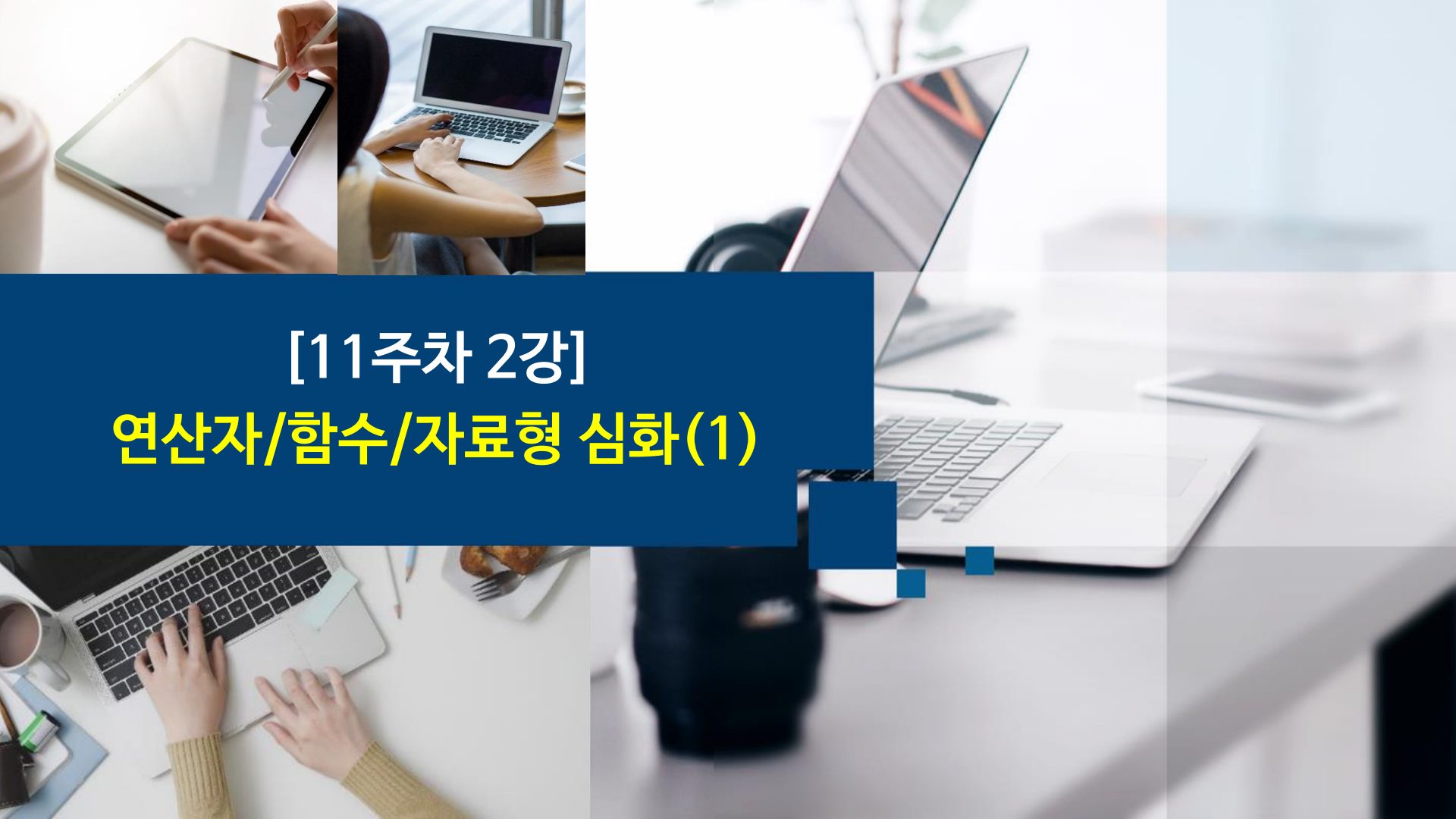

[11주차 2강]
연산자/함수/자료형 심화(1)

학습 내용

- 13.1 비트연산자

학습 목표

- 13.1 비트연산자를 익히고 활용할 수 있다.

C프로그래밍및실습

1. 비트연산자
2. 재귀함수
3. 라이브러리 활용

1. 비트연산자

2진수와 16진수

- 컴퓨터의 기본 정보단위는 bit와 byte로 표현
- 2진수, 16진수 표기가 편리한 경우가 있음
  - ✓ 1 bit → 2진수 한 자리
  - ✓ 1 byte → 16진수 두 자리

- 2진수 :

0	1	1	0	= 2^2x1 + 2^1x1 $_{(10)}$
↑	↑	↑	↑	= 6 $_{(10)}$
2^3	2^2	2^1	2^0	

- 16진수 :

0	0	A	F	= 16^1x10 + 15 $_{(10)}$
↑	↑	↑	↑	= 175 $_{(10)}$
16^3	16^2	16^1	16^0	

1. 비트연산자

2진수와 16진수의 관계

- 2진수 4자리 = 16진수 1자리

  $1010\ \ 1111_{(2)} = AF_{(16)}$

2진수	16진수	10진수
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7

2진수	16진수	10진수
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

1. 비트연산자

비트 연산이란?

- **비트 단위**로 처리되는 연산
- 비트 1은 참, 비트 0은 거짓을 의미
- 예) 두 이진수의 비트 단위 논리곱(bitwise AND)
  - ✓ 해당 자리의 비트가 모두 1인 경우에만 참

```
          0010 1010
  AND     1010 1101
          ---------
(결과)    0010 1000
```

- 비트 연산의 종류
  - ✓ 비트 단위 논리 연산
  - ✓ 비트 단위 이동 연산

1. 비트연산자

비트 단위 논리 연산

- 예) 비트 AND 연산자 : x & y
  - ✓ 두 정수 x와 y의 비트단위 논리곱
- 비트 연산을 할 때는 16진수로 표현하는 것이 이해하기 쉬움

```
unsigned int x, y, z;

x = 0X2A;          // x = 0000 0000 … 0010 1010
y = 0XAD;          // y = 0000 0000 … 1010 1101
z = x & y;         // z = 0000 0000 … 0010 1000

printf("%#X", z);  // #: 16진수임을 나타내는
                   // '0X'를 앞에 출력
결과:
0X28               ➔ 16진수 28을 의미
```

1. 비트연산자

C언어 비트 연산자 (논리 연산)

연산자	연산자 기능	예시
x & y	x와 y의 비트 단위 AND 연산	0110 1100 & 0100 1010 → 0100 1000
x ¦ y	x와 y의 비트 단위 OR 연산	0110 1100 ¦ 0100 1010 → 0110 1110
x ^ y	x와 y의 비트 단위 XOR 연산	0110 1100 ^ 0100 1010 → 0010 0110
~ x	x에 대한 비트 단위 NOT 연산	~ 0110 1100 → 1001 0011

1. 비트연산자

비트 단위 이동 연산

- 예) 왼쪽 이동 연산자 : x << k
  - ✓ x를 비트 단위로 왼쪽으로 k 만큼 이동
  - ✓ 오른쪽 빈 자리는 k개의 0으로 채움

```
unsigned int x, z;

x = 0X2A01234C;        // x = 0010 1010 … 0000 1100
z = x << 4;            // z = 1010 0000 … 1100 0000
                               (왼쪽으로 4칸 이동)
printf("%#X", z);

결과:
0XA01234C0
```

1. 비트연산자

C언어 비트 연산자 (이동 연산)

연산자	연산자 기능	예시
x << n	X를 n 비트 **왼쪽 이동**	0100 1010 << 2 → 0010 1000
x >> n	X를 n 비트 **오른쪽 이동**	0100 1010 >> 2 → 0001 0010

※ 실습하기

[예제 13.1] 다음을 작성하여 앞 슬라이드의 예시를 확인해보자.

① **unsigned char** 형 변수 3개 선언 (8비트 수 표현)

② 두 개의 변수에 **0110 1100**과 **0100 1010**을 16진수로 표현하여 대입

③ 각 논리 연산에 대해, 연산 결과를 나머지 하나의 변수에 대입하고 16진수로 출력

비트 연산의 피연산자

- 피 연산자는 정수형(char, int, long 등) 에 대해서만 연산 가능
- 특별한 이유가 없으면 unsigned 정수 사용
  - ✓ 이동 연산에서 빈 자리는 0으로 채우는 것이 기본이지만, signed 정수는 1로 채워지는 경우도 있음 (표준은 아님)

```
signed int xs, zs;
unsigned int xu, zu;

xs = xu = 0XA1234567;       // xs = xu = 1010 0001 …
zs = xs >> 4;               // zs = 1111 1010 0001 …
zu = xu >> 4;               // zu = 0000 1010 0001 …
printf("signed = %#X, unsigned = %#X\n", zs, zu);

결과:
signed = 0XFA123456, unsigned = 0XA123456
```

※ 실습하기

[예제 13.2] 어떤 정수 N을 이진수로 표기했을 때, 오른쪽에서 10번째 자리의 비트 값을 출력 (비트 연산의 응용)

✓ 가장 오른쪽 비트를 0번째 비트라고 가정

```
unsigned int x = 0X71234567;      // … 0100 0101 0110 0111

x = x >> 10;          // 오른쪽에서 10번째 비트를
                      // 가장 오른쪽에 위치시킴
x = x & 1;            // x와 … 0001 의 AND
                      // 두 문장을 하나로 합치면 (x >> 10) & 1

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

결과:
1                  ➔ 0000 … 0001
```

학습 정리

- **비트 연산**이란 비트 단위로 처리하는 연산
- 비트 연산에는 **비트 단위 논리 연산**과 **비트 단위 이동 연산**이 있음
- 비트 연산의 비연산자는 정수형만 가능하고, 특별한 이유가 없으면 unsigned 를 권장