

연산자 이해하기

제주대학교 컴퓨터공학과

변영철 교수

(ycb@jejunu.ac.kr)

이 장을 공부하면

- 산술, 논리, 관계 연산자를 이해할 수 있다.
- C 언어에서 사용하는 기타 다른 연산자도 이해할 수 있다.
- C 프로그램을 작성 시 연산자를 적절히 활용할 수 있다.

연산자 둘러보기

```
#include <stdio.h>

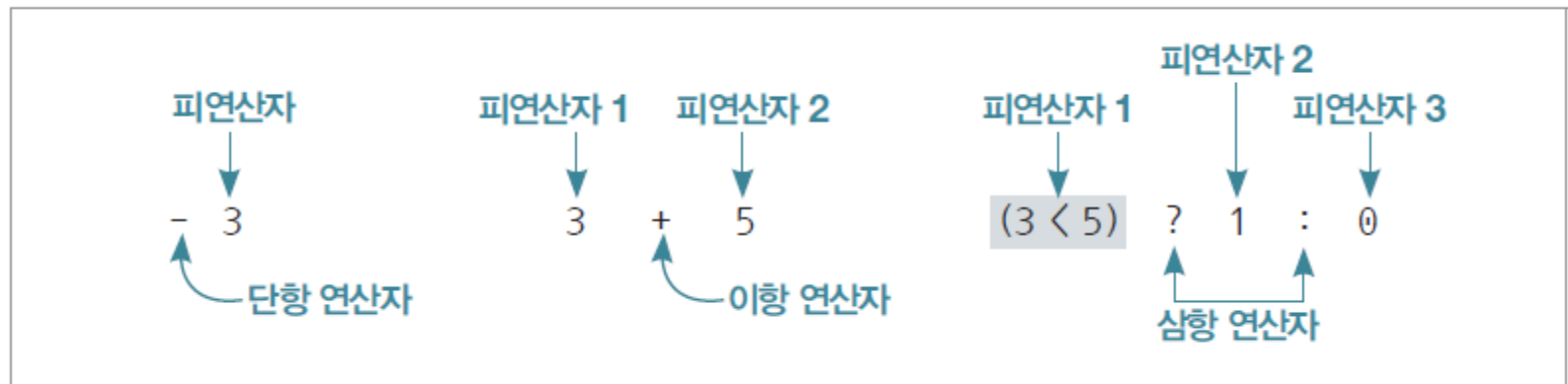
void main()
{
    int a = -3;
    int b = 3 + 5;
    int c = (3 < 5) ? 1 : 0;

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

    getchar();
}
```

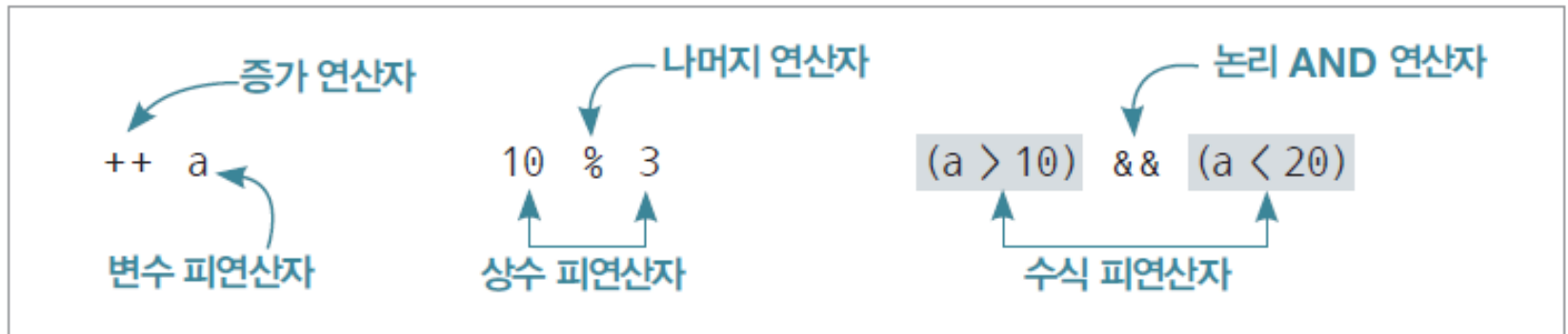
연산자 둘러보기

- 피연산자(항) 수에 따라
 - 단항 연산자, 2항 연산자, 3항 연산자



연산자 둘러보기

- 연산자 종류에 따라
 - 산술 연산자, 관계 연산자, 논리 연산자



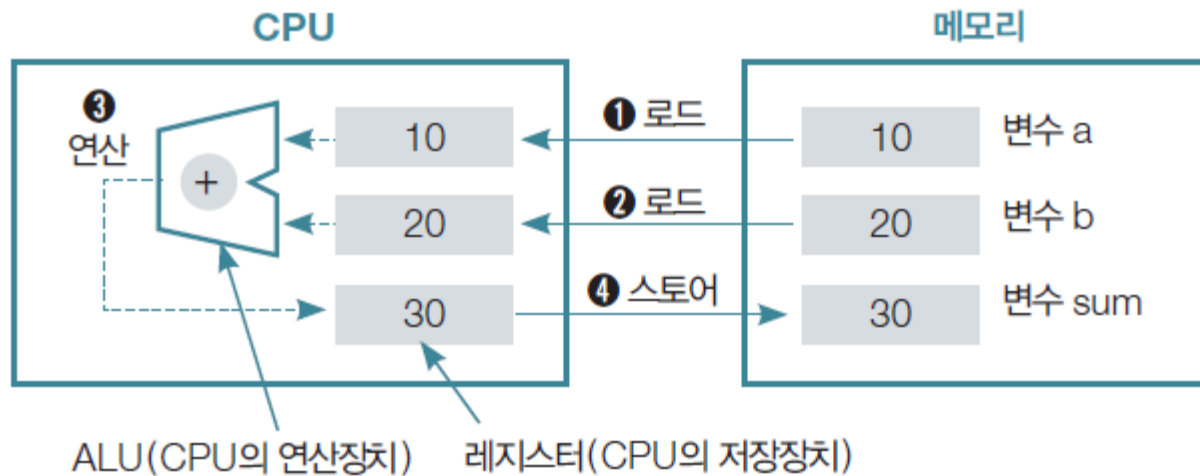
산술 연산자

- CPU = CU + **PU**(ALU = 산술 논리 유닛)
- 산술 연산자 → +, -, *, /, %, 증감
- 논리 연산자 → ||, &&, !
- 대입 연산자 (=)
 - 산술 연산 결과를 대입

산술 연산자

ALU와 산술 연산 처리과정

수식 'sum = a + b'의 연산과정



산술 연산자

- 나눗셈 연산자와 나머지 연산자

예제 4-2 몫과 나머지를 구하는 연산

```
1. #include <stdio.h>
2.
3. int main(void)
4. {
5.     double apple;
6.     int banana;
7.     int orange;
8.
9.     apple = 5.0 / 2.0;           // 실수와 실수의 나눗셈 연산
```


산술 연산자

```
10.    banana = 5 / 2;           // 정수와 정수의 나눗셈 연산
11.    orange = 5 % 2;           // 정수와 정수의 나머지 연산
12.
13.    printf("apple : %.1lf\n", apple);
14.    printf("banana : %d\n", banana);
15.    printf("orange : %d\n", orange);
16.
17.    return 0;
18. }
```

실행
결과

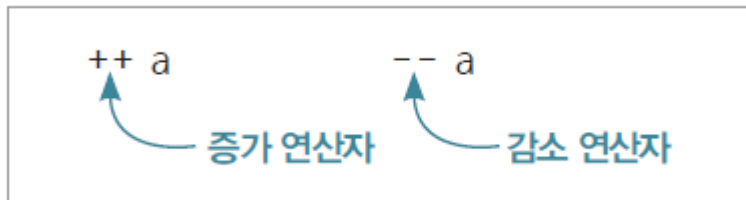
```
apple : 2.5
banana : 2
orange : 1
```

산술 연산자

- 증감 연산자 (++ , --)

++a; $\Rightarrow a = a + 1;$

--a; $\Rightarrow a = a - 1;$



산술 연산자

예제 4-3 증감 연산자의 연산

```
1. #include <stdio.h>
2.
3. int main(void)
4. {
5.     int a = 10, b = 10;
6.
7.     ++a;           // 변수의 값을 1만큼 증가
8.     --b;           // 변수의 값을 1만큼 감소
9.
10.    printf("a : %d\n", a);
11.    printf("b : %d\n", b);
12.
13.    return 0;
14. }
```

실행
결과
a : 11
b : 9

산술 연산자

- 증감 연산자와 우선순위

예제 4-4 전위형과 후위형을 사용한 증감 연산

```
1. #include <stdio.h>
2.
3. int main(void)
4. {
5.     int a = 5, b = 5;
6.     int pre, post;
7.
8.     pre = (++a) * 3;      // 전위형 증감 연산자
9.     post = (b++) * 3;    // 후위형 증감 연산자
10.
11.     printf("초깃값 a = %d, b = %d\n", a, b);
12.     printf("전위형: (++a) * 3 = %d, 후위형: (b++) * 3 = %d\n", pre, post);
13.
14.     return 0;
15. }
```

실행 결과
초깃값 a = 6, b = 6
전위형: (++a) * 3 = 18, 후위형: (b++) * 3 = 15

관계 연산자

- 작으냐 (<), 크냐 (>)
- 같은가 (==), 같지 않은가 (!=)
- 피연산자 2개 사용하며, 연산의 결과값은 1 또는 0
- 컴파일러는 0은 거짓으로, 0이 아닌 값은 참(true)으로 판단
- 관계식을 실행 조건 검사에 사용 가능

관계 연산자

- 관계 연산자

표 4-2 관계 연산자

연산식	결과값
$a > b$	a가 b보다 크면 1(참, true) 그렇지 않으면 0(거짓, false)
$a \geq b$	a가 b보다 크거나 같으면 1(참) 그렇지 않으면 0(거짓)
$a < b$	a가 b보다 작으면 1(참) 그렇지 않으면 0(거짓)
$a \leq b$	a가 b보다 작거나 같으면 1(참) 그렇지 않으면 0(거짓)
$a == b$	a와 b가 같으면 1(참) 다르면 0(거짓)
$a != b$	a와 b가 다르면 1(참) 같으면 0(거짓)

관계 연산자

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

```
void main()
```

```
{
```

```
    printf("%d\\n", 10 > 20);
```

```
    printf("%d\\n", 10 < 20);
```

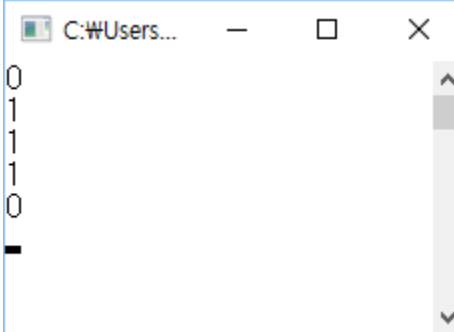
```
    printf("%d\\n", 10 <= 10);
```

```
    printf("%d\\n", 10 == 10);
```

```
    printf("%d\\n", 10 != 10);
```

```
    getchar();
```

```
}
```



```
C:\Users...  
0  
1  
1  
1  
0
```

논리 연산자



표 4-3 논리 연산자

연산식	논리관계	결과값
a && b	논리곱(AND)	a와 b가 모두 참이면 1 그렇지 않으면 0
a b	논리합(OR)	a와 b 중 하나라도 참이면 1 그렇지 않으면 0
!a	논리부정(NOT)	a가 거짓이면 1 참이면 0

논리 연산자

```
#include <stdio.h>

void main()
{
    int a = 7;

    int result = (a > 1) && (a < 3);
    printf("%d\n", result);

    result = a < 3 || a > 5;
    printf("%d\n", result);

    getchar();
}
```

그 외의 멋진 연산자

표 4-4 기타 연산자

연산자	연산식 예	결과값
형변환 연산자	res = (int) 10.7;	res값은 10
sizeof 연산자	res = sizeof(double);	res값은 8
복합대입 연산자	a += 10;	a의 값을 10 증가
콤마 연산자	res = (a, b);	res에 b값 저장
조건 연산자	res = (a > b) ? a : b;	a가 b보다 크면 res값은 a 작거나 같으면 res값은 b
비트 연산자	a & b; a ^ b; a ! b; ~a; a << b; a >> b;	a와 b의 비트 상태에 따라 결과값이 다름

그 외의 멋진 연산자

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

```
void main()
```

```
{
```

```
    int a = 12;
```

```
    printf("%d\\n", sizeof(int));
```

```
    printf("%d\\n", sizeof(a));
```

```
    printf("%d\\n", sizeof(3));
```

```
    getchar();
```

```
}
```

그 외의 멋진 연산자

- 복합 대입 연산자 (산술 연산자의 일종)

표 4-5 산술 복합대입 연산자

복합대입 연산식	동일한 연산식
$a += b$	$a = a + b$
$a -= b$	$a = a - b$
$a *= b$	$a = a * b$
$a /= b$	$a = a / b$
$a \% = b$	$a = a \% b$

그 외의 멋진 연산자

예제 4-10 복합대입 연산자

```
1. #include <stdio.h>
2.
3. int main(void)
4. {
5.     int a = 10, b = 20;
6.     int res = 2;
7.
8.     a += 20;
9.     res *= b + 10;
10.
11.     printf("a = %d, b = %d\n", a, b);
12.     printf("res = %d\n", res);
```

실행
결과 a = 30, b = 20
res = 60

그 외의 멋진 연산자

- 콤마 연산자
 - 왼쪽부터 오른쪽으로 차례로 연산 수행
 - 가장 오른쪽 피연산자가 최종 결과값

그 외의 멋진 연산자

```
#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);
```

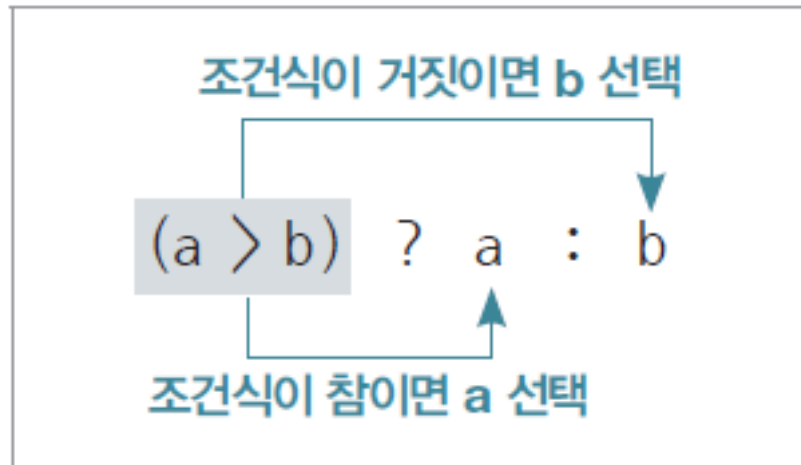
```
    return 0;
```

```
}
```

실행
결과 a:11, b:21
res:21

그 외의 멋진 연산자

- 조건 연산자



그 외의 멋진 연산자

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

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

    return 0;
}
```

실행
결과
큰 값 : 20

그 외의 멋진 연산자

- 비트 연산자
 - 비트 논리 연산자 - 비트 단위로 논리연산 수행
 - 비트 이동 연산자 - 비트들 좌우로 이동

그 외의 멋진 연산자

표 4-6 비트 연산자 종류

구분	연산자	연산 기능
비트 논리 연산자	&	비트 단위 논리곱(and) 연산자
	^	비트 단위 배타적 논리합(xor) 연산자
		비트 단위 논리합(or) 연산자
	~	비트 단위 부정(not) 연산자
비트 이동 연산자	<<	왼쪽 비트 이동 연산자
	>>	오른쪽 비트 이동 연산자

그 외의 멋진 연산자

- 비트 논리 연산자

```
int a = 10; // 비트열 00000000 00000000 00000000 00001010
int b = 12; // 비트열 00000000 00000000 00000000 00001100
```

& 연산은 두 비트가 모두 1인 경우만 1로 계산합니다.

```
00000000 00000000 00000000 00001010 (a: 10)
& 00000000 00000000 00000000 00001100 (b: 12)
-----
00000000 00000000 00000000 00001000 (a & b: 8)
```

! 연산은 두 비트 중에서 하나라도 참이면 1로 계산합니다.

```
00000000 00000000 00000000 00001010 (a: 10)
! 00000000 00000000 00000000 00001100 (b: 12)
-----
00000000 00000000 00000000 00001110 (a ! b: 14)
```

^ 연산은 두 비트가 서로 다른 경우만 1로 계산합니다.

```
00000000 00000000 00000000 00001010 (a: 10)
^ 00000000 00000000 00000000 00001100 (b: 12)
-----
00000000 00000000 00000000 00001110 (a ^ b: 6)
```

~ 연산은 1은 0으로 바꾸고 0은 1로 바꿉니다.

```
~ 00000000 00000000 00000000 00001010 (a: 10)
-----
11111111 11111111 11111111 11110101 (~a: -11)
```

그 외의 멋진 연산자

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

    printf("a & b : %d\n", a & b);
    printf("a ^ b : %d\n", a ^ b);
    printf("a | b : %d\n", a | b);
    printf("~a : %d\n", ~a);
    printf("a << 1 : %d\n", a << 1);
    printf("a >> 2 : %d\n", a >> 2);

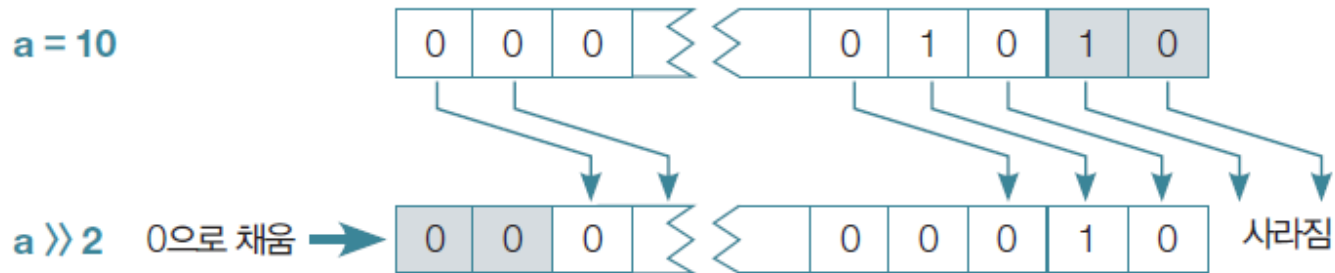
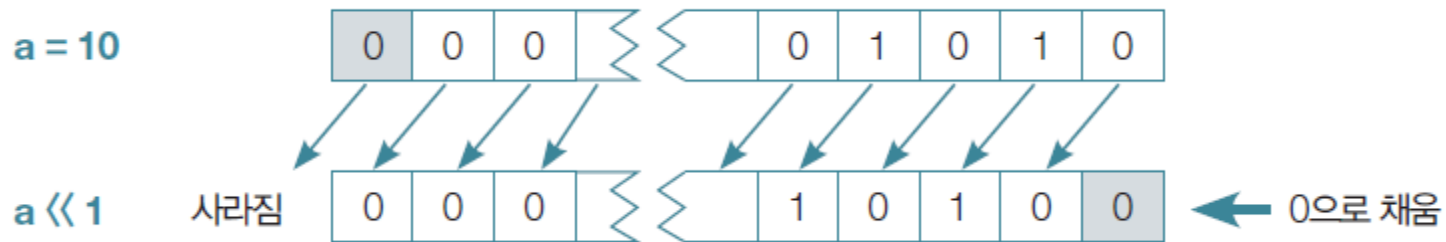
    return 0;
}
```

실행
결과

```
a & b : 8
a ^ b : 6
a | b : 14
~a : -11
a << 1 : 20
a >> 2 : 2
```

그 외의 멋진 연산자

- 비트 이동 연산자 (<<, >>)



그 외의 멋진 연산자

- 복합 대입 연산자

$a = a \ll 2;$  $a \ll 2;$

- 복합 대입 연산자의 종류

$\ll =, \gg =, \& =, \wedge =, |=$

연산자 우선순위

$a = 3.14 * 10 - 5;$
 $a = 3.14 * (10 - 5)$

실행
결과
res = 4
res = 33
res = 1
res = 0

연산자 우선순위

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

```
int main(void)
```

```
{
```

```
    int a = 10, b = 5;
```

```
    int res;
```

```
    res = a / b * 2;           // 우선순위가 같으므로 왼쪽부터 차례로 연산
```

```
    res = ++a * 3;            // a의 값을 1증가시키고 3을 곱한다.
```

```
    res = a > b && a != 5;     // a > b의 결과와 a != 5의 결과를 && 연산
```

```
    res = a % 3 == 0;         // a % 3의 값이 0과 같은지 확인
```

```
    return 0;
```

```
}
```

연산자 우선순위

```
#include <stdio.h>

void main()
{
    int a = 6;
    int result = a % 3 == 0;
    printf("%d\n", result);

    getchar();
}
```

연산자 정리



종류	연산자(괄호의 숫자는 우선순위)
1차 연산자	() [] . ->
단항 연산자	- ++ -- ~ ! * & sizeof (type)
산술 연산자	*(3) /(3) %(3) +(4) -(4)
비트 이동 연산자	<< >>
관계 연산자	< <= > >=
동등 연산자	= = !=
비트 논리 연산자	&(8) ^(9) !(10)
논리 연산자	&&(11) (12)
조건 연산자	? :
대입 연산자	= += -= *= /= %= &= ^= = <<= >>=
кома 연산자	,