

Basic Computer Programming

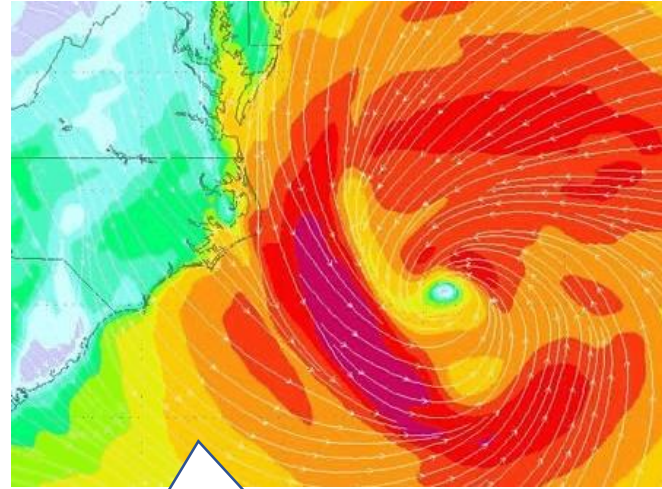
Lecture 4

Electrical & Electronics Engineering
Chung-Ang University

Contents

- Learn about arithmetic operators such as adding, subtracting, multiplying, and dividing.
- Learn about the remaining operators.
- Learn substitution (allocation) operators and complex operators.
- Understand the concept of priority.

What can we do with computation?



When predicting the weather, you have to calculate with a supercomputer.

The program that we are going to make in this chapter.

```
Microsoft Visual Studio 디버그 콘솔

투입한 돈: 1000
물건값: 270
거스름돈: 730

100원 동전의 개수: 7
10원 동전의 개수: 3
```

```
Microsoft Visual Studio 디버그 콘솔

 $y = 3.0 * x * x + 7.0 * x + 9.0 = 35.000000$ 
```

```
Microsoft Visual Studio 디버그 콘솔

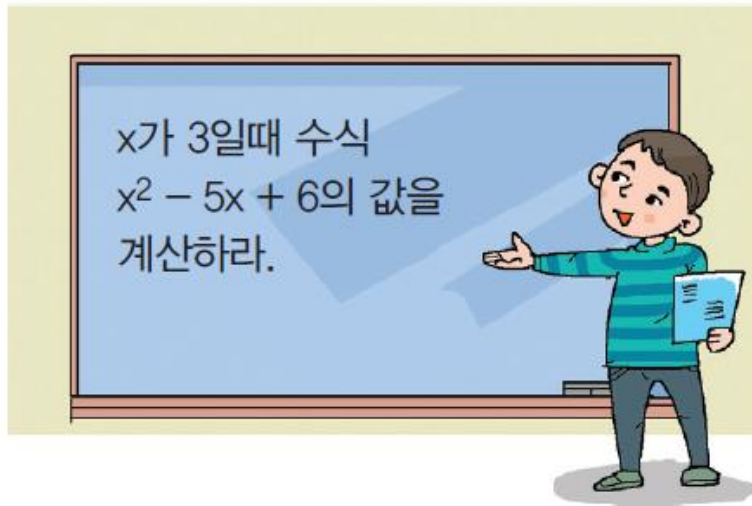
초기저금액: 24.0
이율: 0.06
저축기간: 382
382년 후의 원리금=111442737812.288422
```

Equations are everywhere!

- Computer graphic scenes in movies are made through computing functions.
- The explosion scene of buildings is displayed the computer-calculated results on the screen using various formulas of physics.



An example of equations



```
int x, y;
```

```
x = 3;
```

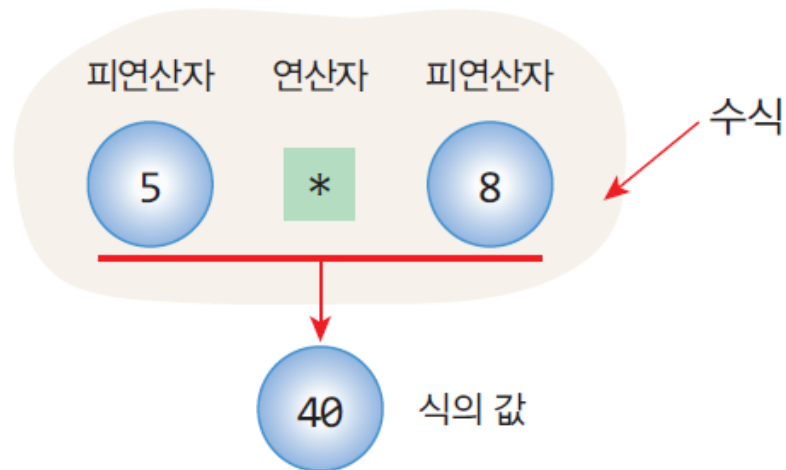
```
y = x*x - 5*x + 6;
```

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

그림 4.1 수식의 예

Definition of equations

- Definition of equations
 - A combination of constants, variables, and operators.
 - It is divided into operators and operands.
 - It has a result value.



Classification of operators according to functions

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
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

Interim check

1. How is an expression defined?
2. Separate the operand from the operator in the formula below.

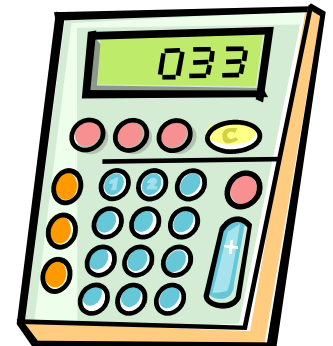
$$y = 10 + 20;$$



Arithmetic operators

- Arithmetic operation: the most basic operation of a computer
- An operator that performs addition, subtraction, multiplication, and division.

연산자	기호	사용예	결괏값
덧셈	+	$7 + 4$	11
뺄셈	-	$7 - 4$	3
곱셈	*	$7 * 4$	28
나눗셈	/	$7 / 4$	1
나머지	%	$7 \% 4$	3



Example

```
#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;           // 뺄셈
```

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

```
    result = x * y;          // 곱셈
```

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

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

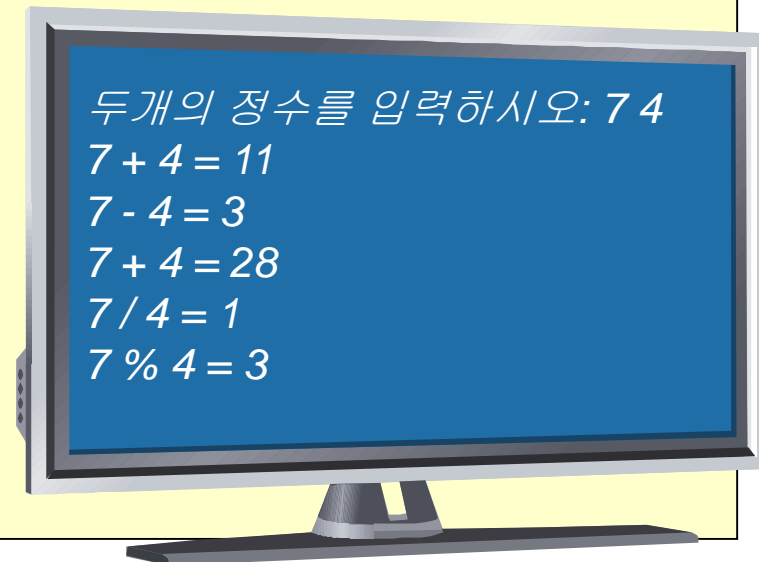
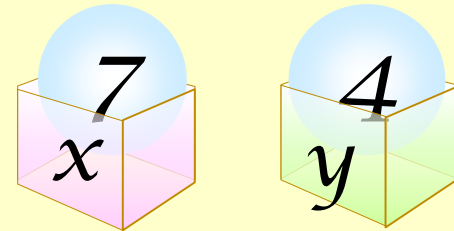
```
    printf("%d / %d = %d", x, y, result);
```

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

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

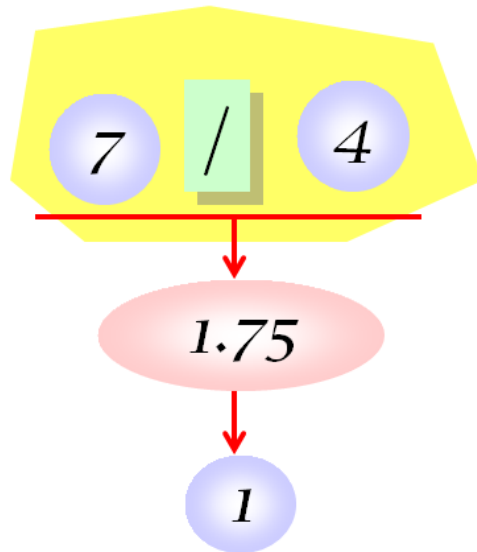
```
    return 0;
```

```
}
```

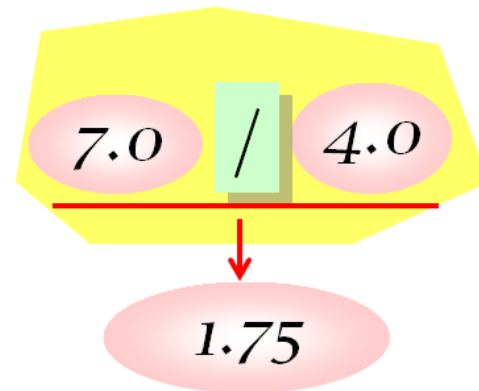


Division operator

- In the division between integers, the result is generated in an integer form, and the floating point type is generated between floating point values.
- In division between integers, less than or equal to decimal points are discarded.



정수와 정수 끼리의 나눗셈.



실수와 실수 끼리의 나눗셈.

Division operator

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

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

    printf("두개의 실수를 입력하시오: ");
    scanf("%lf %lf", &x, &y); // double형을 입력받으려면 %lf를 사용한다.

    result = x + y; // 덧셈 연산을 하여서 결과를 result에 대입
    printf("%lf + %lf = %lf\n", x, y, result);

    result = x - y; // 뺄셈 연산
    printf("%lf - %lf = %lf\n", x, y, result);

    result = x * y; // 곱셈 연산
    printf("%lf * %lf = %lf\n", x, y, result);

    result = x / y; // 나눗셈 연산
    printf("%lf / %lf = %lf\n", x, y, result);

    return 0;
}
```

Results

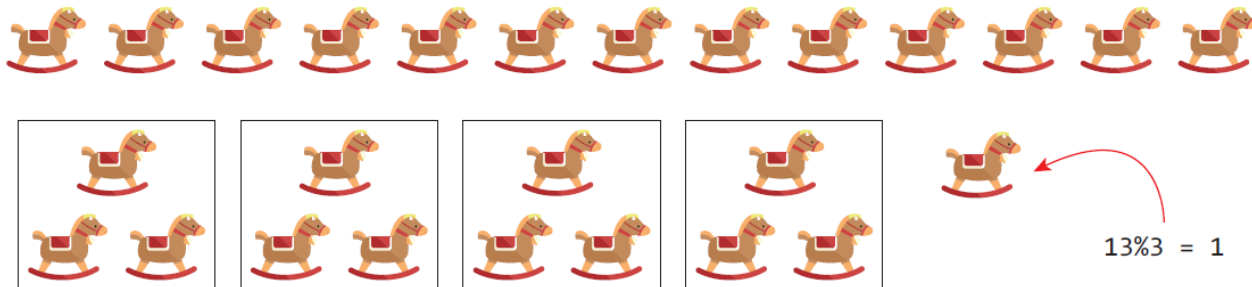


The image shows a screenshot of the Microsoft Visual Studio Debug Console window. The window has a title bar with the text "Microsoft Visual Studio 디버그 콘솔" and standard window control buttons (minimize, maximize, close). The console output displays the results of arithmetic operations on the input values 7.0 and 4.0. The text is as follows:

```
두개의 실수를 입력하시오: 7.0 4.0  
7.000000 + 4.000000 = 11.000000  
7.000000 - 4.000000 = 3.000000  
7.000000 * 4.000000 = 28.000000  
7.000000 / 4.000000 = 1.750000
```

Other operators

- The remainder operator calculates the remainder where the first value is divided by the second value.
- $13\%3=1$



Other operators

- (e.g.) Separate even numbers and odd numbers using the remainder operators.
If $x\%2$ is 0, it's even.
- (e.g.) Determining multiple of 5 using the remainder operators.
If $x\% 5$ is 0, multiple of 5.


```
// 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.
```

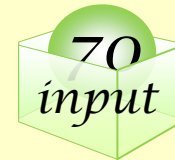
```
    minute = input / SEC_PER_MINUTE; // 몇 분
```

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

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

```
    return 0;
```

```
}
```



Microsoft Visual Studio 디버그 콘솔

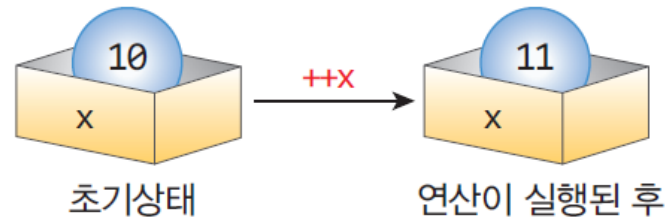
초단위의 시간을 입력하시요:(32억초이하) 1000
1000초는 16분 40초입니다.

Increase/decrease operator

- Increase/decrease operators: ++, --
- An operator that increases or decreases the value of a variable.
- ++ operator can be used as follows. The following two sentences have the same meaning.

`x++;`

`x = x + 1;`



- -- operator can be used as follows. The following two sentences have the same meaning.

`x--;`

`x = x - 1;`



Increase/decrease operator

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.

Difference between ++x and x++

```
x = 1;
```

```
y = 1;
```

```
nextx = ++x;    // x의 값이 증가된 후에 사용된다. nextx는 2가 된다.
```

```
nexty = y++;    // y의 값이 사용된 후에 증가된다. nexty는 1이 된다.
```

Increase/decrease operator

```
#include <stdio.h>

int main(void)
{
    int x, y; // 변수 x, y를 선언한다.

    x = 1; // x는 1이다.
    y = ++x; // x는 2이고 y는 2이다.
    printf("x=%d y=%d \n", x, y);

    y = x++; // x는 3이고 y는 2이다.
    printf("x=%d y=%d \n", x, y);

    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

```
x=2 y=2
x=3 y=2
```

Lab: Converting working days to years, weeks, and days

- We want to know how long we have been working. If you enter the number of working days, let's write a program that calculates how many years, months, and days it corresponds to. For example, 389 days correspond to 1 year, 3 weeks and 3 days.



```
Microsoft Visual Studio 디버그 콘솔
총 일수를 입력하시오: 389
1년 3주 3일
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main(void)
{
    int days, years, weeks;
    printf("총 일수를 입력하시오: ");
    scanf("%d", &days);

    years = (days / 365);           // 년수
    weeks = (days % 365) / 7;      // 주수
    days = days - ((years * 365) + (weeks * 7)); // 남은 일수

    printf("%d년 ", years);
    printf("%d주 ", weeks);
    printf("%d일 \n", days);
    return 0;
}
```

Lab: Vending Machine Program

- Let's write a program to simulate vending machines. The vending machine receives input of money and goods from the user.

```
Microsoft Visual Studio 디버그 콘솔
투입한 돈: 1000
물건값: 270
거스름돈: 730

100원 동전의 개수: 7
10원 동전의 개수: 3
```



Sol: Vending Machine Program

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

int main(void)
{
    int money, price, change;

    printf("투입한 돈: ");
    scanf("%d", &money);

    printf("물건값: ");
    scanf("%d", &price);

    change = money - price;
    printf("거스름돈: %d\n\n", change);
```

Sol: Vending Machine Program

```
int coin100s = change / 100; // 거스름돈에서 100원 짜리의 개수 계산  
change = change % 100;      // 거스름돈에서 100원 짜리를 내주고 남은 돈
```

```
int coin10s = change / 10;    // 거스름돈에서 10원 짜리의 개수 계산  
change = change % 10;        // 거스름돈에서 10원 짜리를 내주고 남은 돈
```

```
printf("100원 동전의 개수: %d\n", coin100s);  
printf("10원 동전의 개수: %d\n", coin10s);
```

```
return 0;
```

```
}
```



도전문제

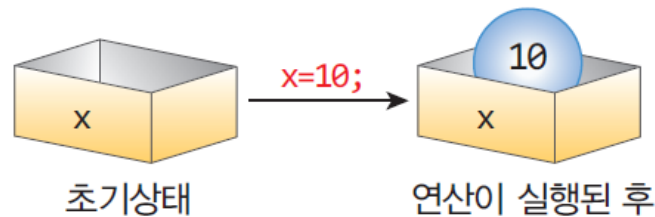
자동판매기가 만약 50원짜리 동전도 거슬러 줄 수 있다면 위의 코드를 어떻게 수정하여야 하는가?

Allocation/assignment operator

Syntax 4.1 대입 연산자

형식 변수(variable) = 수식(expression);

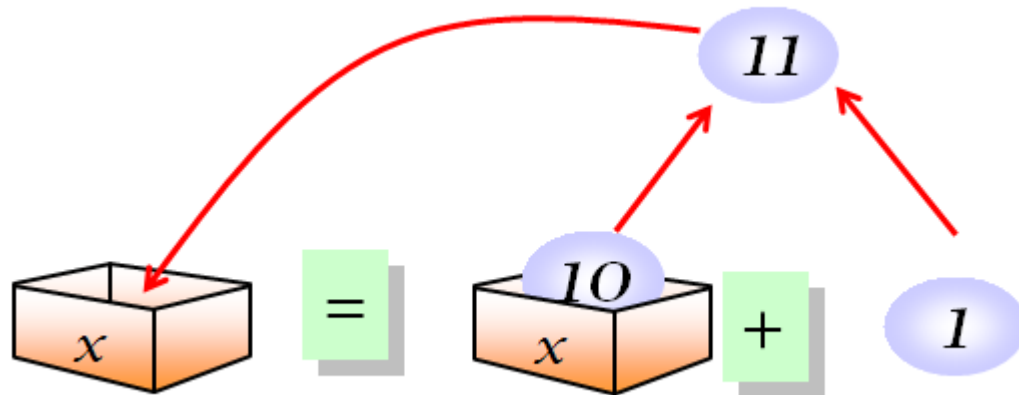
예 x = 10;



Allocation/assignment operator

$x = x + 1$

Increase the value of the variable
by 1.



Allocation/assignment operator

- Substituting operators can be used continuously. For example, a statement that substitutes 0 for variables x, y, and z can be written as follows.

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

- $z = 3$ is executed first, and the value of this equation, 3, is substituted back into y. Then, 3 is assigned to x.

Compound operators

- Compound operators are an operator that combines assignment operator = and arithmetic operator like +=.
- You can make the code simple.

복합 대입 연산자	의미
$x += y$	$x = x + y$
$x -= y$	$x = x - y$
$x *= y$	$x = x * y$
$x /= y$	$x = x / y$
$x \% = y$	$x = x \% y$

Quiz

- If you solve the following formula and rewrite it?

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

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



Compound operators

```
#include <stdio.h>
int main(void)
{
    int x = 10, y = 10;
    printf("x = %d y = %d \n", x, y);
    x += 1;
    printf("(x += 1)이후 x = %d \n", x);
    y *= 2;
    printf("(y *= 2)이후 y = %d \n", y);
    return 0;
}
```

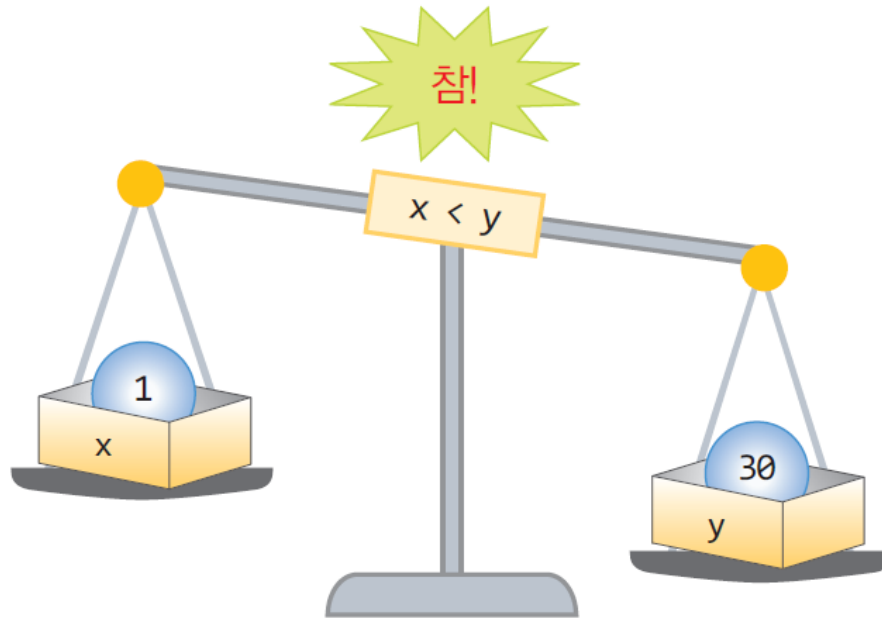


Microsoft Visual Studio 디버그 콘솔

```
x = 10    y = 10
(x += 1)이후 x = 11
(y *= 2)이후 y = 20
```


Relational operator

- A relational operator is used to compare two operands.
- For example, "Is the variable x equal to 0" and "Is the variable y less than 10"? The result of the relation operator is calculated as true or false.



Relational operator

연산	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?

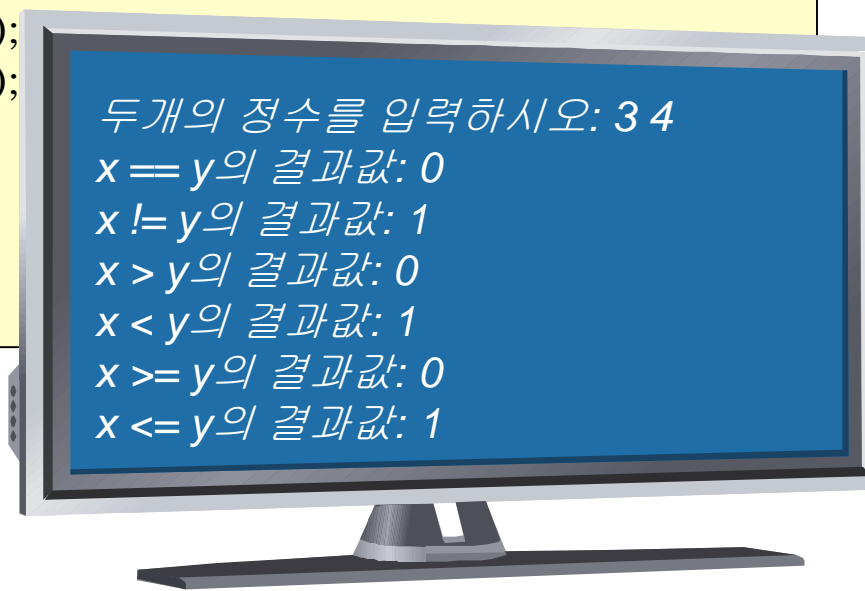
Example

```
#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

Cautions!

- $(x = y)$

Substitutes the value of y into x . The value of this equation is the value of x .

- $(x == y)$

If x and y are the same, 1, and if they are different, 0 is the value of the equation.

Be careful not to miswrite $(x == y)$ as $(x = y)$.

Interim check

1. What values can be generated as a result of the relationship equation?
2. What's the value of $(3 \geq 2) + 5$?

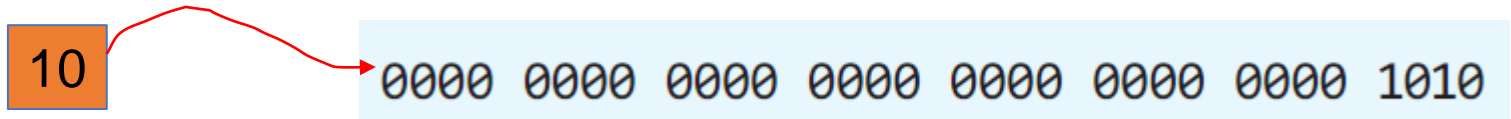


Logic and conditional operators.

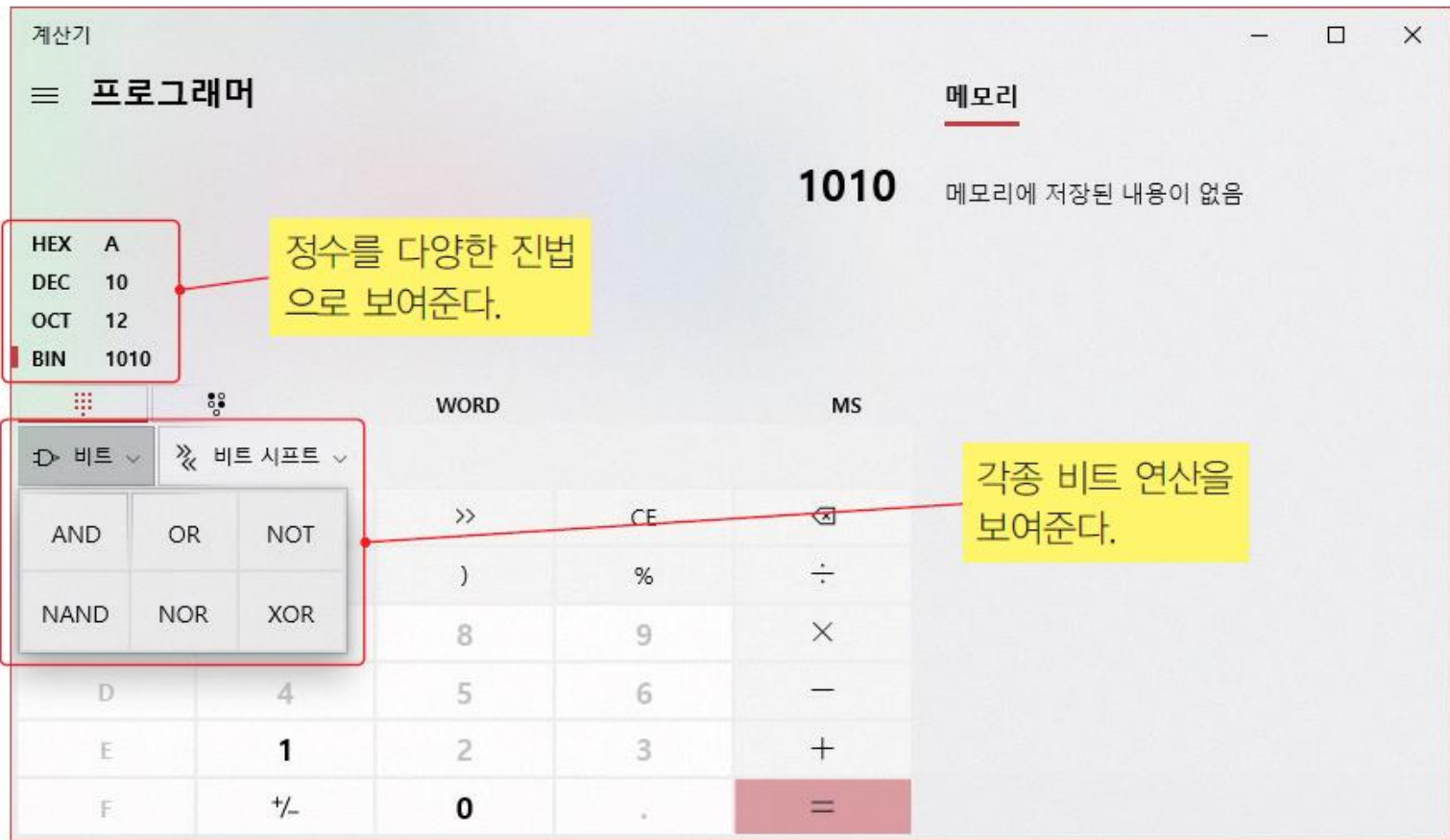
- Logic operators and conditional operators will be addressed in Chapter 5!

Bit operators

- In a computer, all data is eventually expressed in bits. Bit is the smallest unit of storing information in a computer. Since the bit corresponds to a binary digit, it may have a value of 0 or 1.
- For example, integer 10 is stored in the computer in the following 32-bit pattern.



Windows's "For Programmers" calculator.



Bit operators

연산자	연산자의 의미	설명
&	비트 AND	두 개의 피연산자의 해당 비트가 모두 1이면 1, 아니면 0
	비트 OR	두 개의 피연산자의 해당 비트중 하나만 1이면 1, 아니면 0
^	비트 XOR	두 개의 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.

Bit AND operator

0 AND 0 = 0
1 AND 0 = 0
0 AND 1 = 0
1 AND 1 = 1

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

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

Bit OR operator

0 OR 0 = 0
1 OR 0 = 1
0 OR 1 = 1
1 OR 1 = 1

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

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

Bit XOR operator

0 XOR 0 = 0
1 XOR 0 = 1
0 XOR 1 = 1
1 XOR 1 = 0

변수1 00000000 00000000 00000000 00001001 (9)

변수2 00000000 00000000 00000000 00001010 (10)

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

Bit NOT operator

NOT 0 = 1
NOT 1 = 0

부호비트가 반전되었기 때문에 음수가 된다.

변수1 00000000 00000000 00000000 00001001 (9)

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

Example: Bit operators

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

```
int main(void)
```

```
{
```

```
    int x = 9;                // x= 1001
```

```
    int y = 10;               // y= 1010
```

```
    printf("%08X & %08X = %08X\n", x, y, x & y); // x&y=1000
```

```
    printf("%08X | %08X = %08X\n", x, y, x | y); // x|y= 1011
```

```
    printf("%08X ^ %08X = %08X\n", x, y, x ^ y); // x^y= 0011
```

```
    printf("~ %08X = %08X\n", x, ~x);           // ~x= 0110
```

```
    return 0;
```

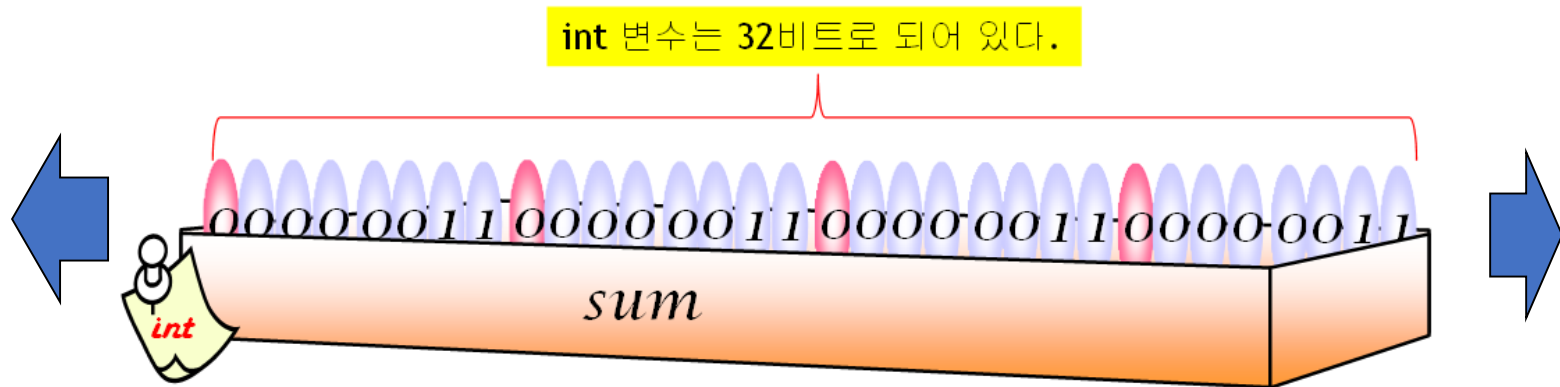
```
}
```

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

```
Microsoft Visual Studio 디버그 콘솔
00000009 & 0000000A = 00000008
00000009 | 0000000A = 0000000B
00000009 ^ 0000000A = 00000003
~ 00000009 = FFFFFFF6
```

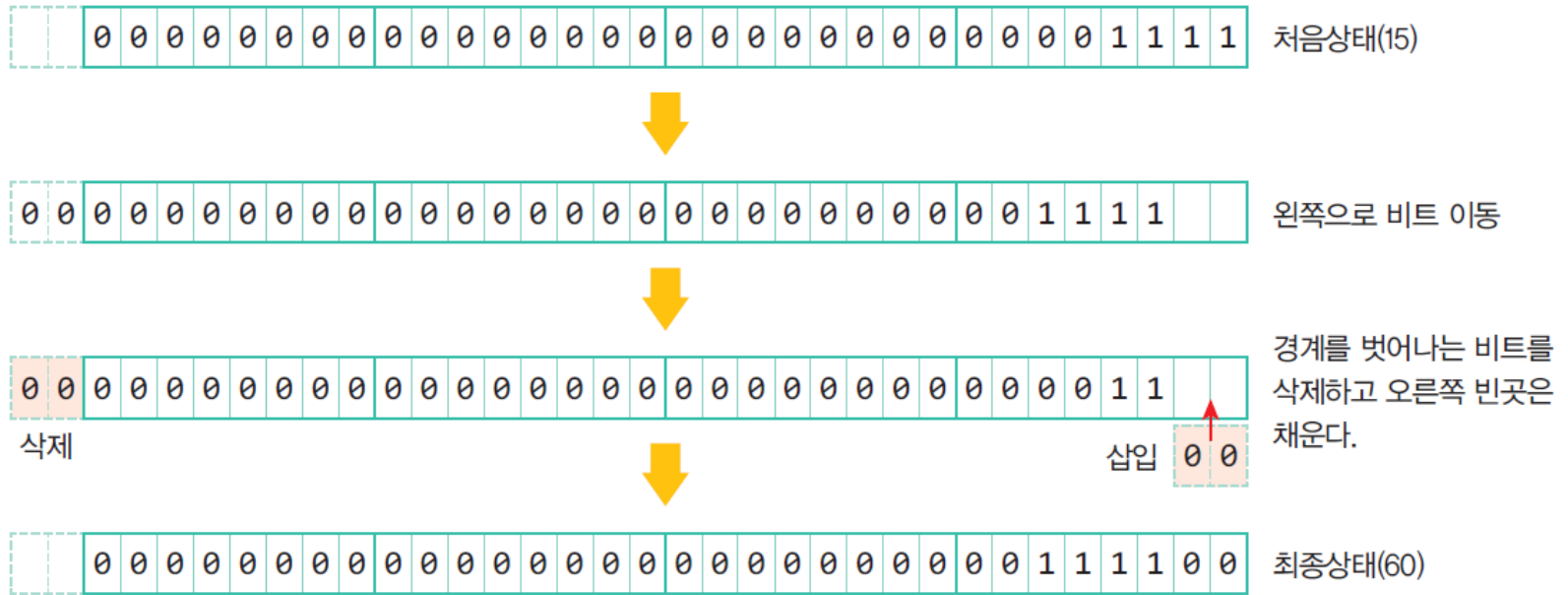
Bit shift operators

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



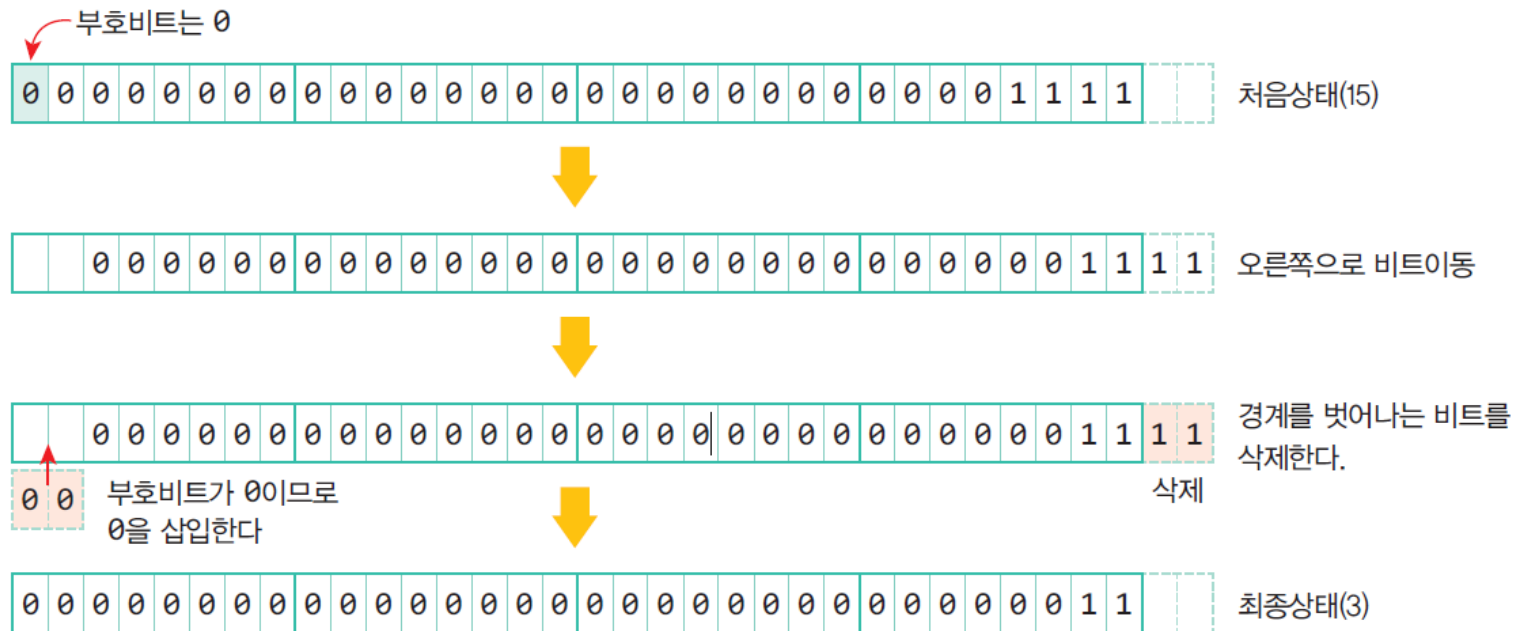
<< operator

- Move the bits to the left.
- The value is doubled.
- (Ex) $15 \ll 2$.



>> operator

- Move the bits to the right.
- The value is 1/2 times.
- (Example) $15 \gg 2$



Example: Bit shift operator

```
#include <stdio.h>
int main(void)
{
    int x = 9; // 1001
    printf("%d << 1 = %d\n", x, x << 1); // 10010
    printf("%d >> 1 = %d\n", x, x >> 1); // 00100
    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

```
9 << 1 = 18
9 >> 1 = 4
```

Interim check

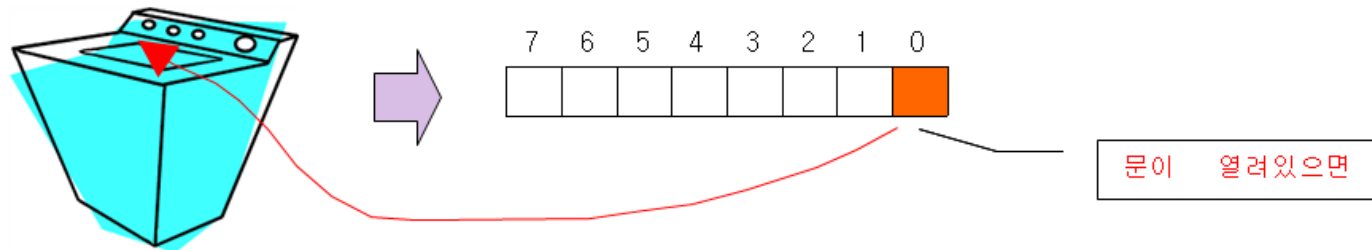


중간점검

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

Lab: Bit operation

- In what case is the bit operation used? It is used for communication between programs and hardware chips. For example, let's say there is a hardware chip that returns the values of the eight sensors in the washing machine in one byte. Let's say we read this byte as a variable called status. It is used to check whether a specific sensor value has reached 1. For example, if the washing machine door is open, let's say bit 0 is 1. Let's write a code that checks whether bit 0 is 0 or 1.



Lab: Bit operation

```
#include <stdio.h>
int main(void)
{
    int status = 0x6f; // 01101111
    printf("문열림 상태=%d \n", (status & 0x01));
    return 0;
}
```



The screenshot shows a console window titled "Microsoft Visual Studio 디버그 콘솔". The output text is "문열림 상태=1".



도전문제

만약 문열림을 나타내는 비트가 비트2라면 위의 프로그램은 어떻게 변경해야 하는가? 비트 이동 연산의 사용도 고려해보자.

Priority

- The rule of which operator to calculate first.

$$x + y * z$$

Diagram illustrating operator precedence for the expression $x + y * z$. A bracket labeled ① groups $y * z$, indicating that multiplication is performed first. A second bracket labeled ② groups the entire expression $x + y * z$, indicating that addition is performed second.

$$(x + y) * z$$

Diagram illustrating operator precedence for the expression $(x + y) * z$. A bracket labeled ① groups $x + y$, indicating that addition is performed first. A second bracket labeled ② groups the entire expression $(x + y) * z$, indicating that multiplication is performed second.

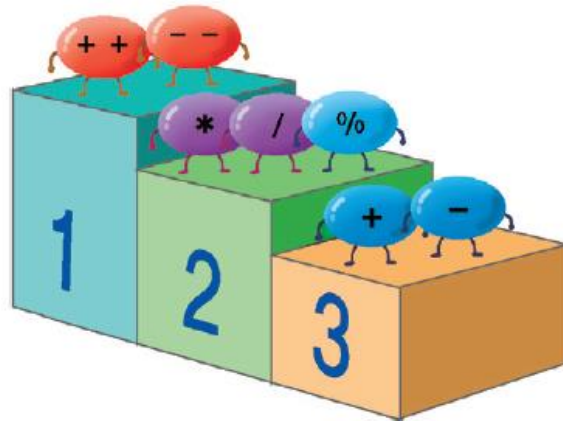


그림 4.8 증감 > 곱셈, 나눗셈, 나머지 > 덧셈, 뺄셈 순의 우선순위를 가진다.

Priority

- The question of which operator to calculate first in the equation.

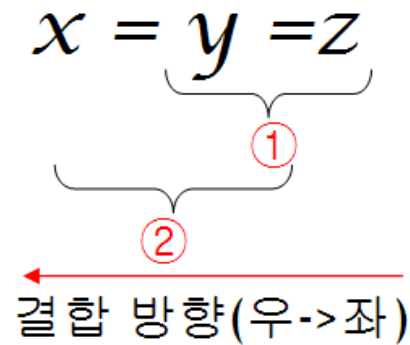
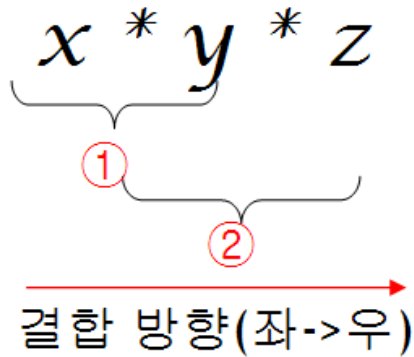
우선순위	연산자	결합규칙
1	() [] -> . ++(후위) --(후위)	→(좌에서 우)
2	sizeof &(주소) ++(전위) --(전위) ~ ! *(억참조) +(부호) -(부호), 형변환	←(우에서 좌)
3	*(곱셈) / %	→(좌에서 우)
4	+(덧셈) -(뺄셈)	→(좌에서 우)
5	<< >>	→(좌에서 우)
6	< <= >= >	→(좌에서 우)
7	== !=	→(좌에서 우)
8	&(비트연산)	→(좌에서 우)
9	^	→(좌에서 우)
10		→(좌에서 우)
11	&&	→(좌에서 우)
12		→(좌에서 우)
13	?(삼항)	←(우에서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

General guidelines for priorities

- Comma < assignment < logic < relationship < arithmetic < unary operators
- The bracket operator has the highest priority.
- All unary operators have higher priorities than binary operators.
- Except for comma operators, substitution operators have the lowest priority.

Combination rule

- Rules on which operations should be performed first if there are multiple operations with the same priority.



Cautions



경고

* 연산자들의 우선순위가 생각나지 않으면 위험을 무릅쓰지 말고 정신적인 안정을 위해서라도 괄호를 이용하여 우선순위를 정확하게 지정해준다. 즉 다음과 같이 +와 <=의 우선 순위가 생각나지 않으면 괄호를 사용해서 먼저 계산되어야 하는 것을 묶어준다.

```
( x + 10 ) <= ( y + 20 )
```

* 관계 연산자는 산술 연산자보다 우선순위가 낮다. 즉 다음과 같은 수식은 마음 놓고 사용하여도 된다.

```
x + 2 == y + 3
```

* 일반적으로 단항 연산자는 이항 연산자보다 우선순위가 높다. 아래의 수식에서 ++은 <=보다 우선순위가 높다.

```
( ++x <= 10 )
```

Example #1

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

The diagram illustrates the evaluation order of the expression $y = a \% b / c + d * (e - f);$ using red lines and numbered circles (1-6) to show the sequence of operations:

- ①: $(e - f)$
- ②: $a \% b$
- ③: $a \% b / c$
- ④: $d * (e - f)$
- ⑤: $d * (e - f) / c$
- ⑥: $a \% b / c + d * (e - f)$

```
#include <stdio.h>

int main(void) {
    int a = 10;
    int b = 20;
    int c = 30;
    int d = 3;
    int result;

    result = a + b * c / d;
    printf("연산값: %d\n", result);
    result = (a + b) * c / d;
    printf("연산값: %d\n", result);
    result = a = b = 1;
    printf("연산값: %d\n", result);
    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

연산값: 210
연산값: 300
연산값: 1

Interim check

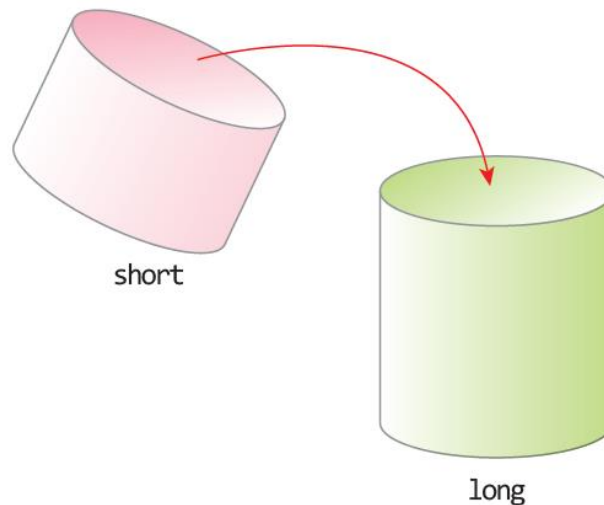


중간점검

1. 연산자 중에서 가장 우선순위가 낮은 연산자는 무엇인가?
2. 단항 연산자와 이항 연산자 중에서 어떤 연산자가 더 우선순위가 높은가?
3. 관계 연산자와 산술 연산자 중에서 어떤 연산자가 더 우선순위가 높은가?

Type casting

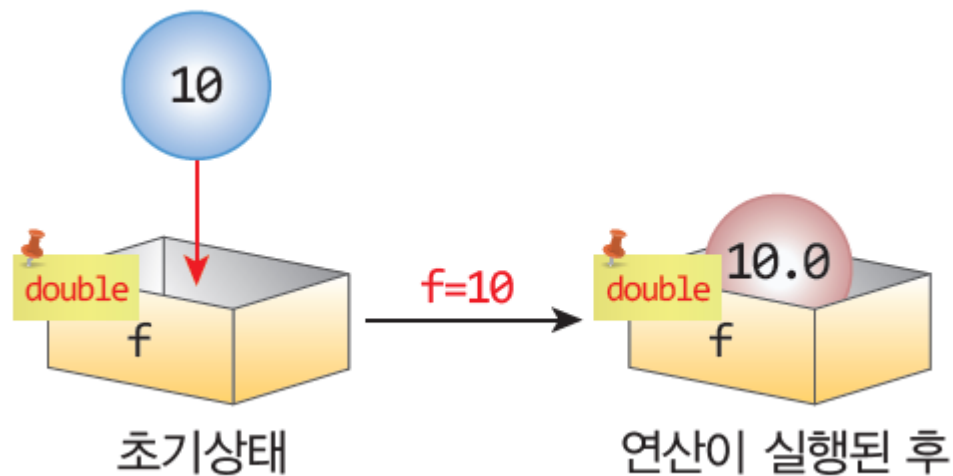
- Type casting is a process of converting a type of data. We can convert the int type into a double type if necessary.
- Automatic casting: automatically (implicitly) performed by a compiler.
- Explicit casting: Programmer explicitly transforms the type of data



Automatic casting

- Up-transformation

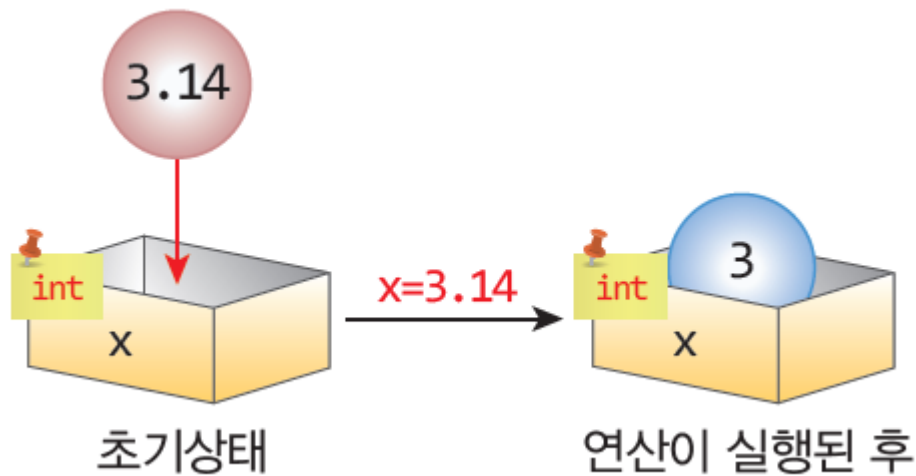
```
double f;  
f = 10;    // f에는 10.0이 저장된다.
```



Automatic casting

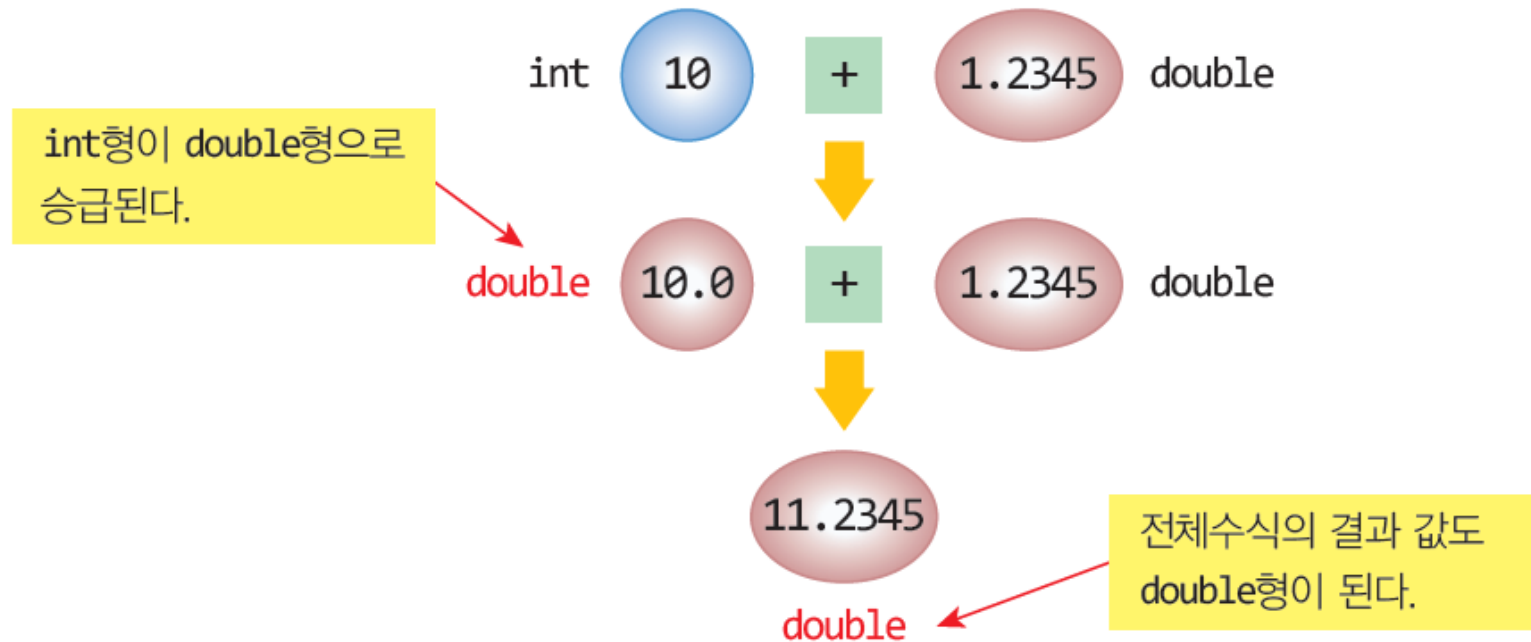
- Down-transformation

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



Automatic casting in formulas.

- When different data types are used in combination, they are unified into larger data types.




Explicit casting

- Type casting: The user changes the data type.
- The type of variable is not changed, but only the type of data is changed by taking out the data of the variable.

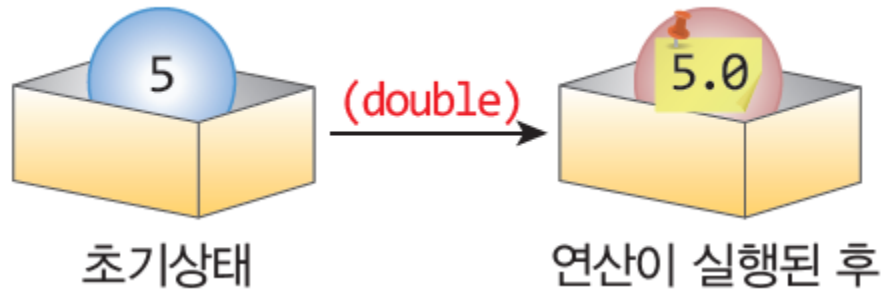
Syntax 4.2 형변환 연산자

형식 (자료형)수식

예 
(int)1.23456 // 상수
(double)x // 변수
(long)(x+1) // 수식

Explicit casting

```
x = 5;  
printf("%lf", (double) x)
```



예

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

```
int main(void) {
```

```
    int i;
```

```
    double f;
```

```
    f = 5 / 4;
```

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

```
    f = (double)5 / 4;
```

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

```
    i = 1.3 + 1.8;
```

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

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

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

```
    return 0;
```

```
}
```

Microsoft Visual Studio 디버그 콘솔

```
(5 / 4) = 1.000000  
(double)5 / 4 = 1.250000  
1.3 + 1.8 = 3  
(int)1.3 + (int)1.8 = 2
```

Lab: Calculation of quadratic functions

- When $x=2$ in a quadratic function, let's calculate the value of the function.



A screenshot of a Windows command prompt window. The title bar is red and contains the text "C:\Windows\system32\cmd.exe" and standard window controls (minimize, maximize, close). The command prompt area is white and contains the text "y=3.0*x*x + 7.0*x + 9.0=35.000000" and "계속하려면 아무 키나 누르십시오 . . .".

```
C:\Windows\system32\cmd.exe
y=3.0*x*x + 7.0*x + 9.0=35.000000
계속하려면 아무 키나 누르십시오 . . .
```

Interim check



중요

형변환을 하였다고 해서 변수의 형이 변경되는 것은 아니다. 변수가 가지고 있는 값을 꺼내서 형을 변경해서 수식에서 임시로 사용하는 것이다.



중간점검

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?

Lab: Calculation of quadratic functions

- If $y = 3x^2 + 7x + 9$ and $x=2$, let's calculate the value of the function. The index can be multiplied by a `pow()` function or just twice.



A screenshot of the Microsoft Visual Studio 디버그 콘솔 (Debug Console) window. The window has a title bar with the text "Microsoft Visual Studio 디버그 콘솔" and standard window controls (minimize, maximize, close). The main area of the console displays the calculation: `y=3.0*x*x + 7.0*x + 9.0=35.000000`. The text is in a monospaced font, and the result is formatted with a trailing zero. A vertical scrollbar is visible on the right side of the console area.

Sol: Calculation of quadratic functions

```
#include <stdio.h>
int main(void)
{
    double x = 2.0;
    double y;
    y = 3.0*x*x + 7.0*x + 9.0;
    printf("y=3.0*x*x + 7.0*x + 9.0=%f \n", y);
    return 0;
}
```