WEEK 04

연산자와 연산식

학습목표

I. 연산자의 기본 개념 이해
II. 연산자의 사용 방법과 연산식의 결과 이해
III. 연산자 우선순위 이해

학습목차

Ⅰ. 연산식과 다양한 연산자

1. 연산자 개요
2. 산술연산자
3. 대입연산자
4. 증감연산자

1. 연산자 개요

◆ 표현식과 평가

❖ 연산자(operator)

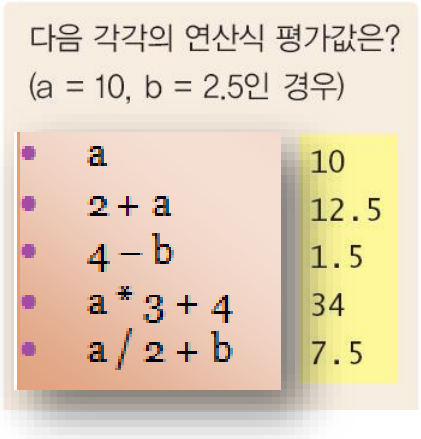
- 산술연산자 +, -, * 기호와 같이 이미 정의된 연산을 수행하는 문자

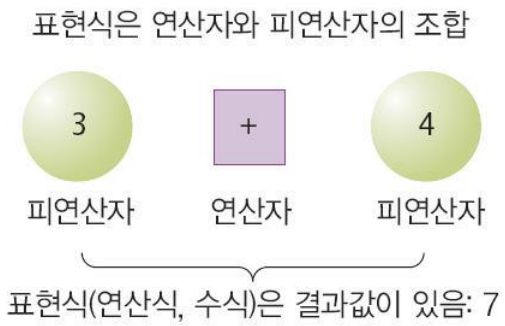
❖ 피연산자(operand)

- 연산(operation)에 참여하는 변수나 상수

❖ 표현식(expression)

- 표현식은 식을 평가(evaluation)하여 항상 하나의 결과값 생성





[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 연산자 개요

◆ 연산자의 분류

❖ 단항, 이항, 삼항 연산자 등

단항연산자

연산자 피연산자 (전위: prefix)

++a: 전위 증가연산자

sizeof (int): sizeof 연산자

(int) 3.46: 자료형 변환연산자

-3: 부호연산자

피연산자 연산자 (후위: postfix)

b--: 후위 감소연산자

a++: 후위 증가연산자

이항연산자

피연산자 1 연산자 피연산자 2

a % 3

3 && 4

5 >= 7

삼항연산자

피연산자 1 ? 피연산자 2 : 피연산자 3

3 ? 4 : 5

(5 >= 7) ? (3+5) : (10/2)

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 산술연산자

◆ 산술연산자와 부호연산자

❖ 산술연산자

다음 각각의 연산식 평가 값은?
(a = 10, b = 2인 경우)

3+a*b	23
2+a-b	10
4-a/b	-1
3.0+a/4	5.0
a%4-b	0

산술연산 5가지

- op1 + op2
- op1 - op2
- op1 * op2
- op1 / op2
- op1 % op2

a = b * n + r (a, b는 정수)이라면

a / b	n
a % b	r
10 / 3	3
10 % 3	1
17 / 5	3
17 % 5	2

```
    3 ← 17 / 5
5 ) 17
    15
     2 ← 17 % 5
```

나머지 연산자 %의 피연산자는 반드시 정수여야 한다.
그러므로 다음은 잘못된 산술연산식이다.

- 3.0 % 4
- 5 % 2.0
- 5.0 % 3.0

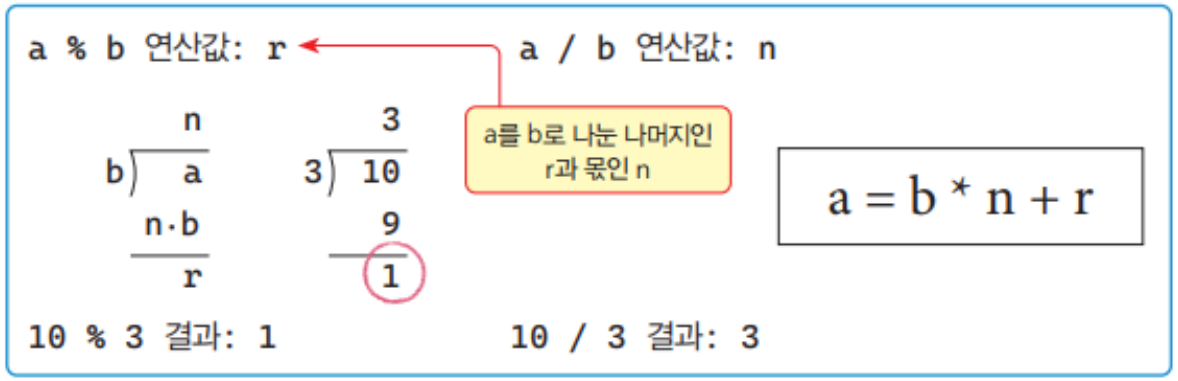
❖ 부호연산자

- 연산식: +3, -4.5, -a
- 연산자 +, -는 피연산자의 부호를 나타내는 연산자

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 산술연산자

◆ 나머지 연산식 a % b



실습예제 5-1 Prj01 01arithop.c 곱하기와 나누기, 나머지 연산자 활용 난이도: ★

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05     int amount = 4000 * 3 + 10000;
06
07     printf(" 총금액 %d 원\n", amount);
08     printf("오천원권 %d 개\n", amount / 5000);
09     printf(" 천원권 %d 개\n", (amount % 5000) / 1000);
10
11     return 0;
12  }
```

정수 / 정수는 정수이므로 결과는 4

결과

```
 총금액 22000 원
오천원권 4 개
 천원권 2 개
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

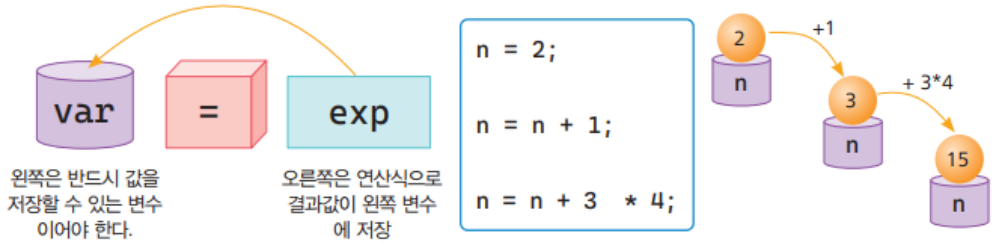
3. 대입연산자

❖ =

- 연산식의 결과값을 변수에 저장하는 연산자
- 할당 또는 치환연산자라고도 부름
- 연산자 오른쪽에 위치한 연산식 exp을 계산하여 그 결과를 왼쪽 변수 var에 저장

❖ l-value와 r-value

- 대입연산자의 왼쪽 부분에는 반드시 하나의 변수만이 올 수 있으며, 이 하나의 변수를 l-value라 하며
- 오른쪽에 위치하는 연산식의 값을 r-value라 함



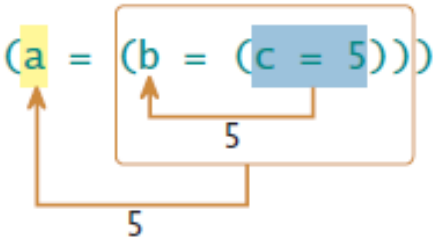
[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 대입연산자

◆ 대입연산자의 이용

❖ 중첩된 대입문

```
int a, b, c;
a = b = c = 5; //(a = (b = (c = 5)))
```



실습예제 5-2 Prj02 02assignop.c 대입 연산자 활용 난이도: ★

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
    int main(void)
    {
       int cred2, cred3;

       cred2 = cred3 = 0;

       printf("2학점과 3학점의 수강 과목 수를 각각 입력 >> ");
       scanf("%d %d", &cred2, &cred3);

       cred2 = 2 * cred2;
       printf("2학점 과목 총 학점: %d\n", cred2);
       printf("3학점 과목 총 학점: %d\n", cred3 = 3 * cred3);
       printf("총 학점: %d\n", cred2 + cred3);

       return 0;
    }
```

대입 연산식의 연산값은 대입된 저장 값

결과

```
2학점과 3학점의 수강 과목 수를 각각 입력 >> 2 4
2학점 과목 총 학점: 4
3학점 과목 총 학점: 12
총 학점: 16
```

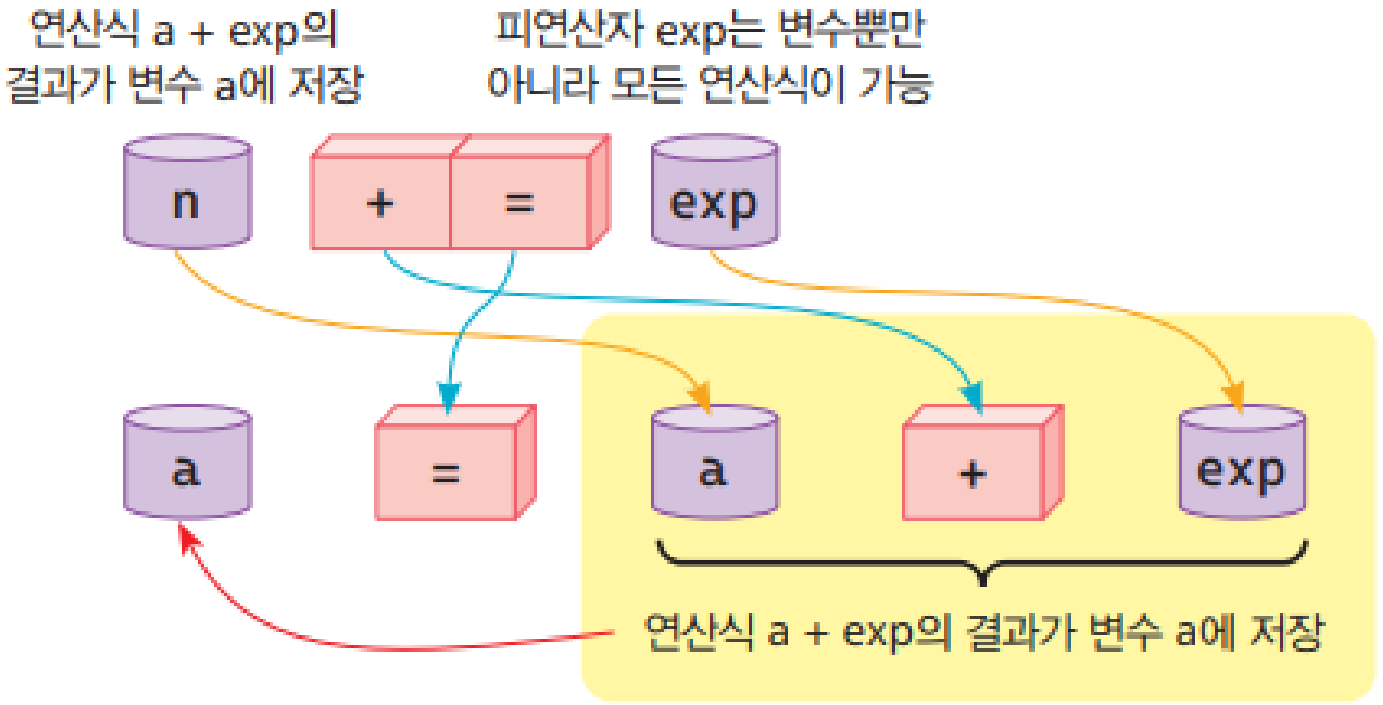
[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 대입연산자

◆ 축약 대입연산자

❖ +=, -=, *=, /=, %=

- 산술연산자와 대입연산자를 이어 붙인 연산자



[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 대입연산자

◆ 축약 대입연산자 이용

❖ +=, -=, *=, /=, %=

산술연산을 간략히 줄인 축약 대입연산자

op1 += op2	op1 = op1 + op2
op1 -= op2	op1 = op1 - op2
op1 *= op2	op1 = op1 * op2
op1 /= op2	op1 = op1 / op2
op1 %= op2	op1 = op1 % op2

a=10, b=2인 경우, 다음 각각의 연산 결과는?

```
a += b + 2; //a = a + (b + 2)   14
a -= b + 2;                      6
a *= b + 2;                     40
a /= b + 2;                      2
a %= b + 2;                      2
```

축약 대입연산자의 왼쪽 피연산자는 반드시 변수여야 하므로 다음은 잘못된 대입연산식

```
++a += b;
a+1 -= b;
a =* b;   //=*가 아니라 *=
a =/a;    //=/가 아니라 /=
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 대입연산자

◆ 축약 대입연산자 활용

실습예제 5-3 | Prj03 | 03compoundassign.c | 복합 대입연산자 활용 | 난이도: ★

```
01  #define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의
02  #include <stdio.h>
03
04  int main(void)
05  {
06     int x = 0, y = 0;
07
08     printf("두 정수를 입력 >> ", &x, &y);
09     scanf("%d%d", &x, &y);
10
11     printf("%d\n", x += y);
12     printf("%d %d\n", x, y);
13     printf("%d\n", x -= y);
14     printf("%d %d\n", x, y);
15
16     return 0;
17  }
```

11줄의 출력 값과 같은 x의 값

결과

```
두 정수를 입력 >> 10 20
30
30 20
10
10 20
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

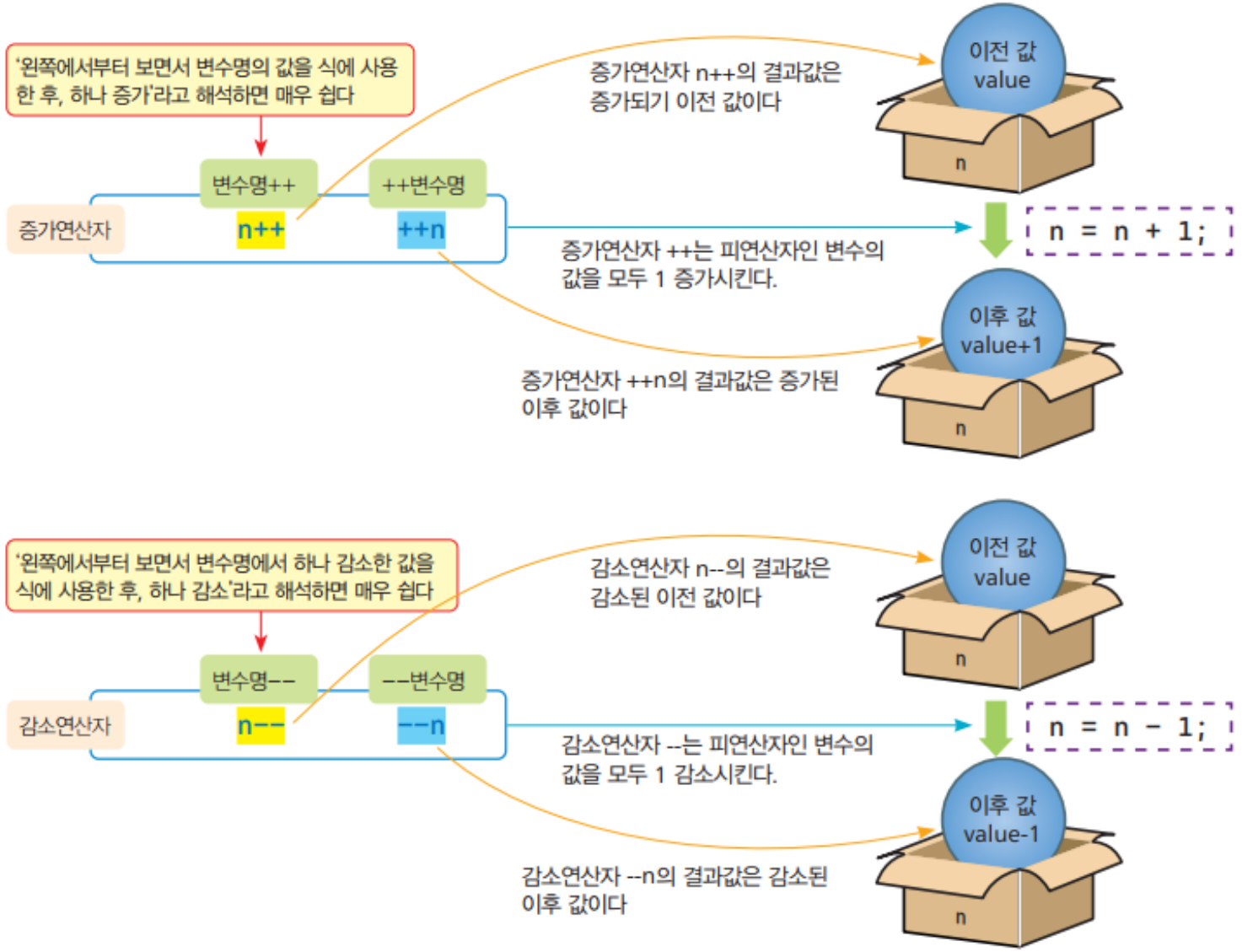
4. 증감연산자

❖ 연산 수행

- 증가연산자 ++는 변수값을 1 증가시키고,
- 감소연산자 --는 1 감소시키는 기능을 수행

❖ 연산 결과 값

- 연산자 n++이면(후위) 1 증가되기전 값이 연산 결과값
- 반대로 ++n과 같이 전위이면 1 증가된 값이 연산 결과값



[출처] 강환수 외, Perfect C 3판, 인피니티북스

4. 증감연산자

◆ 주의점

- 증감연산자는 ++, --는 연산자 기호 중간에 공백이 들어갈 수 없으며
- 다른 연산자보다 그 평가를 먼저 수행
- 증감연산자는 변수만을 피연산자로 사용할 수 있으며
- 상수나 일반 수식을 피연산자로 사용 불가능

```
int n = 10;

                        출력
printf("%d\n", n++);    10
printf("%d\n", n);      11
```

```
int n = 10;

                        출력
printf("%d\n", ++n);    11
printf("%d\n", n);      11
```

```
int n = 10;

                        출력
printf("%d\n", n--);    10
printf("%d\n", n);      9
```

```
int n = 10;

                        출력
printf("%d\n", --n);    9
printf("%d\n", n);      9
```

잘못된 사용 예

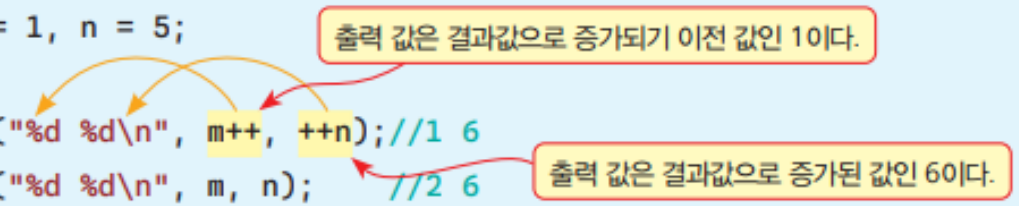
```
int a = 10;
++300;                  // 상수에는 증감연산자를 사용할 수 없다.
(a+1)--;                // 일반 수식에는 증감연산자를 사용할 수 없다.
a = ++a * a--;          // 하나의 연산식에 동일한 변수의 증감연산자는사용하지 말자.
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

4. 증감연산자

실습예제 5-4 | Prj04 | 04incdecop.c | 증가연산자 ++와 감소연산자 -- | 난이도: ★

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      int m = 1, n = 5;
06
07      printf("%d %d\n", m++, ++n);//1 6
08      printf("%d %d\n", m, n);    //2 6
09      printf("%d %d\n", m--, --n);//2 5
10      printf("%d %d\n", m, n);    //1 5
11
12      return 0;
13  }
```



결과

```
1 6
2 6
2 5
1 5
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Ⅰ. 연산식과 다양한 연산자

1교시 수업을 마치겠습니다.

Ⅱ. 관계와 논리, 조건, 비트 연산자

1. 관계와 논리연산자
2. 조건연산자
3. 비트연산자

1. 관계와 논리연산자

◆ 관계연산자

❖ 비교 결과가 참이면 0, 거짓이면 1

- 두 피연산자의 크기를 비교하기 위한 연산자

결과값은 참이면 1, 거짓이면 0이다.

3 > 4	0
56 >= 78	0
"A" < "B"	1
3.46 <= 4.58	1
2 != 2	0
3.4F == 3.4	1

관계연산자는 모두 6가지이다.

- op1 > op2
- op1 >= op2
- op1 < op2
- op1 <= op2
- op1 != op2
- op1 == op2

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 관계와 논리연산자

◆ 관계연산자

실습예제 5-5 | Prj05 | 05relationop.c | 관계 연산 활용 | 난이도: ★

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      int speed = 80;
06      printf("%d ", (60 <= speed));
07      printf("%d\n", (60 > speed));
08
09      printf("%d ", ('a' > 'b'));
10      printf("%d\n", ('Z' <= 'a'));
11      printf("%d ", (4 == 4.0));
12      printf("%d\n", (4.0F != 4.0));
13
14      return 0;
15  }
```

모든 소문자는 대문자보다 코드 값이 크므로 관계 연산도 참을 의미하고 결과는 1

4와 4.0은 같은 것으로 평가

결과

```
1 0
0 1
1 0
```

연산자	연산식	의미	예제	연산(결과)값
>	x > y	x가 y보다 큰가?	3 > 5	0(거짓)
>=	x >= y	x가 y보다 크거나 같은가?	5-4 >= 0	1(참)
<	x < y	x가 y보다 작은가?	'a' < 'b'	1(참)
<=	x <= y	x가 y보다 작거나 같은가?	3.43 <= 5.862	1(참)
!=	x != y	x와 y가 다른가?	5-4 != 3/2	0(거짓)
==	x == y	x가 y가 같은가?	'%' == 'A'	0(거짓)

1. 관계와 논리연산자

◆ 논리연산자

❖ &&, ||, !는 각각

- and, or, not의 논리연산을 의미
- 그 결과가 참이면 1, 거짓이면 0을 반환

❖ C 언어에서 논리 유형은 따로 없으므로

- 0, 0.0, '₩0'은 거짓을 나타내며
- 0이 아닌 모든 정수와 실수, 그리고 널문자 '₩0'가 아닌 모든 문자는 모두 참을 의미

x	y	x && y	x || y	!x
0(거짓)	0(거짓)	0(거짓)	0(거짓)	1(참)
0(거짓)	0 아닌 값(참)	0(거짓)	1(참)	1(참)
0 아닌 값(참)	0(거짓)	0(거짓)	1(참)	0(거짓)
0 아닌 값(참)	0 아닌 값(참)	1(참)	1(참)	0(거짓)

```
21 && 3          1
!2 && 'a'        0
3>4 && 4>=2      0
1 || '\0'        1
2>=1 || 3 <=0    1
0.0 || 2-2       0
!0               1
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 관계와 논리연산자

◆ 논리연산자의 이용

실습예제 5-6 | Prj06 | 06logicop.c | 논리 연산자 && || ! | 난이도: ★

```c
#include <stdio.h>

int main(void)
{
   double grade = 4.21;

   printf("%d ", (4.0 < grade) && (grade <= 5)); //1
   printf("%d ", 0.0 || (4.0 > grade)); //0
   printf("%d\n", (4.2 < grade) || !0.0); //1
   printf("%d ", 'a' && 3.5); //1
   printf("%d ", '\0' || "C"); //1
   printf("%d\n", "java" && '\0'); //0

   return 0;
}
```

결과

```
1 0 1
1 1 0
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 관계와 논리연산자

◆ 논리연산자 &&와 ||의 단축평가

❖ 피연산자 두 개 중에서 왼쪽 피연산자만으로 논리연산 결과가 결정된다면 오른쪽 피연산자는 평가하지 않는 방식

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 관계와 논리연산자

◆ 단축평가 이용

실습예제 5-7 | Prj07 | 07shorteval.c | && 연산자로 일정액 이상의 구매액에 쿠폰 발행과 할인 계산 | 난이도: ★★

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  int main(void)
05  {
06     int amount = 0;
07     int coupons = 10; //각각 10 이상과 10 미만을 입력
08
09     printf("총 금액 >> ");
10     scanf("%d", &amount); // 각각 10000원 이상과 미만을 입력
11
12     int sale = (amount >= 10000) && (coupons++ >= 10);
13     printf("할인: %d, 쿠폰 수: %d\n", sale, coupons);
14
15     return 0;
16  }
```

&&의 왼쪽 (amount >= 10000)가 만족되어야 (coupons++ >= 10)를 실행하며, 이것도 만족해야 최종 결과가 1이 되어 할인이 가능하다.

결과

```
int coupons = 10;
총 금액 >> 9000
할인: 0, 쿠폰 수: 10
총 금액 >> 10000
할인: 1, 쿠폰 수: 11
int coupons = 8;
총 금액 >> 9000
할인: 0, 쿠폰 수: 8
총 금액 >> 10000
할인: 0, 쿠폰 수: 9
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

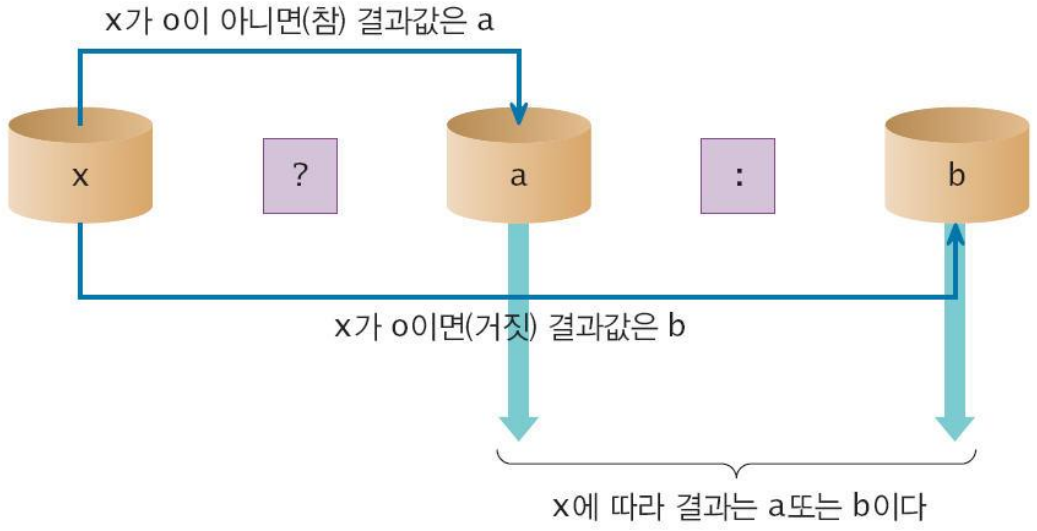
2. 조건연산자

❖ ? :

- 조건에 따라 주어진 피연산자가 결과값이 되는 삼항연산자

❖ 연산식

- 피연산자는 x, a, b 세 개이며
  - 첫 번째 피연산자인 x가 0이 아니면(참) 결과는 a이며
  - x가 0이면(거짓) 결과는 b



[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. 조건연산자

실습예제 5-8 | Prj08 | 08condop.c | 조건 연산자 활용 | 난이도: ★

```
01  #define _CRT_SECURE_NO_WARNINGS
02  #include <stdio.h>
03
04  int main(void)
05  {
06     int a = 0, b = 0;
07
08     printf("두 정수 입력 >> ");
09     scanf("%d%d", &a, &b);
10
11     printf("최대값: %d ", (a > b) ? a : b);
12     printf("최소값: %d\n", (a < b) ? a : b);
13     printf("절대값: %d ", (a > 0) ? a : -a);
14     printf("절대값: %d\n", (b > 0) ? b : -b);
15
16     ((a % 2) == 0) ? printf("짝수 ") : printf("홀수 ");
17     printf("%s\n", ((b % 2) == 0) ? "짝수" : "홀수");
18
19     return 0;
20  }
```

조건 삼항연산자의 두 번째와 세 번째 피연산자는 문장도 가능

조건 삼항연산자의 두 번째와 세 번째 피연산자는 문자열을 비롯하여 모든 자료형도 가능

결과

```
두 정수 입력 >> 8 -9
최대값: 8 최소값: -9
절대값: 8 절대값: 9
짝수 홀수
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 비트연산자

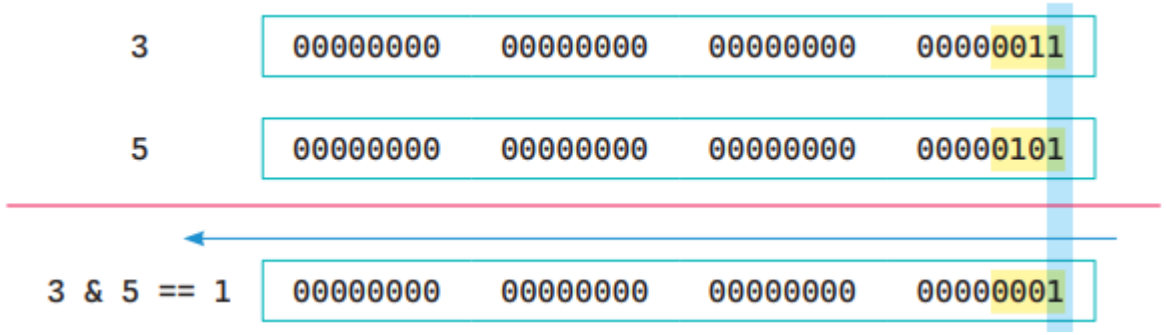
◆ 연산자 &, |, ^, ~ 4가지

연산자	연산자 이름	사용	의미
&	비트 AND	op1 & op2	비트가 모두 1이면 결과는 1, 아니면 0
|	비트 OR	op1 | op2	비트가 적어도 하나 1이면 결과는 1, 아니면 0
^	비트 배타적 OR(XOR)	op1 ^ op2	비트가 서로 다르면 결과는 1, 같으면 0
~	비트 NOT(Negation) 또는 보수(complement)	~op1	비트가 0이면 결과는 1, 0이면 1

❖ 보수 연산자(bitwise complement operator) ~

- 각 비트에서 0은 1, 1은 0이 결과

피연산자		보수 연산	
수	비트표현(2진수)	보수 연산 결과	10진수
1	00000000 00000000 00000000 00000001	11111111 11111111 11111111 11111110	~1 = -2
4	00000000 00000000 00000000 00000100	11111111 11111111 11111111 11111011	~4 = -5



[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 비트연산자

◆ 비트연산자 이용

실습예제 5-9 | Prj09 | 09bitop.c | 비트 연산자 & | ^ ~ | 난이도: ★

```c
#include <stdio.h>

int main(void)
{
   int x = 15; // 1111

   printf("%9x\n", -1);              // 1111
   printf("%3d\n", 10 & -1);         // 10
   printf("%3d\n\n", 10 | 0);        // 10

   printf("%3d %08x\n", x, x);       // 1111
   printf("%3d %08x\n", 1, x & 1);   // 1111 & 0001
   printf("%3d %08x\n", 15, x | 1);  // 1111 | 0001
   printf("%3d %08x\n", 14, x ^ 1);  // 1111 ^ 0001
   printf("%3d %08x\n", ~x, ~x);     // -16

   return 0;
}
```

정수 15는 이진수로 1111

15 & 1의 결과는 1

결과

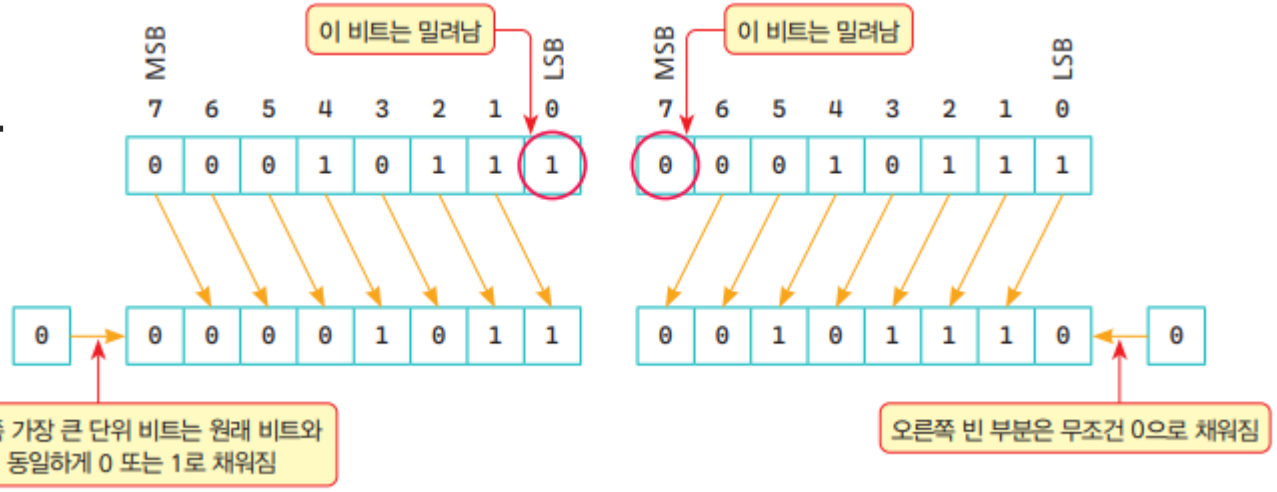
```
ffffffff
 10
 10

 15 0000000f
  1 00000001
 15 0000000f
 14 0000000e
-16 fffffff0
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

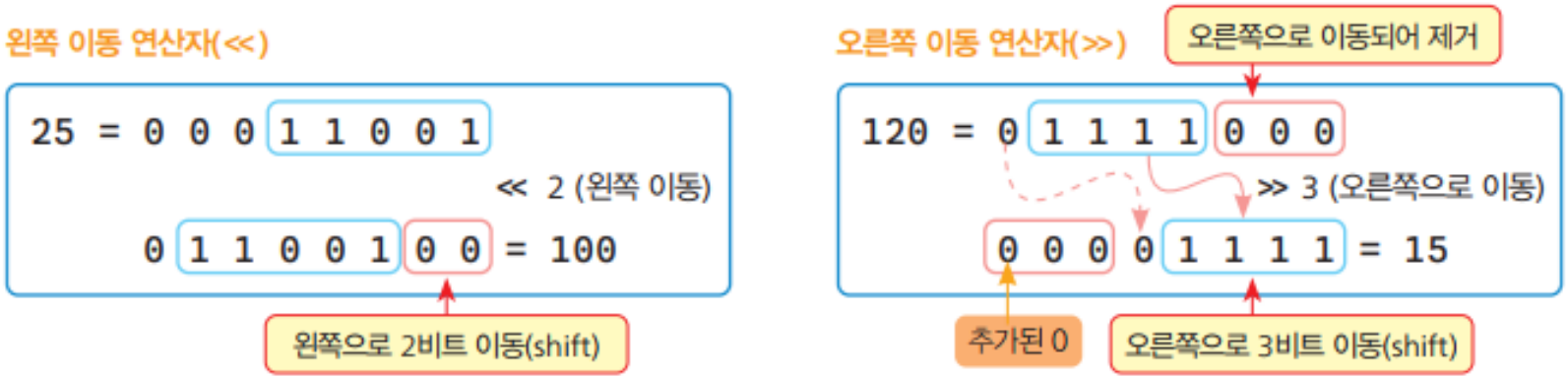
3. 비트연산자

◆ 비트 이동연산자

- Shift right >>
- Shift left <<



연산자	이름	사용	연산 방법	새로 채워지는 비트
>>	shift left	op1 >> op2	op1을 오른쪽으로 op2 비트만큼 이동	가장 왼쪽 비트인 부호 비트는 원래의 부호 비트로 채움
<<	shift right	op1 << op2	op1을 왼쪽으로 op2 비트만큼 이동	가장 오른쪽 비트를 모두 0으로 채움



[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 비트연산자 ◆ 비트 이동연산자

실습예제 5-10 | Prj10 | 10shiftop.c | 비트 이동 연산자 | 난이도: ★

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      int x = 0xffff; //정수 65535
06
07      printf("%6d %08x\n", x, x);           // 1111(f) 1111(f) 1111(f) 1111(f)
08      printf("%6d %08x\n", x >> 1, x >> 1); // 0111(7) 1111(f) 1111(f) 1111(f)
09      printf("%6d %08x\n", x >> 2, x >> 2); // 0011(3) 1111(f) 1111(f) 1111(f)
10      printf("%6d %08x\n", x >> 3, x >> 3); // 0001(1) 1111(f) 1111(f) 1111(f)
11
12      printf("%6d %08x\n", x << 1, x << 1); // 0001(1) 1111(f) 1111(f) 1111(f) 1110(e)
13      printf("%6d %08x\n", x << 2, x << 2); // 0011(3) 1111(f) 1111(f) 1111(f) 1100(c)
14
15      return 0;
16  }
```

설명

```
05      정수 65535는 16진수로 0000ffff
08~10   정수 >> n은 정수를 n번 2로 나눈 효과
12~13   정수 << n은 정수를 n번 2로 곱한 효과
```

결과

```
 65535 0000ffff
 32767 00007fff
 16383 00003fff
  8191 00001fff
131070 0001fffe
262140 0003fffc
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Ⅱ. 관계와 논리, 조건, 비트연산자

2교시 수업을 마치겠습니다.

Ⅲ. 형변환 연산자와 연산자 우선순위

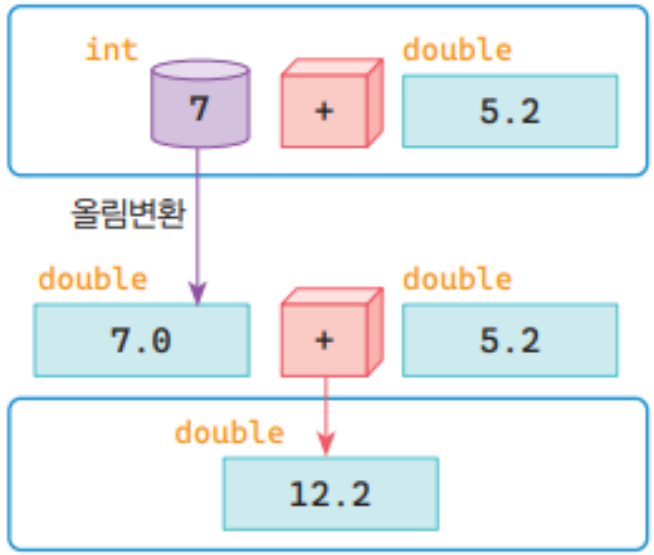
1. 형변환 연산자
2. sizeof 연산자와 콤마연산자
3. 연산자 우선순위

1. 형변환 연산자

◆ 내림변환과 올림변환

❖ 올림변환(형 넓히기)

- 작은 범주의 int형에서 보다 큰 범주인 double 형으로의 형 변환



문자 'a'의 아스키코드값의 int로 변환

다양한 올림변환의 예

'a' + 2	int + int
3 * 4.1F	float * float
4.45F / 3.81	double / double
9.34 - 2	double - double
3 * 4.28	double * double

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 형변환 연산자

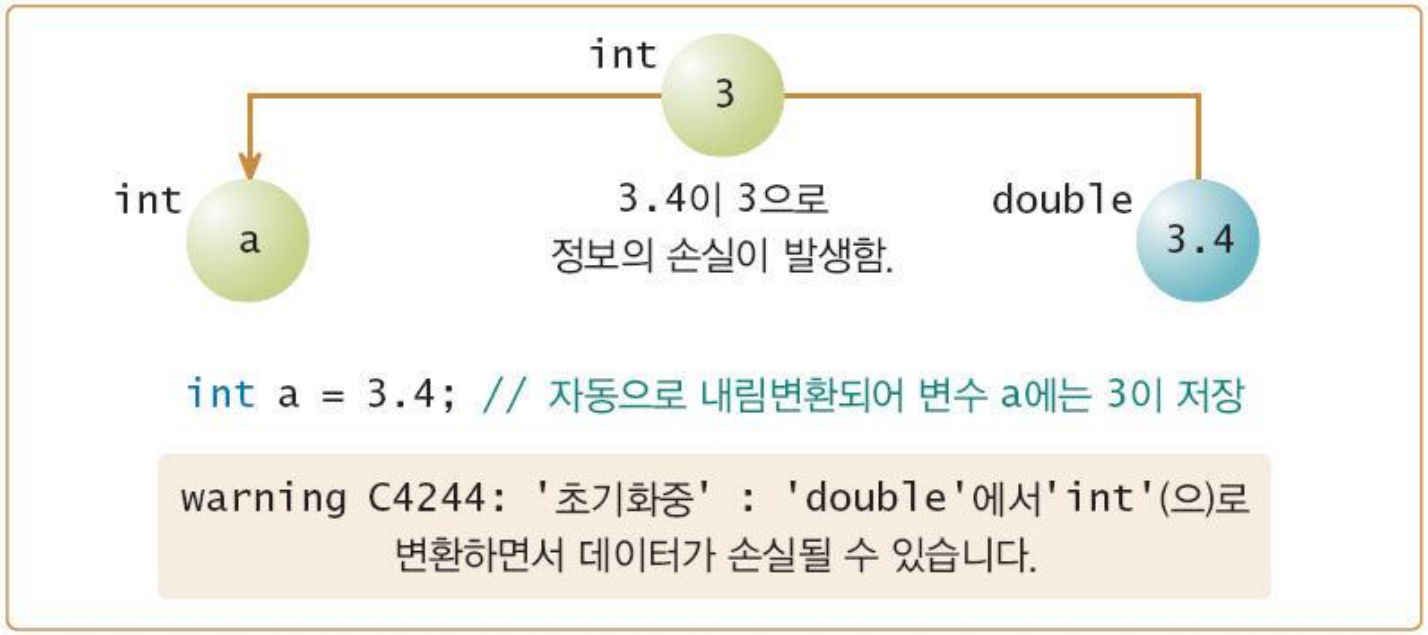
◆ 내림변환과 올림변환

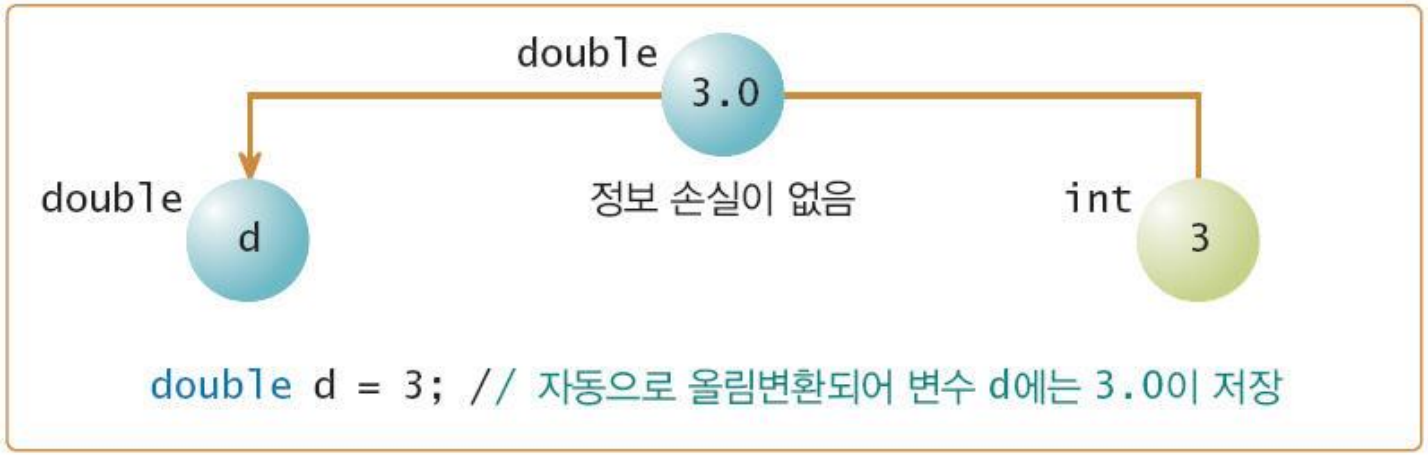
❖ 내림변환(형 좁히기)

- 큰 범주의 형에서 보다 작은 범주로의 형 변환
- 컴파일러가 스스로 시행하는 내림변환의 경우 정보의 손실이 일어날 수 있으므로 경고 발생

❖ 묵시적 형변환(implicit type conversion)

- 컴파일러가 자동으로 수행하는 형변환



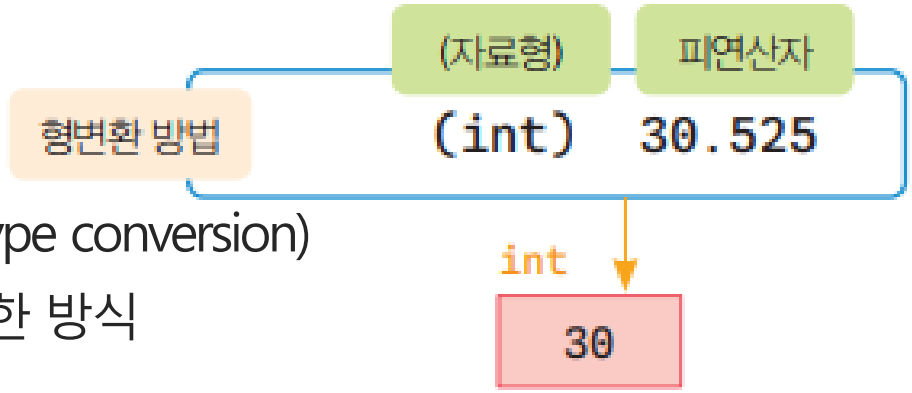


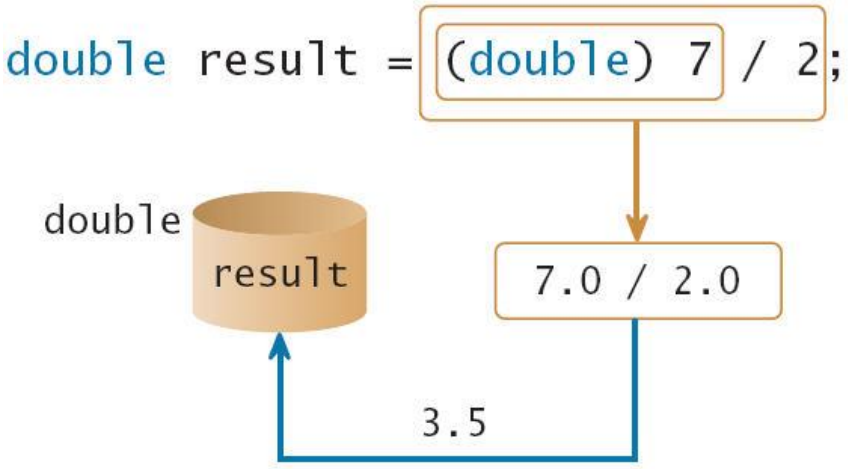
[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 형변환 연산자

❖ (type) 변수_또는_상수

- 명시적 형변환(explicit type conversion)
  - 형변환연산자를 사용한 방식





다양한 형변환연산 예

(int) 3.8 + 5.7	8.7
3.8 + (int) 5.7	8.8
(int) 3.8 + (int) 5.7	8
(int) (3.8 + 5.7)	9
7 / 2	3
7.0 / 2	3.5
7 / (double) 2	3.5
(double) (7/2)	3.0

[출처] 강환수 외, Perfect C 3판, 인피니티북스

1. 형변환 연산자

◆ 형변환 연산자의 활용

실습예제 5-11 | Prj11 | 11typecast.c | 형변환 연산자 활용 | 난이도: ★

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      int a = 7.8;        //자동으로 내림변환되어 변수 a에는 7.8이 저장
06      double b = 5;       //자동으로 올림변환되어 변수 b에는 5.0이 저장
07
08      printf("%d %f ", a, b);
09      printf("%d %f ", (int) 3.56, (double) 3);
10      printf("%f %d\n ", 3.56 + 7.87, (int)(3.56 + 7.87));
11
12      printf("%d %f %f\n", 5 / 2, (double) 5 / 2, (double) (5 / 2));
13
14      return 0;
15  }
```

결과는 정수인 2

5를 먼저 double로 변환한 후 5.0 / 2를 연산하므로 결과는 2.5

결과

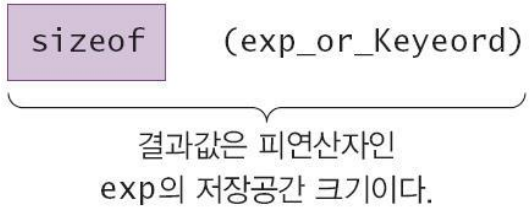
```
7 5.000000 3 3.000000 11.430000 11
2 2.500000 2.000000
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. sizeof 연산자와 콤마연산자

❖ 연산자 sizeof

- 표현식 또는 자료형의 저장장소 크기, 바이트 단위의 정수




```
sizeof(int)     결과는 4
sizeof(3.14)    결과는 8
sizeof a        결과는 2(short a;)
```

❖ 반환 값 자료형

- 자료형 size_t
- printf()에서 형식제어문자 %zu로 출력

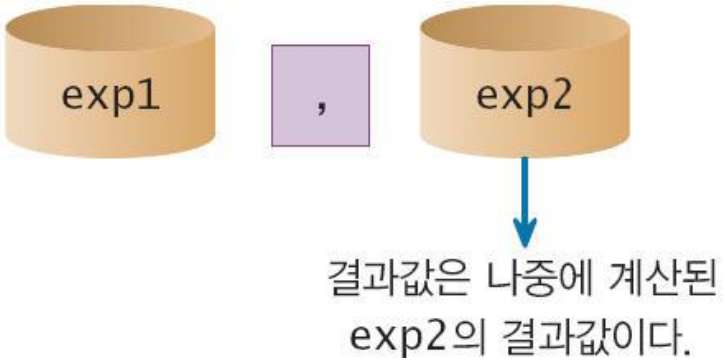
```
size_t sz = sizeof (short);
printf("%zu\n", sz);
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. sizeof 연산자와 콤마연산자

❖ 콤마연산자

- 왼쪽과 오른쪽 연산식을 각각 계산하며 결과값은 오른쪽에서 수행한 연산의 결과
- 간단히 연산식 2, 4의 결과값은 4




```
3 + 4, 5 - 10                    결과는 -5

3 + 4, 5 - 10, 2 * 3             결과는 6
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

2. sizeof 연산자와 콤마연산자

실습예제 5-12 | Prj12 | 12sizeofcomma.c | 연산자 sizeof와 콤마 연산자의 활용 | 난이도: ★

```c
#include <stdio.h>

int main(void)
{
   int a, x;
   a = x = 0;

   x = 3 + 4, 2 * 3;      // (x = 3+4), 2*3;
   printf("x = %d ", x);
   x = (3 + 4, 2 * 3);    // x = (3+4, 2*3);
   printf("x = %d\n", x);

   int byte = sizeof (double);
   printf("double 형: %d bytes, %d bits\n", byte, byte * 8);
   int bit = (byte = sizeof a, byte * 8);
   printf("int 형: %d bytes, %d bits\n", byte, bit);

   size_t sz = sizeof (short);
   printf("%zu\n", sz);

   return 0;
}
```

연산자 sizeof (자료형)에서 괄호는 필수이며, 결과 자료형은 size_t이나 간단히 int형 자료형에 저장도 가능

변수 bit에 저장되는 것은 나중에 계산된 byte * 8

결과

```
x = 7 x = 6
double 형: 8 bytes, 64 bits
int 형: 4 bytes, 32 bits
2
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 연산자 우선순위

◆ 연산규칙

- ❖ 첫 번째 규칙은 괄호가 있으면 먼저 계산
- ❖ 두 번째 규칙으로 연산의 우선순위(priority)
- ❖ 세 번째 규칙은 동일한 우선순위인 경우, 연산을 결합하는 방법인 결합성(또는 결합규칙)
  - 왼쪽부터 오른쪽으로 차례로 계산
  - 제곱승과 같은 정해진 연산은 오른쪽에서 왼쪽으로 차례로 계산

이항연산자

콤마 < 대입 < 조건(삼항) < 논리 < 관계 < 산술 < 단항 < 괄호와 대괄호

- 괄호와 대괄호는 무엇보다도 가장 먼저 계산한다.
- 모든 단항연산자는 어느 이항연산자보다 먼저 계산한다.
- 산술연산자 *, /, %는 +, -보다 먼저 계산한다.
- 산술연산자는 이항연산자 중에서 가장 먼저 계산한다.
- 관계연산자는 논리연산자보다 먼저 계산한다.
- 조건 삼항연산자는 대입연산자보다 먼저 계산하나, 다른 대부분의 연산보다는 늦게 계산한다.
- 조건 > 대입 > 콤마연산자 순으로 나중에 계산한다.

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Ⅲ. 형변환 연산자와 연산자 우선순위

표 5-9 C 언어의 연산자 우선순위

우선순위	연산자	설명	분류	결합성(계산방향)
1	( ) [ ] . -> a++ a--	함수 호출 및 우선 지정 인덱스 필드(유니온) 멤버 지정 필드(유니온)포인터 멤버 지정 후위 증가, 후위 감소	단항	-> (좌에서 우로)
2	++a --a ! ~ sizeof - + & *	전위 증가, 전위 감소 논리 NOT, 비트 NOT(보수) 변수, 자료형, 상수의 바이트 단위 크기 음수 부호, 양수 부호 주소 간접, 역참조		<- (우에서 좌로)
3	(형변환)	형변환		
4	* / %	곱하기 나누기 나머지	산술	-> (좌에서 우로)
5	+ -	더하기 빼기		-> (좌에서 우로)
6	<< >>	비트 이동	이동	-> (좌에서 우로)
7	< > <= >=	대소 비교	관계	-> (좌에서 우로)
8	== !=	동등 비교		-> (좌에서 우로)
9	&	비트 AND 또는 논리 AND	비트	-> (좌에서 우로)
10	^	비트 XOR 또는 논리 XOR		-> (좌에서 우로)
11	|	비트 OR 또는 논리 OR		-> (좌에서 우로)
12	&&	논리 AND(단락 계산)	논리	-> (좌에서 우로)
13	||	논리 OR(단락 계산)		-> (좌에서 우로)
14	? :	조건	조건	<- (우에서 좌로)
15	= += -= *= /= %= <<= >>= &= |= ^=	대입	대입	<- (우에서 좌로)
16	,	콤마	콤마	-> (좌에서 우로)

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 연산자 우선순위

실습예제 5-13 | Prj13 | 13oppriority.c | 연산자 우선순위에 위한 계산 | 난이도: ★★

```c
#include <stdio.h>

int main(void)
{
   int speed = 90;
   int x = 1, y = 2, z = 3;

   printf("%d ", 60 <= speed && speed <= 80 + 20); //산술 > 관계 > 논리
   printf("%d ", ( (60 <= speed) && (speed <= (80 + 20)) ));

   printf("%d ", x % 2 == 0 ? y + z : y * z); //산술 > 관계 > 조건
   printf("%d ", (x % 2 == 0) ? (y + z) : (y * z));

   printf("%d ", speed += ++x && y - 2); //단항++ > 산술 > 논리 > 대입
   printf("%d\n", speed += ( (++x) && (y - 2) ));

   return 0;
}
```

결과

```
1 1 6 6 90 90
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

3. 연산자 우선순위

실습예제 5-14 | Prj14 | 14opassociation.c | 연산자의 결합성에 따른 계산 순서 확인 | 난이도: ★★

```c
#include <stdio.h>

int main(void)
{
   int m = 5, n = 10;
   printf("%d\n", m += n /= 3); //우측에서 좌측으로 결합, (m += (n /= 3))
   printf("%d %d\n", m, n); //8, 3

   //우측에서 좌측으로 결합
   printf("%d ", 3 > 4 ? 3 - 4 : 3 > 4 ? 3 + 4 : 3 * 4); //12
   printf("%d\n", 3 > 4 ? 3 - 4 : (3 > 4 ? 3 + 4 : 3 * 4)); //위와 같은 12

   printf("%d ", 10 * 3 / 2);   //좌측에서 우측으로 결합, 15
   printf("%d\n", 10 * (3 / 2)); //우측에서 좌측으로 결합, 10

   return 0;
}
```

설명

06 연산식 m += n /= 3은 (m += (n /= 3))이므로 결과값은 m에 대입된 8 출력
10 조건연산자는 결합성이 오른쪽에서 왼쪽으로
13~14 연산식 10 * 3 / 2은 (10 * 3) / 2이므로 10 * (3 / 2)과 결과가 다름

결과

```
8
8 3
12 12
15 10
```

[출처] 강환수 외, Perfect C 3판, 인피니티북스

Ⅲ. 형변환 연산자와 연산자 우선순위

3교시 수업을 마치겠습니다.