

제 7 장. 연 산 자

연산자의 종류

- 연산자(operator)
 - 산술 연산자인 $+$, $-$, $*$, $/$ 와 같이 이미 정의된 연산을 수행하는 기호
- 피연산자(operand)
 - 연산(operation)에 참여하는 변수나 값
- 연산자와 피연산자를 구분하기 위한 예

```
int x = 3, y = 4, z;  
z = x + y;
```

연산자의 종류

- 연산자의 종류

분류	예
산술 연산자	+, -, *, /, %
관계 연산자	>, <, ==, !=, >=, <=
증감 연산자	++, --
논리 연산자	&&, , !
조건 연산자	?:
비트 논리 연산자	&, , ^, ~
비트 이동 연산자	<<, >>
대입 연산자	=

대입 연산자

- 대입(할당) 연산자
 - 일반적인 수학에서 사용하는 = 기호
 - 연산자를 이용하여 얻은 의미 있는 문장을 실행할 때 사용
- 연산 진행 순서

```
x = 5 + 6;
```

- 연산자: +, =
- 피연산자: 5, 6, x

5+6(산술 연산자) → =(대입 연산자)
5와 6을 더한 후 변수 x에 저장

```
x = x + 2;
```

- 연산자: +, =
- 피연산자: x, 2

x+2(산술 연산자) → =(대입 연산자)
변수 x 값에 2를 더한 후 변수 x에 저장

대입 연산자

- lvalue와 rvalue의 구별

```
lvalue = rvalue;
```

- lvalue(left value) : 대입 연산자를 기준으로 왼쪽에 있는 피연산자(오로지 변수만 올 수 있음)
- rvalue(right value) : 대입 연산자를 기준으로 오른쪽에 있는 피연산자(값, 변수, 수식이 올 수 있음)

구분	내용	예
올바르게 사용한 예	rvalue에 값이 온 경우	$x = 3;$
	rvalue에 변수가 온 경우	$x = y;$
	rvalue에 수식이 온 경우	$x = 3 + 4;$
잘못 사용한 예	lvalue에 값이 온 경우	$5 = 3 + 2;$
	lvalue에 수식이 온 경우	$x + 3 = y + 5;$

대입 연산자

- 예제) 대입 연산자의 사용 예제

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int a, b = 0;
06
07     a = 4;
08     printf("a = 4 문장을 실행한 후 a의 값은 %d\n", a);
09
10     a = a + 3;
11     printf("a = a + 3 문장을 실행한 후 a의 값은 %d\n", a);
12
13     b = a + 2;
14     printf("b = a + 2 문장을 실행한 후 b의 값은 %d\n", b);
15
16     return 0;
17 }
```



단항 연산자



- 산술 연산자
 - 산술 연산을 수행하는 연산자
 - 단항 연산자와 이항 연산자로 나뉨
- 단항 연산자
 - 피연산자를 1개만 사용하는 산술 연산자
- 증감 연산자
 - 단항 연산자 중 ++, --
 - 변수값을 1씩 증가 또는 감소시킴

단항 연산자

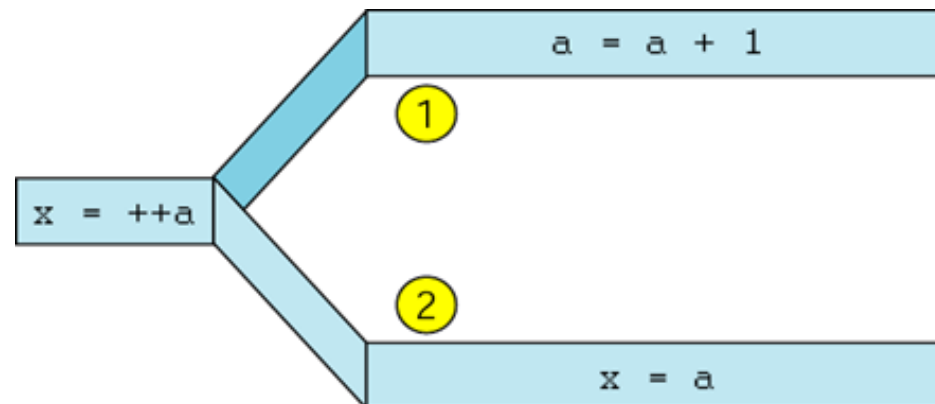
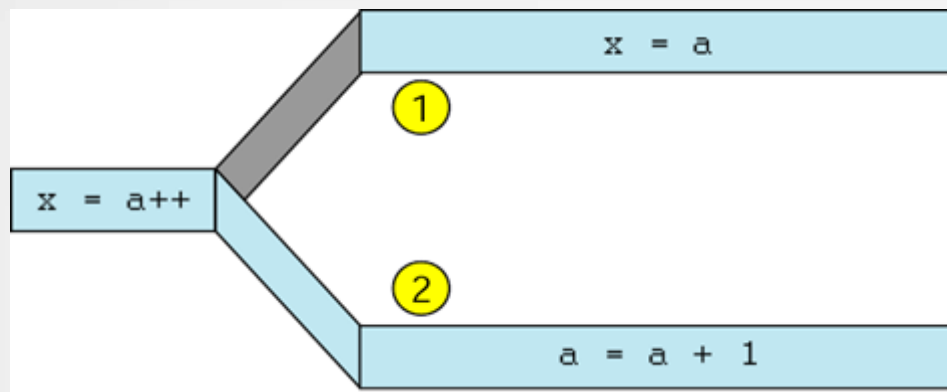
- 증감 연산자의 종류

연산자	연산식	의미
++	a ++	변수값을 수식에 먼저 적용시킨 후 최종 변수값을 1만큼 증가시킨다.
	++ a	변수값을 1만큼 먼저 증가시킨 후 최종 변수값을 수식에 적용시킨다.
--	a --	변수값을 수식에 먼저 적용시킨 후 최종 변수값을 1만큼 감소시킨다.
	-- a	변수값을 1만큼 먼저 감소시킨 후 최종 변수값을 수식에 적용시킨다.

- 증감 연산자의 사용 예 (a=10, b=10)

연산식	실행절차	실행결과
b = ++a	a에 1을 더한 후 그 값을 b에 대입한다.	a = 11, b = 11
b = --a	a에 1을 뺀 후 그 값을 b에 대입한다.	a = 9, b = 9
b = a++	a를 b에 대입한 후 a에 1을 더한다.	a = 11, b = 10
b = a--	a를 b에 대입한 후 a에서 1을 뺀다.	a = 9, b = 10

단항 연산자



단항 연산자

- 예제) 전치 연산과 후치 연산의 차이 비교한 예제

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int a, b, c;
06     a = 10; b = 20; c = 30;
07
08     a = ++b;
09
10     c = b++;
11
12     printf("a = %d, b = %d, c = %d\n\n", a, b, c);
13
14     return 0;
15 }
```

이항 연산자

- 이항 연산자
 - 피연산자가 2개 필요한 산술 연산자
 - 사칙 연산과 나머지 연산자를 포함
- 산술 연산자의 종류

연산자	의미	기능	연산자	의미	기능
+	덧셈	$a = b + c$	-	뺄셈	$a = b - c$
*	곱셈	$a = b * c$	/	나눗셈	$a = b / c$
%	나머지	$a = b \% c$			

산술 연산자

- 나눗셈 연산자

```
int x , y = 9;  
x = y / 2;      // 결과 x에 4가 저장된다.  
x = y / 4;      // 결과 x에 2가 저장된다.
```

몫만 저장!

- 나머지 연산자

```
int x , y = 9;  
x = y % 4;      // 결과 x에 1이 저장된다.  
x = y % 6;      // 결과 x에 3이 저장된다.
```

나머지만 저장!

관계 연산자

- 관계 연산자
 - 두 수 사이의 대소 관계와 특정 조건을 검사할 때 사용하는 연산자
 - 관계가 성립되면 참(true 또는 1)
 - 관계가 성립되지 않으면 거짓(false 또는 0)
- 관계 연산자의 종류와 사용 예

연산자	의미	사용 방법	설명
>	~ 보다 크다.	$a = (b > c)$	b가 c보다 크면 $a = 1$, 그렇지 않으면 $a = 0$
<	~ 보다 작다.	$a = (b < c)$	b가 c보다 작으면 $a = 1$, 그렇지 않으면 $a = 0$
>=	~ 보다 크거나 같다.	$a = (b >= c)$	b가 c보다 크거나 같으면 $a = 1$, 그렇지 않으면 $a = 0$
<=	~ 보다 작거나 같다.	$a = (b <= c)$	b가 c보다 작거나 같으면 $a = 1$, 그렇지 않으면 $a = 0$
==	같다.	$a = (b == c)$	b와 c가 같으면 $a = 1$, 그렇지 않으면 $a = 0$
!=	같지 않다.	$a = (b != c)$	b와 c가 같지 않으면 $a = 1$, 같으면 $a = 0$

관계 연산자

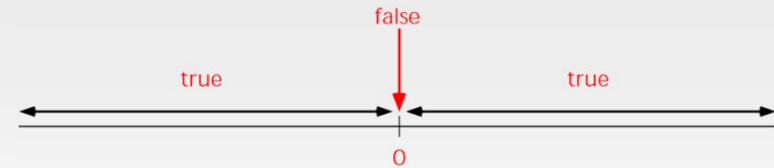
- 예제) 합격과 불합격을 판정하는 예제

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int score, grade;
06
07     printf("점수를 입력하세요 : ");
08     scanf("%d", &score);
09
10     grade = (score >= 90);
11
12     if (grade == 1)
13         printf("결과는 %d, 합격\n", grade);
14     else
15         printf("결과는 %d, 불합격\n", grade);
16
17     return 0;
18 }
```

논리 연산자

- 논리 연산자

- 조건 여러 개를 결합하여 판정하는 연산자
- AND, OR, NOT의 논리연산을 수행
- 관계가 성립되면 참(true 또는 1)을 표시
- 성립되지 않으면 거짓(false 또는 0)을 표시



- 논리 연산의 진리표

X	Y	AND(&&)	OR()	NOT X(!X)
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

논리 연산자

- 논리 연산자의 종류와 사용 예

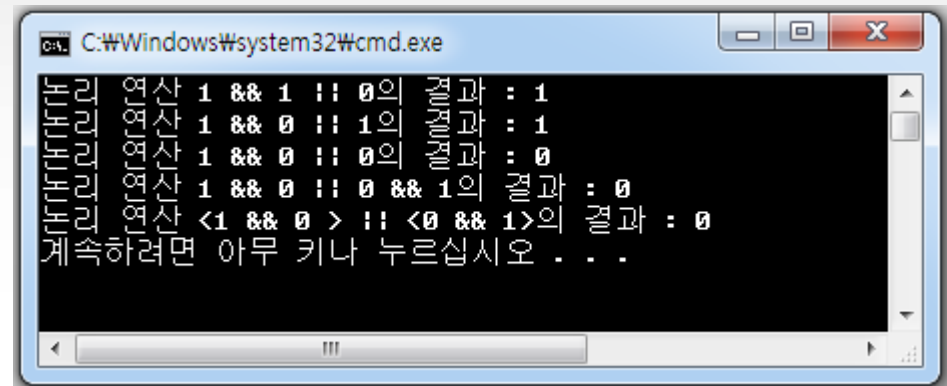
연산자	의미	사용 방법	설명
논리곱(&&)	AND	<code>a = b && c</code>	b와 c가 모두 참이면 a = 1, 아니면 a = 0
논리합()	OR	<code>a = b c</code>	b와 c가 모두 거짓이면 a = 0, 아니면 a = 1
논리부정(!)	NOT	<code>a = !b</code>	b가 참이면 a = 0, b가 거짓이면 a = 1

- 논리 연산자의 우선순위
 - 단항 연산자인 !이 가장 높고 다음이 &&이며 ||가 가장 낮음.
(논리부정 > 논리곱 > 논리합)
 - &&과 ||이 같이 쓰이면 &&이 먼저 수행됨.

논리 연산자

- 예제) 논리 연산자 2개 이상 사용

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int a = 0, b = 1, c = 1;
06
07     printf("논리 연산 1 && 1 || 0의 결과 : %d\n", 1 && 1 || 0);
08     printf("논리 연산 1 && 0 || 1의 결과 : %d\n", 1 && 0 || 1);
09     printf("논리 연산 1 && 0 || 0의 결과 : %d\n", 1 && 0 || 0);
10     printf("논리 연산 1 && 0 || 0 && 1의 결과 : %d\n", b && a || a && c);
11     printf("논리 연산 <1 && 0 > || <0 && 1>의 결과 : %d\n", (b && a) || (a && c));
12
13     return 0;
14 }
```

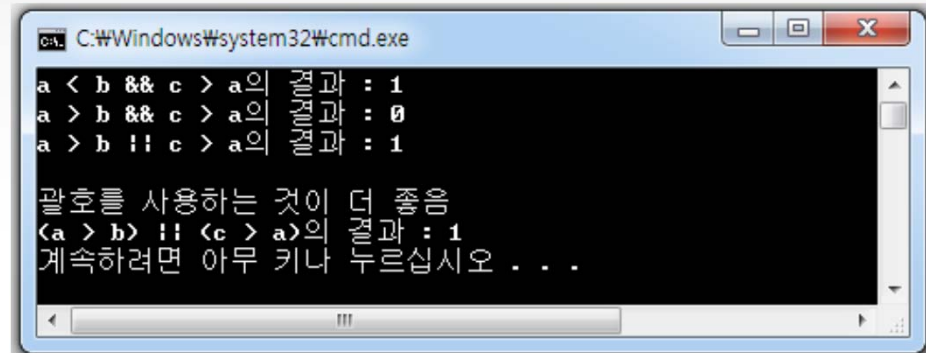


```
C:\Windows\system32\cmd.exe
논리 연산 1 && 1 || 0의 결과 : 1
논리 연산 1 && 0 || 1의 결과 : 1
논리 연산 1 && 0 || 0의 결과 : 0
논리 연산 1 && 0 || 0 && 1의 결과 : 0
논리 연산 <1 && 0 > || <0 && 1>의 결과 : 0
계속하려면 아무 키나 누르십시오 . . .
```

논리 연산자

- 예제) 논리 연산자와 관계 연산자 결합

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int a = 0, b = 1, c = 1;
06
07     printf("a < b && c > a의 결과 : %d\n", a < b && c > a);
08     printf("a > b && c > a의 결과 : %d\n", a > b && c > a);
09     printf("a > b || c > a의 결과 : %d\n\n", a > b || c > a);
10
11     printf("괄호를 사용하는 것이 더 좋음\n");
12     printf("(a > b) || (c > a)의 결과 : %d\n", (a > b) || (c > a));
13
14     return 0;
15 }
```



```
C:\Windows\system32\cmd.exe
a < b && c > a의 결과 : 1
a > b && c > a의 결과 : 0
a > b || c > a의 결과 : 1

괄호를 사용하는 것이 더 좋음
(a > b) || (c > a)의 결과 : 1
계속하려면 아무 키나 누르십시오 . . .
```

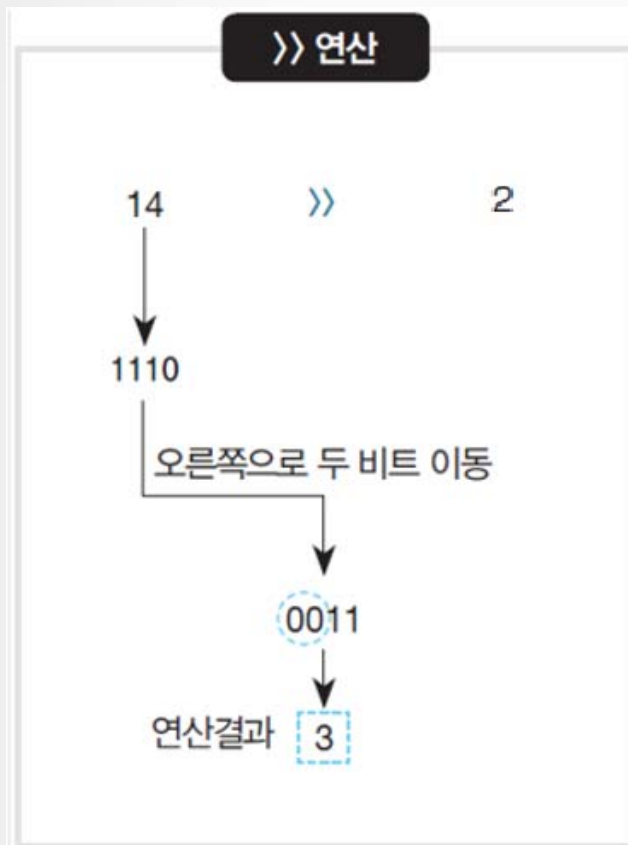
비트 연산자

- 비트 연산자
 - 피연산자 정수값을 비트 단위로 논리 연산을 수행하는 연산자
 - AND, OR, NOT의 기본 논리식으로 이루어짐
- 비트 연산자의 종류와 사용 예

연산자	의미	사용 방법	설명
&	비트 곱(AND)	$a = b \& c$	b와 c를 AND 연산하여 a에 대입한다.
	비트 합(OR)	$a = b c$	b와 c를 OR 연산하여 a에 대입한다.
^	배타적 논리합(XOR)	$a = b \wedge c$	b와 c를 XOR 연산하여 a에 대입한다.
~	비트 반전(1의 보수)	$a = \sim b$	b의 각 비트를 반전하여 a에 대입한다.
<<	왼쪽으로 이동(shift)	$a \ll b$	a를 b만큼 왼쪽으로 비트 이동한다.
>>	오른쪽으로 이동(shift)	$a \gg b$	a를 b만큼 오른쪽으로 비트 이동한다.

비트 연산자

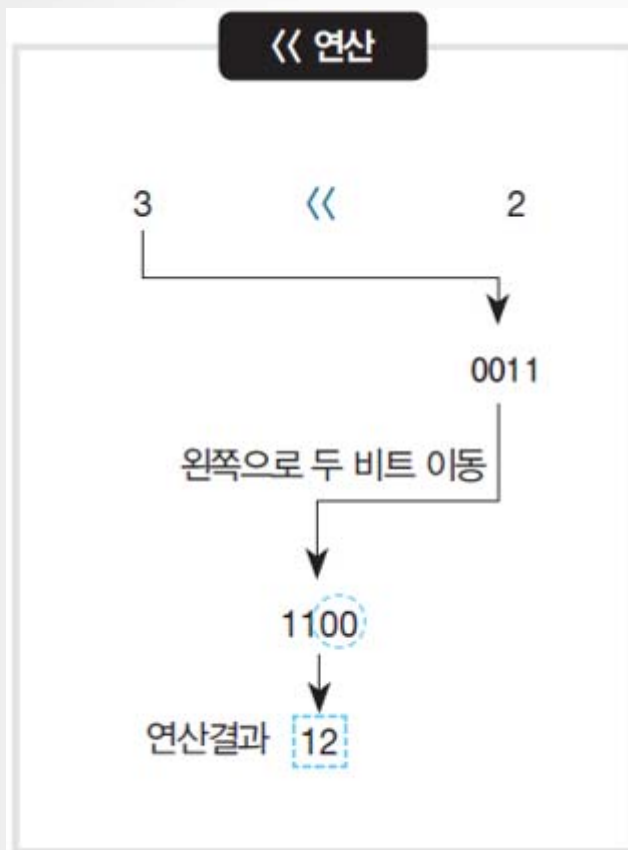
- shift 연산
 - 비트들을 오른쪽이나 왼쪽으로 이동 시킴



- \gg 연산
 - $a \gg b$ 연산의 의미
 - a를 b만큼 오른쪽으로 비트를 이동 시킴
 - $a \gg b$ 연산의 값
 - a를 2^b 으로 나누기한 값($a/2^b$)
 - $14 \gg 2$
 - $14/2^2$ (정수형이므로 소수점은 없 어지고 3이 됨)

비트 연산자

- shift 연산
 - 비트들을 오른쪽이나 왼쪽으로 이동 시킴



- \ll 연산
 - $a \ll b$ 연산의 의미
 - a를 b만큼 왼쪽으로 비트를 이동시킴
 - $a \ll b$ 연산의 값
 - a에 2^b 을 곱한 값($a * 2^b$)
 - $3 \ll 2$
 - $3 * 2^2$ (12가 됨)

비트 연산자

- 예제) 비트 논리 연산자의 사용 예제

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int a, b, c, d;
06     int x = 10, y = 7;
07
08     a = x & y;
09     b = x | y;
10     c = x ^ y;
11     d = ~x;
12
13     printf("비트곱 x & y의 결과 : %d\n", a);
14     printf("비트합 x | y의 결과 : %d\n", b);
15     printf("배타적 논리합 x ^ y의 결과 : %d\n", c);
16     printf("비트 반전 ~x의 결과 : %d\n", d);
17
18     return 0;
19 }
```

축약 연산자

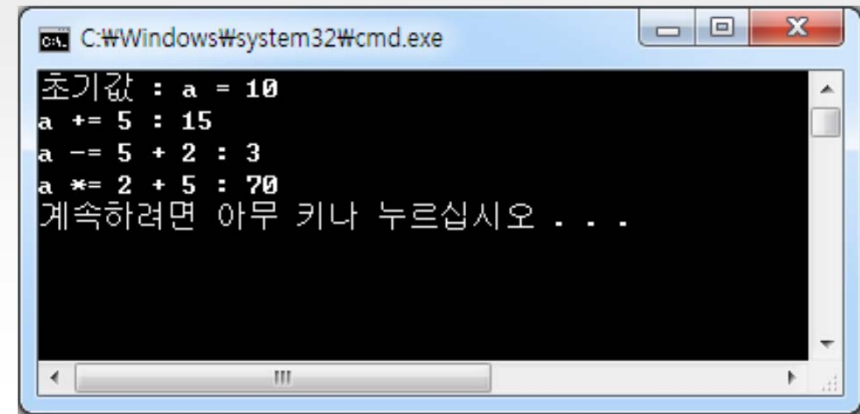
- 축약 연산자
 - 연산 2개를 동시에 수행하여 값을 할당하는 연산자
- 축약 연산자의 종류와 사용 예

연산자	의미	사용 방법
<code>+=</code>	a와 b를 더해서 a에 대입한다.	<code>a += b</code>
<code>-=</code>	a에서 b를 뺀 결과 값을 a에 대입한다.	<code>a -= b</code>
<code>*=</code>	a와 b를 곱해서 a에 대입한다.	<code>a *= b</code>
<code>/=</code>	a를 b로 나눈 몫을 a에 대입한다.	<code>a /= b</code>
<code>%=</code>	a를 b로 나눈 나머지를 a에 대입한다.	<code>a %= b</code>
<code><<=</code>	a를 b비트만큼 좌측으로 이동시켜 a에 대입한다.	<code>a <<= b</code>
<code>>>=</code>	a를 b비트만큼 우측으로 이동시켜 a에 대입한다.	<code>a >>= b</code>
<code>&=</code>	a와 b를 비트별로 AND 연산하여 a에 대입한다.	<code>a &= b</code>
<code> =</code>	a와 b를 비트별로 OR 연산하여 a에 대입한다.	<code>a = b</code>
<code>^=</code>	a와 b를 비트별로 XOR 연산하여 a에 대입한다.	<code>a ^= b</code>

축약 연산자

- 예제) 축약 연산을 활용한 예제

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     int a = 10;
06     printf("초기값 : a = 10\n");
07
08     a += 5;
09     printf("a += 5 : %d\n", a);
10
11     a = 10; a -= 5 + 2;
12     printf("a -= 5 + 2 : %d\n", a);
13
14     a = 10; a *= 2 + 5;
15     printf("a *= 2 + 5 : %d\n", a);
16     return 0;
17 }
```



```
C:\Windows\system32\cmd.exe
초기값 : a = 10
a += 5 : 15
a -= 5 + 2 : 3
a *= 2 + 5 : 70
계속하려면 아무 키나 누르십시오 . . .
```

연산자 우선순위

- 연산자의 분류에 따른 우선순위
 - 단항 > 산술 > 이동 > 관계 > 비트 > 논리 > 조건 > 대입
- 연산자 우선순위의 원칙
 - () 안의 내용이 최우선순위
 - 단항 연산자가 이항 연산자보다 먼저 처리됨
 - 위치에 따라 처리되는 순서가 다르지만 대개 좌측에서 우측으로 실행

혼동된다면, 먼저 처리하려는 연산에 ()!!

연산자 우선순위

- 연산자 우선순위

우선순위	연산자 종류		연산자	결합성
고 ↑ ↓ 저	식, 구조체, 공용체		()[], ->	좌→우
	단항 연산자		! ~ - + ++ -- & *	좌→우
	이항 연산자	승제	*/%	좌→우
		가감	+ -	
		이동	<< >>	
		비교	< <= > >=	
		등가	== !=	
		비트 AND	&	
		비트 XOR	^	좌→우
		비트 OR		
		논리 AND	&&	
		논리 OR		
	조건 연산자		?:	좌→우
	대입 연산자		= += -= *= %= <<= >>= &=	좌→우
	coma 연산자		,	좌→우