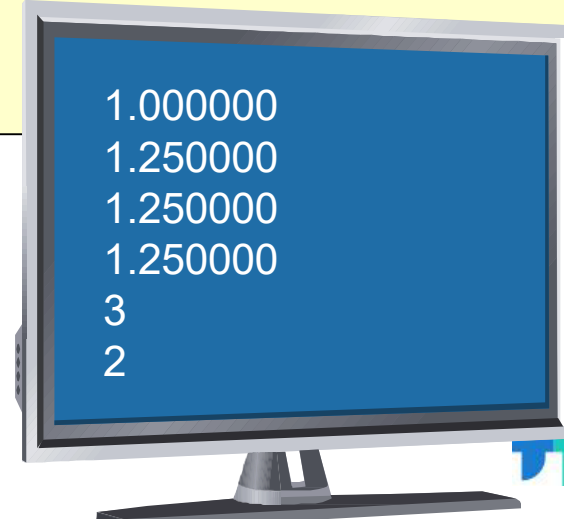
```
i = 1.3 + 1.8;
printf("%d\n", i);
```

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

1.3이 1이 되고 1.8도 1이 되어서 최종 결과는 2가 된다.

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

```
}
```



```
1.000000
1.250000
1.250000
1.250000
3
2
```

연산자의 우선 순위

- 어떤 연산자를 먼저 계산할 것인지에 대한 규칙
- 우선 순위 : 산술 > 관계 > 논리 > 대입
- 괄호 연산자는 가장 우선순위가 높다.
- 단항 연산자들은 이항 연산자들보다 우선순위가 높다.
- 연산자들의 우선 순위가 생각나지 않으면 괄호를 이용
 - `(x <= 10) && (y >= 20)`

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)	

단항

연산자의 우선 순위

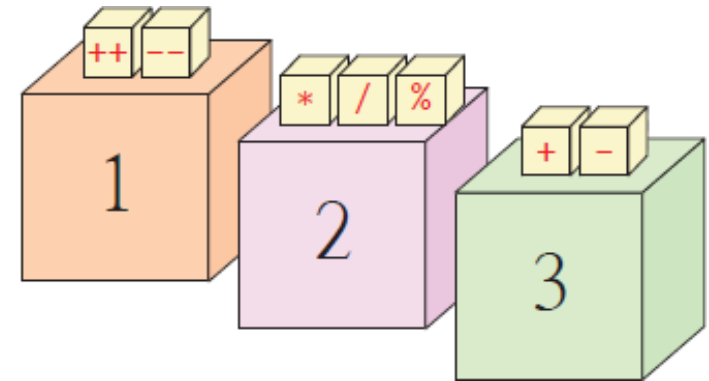
3	* / %	곱셈, 나눗셈, 나머지	} 산술	→ (좌에서 우)
4	+ -	덧셈, 뺄셈		
5	<< >>	비트 이동 연산자		
6	< <=	관계 연산자	} 관계	
	> >=	관계 연산자		
7	== !=	관계 연산자		
8	&	비트 AND	} 비트	
9	^	비트 XOR		
10		비트 OR		
11	&&	논리 AND 연산자	} 논리	
12		논리 OR 연산자		
13	?:	삼항 조건 연산자	} 대입	← (우에서 좌)
14	=	대입 연산자		
	+= -=	복합 대입 연산자		
	*= /= %=	복합 대입 연산자		
	<<= >>=	복합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

$$x + y * z$$

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

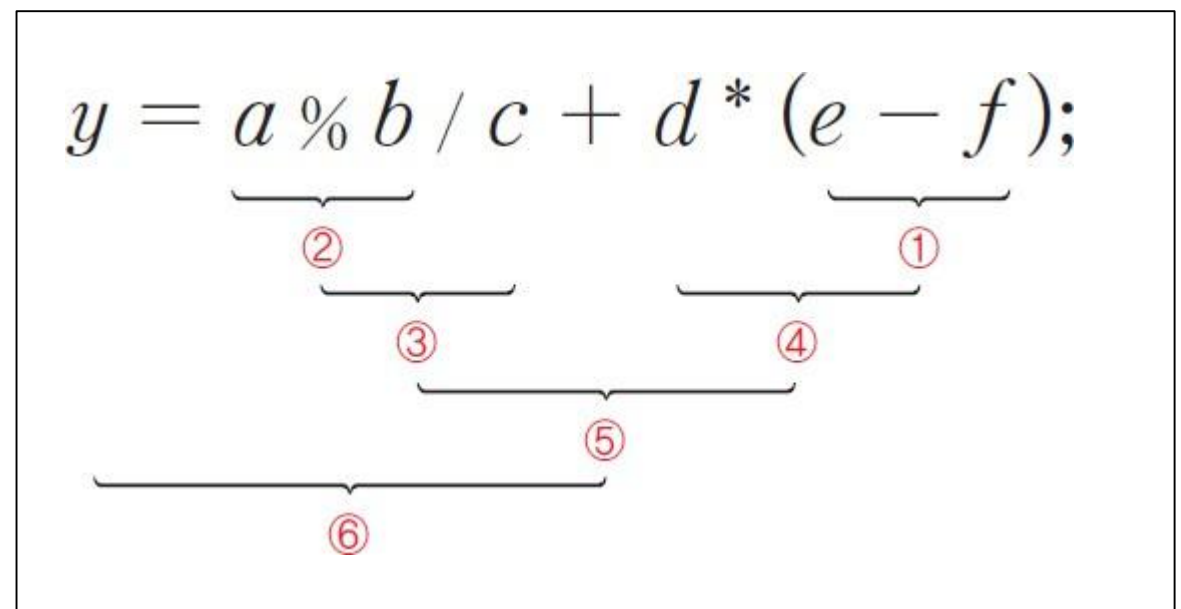
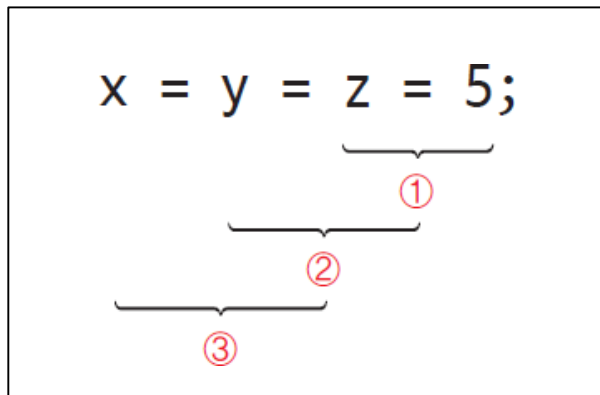
$$(x + y) * z$$

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



결합 규칙

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



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-;
    printf("%d", result);

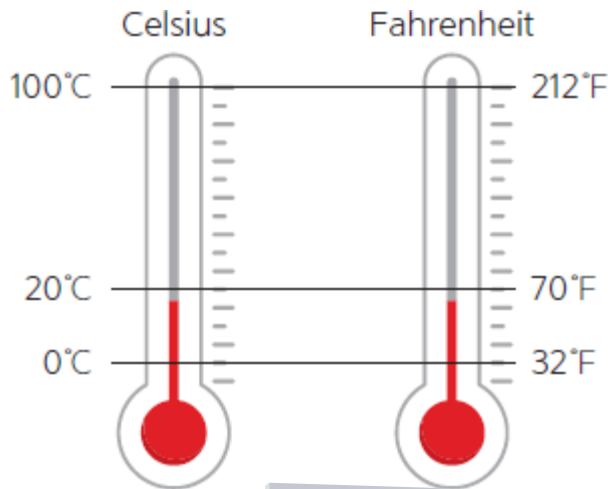
    return 0;
}
```



실습 : 화씨 온도를 섭씨로 바꾸기

○ 화씨 온도를 섭씨 온도로 바꾸는 프로그램을 작성하여 보자.

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



화씨온도를 입력하시오 90
섭씨온도는 32.22222입니다

```
#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;
}
```

프로그램
수정하기!

화씨온도를 입력하시오: 90
섭씨온도는 0.000000입니다.

5/9가 먼저 계산
되어서 0이 된다.

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

퀴즈

1. 비트를 지정된 숫자만큼 왼쪽으로 이동시키는 연산자는 _____이다.
2. 비트의 값을 0에서 1로, 1에서 0으로 바꾸는데 사용하는 연산자는 _____이다.
3. 변수 x 의 값을 2배로 하려면 _____쪽으로 비트를 이동시키면 된다.
4. 변수 x 의 값을 $1/2$ 배로 하려면 _____쪽으로 비트를 이동시키면 된다.



2강 - 정리 요약

○ 자료형

- 문자(char)
- 숫자 : 정수(int) / 실수(float, double)

```
변수 x의 크기 : 4
char 형의 크기 : 1
int 형의 크기 : 4
short 형의 크기 : 2
long 형의 크기 : 4
long long 형의 크기 : 8
float 형의 크기 : 4
double 형의 크기 : 8
```

○ 자료형과 길이

```
#include <stdio.h>

void main()
{
    int x;
    printf("변수 x의 크기 : %d Wn", sizeof(x));

    printf("char 형의 크기 : %d Wn", sizeof(char));
    printf("int 형의 크기 : %d Wn", sizeof(int));
    printf("short 형의 크기 : %d Wn", sizeof(short));
    printf("long 형의 크기 : %d Wn", sizeof(long));
    printf("long long 형의 크기 : %d Wn", sizeof(long long));
    printf("float 형의 크기 : %d Wn", sizeof(float));
    printf("double 형의 크기 : %d Wn", sizeof(double));
}
```

2강 - 정리 요약

○ 자료형

- 문자(char)
- 숫자 : 정수(int) / 실수(float, double)

```
변수 x의 크기 : 4
char 형의 크기 : 1
int 형의 크기 : 4
short 형의 크기 : 2
long 형의 크기 : 4
long long 형의 크기 : 8
float 형의 크기 : 4
double 형의 크기 : 8
```

○ 자료형과 길이

```
#include <stdio.h>

void main()
{
    int x;
    printf("변수 x의 크기 : %d Wn", sizeof(x));

    printf("char 형의 크기 : %d Wn", sizeof(char));
    printf("int 형의 크기 : %d Wn", sizeof(int));
    printf("short 형의 크기 : %d Wn", sizeof(short));
    printf("long 형의 크기 : %d Wn", sizeof(long));
    printf("long long 형의 크기 : %d Wn", sizeof(long long));
    printf("float 형의 크기 : %d Wn", sizeof(float));
    printf("double 형의 크기 : %d Wn", sizeof(double));
}
```