KG 아이티뱅크

C 언어

C LANGUAGE

연산자

연산자(operator)

❖ **연산자 : 피연산자에 대해 수행할 작업을 나타내는 기호**

① **산술 연산자** : 수학적 계산을 하기 위한 연산자

② **대입 연산자** : 값의 저장 등을 위한 연산자

③ **증감 연산자** : 1씩 증가, 감소시키는 연산자

④ **관계 연산자** : 비교하여 참과 거짓을 돌려주는 연산자

⑤ **논리 연산자** : 참과 거짓 간의 연산에 사용되는 연산자

⑥ **조건 연산자** : 참과 거짓에 따른 분기를 제공하는 연산자

⑦ **비트 연산자** : 비트 단위 연산에 사용되는 연산자

⑧ **기타 연산자** : 콤마(,)와 sizeof( )

❖ **이해해야 사용할 수 있고, 그리고 나서 외워야 함**

➢ 각 연산자는 어떤 용도, 어떤 역할에 사용되나?

➢ 왜 이걸 써야 하나?

연산자(operator)

❖ **산술 연산자 : 산술연산에 사용되는 연산자**

- 기본적으로 더 큰 자료형에 맞춰서 자동 형변환이 진행됨
- 정수와 정수간의 연산은 정수
- 정수와 실수간의 연산은 실수
- 실수와 실수간의 연산은 실수

산술 연산자	사용 예	의미
+	a + b	두 수의 **합**
-	a - b	두 수의 **차**
*	a * b	두 수의 **곱**
/	a / b	나누기 **몫**
%	a % b	나누기 **나머지**

%는 정수간의 연산만 지원하는 특징을 가짐

연산자(operator)

❖ **형변환(casting) : 자료형을 바꾸는 것**

➢ 값의 **손실 / 변환**을 통해 결과를 유도하는 방법

① 암묵적(**자동**) 형변환 : 컴파일러가 **자동으로 처리**

- 주로 **연산**이나 **변수에 값을 저장하는 과정**에서 발생

② 명시적(**강제**) 형변환 : 사용자가 **직접 변환**

- 필요한 상황에서 한 순간만 바꿔서 취급
- (자료형)변수 형태로 사용

❖ **예시**

```
double num = 3;
```

자동 형변환(영구)
정수 3 -> 실수 3.0

```
printf("%d₩n", (int)num);
```

강제 형변환(일시적)
실수 3.0 -> 정수 3

표준출력함수

< 파일이름 : operEX1.c >

실습문제1. 아래의 조건을 만족하는 코드를 작성하세요.

조건

실수형 변수를 선언하고, 값은 66.15로 초기화합니다.
1. 초기화가 완료된 변수를 출력합니다.
2. 정수형으로 강제형변환한 결과를 출력합니다.
3. 문자형으로 자동형변환한 결과를 출력합니다.

결과

```
최초 실수값     : 66.15
강제형변환      : 66
자동형변환      : B
```

표준출력함수

< 파일이름 : operEX2.c >

실습문제2. 아래의 조건을 만족하는 코드를 작성하세요.

조건

정수형 변수 두개를 선언하고 각각
1717, 19로 초기화합니다.
두 변수를 이용해 산술연산 결과를 출력하세요.

결과

```
(값1) + (값2) = (결과1)
(값1) - (값2) = (결과2)
(값1) * (값2) = (결과3)
(값1) / (값2) = (결과4) # 몫 연산
(값1) / (값2) = (결과5) # 소수점 연산
(값1) % (값2) = (결과6)
```

연산자(operator)

❖ **대입 연산자 : 값의 저장 및 갱신에 사용되는 연산자**

- 연산도 함께 할 수 있는 복합대입 연산자도 있음
- 복합대입 연산자는 초기화 용도로 사용불가

대입 연산자	사용 예	의미
=	a = b	a에 b를 대입

복합대입 연산자	사용 예	의미
+=	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

연산자(operator)

❖ **증감 연산자 : 하나씩 감소, 증가시키는 경우에 사용**

- 단항 연산자이며, 앞에 붙이거나 뒤에 붙일 수 있음
- 전치를 하느냐 후치를 하느냐에 따라 연산 결과가 다름

증감 연산자	사용 예	의미
++	++a	1증가 후 연산
++	a++	연산 후 1증가
--	--a	1감소 후 연산
--	a--	연산 후 감소

연산자(operator)

❖ **관계 연산자 : 좌측 값 기준으로 두 값의 관계를 비교**

- 연산을 하면 결과로써 **참(1)**과 **거짓(0)**을 돌려줌
- 돌려준 결과의 자료형은 정수형 자료형을 가짐

비교 연산자	사용 예	의미
<	a < b	a는 b보다 작다
>	a > b	a는 b보다 크다
<=	a <= b	a는 b보다 작거나 같다
>=	a >= b	a는 b보다 크거나 같다
==	a == b	a는 b와 같다
!=	a != b	a는 b와 다르다

연산자(operator)

❖ 논리 연산자 : 참과 거짓을 연산하는 연산자

비교 연산자	사용 예	의미
||	A || B	논리**합**(OR)
&&	A && B	논리**곱**(AND)
!	! A	부정(NOT)

변수 값		결과	
A	B	||(논리**합**)	&&(논리**곱**)
1	1	1	1
1	0	1	0
0	1	1	0
0	0	0	0

연산자(operator)

❖ **조건 연산자 : 참과 거짓에 따른 분기를 제공하는 연산자**

- 제어문을 극단적으로 축약시킨 형태이며 단순화시킴
- 각 조건에서 실행되는 코드에 한계가 있음

구조 : (조건식) **?** 참일 때 값 **:** 거짓일 때 값;

```c
#include <stdio.h>

int main(){
    int a_true=1;    //참
    int b_false=0;   //거짓
    int c = ( a_true > b_false ) ? 8 : 4; //연산 진행
    printf( "%d", c );
    return 0;
}
```

연산자(operator)

❖ **비트 연산자 : 비트 단위 연산에 사용되는 연산자**

- 비트단위 별 논리연산 / 이동에 사용
- 10진수를 2진수로 자동으로 변환하여 처리함
- 주로 하드웨어에 대한 직접 접근 / 제어 용도로 사용됨

비트 연산자	사용 예	의미
|	a | b	한쪽 비트라도 1일 때 1 (OR)
&	a & b	양쪽 비트가 1일 때 1 (AND)
^	a ^ b	한쪽 비트만 1일 때 1 (XOR)
~	~a	비트가 1이면 0, 0이면 1 (NOT)
>>	a >> 2	우측으로 2칸 이동
<<	a << 2	좌측으로 2칸 이동

연산자(operator)

❖ **기타 연산자 : 콤마(,)와 sizeof( ) 연산자**

① **콤마(,)**

- 연산식들을 콤마로 구분하여 좌측부터 실행
- 가장 우측식이 연산 결과가 됨
- 우선순위가 가장 낮은 이항 연산자

② **sizeof( )**

- 입력한 내용 또는 변수의 크기를 **바이트 단위**로 표기
- 크기를 직접 확인하는데 사용
- 함수가 아닌 연산자

연산자(operator)

높음 ⟶ 낮음		우선순위
종류	연산자	높음 ↓ 낮음
	( ), [ ], ->, 마침표(.)	
단항	**sizeof, (type), &, *, -, +, --, ++, ~, !**	
산술	*, /, %, +, -	
비트	<<, >>	
비교	<, <=, >, >=, ==, !=	
비트	&, ^, |	
논리	&&, ||	
삼항	? :	
대입	%=, /=, *=, -=, +=, =	
콤마	콤마(,)	