C 언어

연산자

- 대입연산자
- 산술연산자
- 증감연산자
- 비교연산자
- 논리연산자

연산자

■ 연산자

- 대입 연산자(= , +=, -=, *=, /=)
- 산술연산자(+ , - , * , / , %)
- 증감 연산자(++, --)
- 비교연산자(== , !=, …)
- 논리연산자(&& (==and), || (==or), ! (==not))

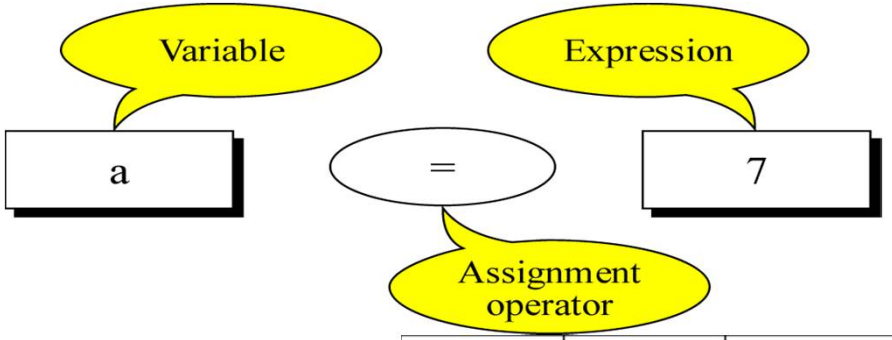
■ 형변환

- 값을 다른 자료형으로 바꾸어 주는 연산
- 실수 --> 정수 , 정수 --> 실수, 정수-->문자, 문자-->정수
- (바꾸려고하는 변수타입)변수명

```
int x = 10;
....
double y = (double) x ;
```

대입연산자

- Assignment expression
  - 수학에서는 “우측의 값과 좌측의 값이 같다”라는 의미
  - C 언어에서는 “우측의 값을 좌측의 저장 장소에 저장하라”라는 의미



  - 분류
    - simple assignment ←
    - 예) a=5, b=x+1, i=i+1등

Contents of Variable x	Contents of Variable y	Expression	Value of Expression	Result of Expression
10	5	x = y + 2	7	x = 7
10	5	x = x / y	2	x = 2
10	5	x = y % 4	1	x = 1

    - Compound assignment
    - 예) x+=y, x*=y, x/=y 등 ←

Compound Expression	Equivalent Simple Expression
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

대입 연산자

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	1
-=	x-=y	2
*=	x*=y	3
/=	x/=y	4
%=	x%=y	5

x = y 는 y 공간에 있는 값을 x 공간에 넣는다.

산술 연산자

연산자	기호	의미	예
덧셈	+	x와 y를 더한다	x+y
뺄셈	-	x에서 y를 뺀다.	x-y
곱셈	*	x와 y를 곱한다.	x*y
나눗셈	/	x를 y로 나눈다.	x/y
나머지	%	x를 y로 나눌 때의 나머지값	x%y

산술 연산자

- Binary expression
  - 형식 : 피연산자 – 연산자 – 피연산자 (operand – operator – operand)
  - Operator가 operand사이에 있음

  - Binary expression 예
    - a+7, 3+4, b–11 등 ← additive expression (+, –)
    - 10*12, a /4, 5%2 등 ← multiplicative expression (*, /, %)
      - %(modulo)는 나머지를 구하는 연산자

실습예제

INPUT

- 2개의 정수 입력
- 원하는 연산 (1 = 더하기, 2=빼기, 3=곱하기, 4=나누기)

Function

- 각 연산을 수행하고 결과값을 리턴 하는 함수
  - 더하기 : a + b
  - 빼기 : a - b
  - 곱하기 : a * b
  - 나누기 : a / b

증감 연산자

- 증가연산(prefix, postfix)과 일반산술연산(infix)의 비교

```c
#include <stdio.h>

int main(void){
        int x = 10;

        printf("%d\n", x++);
        printf("%d\n", x);
        printf("%d\n", --x);
        printf("%d\n", x);

        return;
}
```

printf	X
	10
10	11
11	11
10	10
10	10

```c
#include <stdio.h>

int main(void){
        int x = 10;

        printf("%d\n", x+1);
        printf("%d\n", x);
        printf("%d\n", x-1);
        printf("%d\n", x);

        return;
}
```

printf	X
	10
11	10
10	10
9	10
10	10

증감 연산자

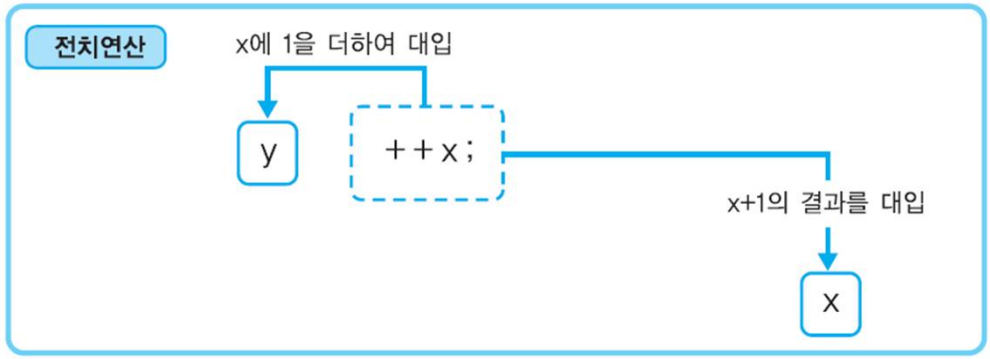
- 증감연산 사용 이유
  - 증감연산자를 이용하면 프로그램 형태가 간결
  - 기계어 코드와 일대일 대응되므로 실행속도가 개선

- 주의점
  - 연산자의 위치에 따라 evaluation value가 다르므로 주의 요구
  - 증감연산자는 ++, --자체가 연산자 기호 ➔ 중간에 공백이 들어가면 안됨
  - 산술연산이나 관계, 논리연산보다 그 평가를 먼저 한다.
  - 증감연산자는 피연산자로 변수를 사용할 수 있지만, 상수나 일반 수식을 피연산자로 사용 불가능
    - 다음과 같은 수식은 잘못된 수식

```
int a = 10;
++300;  /* 상수에는 증가 연산자를 사용할 수 없다 */
(a+1)--; /* 일반 수식에는 증가 연산자를 사용할 수 없다 */
```

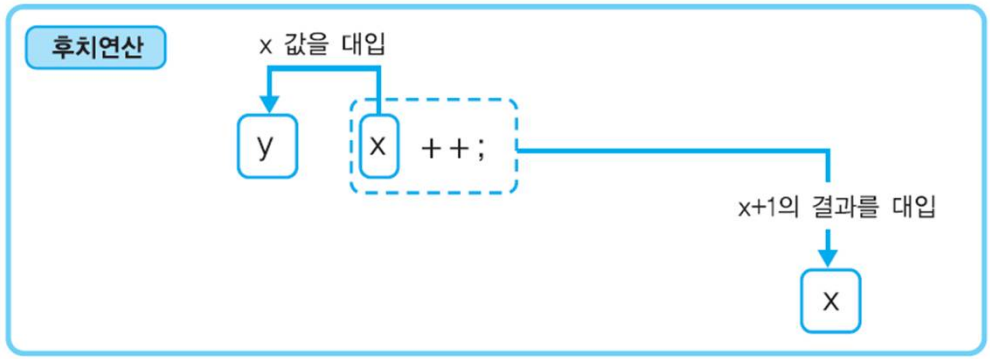
증감 연산자

- 증가 연산자(++)와 감소 연산자(--)




```
var x = 3;
var y = ++x;
printf("%d",x); // ?
printf("%d",y); // ?
```

++x; <===> x=x+1; x;



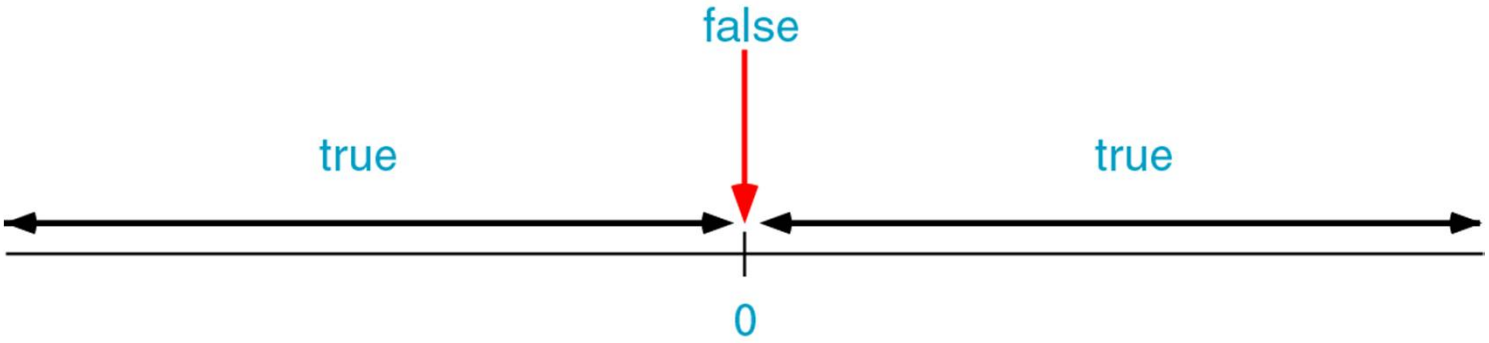

```
var x = 3;
var y = x++;
printf("%d",x); // ?
printf("%d",y); // ?
```

x++; <===> x; x=x+1;

논리연산자

- 선택(selection) 은 어떤 상황에서 판단을 하여 다른 흐름을 만드는 방법
- 선택은 논리적 판단에 기반함

  그러나 전통적으로 C에는 논리형(logical type)이 없음
  - 데이터 값이 '0(zero)'이면 'false', (nonzero) 이면 'true'로 봄
  - C99에서 소개된 Boolean 데이타 타입 사용
- 정수, 실수 등에서는 0, 문자에서는 '\0'가 false에 해당

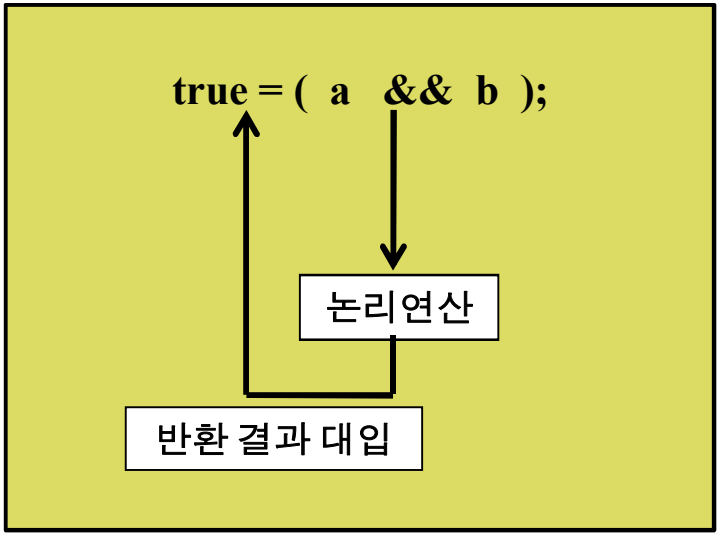


논리연산자

- 예제프로그램 – 논리연산자

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

int main(void){
        bool a = true;
        bool b = true;
        bool c = false;

        printf("    %2d AND     %2d : %2d\n", a, b, a && b);
        printf("    %2d AND     %2d : %2d\n", a, c, a && c);
        printf("    %2d AND     %2d : %2d\n", c, a, c && a);
        printf("    %2d OR      %2d : %2d\n", a, c, a || b);
        printf("    %2d OR      %2d : %2d\n", c, a, c || a);
        printf("    %2d OR      %2d : %2d\n", c, c, c || c);
        printf("NOT %2d AND NOT %2d : %2d\n", a, c, !a && !c);
        printf("NOT %2d AND     %2d : %2d\n", a, c, !a && c);
        printf("    %2d AND  NOT%2d : %2d\n", a, c, a && !c);

        return 0;
}
```

a 는 true 이므로 1,
b 도 true 이므로 1.




```
    1 AND     1 :  1
    1 AND     0 :  0
    0 AND     1 :  0
    1 OR      0 :  1
    0 OR      1 :  1
    0 OR      0 :  0
NOT 1 AND NOT 0 :  0
NOT 1 AND     0 :  0
    1 AND  NOT 0 :  1
```

논리연산자

- 논리연산자
  - **and(&&), or(||), not(!)**을 표현하는 연산자
  - 두 피연산자의 참과 거짓에 따라 연산의 결과값을 결정
    - **&&** : 피연산자가 모두 참이면 **true**를 반환
    - || : 피연산자 중 하나라도 참이면 **true**를 반환
    - ! : 피연산자가 **true**이면 **false**를, **false**이면 **true**를 반환

- 논리 연산자의 연산 방법
  - 연산에 참여하는 자료 값 중 **0**은 거짓**(false)**을 의미
    1(0이 아닌 값**)**은 참**(true)**을 의미
  - 정수만을 이용하는 것은 아니고
    실수나 다른 유형의 자료 값에도 이용
    - 평가의 결과는 반드시 **0(false)**이거나 **1(true)**

논리연산자

연산자 기호	사용 예	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	&& 연산	|| 연산	! 연산
		x && y	x || y	! x
0(false)	0			true(1)
0	1(true)			true(1)
1	0			false(0)
1	1			false(0)

<Logical Operators Truth Table>

관계연산자

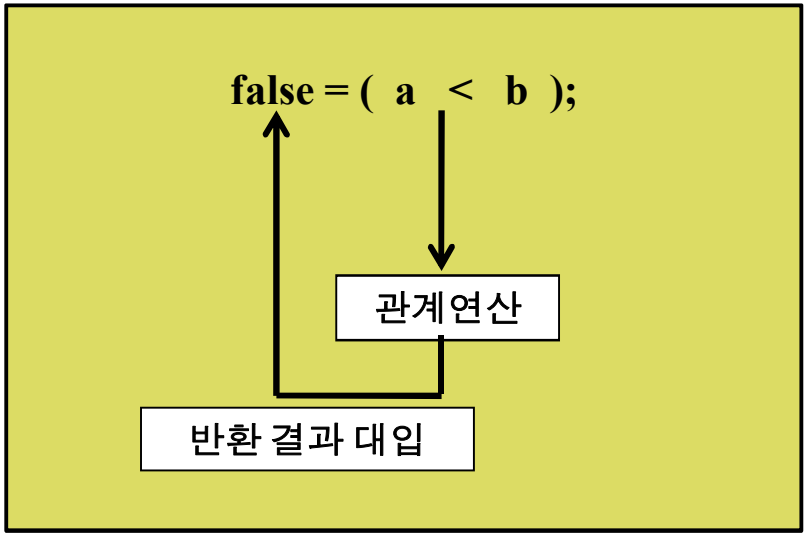
- 관계연산자
  - 두 값의 크기를 비교하는 연산자
    - <, >, >=, <=, ==, !=
    - 두 기호의 순서가 바뀌면 에러 발생(=<, =!)

연산자 기호	의미	사용 예
==	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?	x <= y

관계연산자

- 예제프로그램 – 관계연산자

```c
#include <stdio.h>

int main(void){
        int a = 5;
        int b = -3;

        printf("%2d  < %2d is %2d\n",a,b,a <b);
        printf("%2d == %2d is %2d\n",a,b,a==b);
        printf("%2d != %2d is %2d\n",a,b,a!=b);
        printf("%2d  > %2d is %2d\n",a,b,a >b);
        printf("%2d <= %2d is %2d\n",a,b,a<=b);
        printf("%2d >= %2d is %2d\n",a,b,a>=b);

        return 0;
}
```




```
gr120100206@cspro:~/CProgramming/chap5$ vi comparative_operator.c
gr120100206@cspro:~/CProgramming/chap5$ gcc -o comparative_operator comparative_operator.c
gr120100206@cspro:~/CProgramming/chap5$ ./comparative_operator
 5  < -3 is  0
 5 == -3 is  0
 5 != -3 is  1
 5  > -3 is  1
 5 <= -3 is  0
 5 >= -3 is  1
```

관계연산자

Original Expression	! (x < y)	! (x > y)	! (x != y)	! (x <= y)	! (x >= y)	! (x == y)
Simplified Expression	x >= y	x <= y	x == y	x > y	x < y	x != y

<Example of simplifying Operator Complements>

Type Conversion

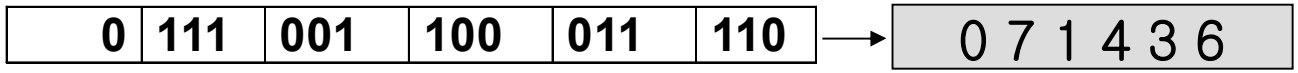
- Type Conversion
  - 일반적으로 C 언어의 연산식에서 여러 피연산자의 자료형이 서로 다른 경우, 하나의 통일된 자료형으로 자동 변환하여 연산을 수행
  - Implicit type conversion (coercion) :
    C 컴파일러가 판단하여 자동으로 데이터형을 변형하는 경우
    - 다음과 같은 경우 C 컴파일러가 판단하여 자동 형 변환을 수행한다.
      - 수식에서 데이터형이 혼합되어 사용되었을 때 값을 변환
      - 특정한 데이터형의 변수에 다른 데이터형의 값을 대입할 때, 값을 변환
  - Explicit type conversion (cast) : 프로그래머가 형 변환자(cast)를 사용하여 강제적으로 변형하는 경우

실습예제-Type Conversion

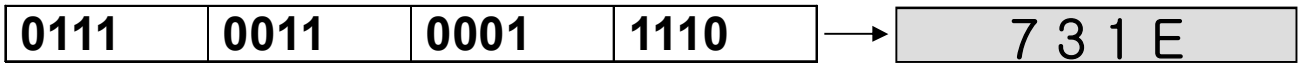
```c
#include<stdio.h>
#define ip 10
int main()
{
    int a=10;
    int b= 15;

    double result;

    result = a / b;
    printf("%lf ₩n", result);

    result = (double)a / b;
    printf("%lf ₩n", result);

    return 0;
}
```

Number Conversion

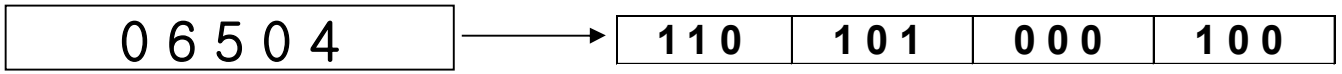
- 2진수, 8진수, 16진수 사이의 변환
  - 2진수를 오른쪽에서부터 3bits로 묶어서 8진수로 변환.
    ex) 0111001100011110



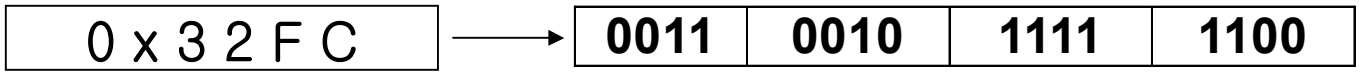
  - 2진수를 오른쪽에서부터 4bits로 묶어서 16진수로 변환
    ex) 0111001100011110



  - 8진수 하나의 숫자는 2진수의 3자리로 변환



  - 16진수 하나의 숫자는 2진수의 4자리로 변환



  - 16진수 ↔ 8진수는 2진수로 변환하여 변환

실습예제- Number Conversion