

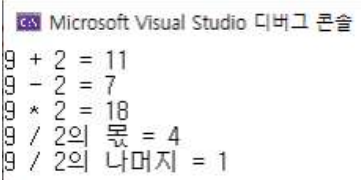
## 6. C언어의 다양한 연산자 소개

### ■ 대입연산자(=)와 산술연산자(+,-,\*,/,%)

연산자	연산자의 기능	결합방향
=	연산자 오른쪽에 있는 값을 연산자 왼쪽에 있는 변수에 대입한다. 예) num = 20;	←
+	두 피연산자의 값을 더한다. 예) num = 4 + 3;	→
-	왼쪽의 피연산자 값에서 오른쪽 피연산자 값을 뺀다. 예) num = 4 - 3;	→
*	두 피연산자의 값을 곱한다. 예) num = 4 * 3;	→
/	왼쪽의 피연산자 값에서 오른쪽 피연산자 값으로 나눈다. 예) num = 4 / 3;	→
%	왼쪽의 피연산자 값을 오른쪽 피연산자 값으로 나눴을 때 얻게 되는 나머지를 반환한다. 예) num = 7 % 3;	→

\*두 개의 피연산자(연산의 대상)를 필요로 하는 연산자

#### ● 실습 예제

	<pre>// 왼쪽과 같이 실행결과를 위해서 아래 프로그램 main() 함수를 작성해보세요!! #include &lt;stdio.h&gt; int main() {     int num1 = 9, num2 = 2; // 임의의 2개의 숫자로 초기화     printf("%d + %d = %d\n", num1, num2, num1 + num2);      return 0; }</pre>
--	---

### ■ 복합대입연산자

#### ● 복합대입연산자 : 다른 연산자와 합쳐진 형태의 대입 연산자

복합 대입 연산자	동일한 연산
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

<pre>// 예제 #include &lt;stdio.h&gt; int main() {     int num1 = 2, num2 = 4, num3 = 3;     num1 += 3;     num2 *= num1;     num2 %= num3;     printf("Result : %d, %d\n", num1, num2);     return 0; }</pre>	<div>실행결과</div>
--	-----------------

## ■ 부호연산의 의미를 갖는 + 연산자와 -연산자

- +, - 연산자는 피연산자가 하나인 단항연산자로서 부호를 뜻하기도 함.
- +3, -7과 같이 숫자앞에 붙는 부호를 뜻하는 것임.

```
// 예제
#include <stdio.h>
int main()
{
    int num1 = +2;
    int num2 = -4;

    num1 = -num1;
    printf("num1 : %d\n", num1);
    num2 = -num2;
    printf("num2 : %d\n", num2);

    return 0;
}
```

실행결과

## ■ 증가, 감소 연산자

- 변수에 저장된 값을 1 증가 및 감소시키는 경우에 사용되는 연산자
- 단항 연산자로서 활용의 빈도가 높으므로 확실히 이해하고 넘어가자!

연산자	연산자의 기능
++num	값을 1 증가 후, 속한 문장의 나머지를 진행(선 증가, 후 연산) 예) var = ++num;
num++	속한 문장을 먼저 진행한 후, 값을 1 증가(선 연산, 후 증가) 예) var = num++;
--num	값을 1 감소 후, 속한 문장의 나머지를 진행(선 감소, 후 연산) 예) var = --num;
num--	속한 문장을 먼저 진행한 후, 값을 1 감소(선 연산, 후 감소) 예) var = num--;

```
// 예제4
#include<stdio.h>
int main()
{
    int num1 = 10, num2 = 10;

    printf("num1 : %d\n", num1);
    printf("num1++ : %d\n", num1++); // 후위증가
    printf("num1 : %d\n", num1);
    printf("num2 : %d\n", num2);
    printf("++num2 : %d\n", ++num2); // 전위증가
    printf("num2 : %d\n", num2);

    return 0;
}
```

- 다음 문장에서 printf()의 출력값은?

```
int num1 = 10;
int num2 = num1--+2;

printf("num1 : %d, num2 : %d\n", num1, num2);
```

## ■ 관계연산자

- 관계연산자 : 대소와 동등의 관계를 따지는 연산자, 비교 연산자라고도 함!
- 종류 : <, >, ==, !=, <=, >=

연산자	연산자의 기능
<	예) n1 < n2 n1 이 n2보다 작은가?
>	예) n1 > n2 n1 이 n2보다 큰가?
==	예) n1 == n2 n1 과 n2가 같은가?
!=	예) n1 != n2 n1 과 n2가 다른가?
<=	예) n1 <= n2 n1 이 n2보다 같거나 작은가?
>=	예) n1 >= n2 n1 이 n2보다 같거나 큰가?

- 관계연산자들은 조건을 만족하면 1을 만족하지 않으면 0을 반환함
- 1은 참(true), 0은 거짓(false)을 의미하는 대표적인 숫자임.(엄밀하게는 0을 제외하면 모두 참)
- 따라서, 조건을 만족하면 참(true)을 만족하지 않으면 거짓(false)을 반환함.

```
// 예제
#include <stdio.h>
int main()
{
    int num1 = 10;
    int num2 = 12;
    int result1, result2, result3,
        result4, result5, result6;

    result1 = (num1 == 10);
    result2 = (num1 <= 9);
    result3 = (12 >= num2);
    result4 = (num1 != num2);
    result5 = (num1 > num2);
    result6 = (12 < num2);

    printf( " result1: %d\n result2: %d\n result3: %d\n"
           result4: %d\n result5: %d\n result6: %d\n",
           result1, result2, result3, result4, result5, result6);

    return 0;
}
```

**실행결과**  
result1:  
result2:  
result3:  
result4:  
result5:  
result6:

## ■ 논리연산자

- 두 개 이상의 조건을 조합하여 논리적인 결과를 확인할 수 있는 연산자
- 종류 : &&, ||, !

연산자	연산자의 기능	
&&	예) A && B A와 B <b>모두</b> '참'이면 연산결과로 '참'을 반환(논리 AND)	0 && 0 => 0 0 && 1 => 0 1 && 0 => 0 1 && 1 => 1
	예) A    B A와 B <b>둘 중 하나라도</b> '참'이면 연산결과로 '참'을 반환(논리 OR)	0    0 => 0 0    1 => 1 1    0 => 1 1    1 => 1
!	예) !A A가 '참'이면 '거짓', A가 '거짓'이면 '참'을 반환(논리 NOT)	!0 => 1 !1 => 0 !2 => 0

// 예제

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

```
int main()
```

```
{
```

```
    int num1 = 10;
```

```
    int num2 = 12;
```

```
    int result1, result2, result3, result4;
```

```
    result1 = (num1 == 10 && num1 <= 9);
```

```
    result2 = (12 >= num2 || num1 != num2);
```

```
    result3 = !(num1 > num2);
```

```
    result4 = !num2;
```

```
    printf(" result1: %d\n result2: %d\n result3: %d\n result4: %d\n",  
           result1, result2, result3, result4);
```

```
    return 0;
```

```
}
```

실행결과

result1:

result2:

result3:

result4:

## ■ 콤마연산자

- 콤마연산자 : 둘 이상의 변수를 동시에 선언하거나, 둘 이상의 문장을 한 행(line)에 삽입하는 경우에 사용되는 연산자임. 또한 둘 이상의 매개변수를 함수에 전달할 때도 매개변수의 구분을 목적으로 사용된다. 연산의 결과가 아닌 '구분'을 목적으로 사용!

// 예제

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

```
int main()
```

```
{
```

```
    int num1 = 1, num2 = 2;
```

```
    printf("학번 : 1234,", printf(" 이름 : 홍길동!\n"));
```

```
    num1++, num2++;
```

```
    printf("%d ", num1), printf("%d ", num2), printf("\n");
```

```
    return 0;
```

```
}
```

실행결과

Microsoft Visual Studio 디버그 콘솔

학번 : 1234, 이름 : 홍길동!  
2 3

## ■ 연산자의 우선순위와 결합방향

- 우선순위가 동일한 두 연산자가 하나의 수식에 존재하는 경우, 어떠한 순서대로 연산하느냐를 결정해 놓은 것이 결합법칙

우선순위	연산자	설명	결합법칙
1	x++ x--	후위 증가 후위 감소	→
	()	함수 호출	
	[]	배열 첨자, 인덱스	
	.	간접지정, 참조에 의한 요소 선택	
	->	직접지정, 포인터를 통해 요소 선택	
2	++x --x	전위 증가 전위 감소	←
	+x, -x	단항 부호연산(양수, 음수의 표현)	
	!	논리적 NOT	
	~	비트단위 NOT	
	(자료형)	자료형 변환	
	*x	간접지정 연산	
	&x	주소 연산	
	sizeof	바이트 단위 크기 계산	
3	* / %	곱셈 나눗셈 나머지	→
4	+ -	덧셈 뺄셈	
5	<< >>	비트 시프트	
6	< > <= >=	대소비교	
7	== !=	동등비교	
8	&	비트 AND	
9	^	비트 XOR (배타적 or)	
10		비트 OR (포함적 or)	
11	&&	논리 AND	
12		논리 OR	
13	?:	3항 연산자 조건부 (참조)	←
14	=	대입연산	
	+= -= *= /= %= <<= >>= &=  = ^=	복합대입연산	
15	,	coma연산	→