

# 혼자 공부하는 c언어 4장



미녀와 함께....

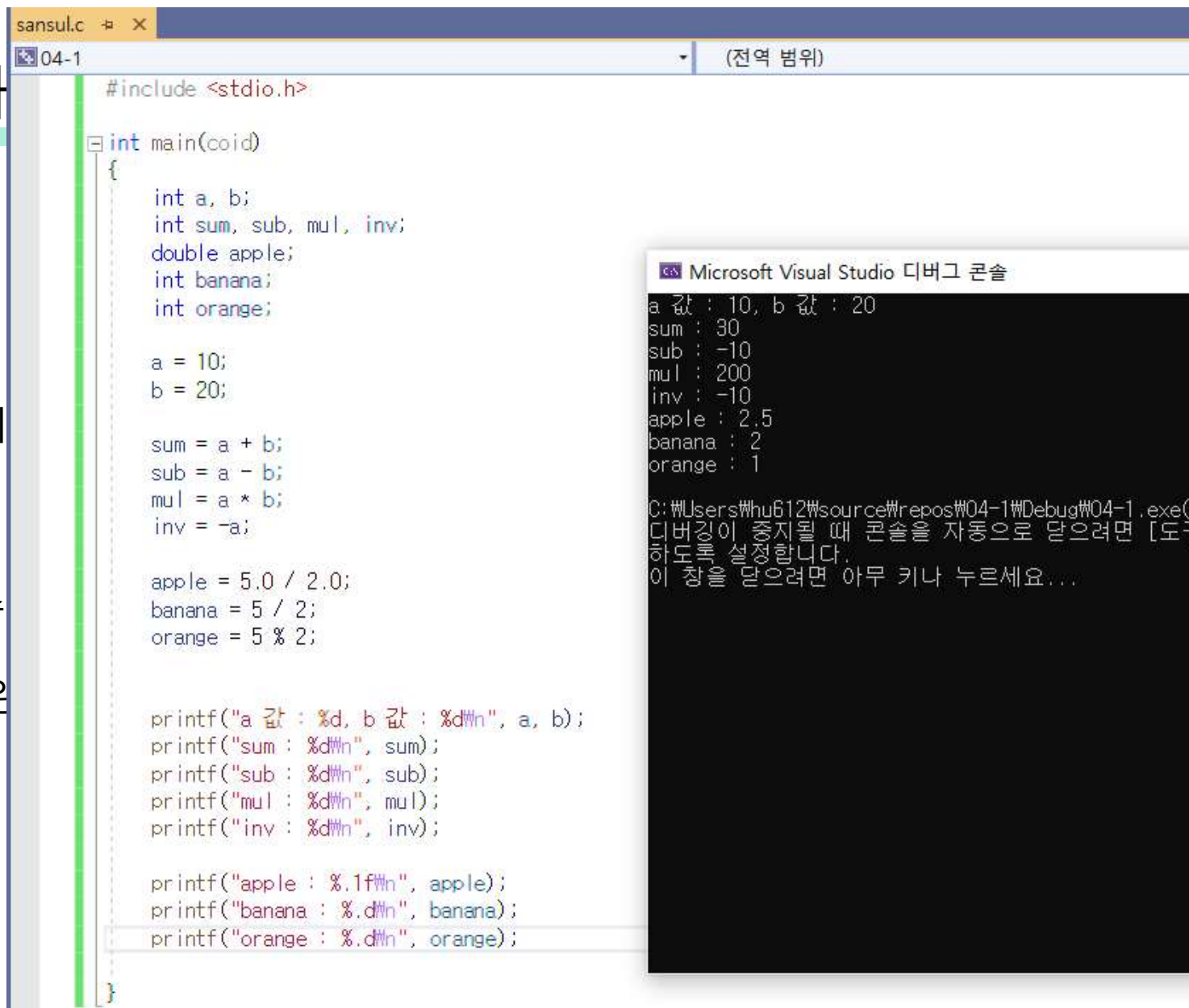
# 1. 산술 연산자 & 대입 연산자

## <산술 연산자 특징>

+, -, \*, /, %가 있다.  
두 개의 피연산자를 사용한다.  
-는 하나만 사용시 피연산자의

## <대입 연산자 특징>

=이 있다.  
오른쪽 수식 결과를 왼쪽 변수  
대입 연산자를 다른 연산자와  
다른 연산을 먼저 수행한 후 온



The image shows a Visual Studio IDE with a C program named 'sansul.c'. The code defines variables for integers and doubles, performs arithmetic operations (sum, subtraction, multiplication, division, and modulo), and prints the results. A debug console window is open, showing the output of the program.

```
#include <stdio.h>

int main(void)
{
    int a, b;
    int sum, sub, mul, inv;
    double apple;
    int banana;
    int orange;

    a = 10;
    b = 20;

    sum = a + b;
    sub = a - b;
    mul = a * b;
    inv = -a;

    apple = 5.0 / 2.0;
    banana = 5 / 2;
    orange = 5 % 2;

    printf("a 값 : %d, b 값 : %d\n", a, b);
    printf("sum : %d\n", sum);
    printf("sub : %d\n", sub);
    printf("mul : %d\n", mul);
    printf("inv : %d\n", inv);

    printf("apple : %.1f\n", apple);
    printf("banana : %d\n", banana);
    printf("orange : %d\n", orange);
}
```

Microsoft Visual Studio 디버그 콘솔

```
a 값 : 10, b 값 : 20
sum : 30
sub : -10
mul : 200
inv : -10
apple : 2.5
banana : 2
orange : 1

C:\Users\hhu612\source\repos\04-1\Debug\04-1.exe(
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

## 2. 증감 연산자

### <증감 연산자 특징>

표시하는 방법은 두가지로 나뉜다. 전위 표기와 후위 표기.

++a, --a(전위 표기)

: 값을 증감 시킨 뒤 연산을 사용하는 방법.

a++, a--(후위 표기)

연산을 사용한 뒤 값을 증감시키는 방법.

단항 연산자로 1증가 또는 1감소 시킨다.

또한 피연산자로 상수 피연산자는 불가능하다.

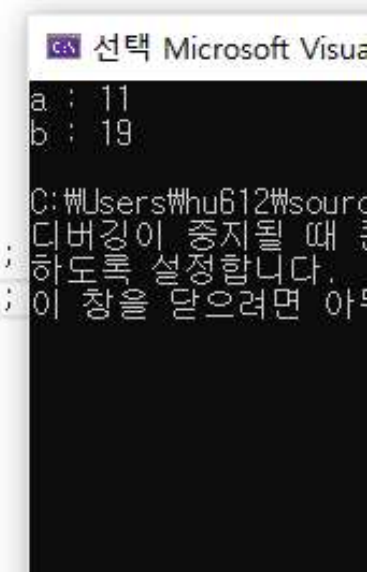
```
#include <stdio.h>

int main(void)
{
    int a = 10, b = 20;

    ++a;
    --b;

    printf("a : %.d\n", a);
    printf("b : %.d\n", b);

    return 0;
}
```



The screenshot shows a C program in a code editor. The program defines two integers, a and b, with initial values 10 and 20. It then increments a by 1 and decrements b by 1. The program uses printf to output the values of a and b. The output window shows a : 11 and b : 19. The code is as follows:

```
#include <stdio.h>

int main(void)
{
    int a = 10, b = 20;

    ++a;
    --b;

    printf("a : %.d\n", a);
    printf("b : %.d\n", b);

    return 0;
}
```

선택 Microsoft Visual  
a : 11  
b : 19  
C:\Users\hhu612\source  
디버깅이 중지될 때  
하도록 설정합니다.  
이 창을 닫으려면 아

### 3. 관계 연산자

관계 연산자는 두가지 종류가 있다.  
대소관계연산자 >, < 와 동등관계연산자 ==, !=  
좌항, 우항 필요하여 피연산자 2개를 사용하고  
결과값은 0(False) 또는 1(True) 중 하나를 낸다.

```
#include <stdio.h>

int main(void)
{
    int a = 10, b = 20, c = 10;
    int res, res1;

    res = (a > b);
    printf("a > b : %d\n", res);
    res1 = (a < b);
    printf("a < b : %d\n", res1);

    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

```
a > b : 0
a < b : 1

C:\Users\whu812\source\repos\04-1\04-1
디버깅이 중지될 때 콘솔을 자동으로
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요
```

## 4. 논리 연산자

논리 연산자는 세가지 종류가 있다.

논리 연산자는 논리곱(&&(AND)), 논리합(::(OR)), 논리부정(!(NOR)) 이다.

좌항, 우항 필요하여 피연산자 2개를 사용한다.

결과값은 0(False) 또는 1(True) 중 하나를 낸다.

- 1) 논리곱(&&(AND)) : 참거짓 여부를 판단하는데 둘 다 참이
- 2) 논리합(::(OR)) : 참거짓 여부를 판단하는데 둘 중 하나만 참
- 3) 논리부정(!(NOR)) : 피연산자 하나를 사용하는데 참과 거짓

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

```
int main(void)
```

```
{  
    int a = 30;  
    int res, res1, res2;
```

```
    res = (a > 10) && (a < 20);  
    printf("(a > 10) && (a < 20) : %d\n", res);
```

```
    res1 = (a < 10) || (a > 20);  
    printf("(a < 10) || (a > 20) : %d\n", res1);
```

```
    res2 = !(a >= 30);  
    printf("(a >= 30) : %d\n", res2);
```

```
    return 0;  
}
```

Microsoft Visual Studio 디버그 콘솔

```
(a > 10) && (a < 20) : 0  
(a < 10) || (a > 20) : 1  
!(a >= 30) : 0
```

## 4. 논리 연산자

```
#include <stdio.h>

int main(void)
{
    int a = 30;
    int res, res1, res2;

    res = (a > 10) && (a < 20);
    printf("(a > 10) && (a < 20) : %d\n", res);

    res1 = (a < 10) || (a > 20);
    printf("(a < 10) || (a > 20) : %d\n", res1);

    res2 = !(a >= 30);
    printf("(a >= 30) : %d\n", res2);

    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

```
(a > 10) && (a < 20) : 0
(a < 10) || (a > 20) : 1
!(a >= 30) : 0
```

1) 논리곱 : 둘 다 > <  
(a > 10) && (a < 20) = 0

2) 논리합 : 둘 중 하나 > <  
(a < 10) || (a > 20) = 1

3) 논리부정  
!(a >= 30) = 0

## 4. 논리 연산자

논리 연산자의 종류 중 논리곱(&&(AND)), 논리합(::(OR)) 에는  
숏 서킷룰이 적용된다.

숏 서킷룰이란, 좌항만으로도 논리곱, 논리합의 결과를 판별하는 기능이다.  
숏 서킷룰 이용시 불필요한 연산을 줄여 실행 속도를 높일 수 있으나,  
예상치 못한 결과를 만들어낼 수 있으니 주의를 바란다.

## 5. 형 변환 연산자

- 피연산자 1개를 원하는 형태로 바꾸는 기능.
- 일시적이라 연산 후 피연산자 형태, 값은 변화 X
- 형태 : (자료형)피연산자

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

```
int main(void)
{
    int a = 50, b = 20;
    int res2;
    double res;

    res = ((double)a) / ((double)b); // 실수
    res2 = a / b;

    printf("res 값 : %.1f\n", res);
    printf("res2 값 : %.d\n", res2);

    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

```
res 값 : 2.5
res2 값 : 2
```

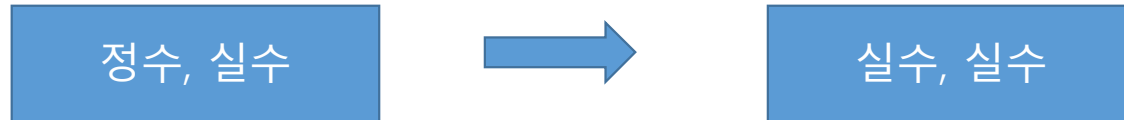
```
C:\Users\whu612\source\repos\04-1\Debug\04-1.exe
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```



## 5. 형 변환 연산자

피연산자가 2개 이상이면 피연산자의 형태가 같아야 한다.  
따라서 컴파일 과정에서 피연산자의 형태가 다르면 자동으로 변환시킨다.  
이를 '**자동 형 변환**(암시적인 형 변환, 묵시적인 형 변환)' 이라고 한다.

기본 규칙 : 크기가 작은 값 -> 크기가 큰 값으로 변환



## 6. sizeof 연산자

Sizeof 연산자는 피연산자를 하나 사용한다.  
크기를 바이트 단위로 계산해서 알려준다.  
표기는 다음과 같다.

Sizeof(피연산자) ; 피연산자 종류에는 변수, 상수, 수식, 자료형 등이 있다.  
데이터 크기 확인 및 메모리 동적 할당 등으로 쓰인다.

```
#include <stdio.h>

int main(void)
{
    int a = 30;
    double b = 3.4;

    printf("int 형 변수의 크기 : %d\n", sizeof(a));
    printf("double 형 변수의 크기 : %d\n", sizeof(b));
    printf("정수 형 변수의 크기 : %d\n", sizeof(10));
    printf("수식 형 변수의 크기 : %d\n", sizeof(1.5+4.3));
    printf("char 자료 형 변수의 크기 : %d\n", sizeof(char));

    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

```
int 형 변수의 크기 : 4
double 형 변수의 크기 : 8
정수 형 변수의 크기 : 4
수식 형 변수의 크기 : 8
char 자료 형 변수의 크기 : 1
```

## 6. sizeof 연산자

구분	자료형	크기	범위
기본형	void	—	—
문자형	(signed) char	1 byte	-128 ~ 127
	unsigned char	1 byte	0 ~ 255
	wchar_t	2 byte	0 ~ 65,535
정수형	bool	1 byte	0 ~ 1
	(signed) short (int)	2 byte	-32,768 ~ 32,767
	unsigned short (int)	4 byte	0 ~ 65,535
	(signed) int	4 byte	-2,147,483,648 ~ 2,147,483,647
	unsigned int	4 byte	0 ~ 4,294,967,295
	(signed) long (int)	4 byte	-2,147,483,648 ~ 2,147,483,647
	unsigned long (int)	4 byte	0 ~ 4,294,967,295
	__int8	1 byte	-128 ~ 127
	__int16	2 byte	-32,768 ~ 32,767
	__int32	4 byte	-2,147,483,648 ~ 2,147,483,647
	__int64	8 byte	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
실수형	float	4 byte	3.4E-38(-3.4*10 <sup>38</sup> ) ~ 3.4E+38(3.4*10 <sup>38</sup> ) (7digits)
	(long) double	8 byte	1.79E-308(-1.79*10 <sup>308</sup> ) ~ 1.79E+308(1.79*10 <sup>308</sup> ) (15digits)

<https://myblog.opendocs.co.kr/archives/1230>

## 7. 복합대입 연산자

대입 연산자(=)와 증감 연산자(++ , --) 제외  
다른 연산자는 연산하고 나서 피연산자의 값을 바꾸지 않는다.  
만약, 연산 결과를 피연산자에 저장할 필요가 있다면  
추가로 대입 연산을 수행해야 한다.  
이때는 '복합대입 연산자'를 사용하면 간단하게 할 수 있다.

복합대입 연산자는 연산 결과를 다시 피연산자에 저장한다.  
종류로는 +=, -=, \*=, /=, %=가 있다.

복합대입 연산자는 대입 연산자의 특징을 그대로 가져온다.

- 1) 왼쪽 피연산자는 반드시 변수가 와야 한다.
- 2) 오른쪽 항의 계산이 모두 끝난 다음에 복합대입 연산자는 가장 마지막에 계산한다.

## 7. 복합대입 연산자

```
#include <stdio.h>

int main(void)
{
    int a = 10, b = 20;
    int res = 2;

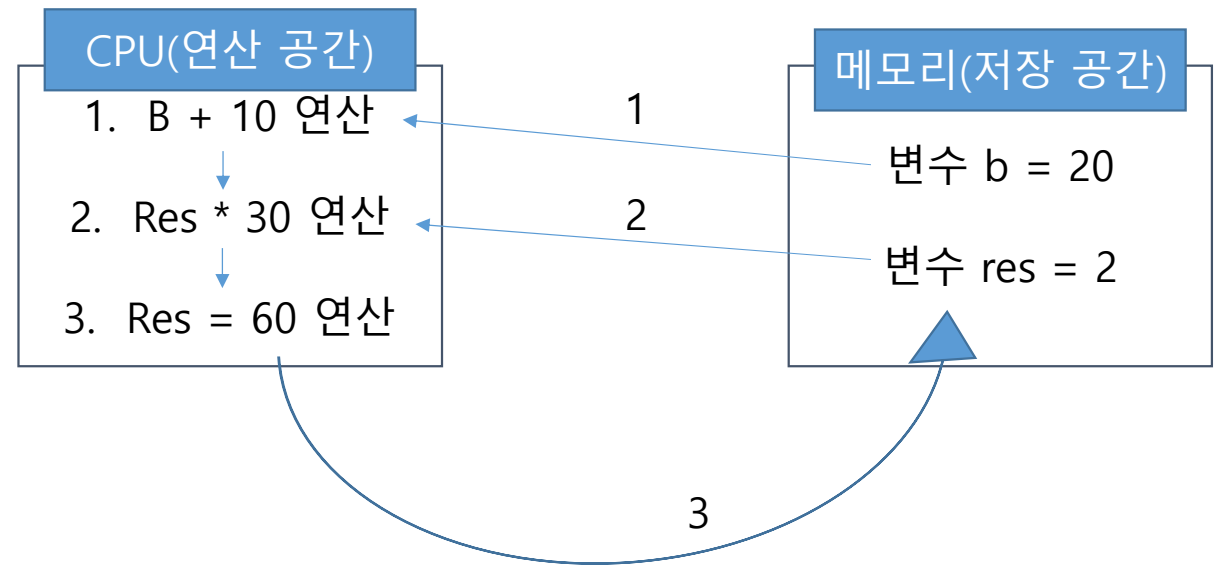
    a += 20;
    res *= b + 10;

    printf("a=%d, b = %d\n", a, b);
    printf("res = %d", res);

    return 0;
}
```

Microsoft Visual Studio

a=30, b = 20  
res = 60



## 8. 콤마 연산자

콤마 연산자는 한 번에 여러 개의 수식을 차례로 나열해야 할 때 사용한다.  
왼쪽부터 오른쪽으로 차례로 연산을 수행  
-> 제일 오른쪽 피연산자가 최종 결과값

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

```
int main(void)
{
    int a = 10, b = 20;
    int res;

    res = (++a, ++b);

    printf("a:%d, b:%d\n", a, b);
    printf("res:%d\n", res);
    return 0;
}
```

Microsoft Visual Stuc

```
a:11, b:21
res:21
```

## 9. 조건 연산자

조건 연산자는 유일한 삼항 연산자로 ?와 :를 쓴다.

표기 : (첫번째 피연산자) ? (두번째 피연산자) : (세번째 피연산자)

첫번째 피연산자 참 -> 두번째 피연산자  
거짓 -> 세번째 피연산자

```
#include <stdio.h>

int main(coid)
{
    int a = 10, b = 20;
    int res;

    res = (a > b) ? a : b;
    printf("큰 값 : %d\n", res);
    return 0;
}
```

Microsoft Visual Studio

큰 값 : 20

```
#include <stdio.h>

int main(coid)
{
    int a = 10, b = 20;
    int res;

    (a > b) ? (res = a) : (res=b);
    printf("큰 값 : %d\n", res);
    return 0;
}
```

Microsoft Visual Stud

큰 값 : 20

## 10. 비트 연산자

데이터를 비트 단위로 연산하는 비트 연산자는 두 종류가 있다. (정수에만 사용 가능)

- 1) 비트 논리 연산자  $\&$ ,  $;$ ,  $\wedge$
- 2) 비트 이동 연산자  $\gg$ ,  $\ll$

### 비트별 논리곱 연산자

```
#include <stdio.h>

int main(void)
{
    int a = 10;
    int b = 12;

    printf("a & b : %d\n", a & b);
    return 0;
}
```

Microsoft Visual Studio

a & b : 8

	00000000	00000000	00000000	00001010	(a = 10)
+	00000000	00000000	00000000	00001100	(b = 12)
	00000000	00000000	00000000	00001000	(a & b = 8)



## 10. 비트 연산자

### 비트별 배타적 논리합 연산자

```
#include <stdio.h>

int main(void)
{
    int a = 10;
    int b = 12;

    printf("a ^ b : %d\n", a ^ b);
    return 0;
}
```

Microsoft Visual Studio  
a ^ b : 6

+  
00000000 00000000 00000000 00001010 (a = 10)  
00000000 00000000 00000000 00001100 (b = 12)  
-----  
00000000 00000000 00000000 00000110 (a ^ b = 6)

### 비트별 논리합 연산자

```
#include <stdio.h>

int main(void)
{
    int a = 10;
    int b = 12;

    printf("a | b : %d\n", a | b);
    return 0;
}
```

Microsoft Visual Studio  
a | b : 14

+  
00000000 00000000 00000000 00001010 (a = 10)  
00000000 00000000 00000000 00001100 (b = 12)  
-----  
00000000 00000000 00000000 00001110 (a | b = 14)

## 10. 비트 연산자

### 비트별 부정 연산자

```
#include <stdio.h>

int main(void)
{
    int a = 10;
    int b = 12;

    printf("~a : %d\n", ~a);
    return 0;
}
```

Microsoft Visual Studio  
~a : -11

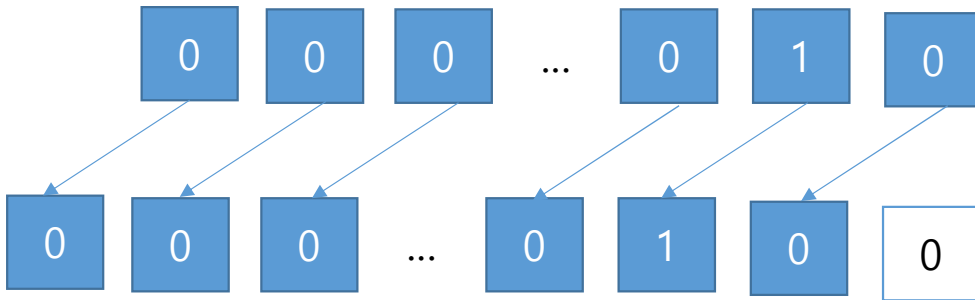
~      00000000 00000000 00000000 00001010 (a = 10)  
11111111 11111111 11111111 11110101 (~a = -11)

## 10. 비트 연산자

### 비트 이동 연산자

<< 비트들은 왼쪽으로 이동하고 >> 비트들은 오른쪽으로 이동한다.  
>>> 지정 수 만큼 비트를 오른쪽으로 이동시키며 새로운 비트는 0이 된다.

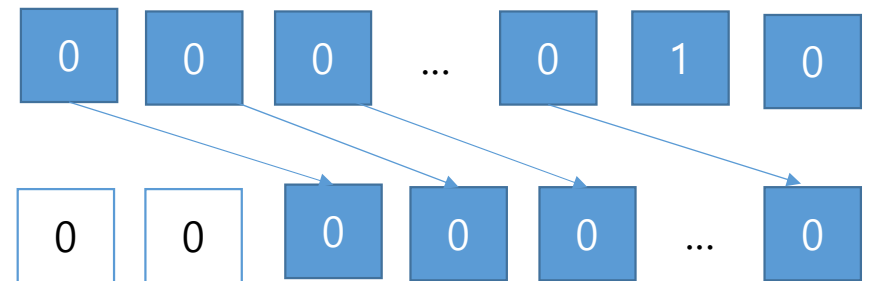
A = 10



A << 1

비트가 왼쪽으로 한 비트씩 이동 => 2 곱해짐

A = 10



A >> 2

비트가 오른쪽으로 한 비트씩 이동 => 2 나눠짐

## 11. 연산자 우선 순위와 연산 방향

두 개 이상의 연산자가 함께 쓰이면 연산자의 우선순위에 따라 연산된다.

단항 연산자 > 이항 연산자 > 삼항 연산자

산술 연산자 > 비트 이동 연산자 > 관계 연산자 > 논리 연산자

순으로 연산된다.

애매할 땐! 괄호를 쓰자 !

```
#include <stdio.h>

int main(void)
{
    int a = 10, b = 5;
    int res, res1, res2, res3;

    res = a / b * 2;
    printf("res 값 : %d\n", res);

    res1 = ++a * 3;
    printf("res1 값 : %d\n", res1);

    res2 = a > b && a != 5;
    printf("res = %d\n", res2);

    res3 = a % 3 == 0;
    printf("Res = %d\n", res3);

    return 0;
}
```

Microsoft Visual Studio

```
res 값 : 4
res1 값 : 33
res = 1
Res = 0
```

```
C:\Users\hu812>
디버깅이 중지됨
하도록 설정함
이 창을 닫으려
```

문제1) 결과가 모두 1이 되게 출력하세요.

```
#include <stdio.h>

int main()
{
    int num1 = 27;

    printf("%d\n", num1 >= 10);
    printf("%d\n", num1 != 5);

    printf("%d\n", num1 >= 27);
    printf("%d\n", num1 == 27);

    printf("%d\n", num1 <= 30);
    printf("%d\n", num1 <= 27);

    return 0;
}
```

끝

