구조체와 공용체

❑ 학습목표

- 구조체와 공용체의 개념과 구조를 이해할 수 있다.
- 구조체의 특성을 이해하고 프로그램에 활용할 수 있다.
- 구조체를 함수의 매개변수로 전달하는 방법을 이해한다.
- 구조체를 다룰 때 포인터를 사용하는 이유를 알 수 있다.

구조체의 선언

■ 구조체의 선언 형식

```
[storage_class] struct [tag_name] {
    data_type member_name1;
    data_type member_name2;
    …
    data_type member_nameN;
} [variable_name];
```

```
struct STUDENT_SCORE {
    char name[20];              /* 학생 이름 */
    int kor, eng, math;         /* 국영수 점수 */
    float avg;                  /* 평균 */
};
```

❑ 구조체 개념 (2/5)

① **struct STUDENT_SCORE** score_data;

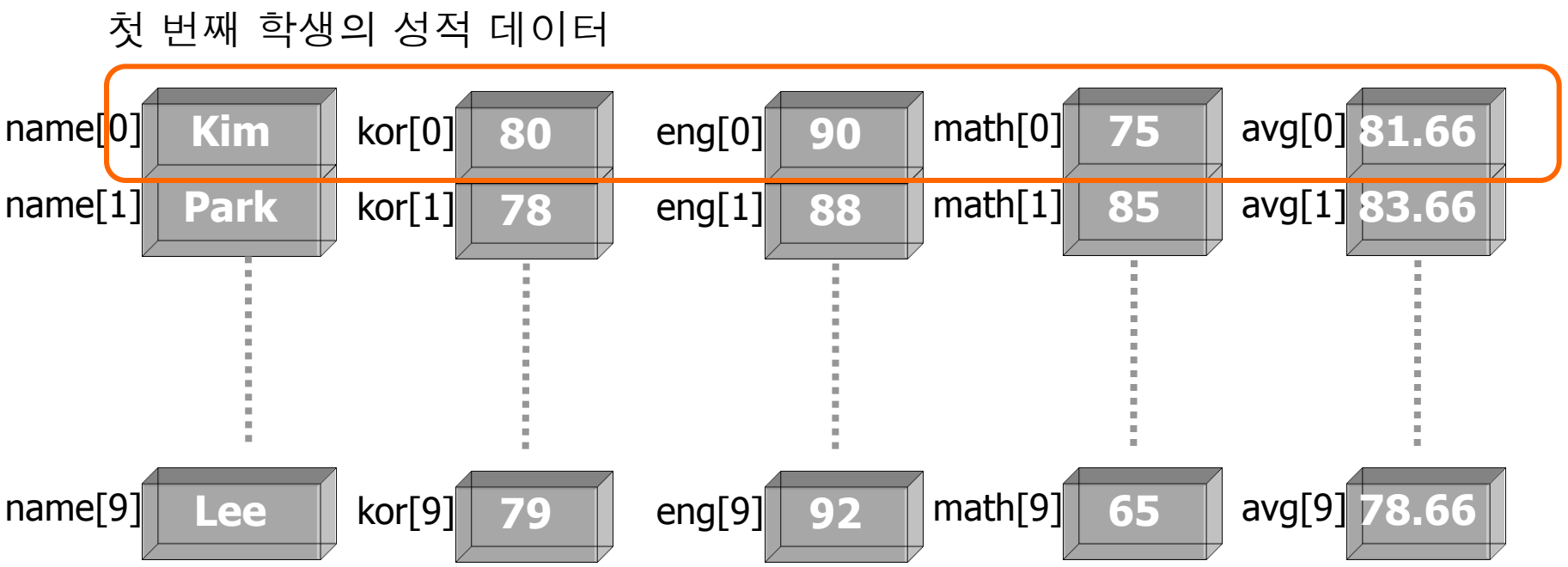
```
/* ②구조체 정의와 동시에 변수 선언 */

struct STUDENT_SCORE {
    char name[20];
    int kor, eng, math;
    float avg;
} score_data;
```

```
/* ③구조체 태그 명을 생략하고 변수 선언 */

struct {
    char name[20];
    int kor, eng, math;
    float avg;
} score_data1, score_data2;
```

❑ 구조체 개념 (3/5)

❖ **10명의 학생 성적 처리 (배열 사용)**

```
char name[10][20];
int kor[10], eng[10], math[10];
float avg[10];
```



❑ 구조체 개념 (4/5)

❖ **10명의 학생 성적 처리 (구조체 사용)**

struct **STUDENT_SCORE** **score_data**;

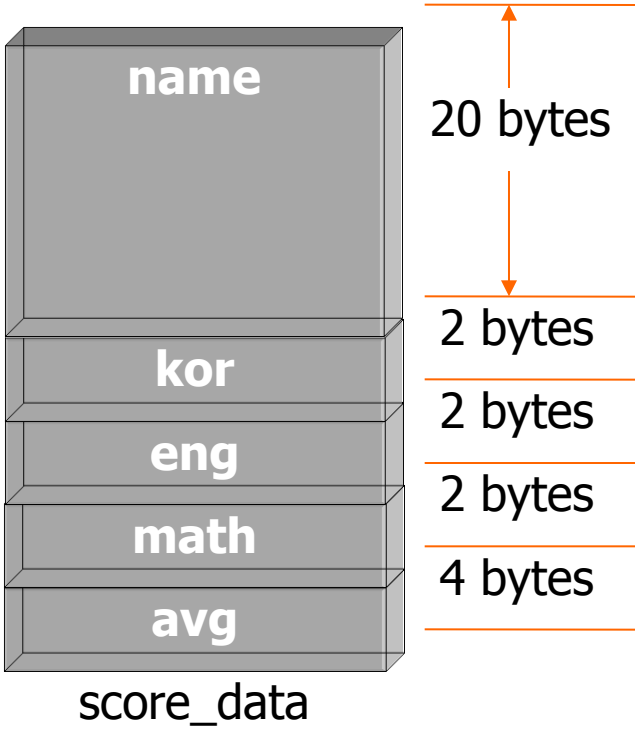
구조체의 크기는 기술된 각 멤버들이 차지하는 메모리 용량의 합과 같다. (**총 30 bytes**)
sizeof(struct STUDENT_SCORE) = 30 Bytes

구조체 멤버의 사용은 구조체변수명과 멤버 이름 사이를 **.** 로 구분하여 사용한다.

구조체변수명.멤버명

```
score_data.kor = 80;
```

score_data.name
score_data.kor
score_data.eng
score_data.math
score_data.avg



구조체의 초기화

```
/* 구조체 정의와 동시에 변수를 선언하고 초기화 */
struct STUDENT_SCORE {
    char name[20];
    int kor, eng, math;
    float avg;
} score_data = { "Kim", 80, 90, 75, 81.66};
```

```
/* 미리 정의된 구조체 타입을 이용하여 변수를 선언하고 동시에 초기화 */
struct STUDENT_SCORE sdata[3] = {
    { "Kim", 80, 90, 75, 81.66},
    { "Park", 78, 88, 85, 83.66},
    { "Lee", 79, 92, 65, 78.66}
};
```

❖ 초기화 시에 명시하지 않은 값들은 자동 초기화 된다. 예를 들어, **int**는 **0**, **float**이나 **double**은 **0.0**, 문자열은 '□**0**'로 자동 초기화 된다. 물론, 초기치 숫자가 멤버의 수나 배열의 수보다 많으면 에러가 발생한다.

구조체 멤버의 참조 방법

- 직접참조
- 간접참조

```
struct STUDENT_SCORE sd;
sd.name = "Kim";                /* 주의 : Error!! */
sd.kor = 80
sd.eng = 90;
sd.math = 75;
sd.avg = sd.kor + sd.eng + sd.math / 3;
```

"배열명은 포인터 상수다!"

```
strcpy(sd.name, "Kim");
```

❑ 구조체 멤버의 정의와 참조, 연산 (2/8)

[예제7-1] 간단한 구조체 사용의 예

```
01  #include <stdio.h>
02  int main()
03  {
04    struct STUDENT_SCORE {          /* 구조체의 정의 */
05      char name[20];
06      int kor, eng, math;
07      float avg;
08    };
09
10    struct STUDENT_SCORE sd;        /* 구조체 변수의 선언 */
11
12    printf("Student name: ");
13    gets(sd.name);
14    printf("KOR score : ");
15    scanf("%d", &sd.kor);
16    printf("ENG score : ");
17    scanf("%d", &sd.eng);
18    printf("MATH score : ");
19    scanf("%d", &sd.math);
20    sd.avg = (float)(sd.kor + sd.eng + sd.math)/3;
21    printf("%s's avg = %3.2f□n", sd.name, sd.avg);
22    printf("sizeof(struct STUDENT_SCORE) = %d□n", sizeof(struct STUDENT_SCORE));
23    return 0;
24  }
```

■ 실행결과

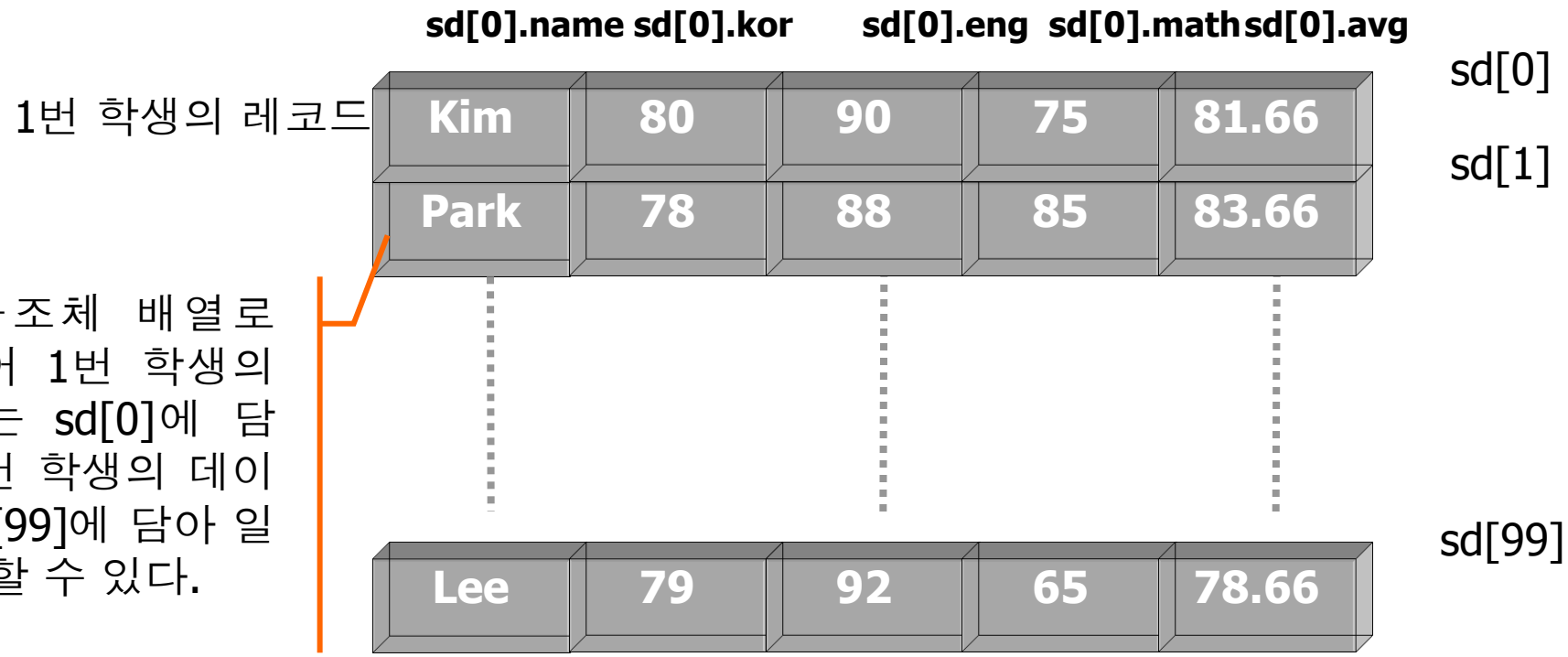
```
Student name: Kim
KOR score : 80
ENG score : 90
MATH score : 75
Kim avg = 81.67
sizeof(struct STUDENT_SCORE) = 30
```

❑ 구조체 멤버의 정의와 참조, 연산 (3/8)

❖ 구조체 배열 사용

```
struct STUDENT_SCORE sd[100];
```

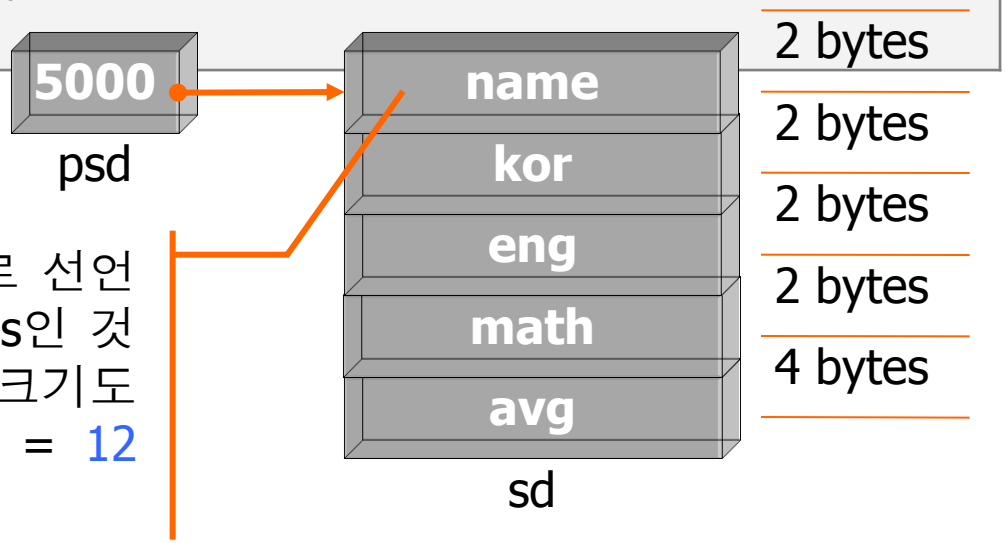
```
for (i=0; i<99; i++)
{
    gets(sd[i].name);
    scanf("%d", &sd[i].kor, &sd[i].eng, &sd[i].math);
      sd[i].avg = (float) (sd[i].kor + sd[i].eng + sd[i].math)/3;
}
```



❑ 구조체 멤버의 정의와 참조, 연산 (4/8)

❖ 구조체 포인터 변수

```
struct STUDENT_SCORE {
    char *name;          /* 문자열 포인터 변수로 선언되었다. */
    int kor, eng, math;
    float avg;
};
struct STDUENT_SCORE sd, *psd;
psd = &sd;
```



name이 배열이 아니라 포인터로 선언되었다. 때문에 그 크기가 2bytes인 것에 주목하자. 때문에 구조체의 크기도 sizeof(struct STUDENT_SCORE) = 12 Bytes 가 된다.

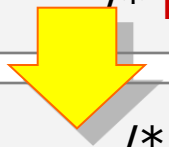
❑ 구조체 멤버의 정의와 참조, 연산 (5/8)

```
psd.name = “Kim”;            /* Error! */
*psd.name = “Kim”;           /* Error! */
strcpy(psd.name, “Kim”);     /* Error! */
psd.kor = 80;                /* Error! */
```

(*구조체포인터변수명).멤버변수명

```
(*psd).name = “Kim”;          /* 적법 */
strcpy((*psd).name, “Kim”);   /* 적법 */
(*psd).kor = 80;              /* 적법 */
```

```
*psd.name = “Kim”;        /* Error : *(psd.name) = “Kim”  으로 해석된다. */
```



```
psd->name = “Kim”;        /* (*psd).name = “Kim”  과 동일하다. */
```

❑ 구조체 멤버의 정의와 참조, 연산 (6/8)

```
scanf("%d", psd->kor);        /* Error! : psd->kor 은 그 자체로 정수형 변수 */
```

```
scanf("%d", &psd->kor);       /* 적법 */
scanf("%s", psd->name);       /* 적법 */
```

■ 구조체 참조 시 포인터 사용을 권하는 이유

● 배열에 비해 구조체 포인터가 메모리 공간을 적게 사용하고 효율적인 처리가능
● 특히, 구조체 자체를 함수의 매개변수로 전달할 경우 구조체 전체를 넘기게 되면 멤버들의 복사 작업에 많은 시간을 소비하게 되고 더불어 메모리의 낭비도 심해진다. 하지만, 구조체 자체가 아닌 포인터만 전달하면(Call by reference) 포인터를 위한 메모리 공간만 소비하게 되므로 실행속도를 향상시킬 수 있고 훨씬 더 효과적으로 관련처리를 할 수 있다.

구조체 복사

```
struct STUDENT_SCORE sd1, sd2 = {“Kim”, 80, 95, 75, 87.66};
sd1 = sd2;        /* 적법 : 구조체 복사 */
```

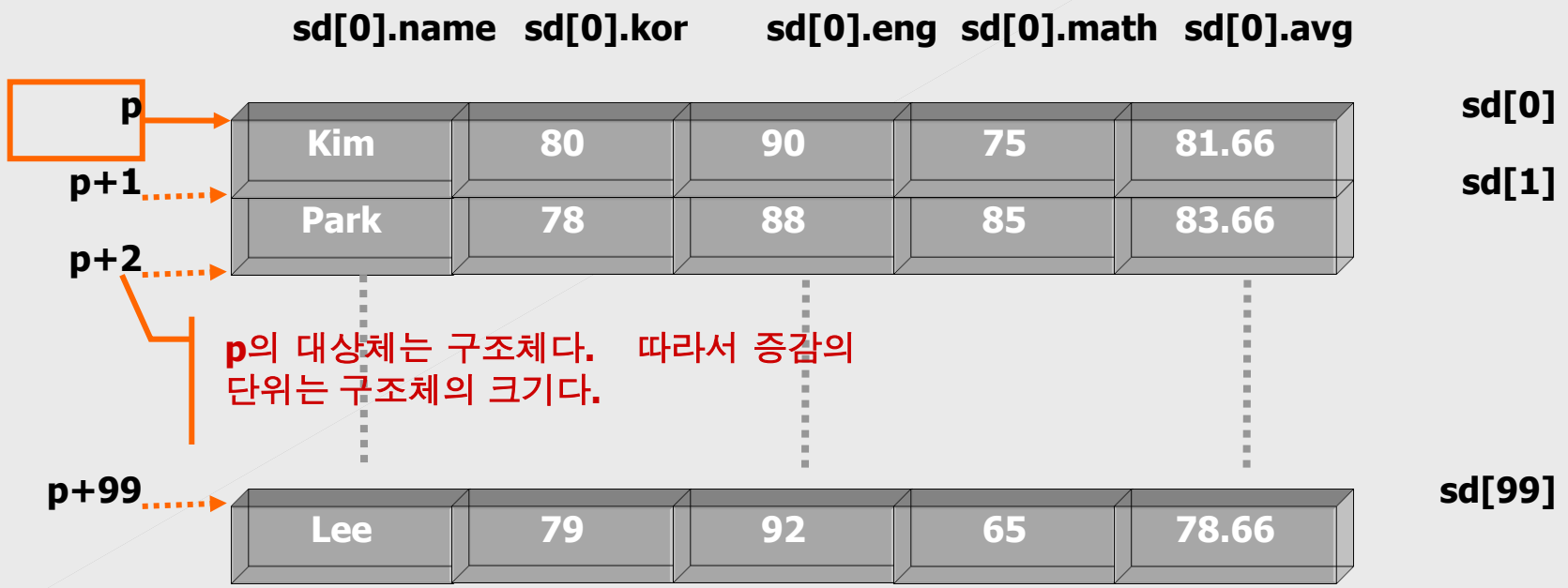
```
struct A {
    int a;
    float b;
} a;
```

```
struct B {
    int a;
    float b;
} b= { 10, 10.32 };
```

```
a = b;          /* Error : 구조체의 타입 불일치!! */
a.a = b.a;      /* 적법 */
a.b = b.b;      /* 적법 */
```

구조체 포인터 변수와 배열과의 관계

```
struct STUDENT_SCORE sd[100], *p;
p = sd;                    /* p는 구조체 배열 sd의 시작주소를 가리킨다. */
```



❑ 중첩 구조체 (1/4)

중첩 구조체의 정의

/* ① 구조체 내부의 멤버로 구조체를 직접 정의 */

```
struct STUDENT {
  int  no;
  char name[20];
  char sex, tel[15], major[30];
  struct SCORE {
    char date[10];
    int kor, eng, math;
    float avg;
  } score;
};
```

/* ② 외부에 선언된 구조체를 정의 */

```
struct SCORE {
  char date[10];
  int kor, eng, math;
  float avg;
};
struct STUDENT {
  int  no;
  char name[20];
  char sex, tel[15], major[30];
  struct SCORE score;
};
```

중첩 구조체의 초기화

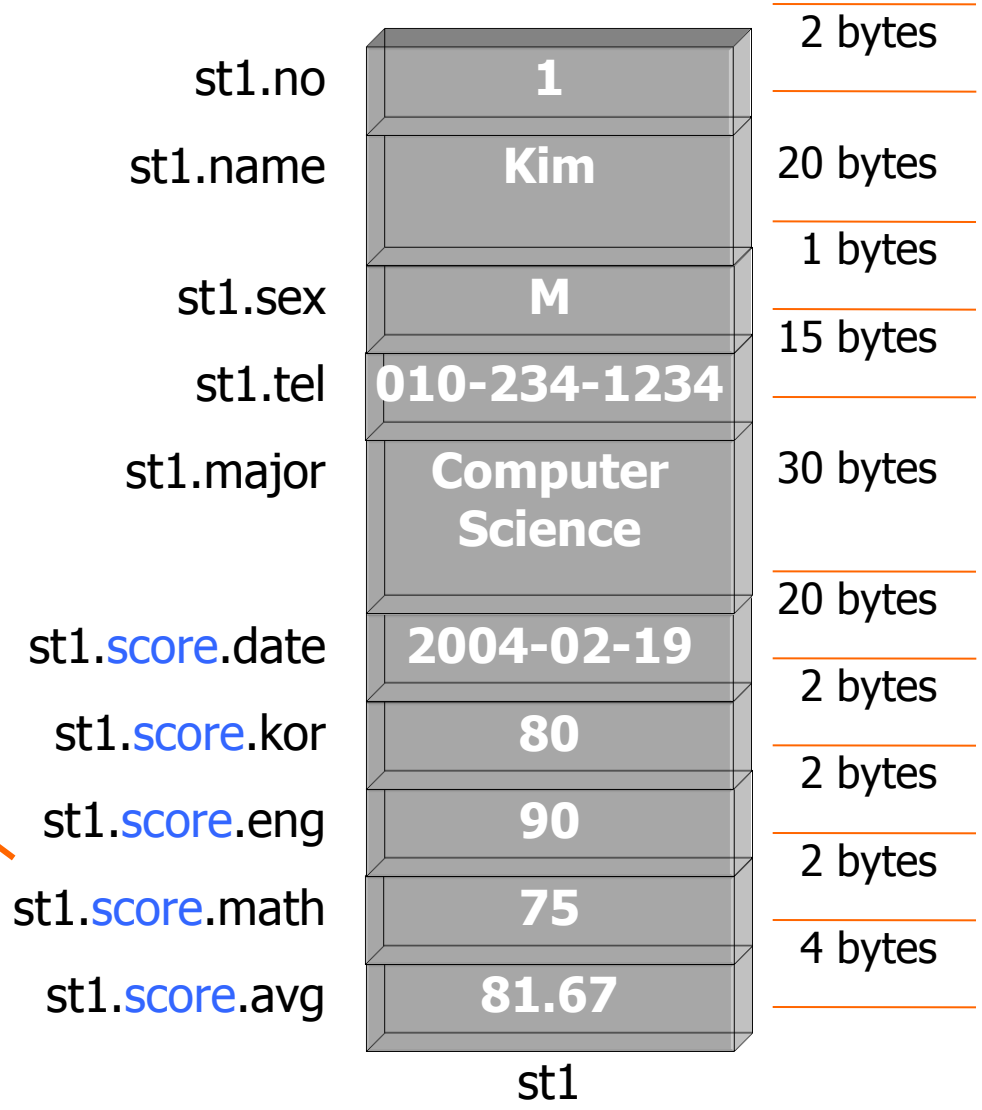
```
struct STUDENT {
    int  no;
    char name[20];
    char sex, tel[15], major[30];
    struct SCORE {
        char date[10];
        int kor, eng, math;
        float avg;
    } score;
} st1 = { 1, "Kim", 'M', "010-234-1234", "Computer Science",
"2004-01-22", 80, 90, 75, 81.67 };

struct STUDENT st2 = { 2, "Park", 'F', "016-123-3034", "Computer
Science", "2004-01-22", 78, 88, 85, 83.67 };
```

❑ 중첩 구조체 (3/4)

❖ 중첩된 구조체와 멤버 참조

```
struct STUDENT {
  int  no;
  char name[20];
  char sex, tel[15], major[30];
  struct SCORE {
    char date[20];
    int kor, eng, math;
    float avg;
  }
} st1 = { 1, "Kim", 'M', "010-234-1234",
"Computer Science", "2004-02-19", 80, 90,
75, 81.67 };
```

중첩된 구조체의 멤버 참조



❑ 중첩 구조체 (4/4)

```
strcpy(st1.score.date, “2004-02-20”);
printf(“%d”, st1.score.kor);
st1.score.eng = 100;
```

```
struct STUDENT st, *p;
p = &st;

①strcpy(p->score.date, “2004-02-20”);  /* 적법 : (*p).score.date 도 같은 표현 */
②printf(“%d”, p->score.kor);          /* 적법 */
③p->score.eng = 100;                  /* 적법 */
④p->score->math = 80;                 /* Error !! : score는 포인터 변수가 아니다 */
```

❑ 구조체를 함수의 인자로 전달 (1/2)

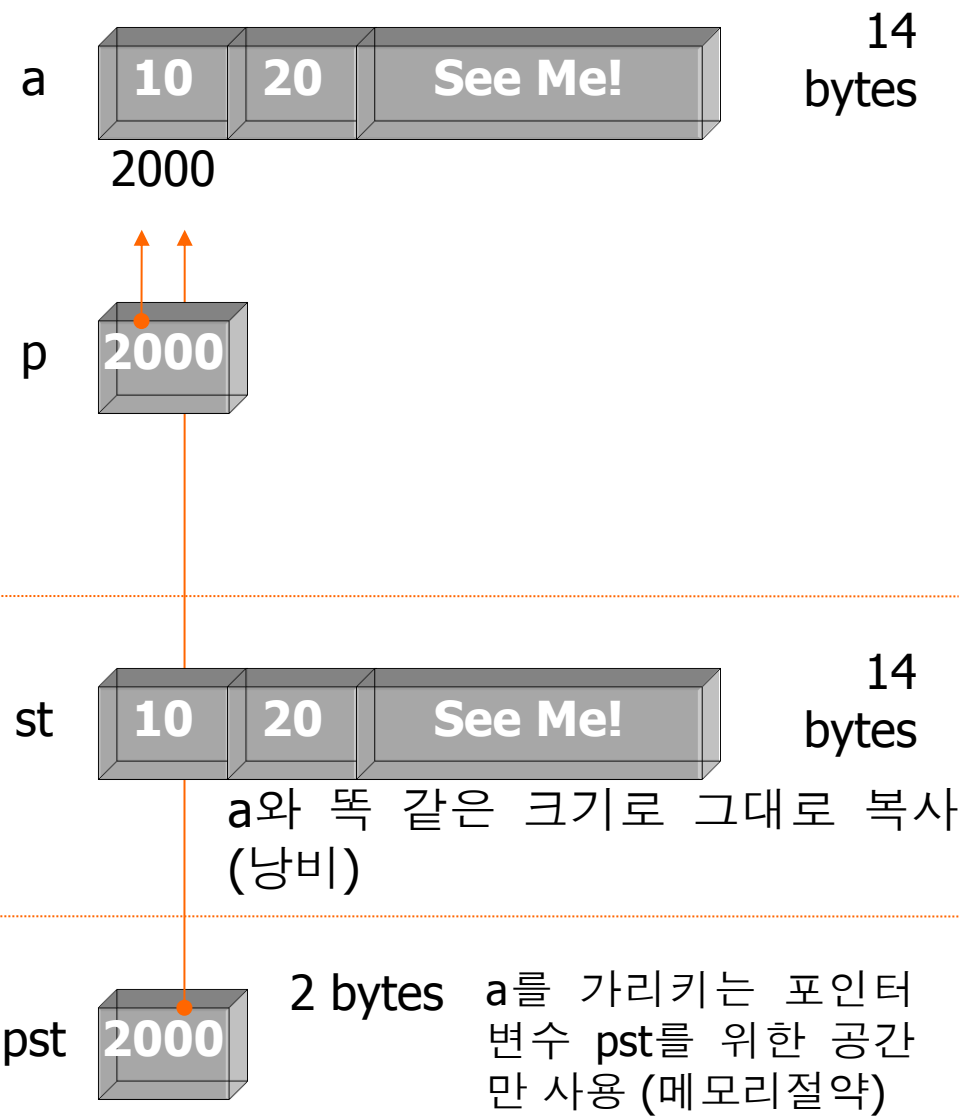
구조체를 함수의 매개변수로 전달하는 방법

- 실인자로 구조체 전체를 전달하는 방법 **(Call by value)**
  - 함수 내부의 형식인자에 그대로 복사해서 사용
  - 구조체의 크기가 필요이상으로 커지게 되면 이것은 심각한 부하 초래
  - 메모리 용량 과다 소모, 실행 속도 저하의 원인
- 실인자로 구조체 포인터를 전달하는 방법 **(Call by reference)**
  - 구조체로의 포인터를 전달
  - 포인터를 위한 **2 bytes** 공간만 소비
  - 프로그램의 효율 면에서 훨씬 좋은 방법

❑ 구조체를 함수의 인자로 전달 (2/2)

```
void main() {
   struct A {
      int  x, y;
      char s[10];
   }  a  =  {10,  20,  "See  Me!"},
*p=&a;

   f1(a);
   f2(p);
}
void f1(struct A st)
{
 ...
}

void f2(struct A *pst)
{
...
}
```

① ②



비트 필드의 이해

- 구조체의 멤버를 비트 단위로 제어할 수 있는 특수한 형태의 구조체

■ 비트필드의 선언 형식
data_type member : size;

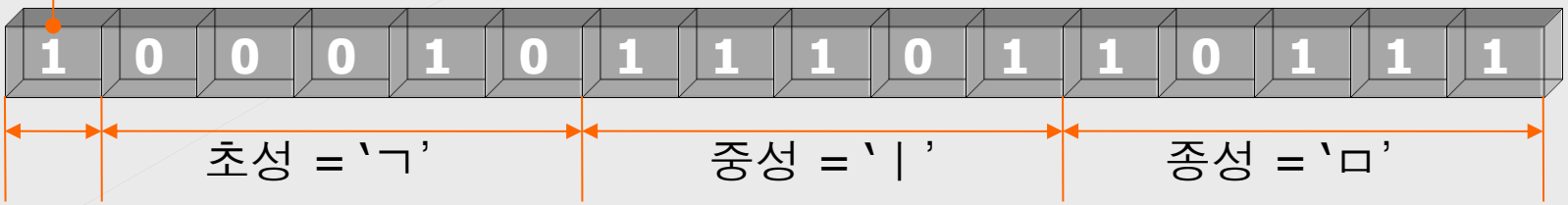
```
struct HANCODE {
    unsigned sign : 1;
    unsigned cho : 5;
    unsigned joong : 5;
    unsigned jong ; 5;
} hcode;
```

0: 영어
1: 한글

hcode.sign = 1

"김"

2 bytes



hcode.cho = 2

hcode.joong = 29

hcode.jong = 17

❑ 공용체(1/3)

공용체 개념

- 하나 이상의 변수가 함께 공유하여 사용할 수 있는 메모리 구조를 정의한 것
- 하나의 메모리 공간을 여러 멤버가 공유

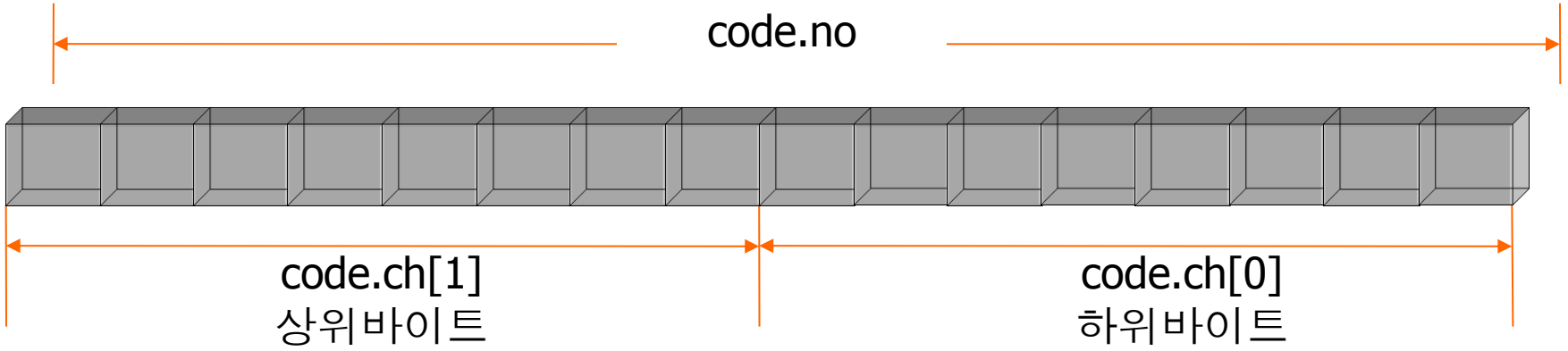
■ 공용체의 선언 형식

```
union [tag_name] {
    data_type member_name1;
    data_type member_name2;
    …
    data_type member_nameN;
} [variable_name];
```

❑ 공용체(2/3)

❖ 공용체 선언 예

```
union MY_CODE {
    int no;
    char ch[2];
} code;
```



NOTE 공용체는 그 특성상 선언과 동시에 초기화 하고자 한다면 첫 번째 필드만 가능하다.

```
union MY_CODE { int no, char ch[2]} = {65, 97}; /* Error */
union MY_CODE { int no, char ch[2]} = {65};    /* 적법 */
```

❑ 공용체(3/3)

❖ 공용체와 구조체의 혼합 사용 예

```
struct xreg {
    unsigned ax, bx, cx, dx, si, di, cflag;
};
struct hreg {
    unsigned char al, ah, bl, bh, cl, ch, di, dh;
};
union REGISTER {
    struct xreg x;
    struct hreg h;
} reg;
```

❑ 요약

- 구조체와 그 필요성
- 구조체의 멤버를 참조하는 두 가지 방법
- 구조체의 멤버를 참조할 때 주의해야 할 점
- 구조체를 이용할 경우 구조체 포인터에 의한 처리를 권하는 이유
- 비트 필드를 사용의 장점과 제약사항
- 공용체와 구조체의 차이점