

Structure, Union, Enumeration

Section 1 구조체 (Structure)

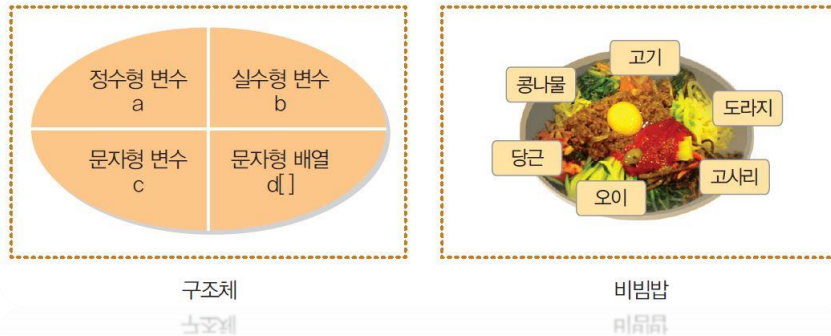
Section 2 공용체 (Union)

Section 3 열거형 (Enumeration)



구조체(Structure)와 배열

● 구조체와 비빔밥의 비교



- 비빔밥 : 별도의 재료가 하나의 그릇에 담겨 있다.
- 구조체 : 서로 다른 여러 가지 변수 형태를 하나의 블록으로 묶는다.

● 구조체와 배열

- 배열: 같은 데이터형을 여러 개 묶어서 처리하는 자료형
- 구조체: 다른 데이터형을 여러 개 묶어서 처리할 수 있는 자료형

구조체를 만들고 사용하기

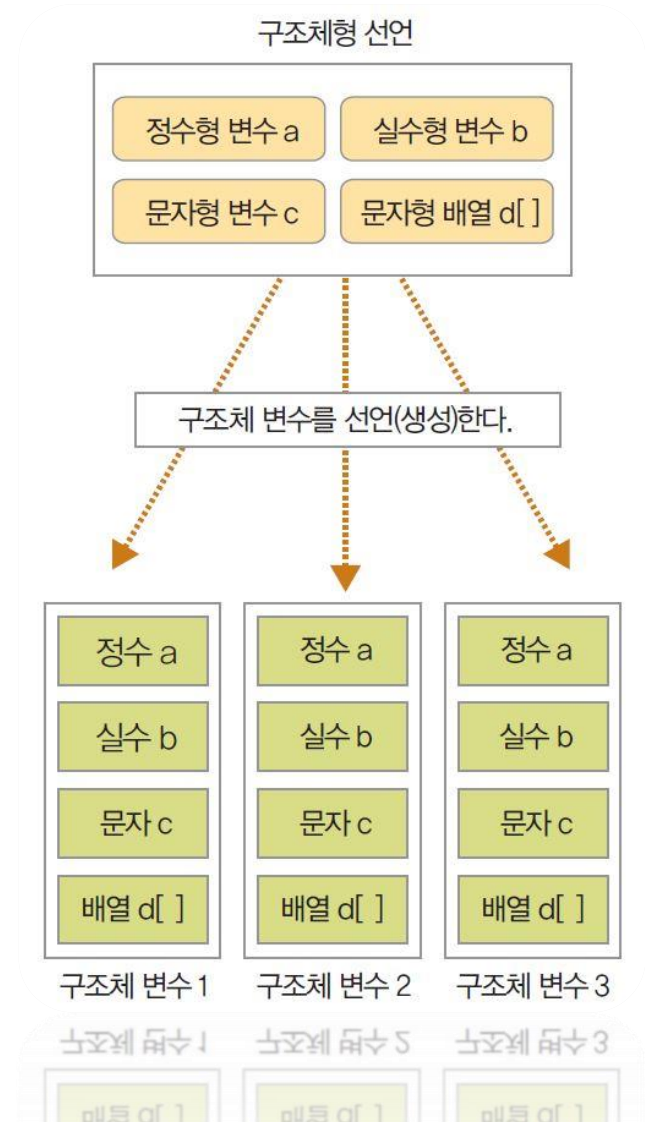
● 구조체 형식의 선언

- 구조체 형태를 먼저 선언한다
- 어떤 변수들이 구조체에 들어가고 이름이 무엇인지

● 구조체 변수의 선언

- 정의된 구조체 형식으로 메모리에 공간을 확보

```
struct bibim {
    int a;
    char b;
    float c;
    char d[5];
};
struct bibim b1;
struct bibim b2;
struct bibim b3;
```



구조체를 만들고 사용하기

● 구조체 형식의 선언

- 구조체 형태를 먼저 선언한다
- 어떤 변수들이 구조체에 들어가고 이름이 무엇인지

● 구조체 변수의 선언

- 정의된 구조체 형식으로 메모리에 공간을 확보

```
struct 구조체형_이름 {  
    데이터_형식  멤버_변수_1;  
    데이터_형식  멤버_변수_2;  
    :  
};  
struct 구조체형_이름 구조체_변수;
```

↓
필수

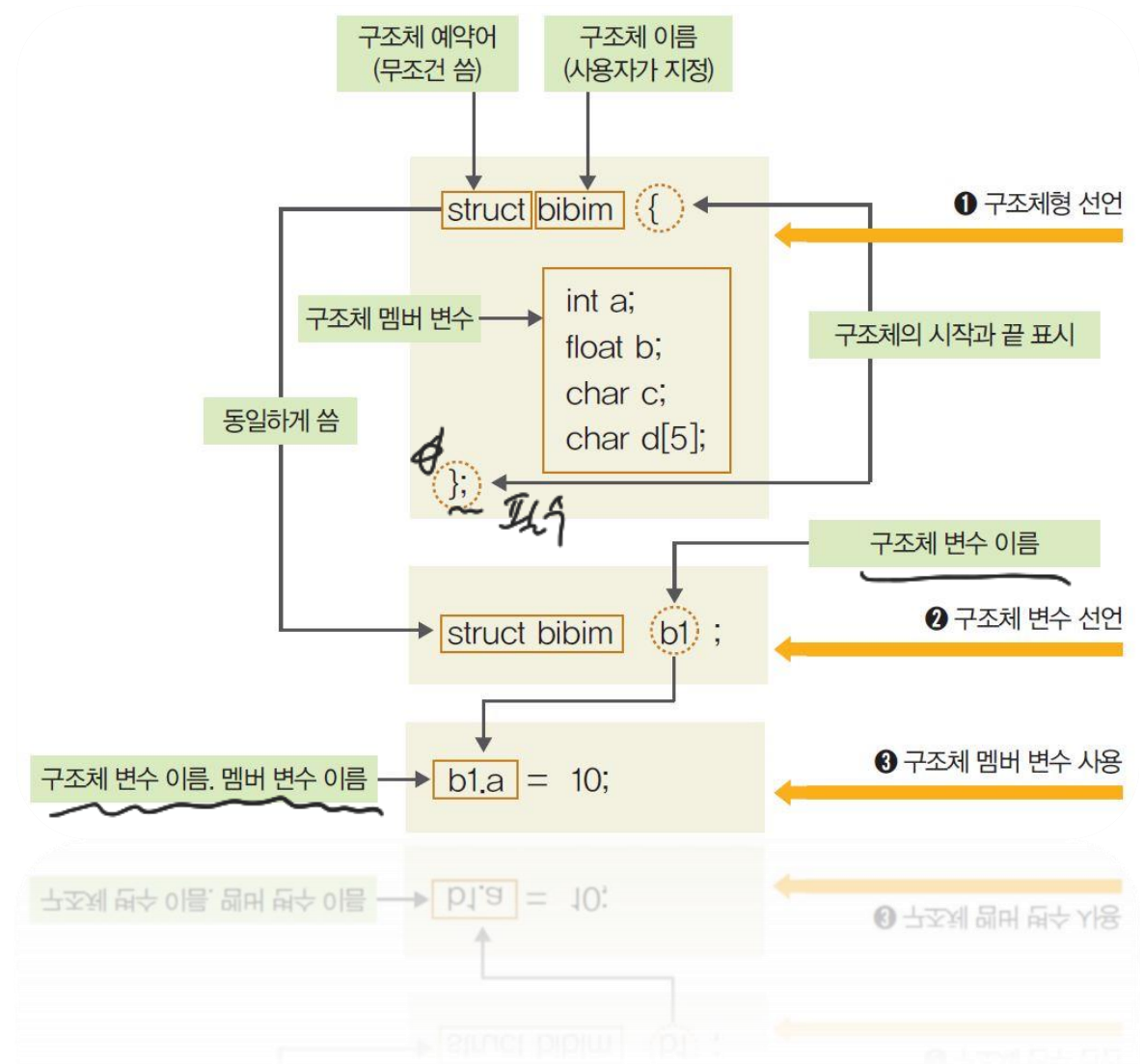
● 구조체 변수 선언시의 유의사항

- bibim이라는 자료형은 존재하지 않는다.
- 따라서, struct임을 bibim 앞에 명시해야 함.

- struct bibim b1;
키워드 변수명

구조체 사용의 문법

- 구조체 자료형의 선언
- 구조체 변수의 선언
- 구조체 변수의 멤버 변수 접근
 - ex) . 연산자: b1.a



구조체의 메모리 할당 구조

● 구조체 변수 선언시의 유의사항

- 구조체형의 마지막은 반드시 "};"로 끝나야 함.
- 정의된 구조체의 멤버 변수 순서대로 구조체 변수가 메모리에 만들어짐
- 구조체 변수는 '구조체 변수 이름.멤버 변수 이름' 형태로 쓰고 일반 변수처럼 사용함

● 메모리에 만들어진 구조체의 모양

- 예제: 잘 듣고 잘 기억하자



구조체 변수의 초기화

● 일반 변수와 구조체 변수의 차이

❶ 일반 변수의 초기값 대입

```
char name[10] = "Woo";
int kor = 90;
int eng = 80;
float avg;
```



❷ 구조체 변수의 초기값 대입

```
struct student {
    char name[10];
    int kor;
    int eng;
    float avg;
};
```

```
struct student s = {"Woo", 90, 80};
```

```
struct student s = {"Woo", 90, 80};
```

구조체 선언의 다른 방법

- 구조체 자료형의 선언과 동시에 변수 선언

int {
char _____

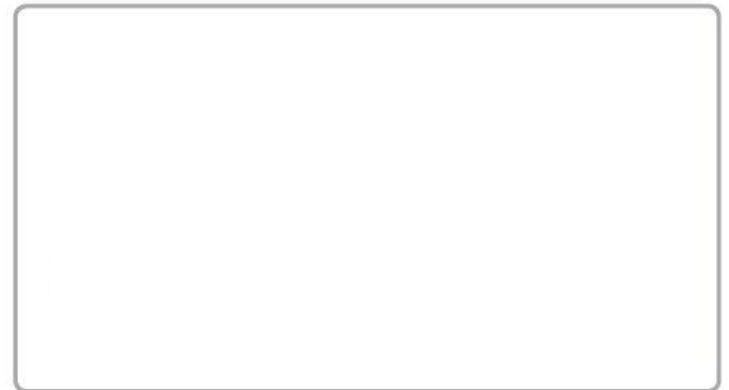
```
struct student {
    char name[10];
    int kor;
    int eng;
    float avg;
};
```

변수선언

- Typedef를 이용한 자료형의 단축을 이용한다.

```
struct 구조체형_이름 {
    데이터_형식 멤버_변수_1;
    데이터_형식 멤버_변수_2;
    :
};
struct 구조체형_이름 구조체_변수;
```

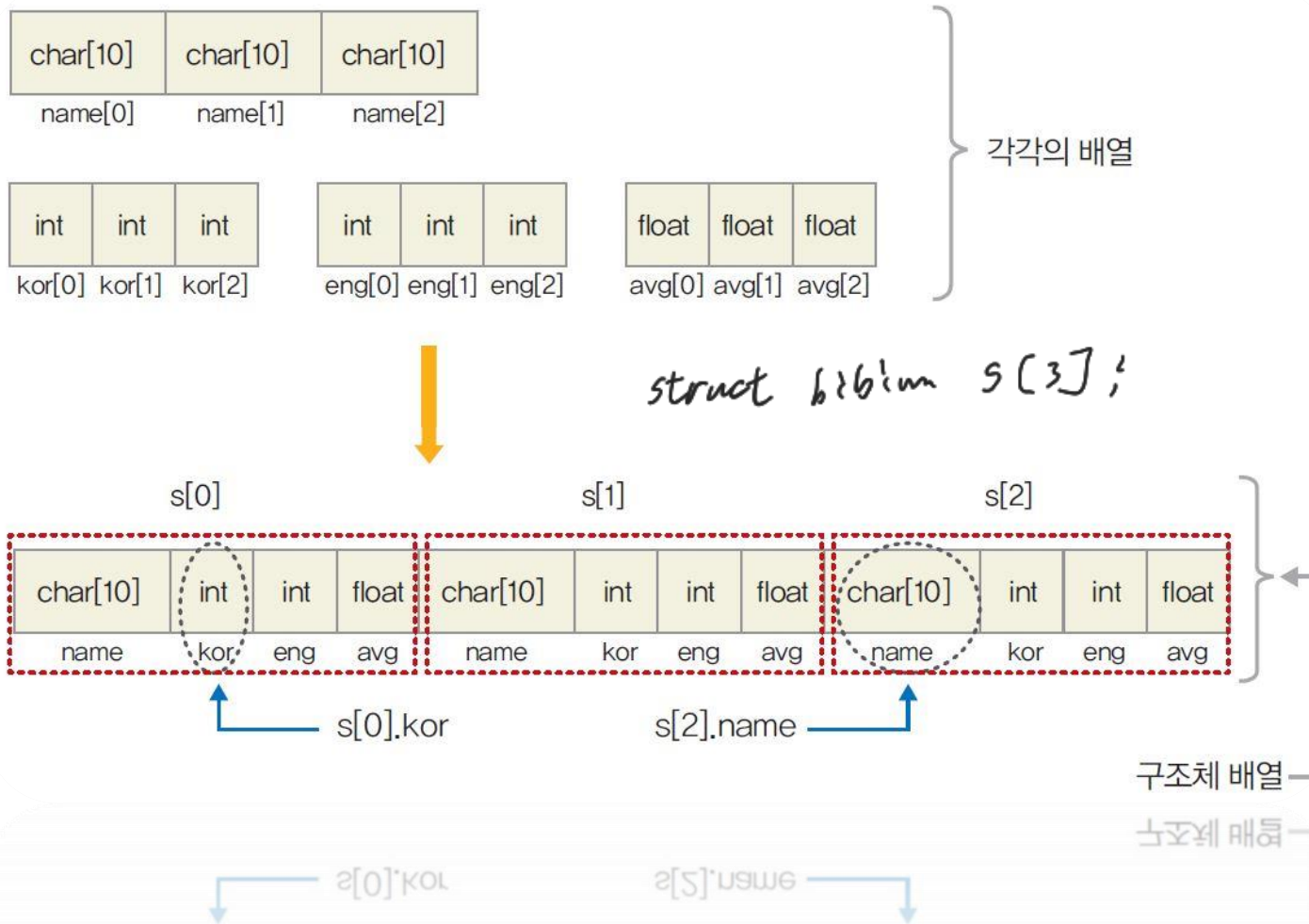
이름을 다시 제정의



1 구조체

구조체의 배열

- 구조체 하나가 하나의 변수
- 구조체 배열은 구조체 여러 개를 한 번에 나열한 것



구조체의 포인터

● 구조체 포인터

- 구조체 포인터 변수도 구조체의 주소를 갖는다.



● 일반 포인터 변수와 구조체 포인터 변수의 차이

- 포인터로 멤버 변수 접근 시
- (*p). 연산을 사용하지 않음
- p-> 의 연산으로 직접 멤버에 접근

① 일반 포인터 변수 사용

```
int a;
int* p;
p = &a;
*p = 100;
```

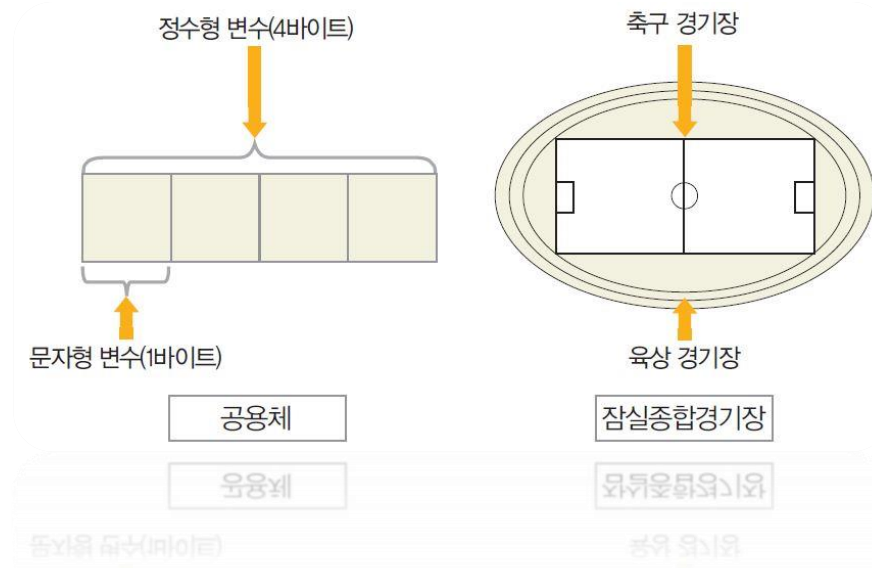
② 구조체 포인터 변수 사용

```
struct student {
    char name[10];
    int kor;
    int eng;
    float avg;
};
struct student s;
struct student *p;
p = &s;
p->kor = 100;
```

같은 메모리를 사용하는 다른 변수들의 집합

● 공용체와 종합경기장의 비교

- 공용체 : 하나의 메모리 공간을 다른 종류의 변수가 공용으로 사용하는 것



- 공용체의 문법

```
union 공용체형_이름 {
    데이터_형식 멤버_변수_1;
    데이터_형식 멤버_변수_2;
    :
};

union 공용체형_이름 공용체_변수;
```

```
union 공용체형_이름 공용체_변수;
```

2 공용체

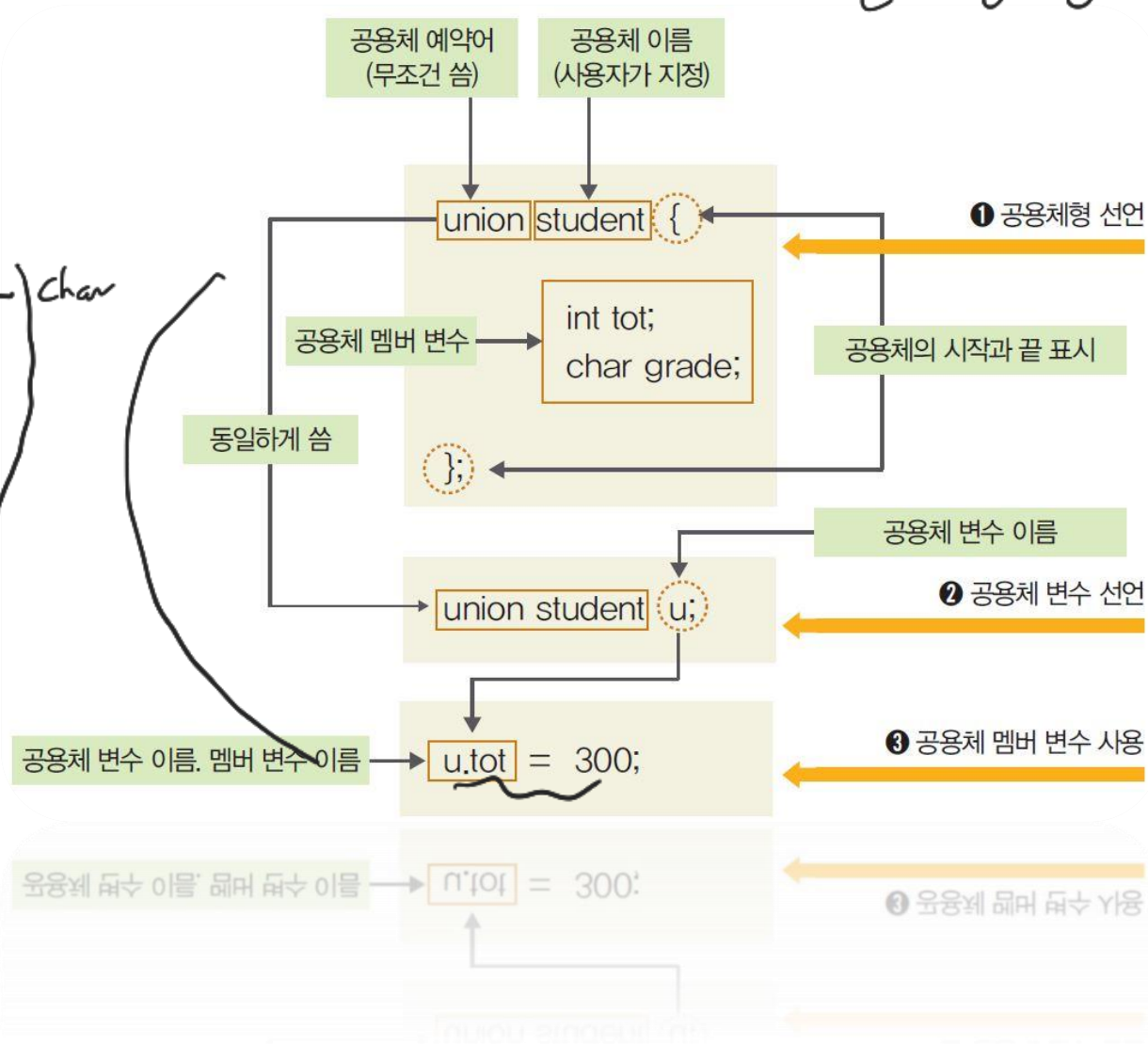
공용체의 사용 방법

● 공용체 사용을 위한 문법

중복가능

1개만 사용가능

int () char



- 공용체의 메모리 구조

- 구조체의 메모리 구조 이해가 필수



열거형의 이해

● 텍스트를 숫자로 취급하는 자료형

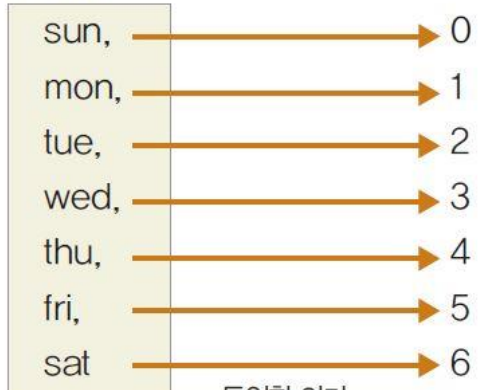
- 열거형의 형식
 - 단순히 1, 2, 3, 4, ... 와 같은 숫자를 기호로 표현한 것

- 요일을 열거형으로 표현
 - 0은 sun, 1은 mon, 2는 tue, ... 등과 같이 의미가 좀더 명확해짐
 - 나열한 데이터의 값은 0에서부터 1씩 차례대로 증가함

```
enum 열거형_이름{
    기호_1;
    기호_2;
    :
};

enum 열거형_이름 열거형_변수;
```

```
enum week {
    sun,
    mon,
    tue,
    wed,
    thu,
    fri,
    sat
};
```



동일한 의미

```
};
```

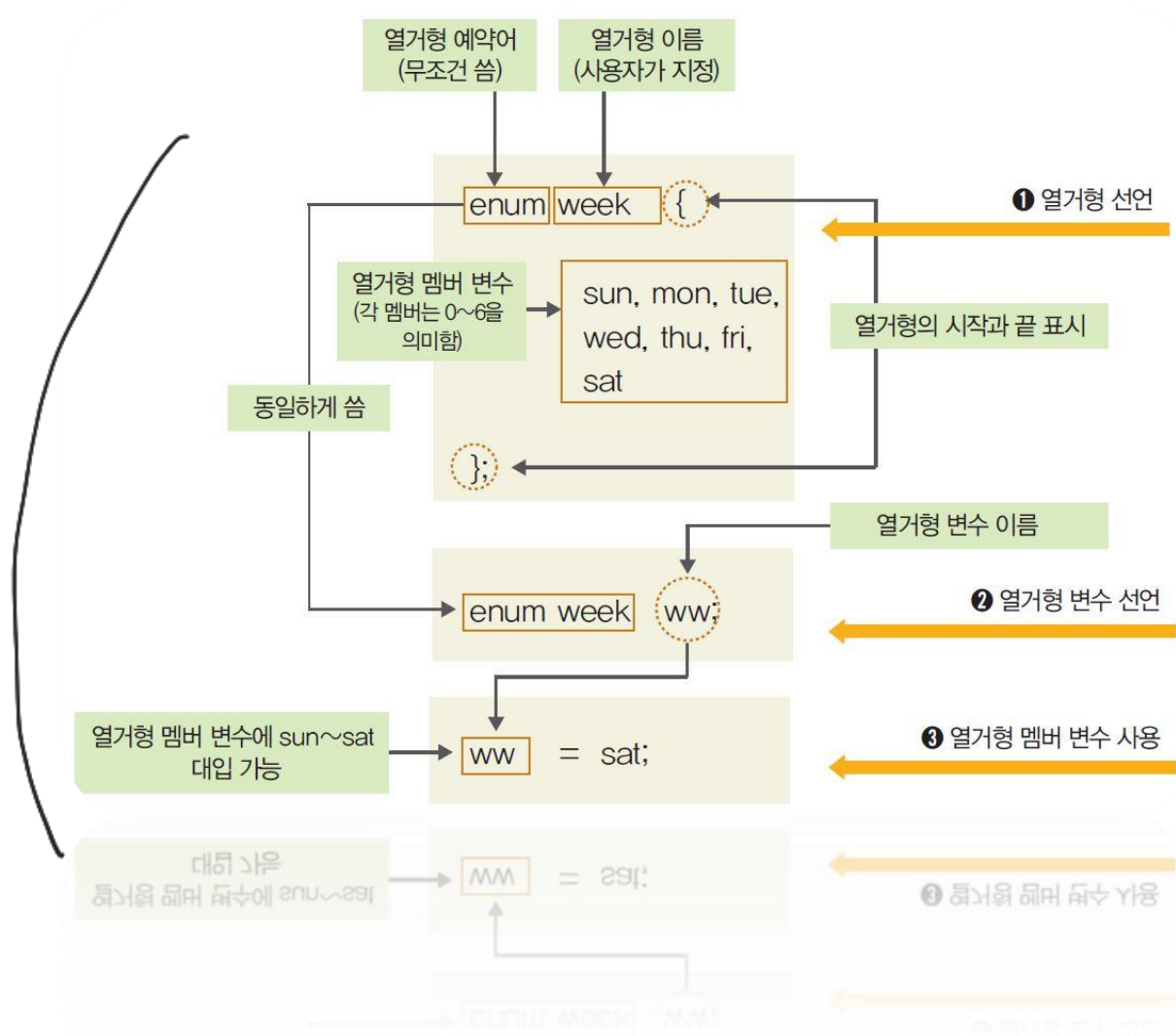


열거형_변수 → 0

3 열거형

열거형의 사용 방법

● 열거형의 문법



서로 다른 데이터 형식의 변수를 하나로 묶어 놓은 것

▼ 구조체의 기본 형식

```
struct 구조체형_이름 {
    데이터_형식  멤버_변수_1;
    데이터_형식  멤버_변수_2;
    :
};

struct 구조체형_이름 구조체_변수;
```

▼ 구조체의 예

```
struct bibim {
    int a;
    float b;
    char c;
};

struct bibim b1;
b1.a = 10;
```

구조체 변수 초기화 방법의 예

```
struct student {
    char name[10];
    int kor;
    int eng;
    float avg;
};

struct student s = {"LeeSan", 90, 80};
```

구조체 변수도 일반 변수처럼 배열과 포인터로 사용할 수 있다.

공용체와 열거형

■ 하나의 공간을 서로 다른 두 변수가 같이 사용하는 것

▼ 공용체의 기본 형식

```
union 공용체형_이름 {
    데이터_형식  멤버_변수_1;
    데이터_형식  멤버_변수_2;
    :
};

union 공용체형_이름 공용체_변수;
```

▼ 공용체의 예

```
union student {
    int tot;
    char grade;
};

union student u;
u.tot = 300;
```

배열 - 구조체

■ 변수가 가질 수 있는 범위가 정해져 있을 때 숫자보다 쉽게 파악할 수 있는 문자나 단어로 표현하는 자료형

▼ 열거형의 기본 형식

```
enum 열거형_이름{
    기호_1;
    기호_2;
    :
};

enum 열거형_이름 열거형_변수;
```

▼ 열거형의 예

```
enum week {
    sun, mon, tue, wed, thu, fri, sat
};

enum week ww;
ww = sat;
```