

컴퓨터 프로그래밍2

구조체

장서윤 pineai@cnu.ac.kr

구조체

▶ 다양한 자료형을 하나로 묶을 수 있는 복합 자료형

표 17-1 구조체의 기본적인 사용법

구분	기능	사용 예
구조체 기본	형 선언	<code>struct student { int num; double grade; };</code>
	변수 선언	<code>struct student s1;</code>
	멤버 접근	<code>s1.num = 315;</code>
다양한 멤버	배열	<code>strcpy(yuni.name, "서하윤");</code>
	포인터	<code>yuni.intro = "안녕하세요";</code>
	다른 구조체	<code>yuni.pf.age = 17;</code>
구조체 대입 함수에 사용	구조체 대입	<code>struct student s1, s2; s2 = s1;</code>
	매개변수	<code>void func(struct student s1);</code>
	반환형	<code>struct student func(void);</code>
비트 필드 구조체	선언 방법	<code>struct children { unsigned int s : 2, d : 2, p : 3; };</code>
	패딩 멤버	<code>unsigned int : 2; unsigned int : 0;</code>

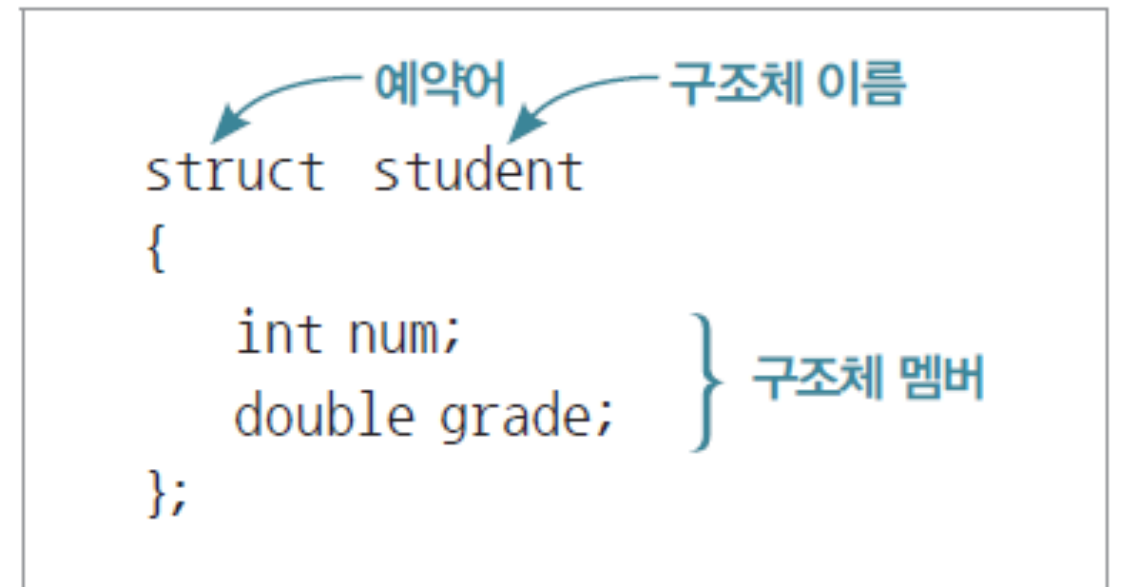
예제 17-1 구조체를 선언하고 멤버를 사용하는 방법

```
1. #include <stdio.h>
2.
3. struct student                // 구조체 선언
4. {
5.     int num;                  // int형 멤버
6.     double grade;            // double형 멤버
7. };                            // 세미콜론 사용
8.
9. int main(void)
10. {
11.     struct student s1;        // struct student형의 변수 선언
12.
13.     s1.num = 2;                // s1의 num 멤버에 2 저장
14.     s1.grade = 2.7;           // s1의 grade 멤버에 2.7 저장
15.     printf("학번 : %d\n", s1.num); // num 멤버 출력
16.     printf("학점 : %.1lf\n", s1.grade); // grade 멤버 출력
17.
18.     return 0;
19. }
```

실행
결과 학번 : 2
학점 : 2.7

구조체 선언과 멤버 사용

.....



➤ 구조체 선언

➤ struct 예약어 사용하여 구조체 형태 작성

➤ 이름은 구조체의 성격에 맞는 적절한 이름

➤ 블록 안에 멤버 나열

➤ 구조체 선언

➤ 구성하는 자료형 종류와 이름을 컴파일러에게 알림

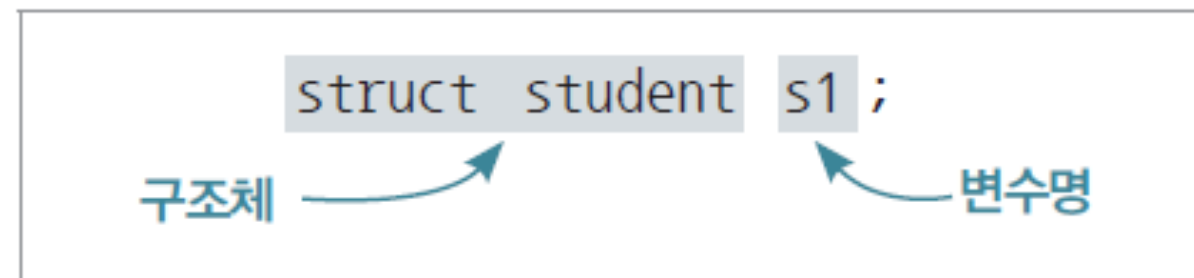
➤ 실제 저장 공간이 할당되는 변수 선언과는 다름

➤ 마지막으로 블록을 닫은 후 반드시 세미콜론

➤ 함수 안에도 선언 가능 (그 함수 안에서만 사용 가능)

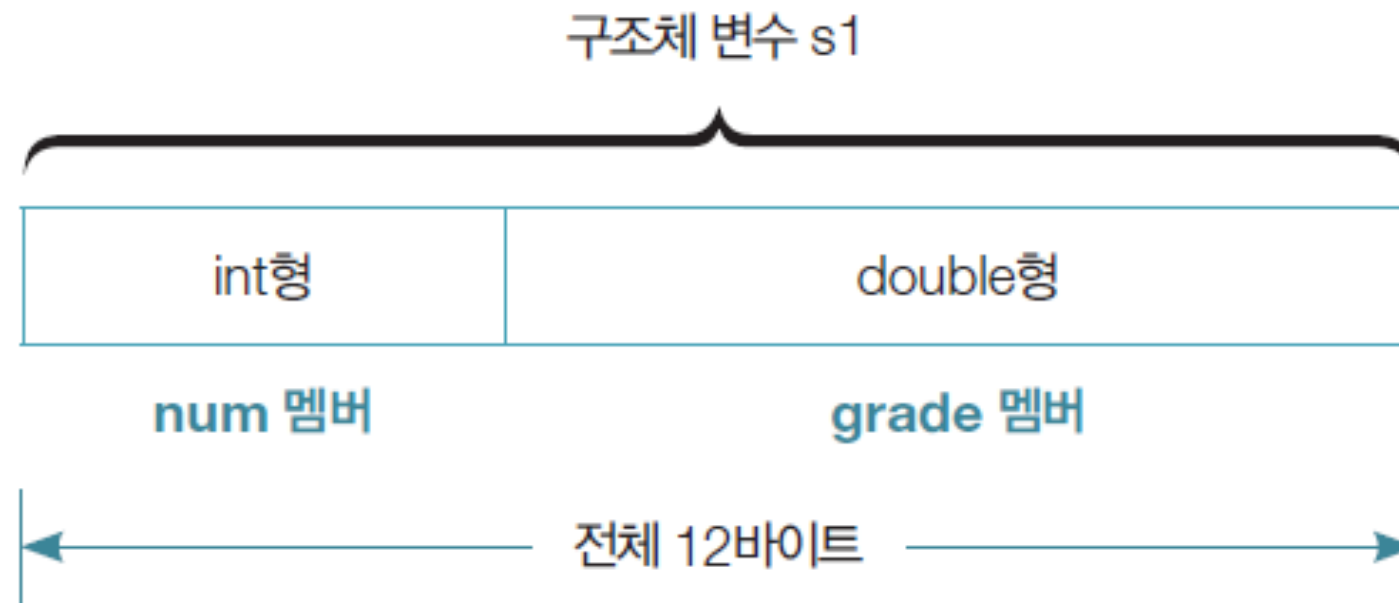
구조체 선언과 멤버 사용

- ▶ 구조체 선언 끝나면 사용자가 정의한 새로운 자료형을 컴파일러가 인식
- ▶ 구조체는 struct 예약어와 구조체 이름을 함께 하나의 자료형 이름으로 사용



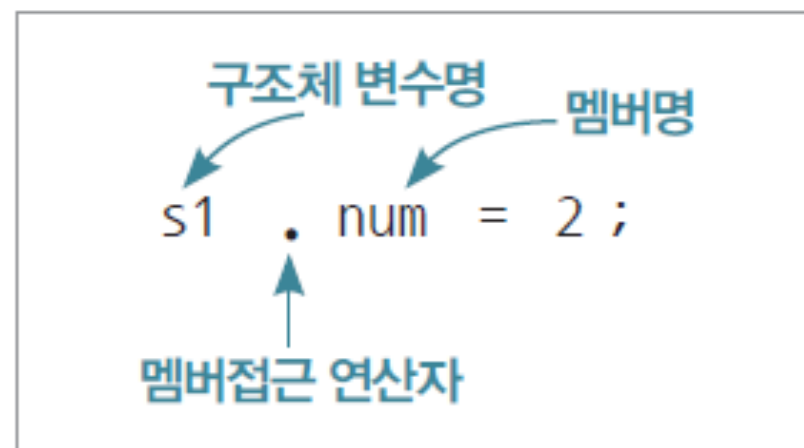
구조체 선언과 멤버 사용

- ▶ 구조체 변수를 선언 - 저장 공간 할당
 - ▶ 각 멤버들의 공간이 메모리에 연속으로 할당
- ▶ 모든 멤버를 더한 전체 저장 공간이 하나의 구조체 변수
 - ▶ 변수의 크기는 각 멤버의 크기 더한 값



구조체 선언과 멤버 사용

- ▶ 선언된 구조체 변수
 - ▶ 그 안에 여러 개의 멤버를 가짐
 - ▶ 특정 멤버 골라서 사용해야 함
 - ▶ 구조체 변수는 사용할 멤버에 접근할 때 . 연산자 사용
- ▶ s1의 멤버 num 사용하는 방법
 - ▶ s1은 구조체 변수지만 s1.num은 int형 변수
 - ▶ 연산하거나 입출력하는 등 하나의 int형 변수로 수행할 수 있는 모든 작업 가능



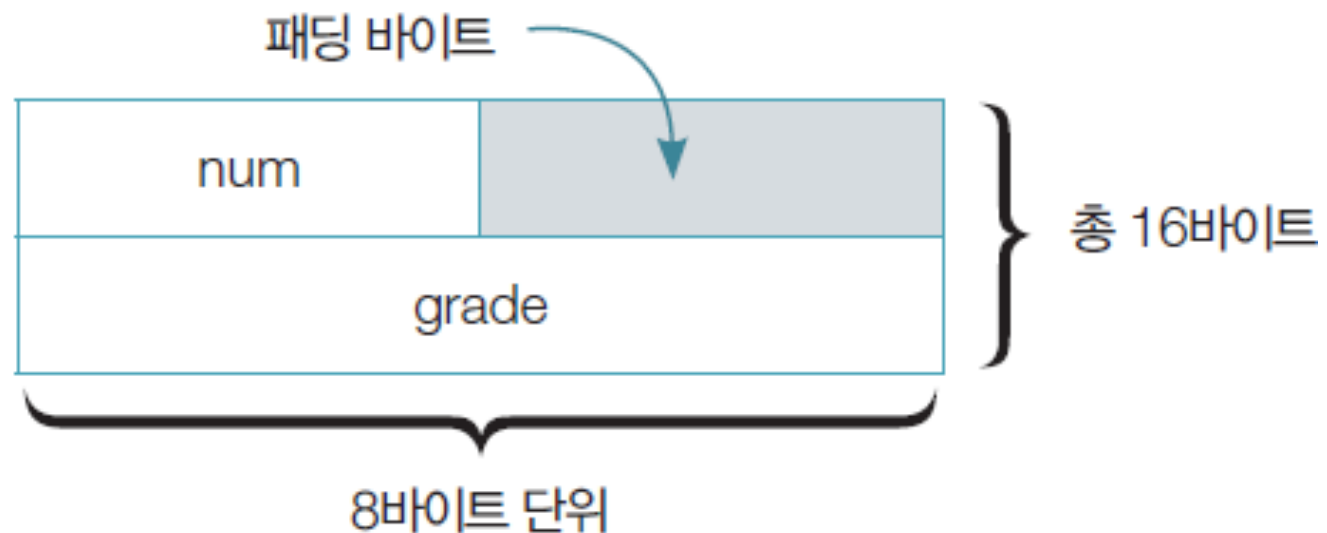
구조체 선언과 멤버 사용

- ▶ 바이트 얼라인먼트(byte alignment)
 - ▶ 모든 시스템은 데이터를 빠르게 읽고 쓰기 위해 일정한 크기 단위로 메모리에 접근
 - ▶ 컴파일러는 구조체 멤버의 크기가 다른 경우 멤버 사이에 패딩 바이트(padding byte) 넣어 멤버 정렬
- ▶ Ex) struct student 구조체의 크기는 패딩 바이트가 추가되어 16바이트로 계산
 - ▶ 크기가 가장 큰 멤버가 메모리 할당 기준 단위
 - ▶ struct student 구조체는 grade 멤버의 크기가 가장 크므로 8바이트가 기준 단위

구조체 선언과 멤버 사용

- ▶ 바이트 어라인먼트(byte alignment)
 - ▶ num 멤버
 - ▶ 첫 번째 8바이트 블록의 처음 4바이트에 할당
 - ▶ grade 멤버
 - ▶ 다음 8바이트 블록 사용
 - ▶ 4바이트의 패딩 바이트 포함
 - ▶ 전체 구조체의 크기는 16바이트

```
struct student
{
    int num;
    double grade;
};
```



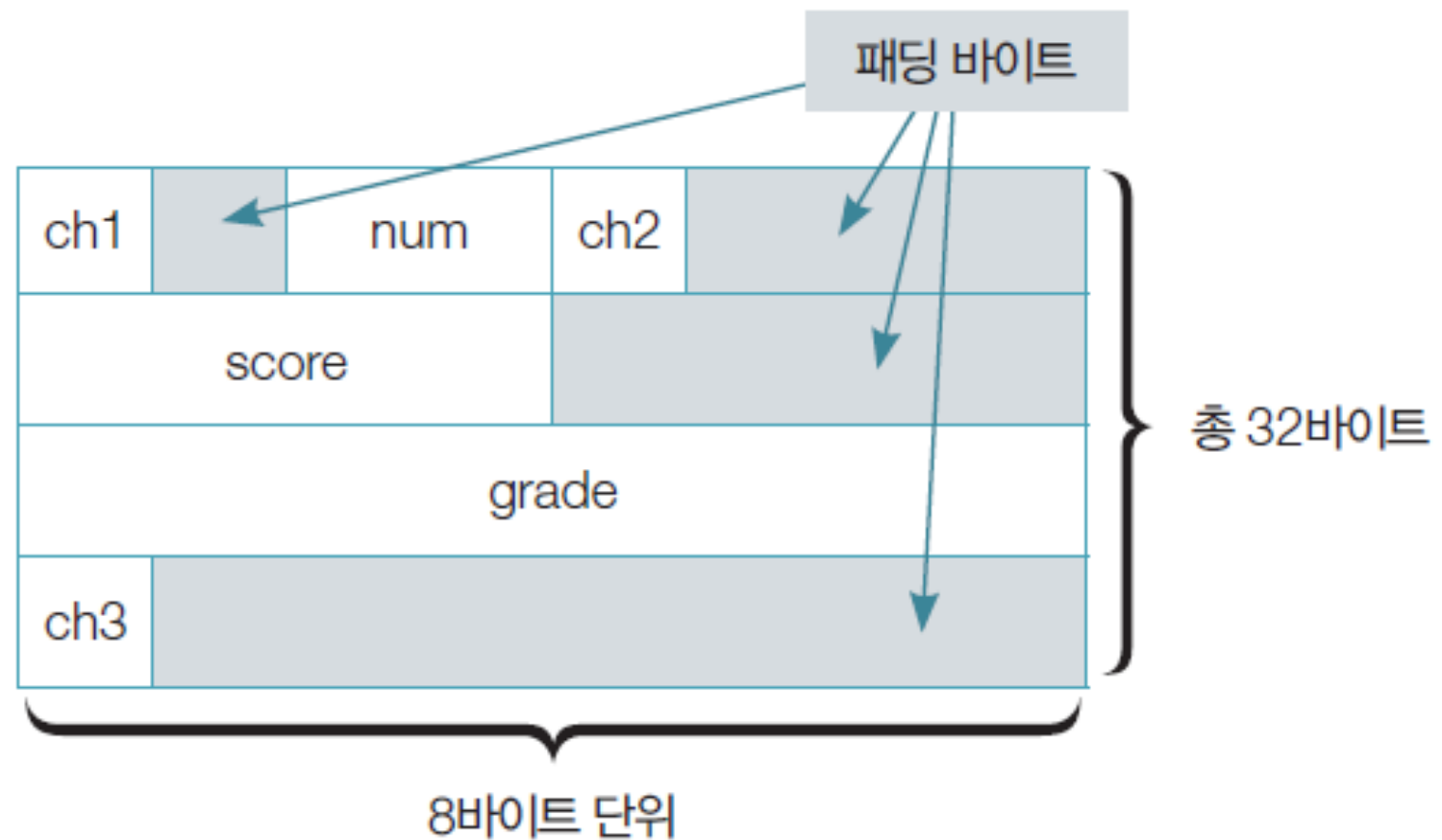
구조체 선언과 멤버 사용

- ▶ 바이트 얼라인먼트(byte alignment)
 - ▶ 기준 블록 내에서 크기가 작은 멤버들은 각 자료형 크기 단위로 할당
 - ▶ char형은 1바이트 단위로 할당 - 모든 위치에서 할당
 - ▶ short형은 2바이트 단위, int형은 4바이트 단위로 끊어서 할당

구조체 선언과 멤버 사용

- ▶ 바이트 어라인먼트(byte alignment)
 - ▶ Ex) 다음과 같은 구조체의 경우 크기는 32바이트

```
struct student
{
    char ch1;
    short num;
    char ch2;
    int score;
    double grade;
    char ch3;
};
```



구조체 선언과 멤버 사용

- ▶ 바이트 얼라인먼트(byte alignment)
 - ▶ 멤버의 순서 따라 구조체의 크기가 달라질 수 있음
 - ▶ 패딩 바이트가 가장 작도록 구조체 선언하면 메모리를 아낄 수 있음
 - ▶ 컴파일러에 패딩 바이트를 넣지 않도록 지시 가능
 - ▶ 데이터 읽고 쓰는 시간은 더 걸릴 수 있음
 - ▶ 구조체의 크기는 모든 멤버의 크기를 더한 것과 같아지므로 사용하는 메모리 크기 최소화

```
#pragma pack(1); // 바이트 얼라인먼트를 1로 설정하면 패딩 바이트가 필요 없음
```

다양한 구조체 멤버

▶ 배열, 포인터, 이미 선언된 다른 구조체도 멤버 사용 가능

예제 17-2 배열과 포인터를 멤버로 갖는 구조체 사용

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4.
5. struct profile                // 신상명세 구조체 선언
6. {
7.     char name[20];            // 이름을 저장할 배열 멤버
8.     int age;                  // 나이
9.     double height;            // 키
10.    char *intro;               // 자기소개를 위한 포인터
11. };
12.
```

```

13. int main(void)
14. {
15.     struct profile yuni;           // profile 구조체 변수 선언
16.
17.     strcpy(yuni.name, "서하윤");    // name 배열 멤버에 이름 복사
18.     yuni.age = 17;                  // age 멤버에 나이 저장
19.     yuni.height = 164.5;            // height 멤버에 키 저장
20.
21.     yuni.intro = (char *) malloc(80); // 자기소개를 저장할 공간 동적 할당
22.     printf("자기 소개 : ");
23.     gets(yuni.intro);               // 할당한 공간에 자기소개 입력
24.
25.     printf("이름 : %s\n", yuni.name); // 각 멤버의 데이터 출력
26.     printf("나이 : %d\n", yuni.age);
27.     printf("키 : %.1lf\n", yuni.height);
28.     printf("자기소개 : %s\n", yuni.intro);
29.     free(yuni.intro);                // 동적 할당 영역 반환
30.
31.     return 0;
32. }

```

실행 결과

```

자기 소개 : 항상 행복하세요.
이름 : 서하윤
나이 : 17
키 : 164.5
자기소개 : 항상 행복하세요.

```

다양한 구조체 멤버

- ▶ 구조체 멤버로 배열 사용 가능
- ▶ profile 구조체는 이름을 저장하는 멤버로 배열 사용
 - ▶ 배열이 구조체 안에 하나의 멤버
 - ▶ 구조체 변수 선언 - 배열 멤버도 그 크기만큼 저장 공간 할당

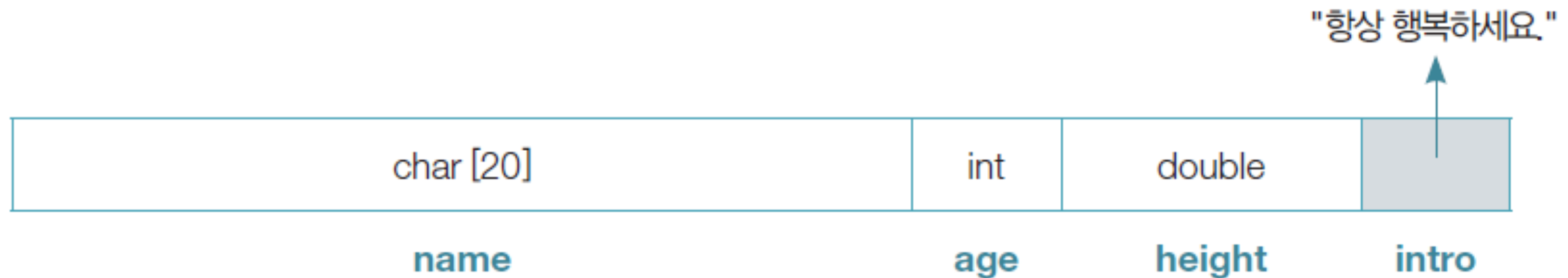
```
struct profile yuni;    // 구조체 변수 선언
```



다양한 구조체 멤버

- ▶ 구조체 멤버로 포인터 사용 가능
 - ▶ 포인터 멤버에 대입 연산으로 간단히 문자열 연결
- ▶ intro 멤버

```
yuni.intro = "항상 행복하세요.";
```



다양한 구조체 멤버

▶ intro 멤버

- ▶ 문자열 상수 대신 키보드로 문자열 바로 입력 불가
- ▶ intro 멤버는 포인터 - 문자열 자체 저장할 공간 없음
 - ▶ 동적 할당 통해 저장 공간 먼저 확보
 - ▶ 확보 후 자기 소개 입력



다양한 구조체 멤버

- ▶ 포인터 멤버는 배열 멤버보다 사용하기 더 번거로움
 - ▶ 길이가 고정적인 배열과 달리 다양한 길이의 공간 동적 할당하여 쓸 수 있음
 - ▶ 필요에 따라 적절한 멤버 사용
- ▶ 구조체의 멤버로 다른 구조체 사용
 - ▶ student 구조체에 신상명세에 대한 부분 추가된다면?
 - ▶ profile 구조체 활용 가능
 - ▶ 물론 student 구조체보다 profile 구조체가 먼저 선언되어 있어야 사용 가능

다양한 구조체 멤버

예제 17-3 다른 구조체를 멤버로 갖는 구조체 사용

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4.
5. struct profile                // 신상명세 구조체 선언
6. {
7.     int age;                  // 나이
8.     double height;           // 키
9. };
10.
11. struct student
12. {
13.     struct profile pf;        // profile 구조체를 멤버로 사용
14.     int id;                   // 학번을 저장할 멤버
15.     double grade;            // 학점을 저장할 멤버
16. };
```

```

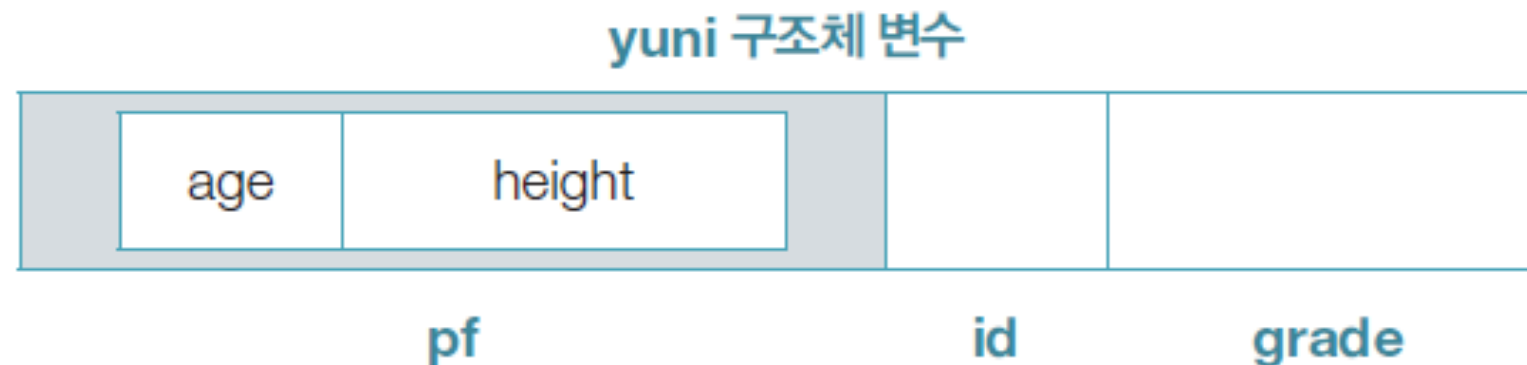
17.
18. int main(void)
19. {
20.     struct student yuni;           // student 구조체 변수 선언 .....
21.
22.     yuni.pf.age = 17;               // age 멤버에 나이 저장
23.     yuni.pf.height = 164.5;        // height 멤버에 키 저장
24.     yuni.id = 315
25.     yuni.grade = 4.3;
26.
27.     printf("나이 : %d\n", yuni.pf.age);    // pf 멤버의 age 멤버 출력
28.     printf("키 : %.1lf\n", yuni.pf.height); // pf 멤버의 height 멤버 출력
29.     printf("학번 : %d\n", yuni.id);        // id 멤버 출력
30.     printf("학점 : %.1lf\n", yuni.grade);  // grade 멤버 출력
31.
32.     return 0;
33. }

```

실행
결과
 나이 : 17
 키 : 164.5
 학번 : 315
 학점 : 4.3

다양한 구조체 멤버

- ▶ student 구조체는 profile 구조체를 멤버로 선언
- ▶ student 구조체는 나이와 키를 저장할 멤버를 일일이 선언하지 않아도 profile 구조체의 멤버를 모두 자신의 데이터로 가질 수 있음

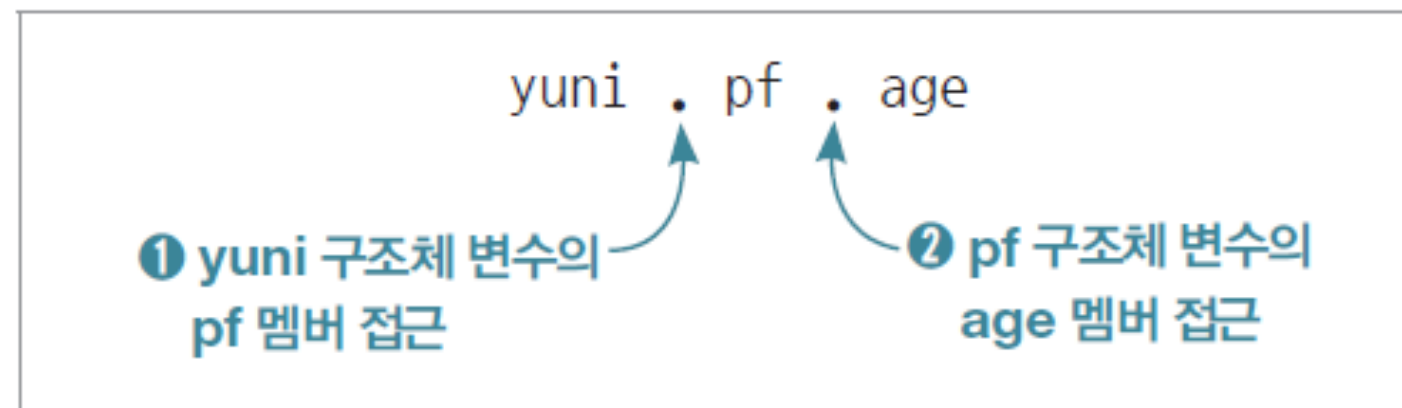


다양한 구조체 멤버

▶ 포함된 구조체인 profile의 멤버 사용

▶ . 연산자 2번 사용

▶ Ex) 나이를 저장하는 age 멤버



구조체 변수의 초기화와 대입 연산

▶ 구조체 변수의 초기화

▶ 초깃값 중괄호로 묶고 각 멤버의 형태에 맞는 값으로 초기화

▶ 여러 개의 멤버 초기화

▶ 배열 초기화와 비슷한 방법 사용

예제 17-4 최고 학점의 학생 데이터 출력

```
1. #include <stdio.h>
2.
3. struct student                                // 학생 구조체 선언 .....
4. {
5.     int id;                                    // 학번
6.     char name[20];                            // 이름
7.     double grade;                             // 학점
8. };
9.
10. int main(void)
11. {
12.     struct student  s1 = {315, "홍길동", 2.4},      // 구조체 변수 선언과 초기화
13.                      s2 = {316, "이순신", 3.7},
14.                      s3 = {317, "세종대왕", 4.4};
15.
16.     struct student max;                            // 최고 학점을 저장할 구조체 변수
17.
18.     max = s1;                                       // s1을 최고 학점으로 가정
19.     if(s2.grade > max.grade) max = s2;             // s2가 더 높으면 max에 대입
20.     if(s3.grade > max.grade) max = s3;             // s3가 더 높으면 max에 대입
21.
22.     printf("학번 : %d\n", max.id);                 // 최고 학점 학생의 학번 출력
23.     printf("이름 : %s\n", max.name);               // 최고 학점 학생의 이름 출력
24.     printf("학점 : %.1lf\n", max.grade);           // 최고 학점 학생의 학점 출력
25.
26.     return 0;
27. }
```

실행
결과

학번 : 317
이름 : 세종대왕
학점 : 4.4

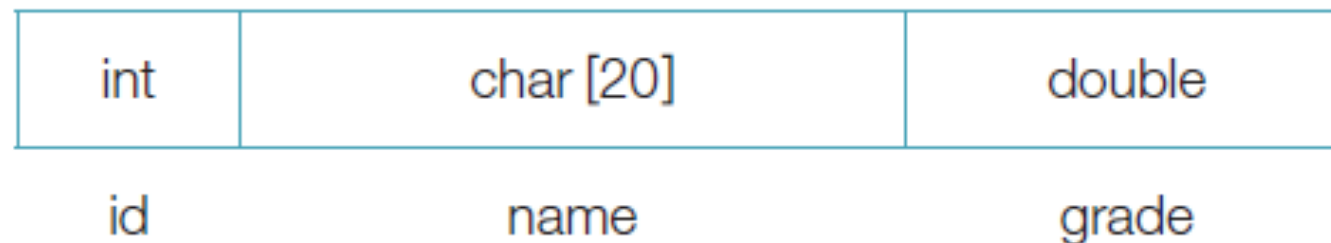
구조체 변수의 초기화와 대입 연산

▶ student 구조체

- ▶ 학번, 이름, 학점을 저장하는 멤버
- ▶ 멤버에 따라 정수, 문자열, 실수가 각각 필요
- ▶ 구조체 변수 선언과 함께 중괄호 사용
 - ▶ 각 멤버의 형태에 맞는 값으로 초기화

```
struct student s1 = { 315 , "홍길동" , 2.4 } ;
```

구조체 변수 **s1**



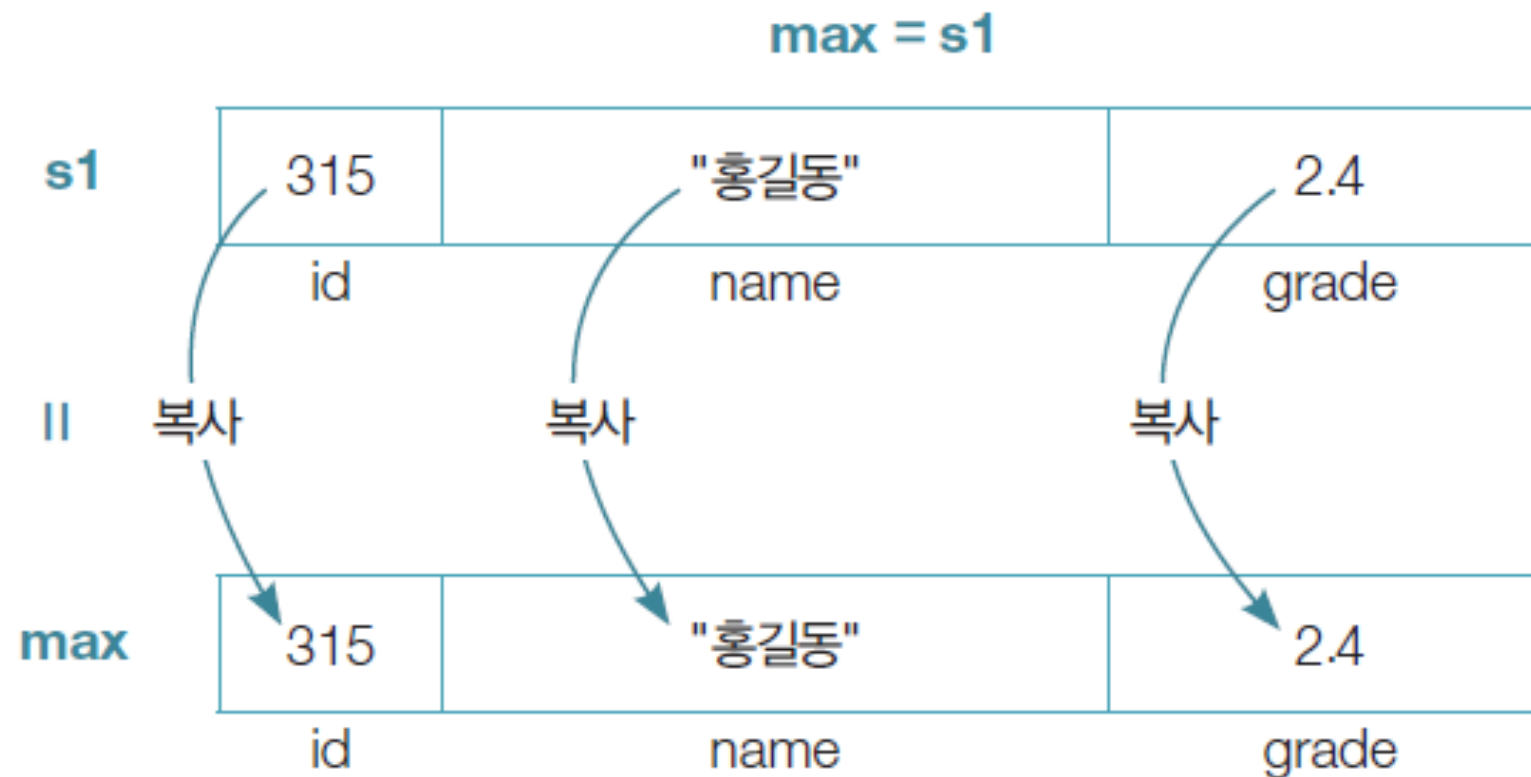
구조체 변수의 초기화와 대입 연산

▶ 구조체 변수끼리 값을 복사하는 과정에 대입 연산 사용

▶ 구조체 변수의 대입

▶ 각 멤버들을 자동으로 다른 구조체 변수에 복사

구조체 변수의 대입은 각 멤버들을 자동으로 다른 구조체 변수에 복사합니다.



구조체 변수의 초기화와 대입 연산

- ▶ 만일 학번, 이름, 학점이 서로 분리되어 있다면?
 - ▶ 복사하는 과정을 따로 수행
- ▶ 구조체로 묶여 있으므로 대입 연산 실행 - 한번에 처리

멤버별 복사 방법

```
max.id = s1.id;  
strcpy(max.name, s1.name);  
max.grade = s1.grade;
```



대입 연산으로 복사

```
max = s1;
```

구조체 변수의 초기화와 대입 연산

- ▶ 구조체 선언과 동시에 변수 선언과 초기화 가능
 - ▶ 구조체 선언 함수 밖에서
 - ▶ 함께 선언되는 변수가 전역 변수
 - ▶ 초기화하지 않을 경우 모든 멤버 0으로 자동 초기화

```
struct student
{
    int id;
    char name[20];
    double grade;
} s1 = {315, "홍길동", 2.4};
```

구조체 선언과 구조체 변수 선언
초기화를 동시에 한다.

구조체 변수를 함수 매개변수에 사용하기

- ▶ 구조체 변수는 대입 연산 가능
 - ▶ 함수에서 여러 개의 값 구조체로 묶어 동시에 반환 가능
- ▶ Ex) 두 변수의 값을 바꾸는 함수는 포인터 필요
 - ▶ 구조체 변수 사용해 값 주고받으면 포인터 없이 두 변수의 값을 바꾸는 함수 구현 가능


예제 17-5 구조체를 반환하여 두 변수의 값 교환

```
1. #include <stdio.h>
2.
3. struct vision                                // 로봇의 시력을 저장할 구조체
4. {
5.     double left;                             // 왼쪽 눈
6.     double right;                            // 오른쪽 눈
7. };
8.
9. struct vision exchange(struct vision);        // 두 시력을 바꾸는 함수
10.
11. int main(void)
12. {
13.     struct vision robot;                     // 구조체 변수 선언
14.
15.     printf("시력 입력 : ");
16.     scanf("%lf%lf", &(robot.left), &(robot.right)); // 시력 입력
17.     robot = exchange(robot);                 // 교환 함수 호출
18.     printf("바뀐 시력 : %.1lf, %.1lf\n", robot.left, robot.right);
19.
20.     return 0;
21. }
22.
```

구조체 변수를 함수 매개변수에 사용하기

```
23. struct vision exchange(struct vision robot)           // 구조체를 반환하는 함수
24. {
25.     double temp;                                       // 교환을 위한 임시 변수
26.
27.     temp = robot.left;                                // 좌우 시력 교환
28.     robot.left = robot.right;
29.     robot.right = temp;
30.
31.     return robot;                                     // 구조체 변수 반환
32. }
```

실행
결과

시력 입력 : 15.5, 20.0 

바뀐 시력 : 20.0, 15.5

구조체 변수를 함수 매개변수에 사용하기

- ▶ 함수 호출할 때 인수로 구조체 변수 사용
 - ▶ 멤버들의 값을 한꺼번에 함수에 넘기는 것 가능
 - ▶ 멤버가 배열이라도 모든 배열 요소의 값이 함수에 복사
- ▶ 구조체 변수 반환할 때도 똑같이 적용
 - ▶ 함수가 여러 개의 값 한 번에 반환 가능

구조체 변수를 함수 매개변수에 사용하기

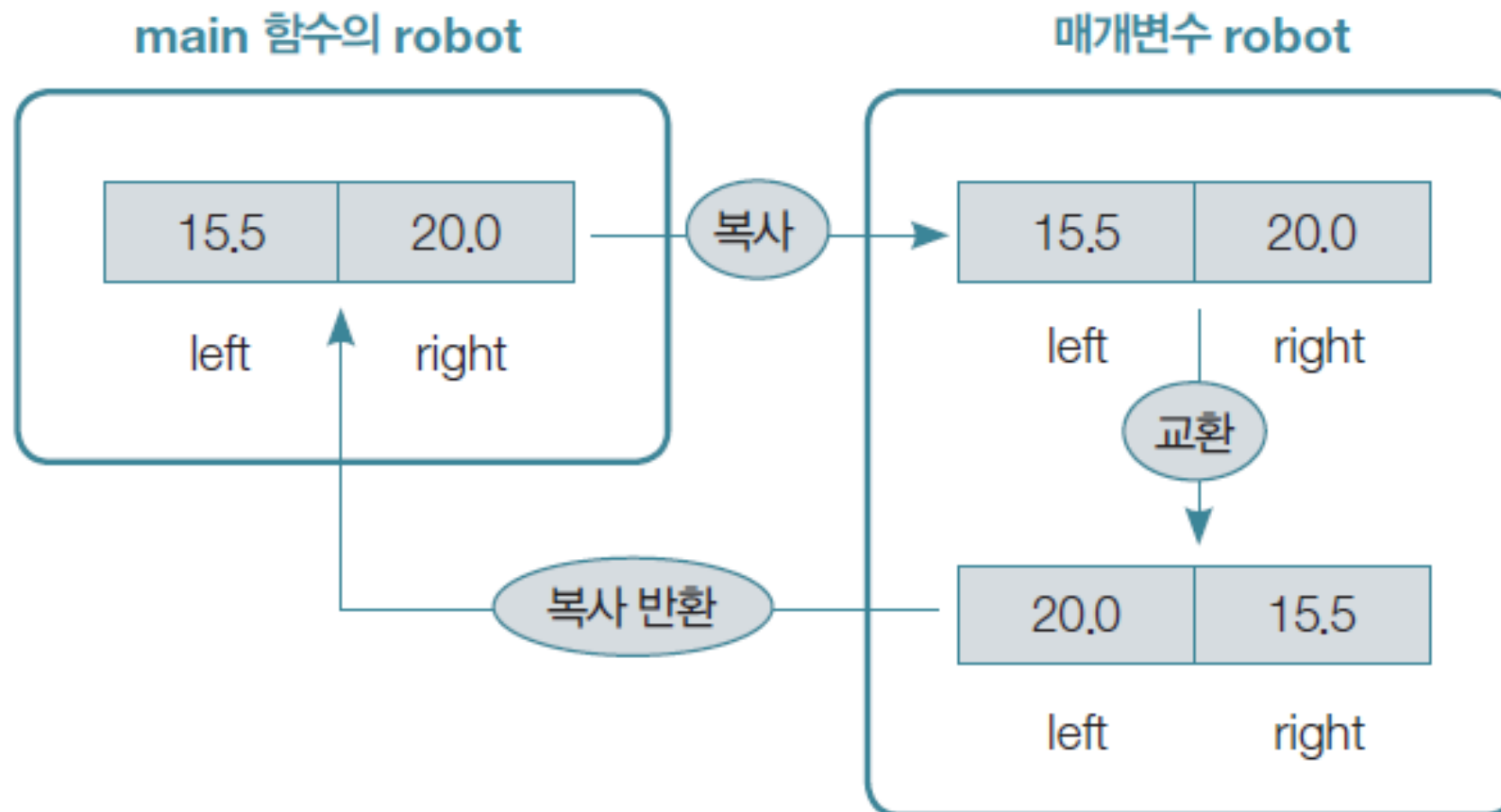
- ▶ 로봇의 좌, 우 시력 입력한 후에 함수로 두 값 바꿔 다시 출력
 - ▶ 함수 호출할 때 구조체 변수를 인수로
- ▶ 함수 내에서 두 값 바꿈
- ▶ 바뀐 두 값은 구조체 변수로 함께 반환
 - ▶ main 함수에서는 바뀐 결과 다시 돌려받음
 - ▶ 함수 선언에서 매개변수와 반환값의 형으로 모두 구조체 사용

```
struct vision exchange ( struct vision ) ;
```

매개변수와 반환형 모두 구조체

구조체 변수를 함수 매개변수에 사용하기

- ▶ robot의 값이 반환
 - ▶ main 함수의 robot에 복사
- ▶ main 함수에 있는 robot의 값이 바뀜



구조체 활용, 공용체, 열거형

.....

표 17-2 구조체 응용, 공용체와 열거형, typedef의 사용법

구분	기능	사용 예
구조체 응용	구조체 포인터	<code>struct student *ps = &s1; ps->num;</code>
	구조체 배열	<code>struct address list[5]; list[0].age = 23;</code>
	자기 참조 구조체	<code>struct list { int num; struct list *next; };</code>
공용체	형 선언	<code>union student { int num; double grade; };</code>
	변수 선언	<code>union student s1;</code>
	멤버 접근	<code>s1.num = 315; s1.grade = 3.4;</code>
	초기화	<code>union student s1 = { 315 };</code>
열거형	형 선언	<code>enum season { SPRING, SUMMER, FALL, WINTER };</code>
	변수 선언	<code>enum season ss;</code>
	변수 사용	<code>ss = SPRING;</code>
typedef	형 재정의	<code>typedef struct student Student;</code>
	재정의 형 사용	<code>Student s1;</code>

구조체 포인터와 -> 연산자

- ▶ 구조체 변수의 주소는 구조체 포인터에 저장
 - ▶ 구조체 변수 전체 가리킴
- ▶ 그 안에 여러 개의 변수를 멤버로 가질 수 있음
 - ▶ 그 자신은 단지 하나의 변수일 뿐
- ▶ 구조체 변수에 주소 연산자 사용
 - ▶ 특정 멤버의 주소가 아니라 구조체 변수 전체의 주소
 - ▶ 그 값 저장할 때는 구조체 포인터 사용

예제 17-7 구조체 포인터의 사용

```
1. #include <stdio.h>
2.
3. struct score                // 구조체 선언
4. {
5.     int kor;                // 국어 점수를 저장할 멤버
6.     int eng;                // 영어 점수
7.     int mat;                // 수학 점수
8. };
9.
10. int main(void)
11. {
12.     struct score yuni = {90, 80, 70}; // 구조체 변수 선언과 초기화
13.     struct score *ps = &yuni;         // 구조체 포인터에 주소 저장
14.
15.     printf("국어 : %d\n", (*ps).kor); // 구조체 포인터로 멤버 접근
16.     printf("영어 : %d\n", ps -> eng); // -> 연산자 사용
17.     printf("수학 : %d\n", ps -> mat);
18.
19.     return 0;
20. }
```

실행
결과

```
국어 : 90
영어 : 80
수학 : 70
```

구조체 포인터와 -> 연산자

- ▶ 구조체 포인터 선언
 - ▶ yuni의 주소로 초기화
- ▶ 구조체 포인터는 가리키는 자료형으로 구조체 사용하여 선언

가리키는 것은 struct score 구조체

```
struct score * ps = & yuni ;
```

ps는 포인터

구조체 포인터와 -> 연산자

➤ yuni는 하나의 변수

➤ 주소 연산을 수행하면 구조체 변수 전체의 주소 구해짐

➤ 이 값을 구조체 포인터 ps에 저장하면 ps가 구조체 변수 yuni를 가리킴

구조체 포인터 ps



구조체 변수 yuni



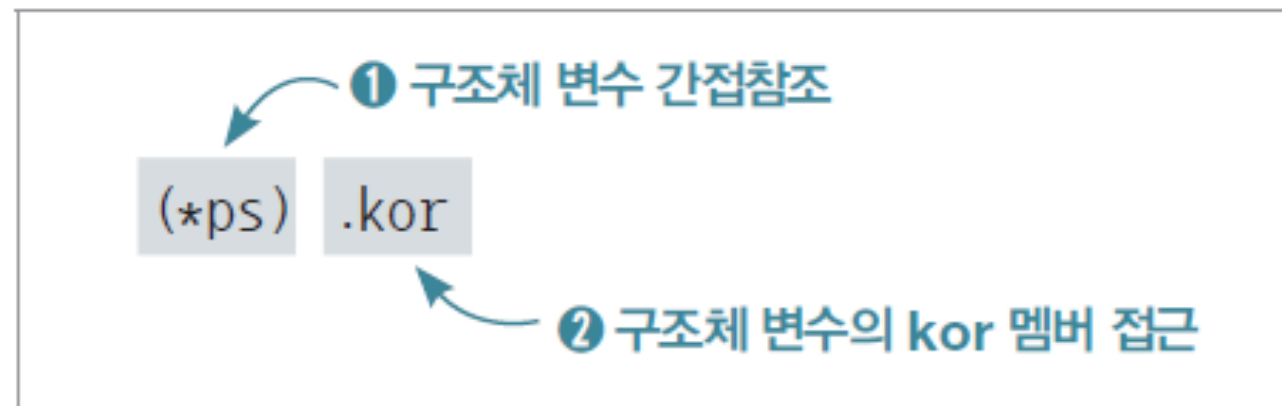
kor

eng

mat

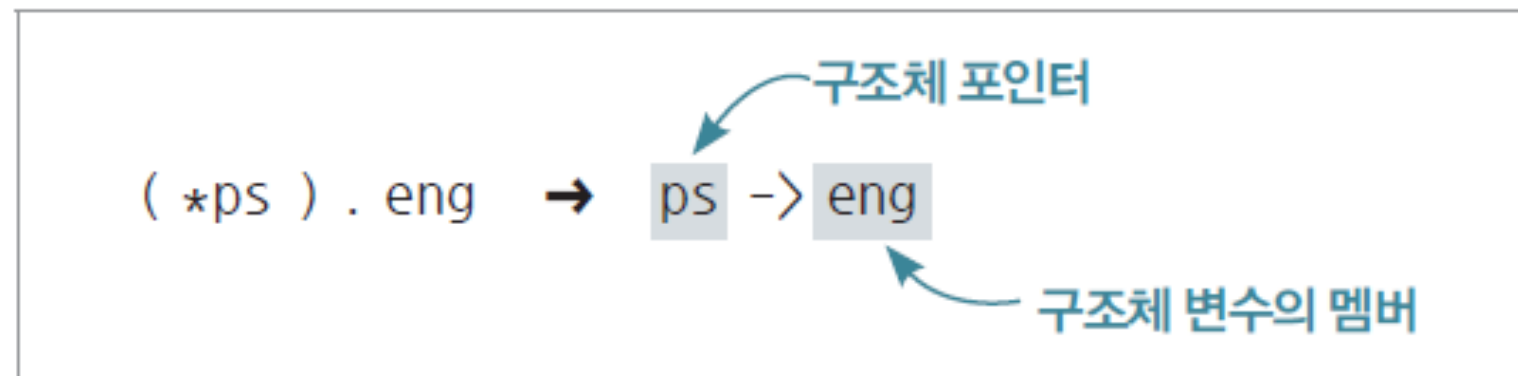
구조체 포인터와 -> 연산자

- ▶ ps에 * 연산을 수행하면 가리키는 변수인 yuni 사용 가능
 - ▶ yuni의 데이터는 멤버가 가지고 있음
 - ▶ 추가로 멤버에 접근하는 과정 필요
- ▶ 멤버 접근하는 . 연산자가 * 연산자보다 우선순위 높음
 - ▶ * 연산자가 먼저 수행될 수 있도록 * 연산자와 구조체 포인터 ps를 괄호로 묶어야 함!



구조체 포인터와 -> 연산자

- ▶ 괄호 쓰지 않으면 우선순위에 따라 ps.kor 먼저 수행
 - ▶ ps는 포인터로 kor 멤버가 없으므로 컴파일 에러 발생
- ▶ 매번 괄호 사용하는 것이 번거롭다면?
 - ▶ 같은 기능을 하는 -> 연산자 사용
- ▶ ps에 -> 연산자 사용
 - ▶ eng 멤버 쉽게 사용하는 예



구조체 배열

- ▶ 구조체 변수는 그 안에 여러 개의 멤버 가질 수 있음
- ▶ 그 자체는 하나의 변수로 취급
- ▶ 같은 형태의 구조체 변수가 많이 필요하면 배열 선언 가능

예제 17-8 구조체 배열을 초기화하고 출력

```
1. #include <stdio.h>
2.
3. struct address                // 주소록을 만들 구조체 선언
4. {
5.     char name[20];            // 이름을 저장할 멤버
6.     int age;                  // 나이를 저장할 멤버
7.     char tel[20];             // 전화번호를 저장할 멤버
8.     char addr[80];            // 주소를 저장할 멤버
9. };
10.
11. int main(void)
12. {
13.     struct address list[5] = { // 요소가 5개인 구조체 배열 선언
14.         {"홍길동", 23, "111-1111", "울릉도 독도"},
15.         {"이순신", 35, "222-2222", "서울 건천동"},
16.         {"장보고", 19, "333-3333", "완도 청해진"},
17.         {"유관순", 15, "444-4444", "충남 천안"},
18.         {"안중근", 45, "555-5555", "황해도 해주"}
19.     };
20.     int i;
21.
22.     for(i = 0; i < 5; i++)      // 배열 요소 수만큼 반복
23.     {
24.         printf("%10s%5d%15s%20s\n",    // 각 배열 요소의 멤버 출력
25.             list[i].name, list[i].age, list[i].tel, list[i].addr);
26.     }
27.
28.     return 0;
29. }
```

실행 결과

홍길동	23	111-1111	울릉도 독도
이순신	35	222-2222	서울 건천동
장보고	19	333-3333	완도 청해진
유관순	15	444-4444	충남 천안
안중근	45	555-5555	황해도 해주

구조체 배열

- ▶ 주소록 데이터 저장할 구조체 선언
 - ▶ 주소록은 기본적으로 많은 사람의 데이터 저장
 - ▶ address 구조체 변수가 많이 필요
- ▶ 구조체 변수들 배열로 선언
 - ▶ 배열 선언하면 배열 요소가 하나의 구조체 변수
 - ▶ 각 요소는 일정한 크기로 연속된 저장 공간에 할당

list 배열					배열 요소는 구조체 변수
list[0]	name	age	tel	addr	
list[1]	name	age	tel	addr	
list[2]	name	age	tel	addr	
list[3]	name	age	tel	addr	
list[4]	name	age	tel	addr	

구조체 배열

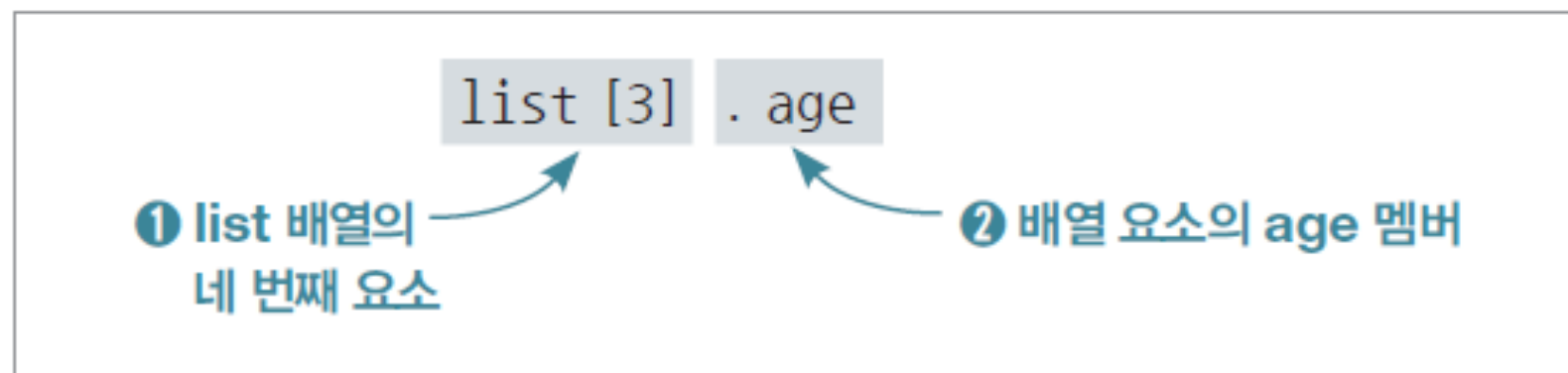
- ▶ 구조체 배열 초기화
 - ▶ 배열 초기화하는 방법 그대로 사용
- ▶ 배열의 요소가 구조체이므로 각각의 초깃값은 구조체를 초기화하는 형식 사용 (중괄호 쌍을 2개 사용)
 - ▶ 첫 번째 중괄호 쌍은 배열 초기화 괄호
 - ▶ 안쪽의 중괄호 쌍은 구조체 초기화 괄호
 - ▶ 구조체 초기화 괄호 안쪽에는 구조체 멤버의 형태에 맞게 초깃값 나열

```
struct address list[5] = { { "홍길동", 23, "111-1111", "울릉도 독도" }, ...
```

배열 초기화 괄호 구조체 초기화 괄호 첫 번째 구조체 변수의 초깃값

구조체 배열

- ▶ 각 배열 요소 사용할 때
 - ▶ 보통의 배열과 마찬가지로 첨자 사용
- ▶ 배열 요소가 구조체 변수이므로 멤버에 접근하기 위해 . 연산자 추가 사용
 - ▶ Ex) list 배열의 네 번째 요소의 age 멤버를 사용할 때는 다음과 같이 배열 요소에 . 연산자 사용



구조체 배열을 처리하는 함수

- ▶ 구조체 배열의 이름은 첫 번째 요소의 주소 - 구조체 변수
 - ▶ 구조체 배열의 이름을 인수로 주는 함수
 - ▶ 구조체 포인터를 매개변수로 선언
- ▶ 앞의 예제의 주소록을 출력하는 부분 함수 구현
 - ▶ 구조체 배열을 함수에서 구조체 포인터로 다루는 방법 습득

구조체 배열을 처리하는 함수

예제 17-9 함수에서 → 연산자를 사용하여 구조체 배열의 값 출력

```
1. #include <stdio.h>
2.
3. struct address                      // 주소록을 만들 구조체 선언
4. {
5.     char name[20];                  // 이름을 저장할 멤버
6.     int age;                        // 나이를 저장할 멤버
7.     char tel[20];                  // 전화번호를 저장할 멤버
8.     char addr[80];                 // 주소를 저장할 멤버
9. };
10.
11. void print_list(struct address *lp);
12.
13. int main(void)
14. {
15.     struct address list[5] = {      // 요소가 5개인 구조체 배열 선언
16.         {"홍길동", 23, "111-1111", "울릉도 독도"},
17.         {"이순신", 35, "222-2222", "서울 건천동"},
```



```

18.     {"장보고", 19, "333-3333", "완도 청해진"},
19.     {"유관순", 15, "444-4444", "충남 천안"},
20.     {"안중근", 45, "555-5555", "황해도 해주"}
21. };
22.
23. print_list(list);
24.
25. return 0;
26. }
27.
28. void print_list(struct address *lp) // 매개변수는 구조체 포인터
29. {
30.     int i;                        // 반복 제어 변수
31.
32.     for(i = 0; i < 5; i++)        // 배열 요소 수 만큼 반복
33.     {
34.         printf("%10s%5d%15s%20s\n", // 각 배열 요소의 멤버 출력
35.             (lp+i)->name, (lp+i)->age, (lp+i)->tel, (lp+i)->addr);
36.     }
37. }

```

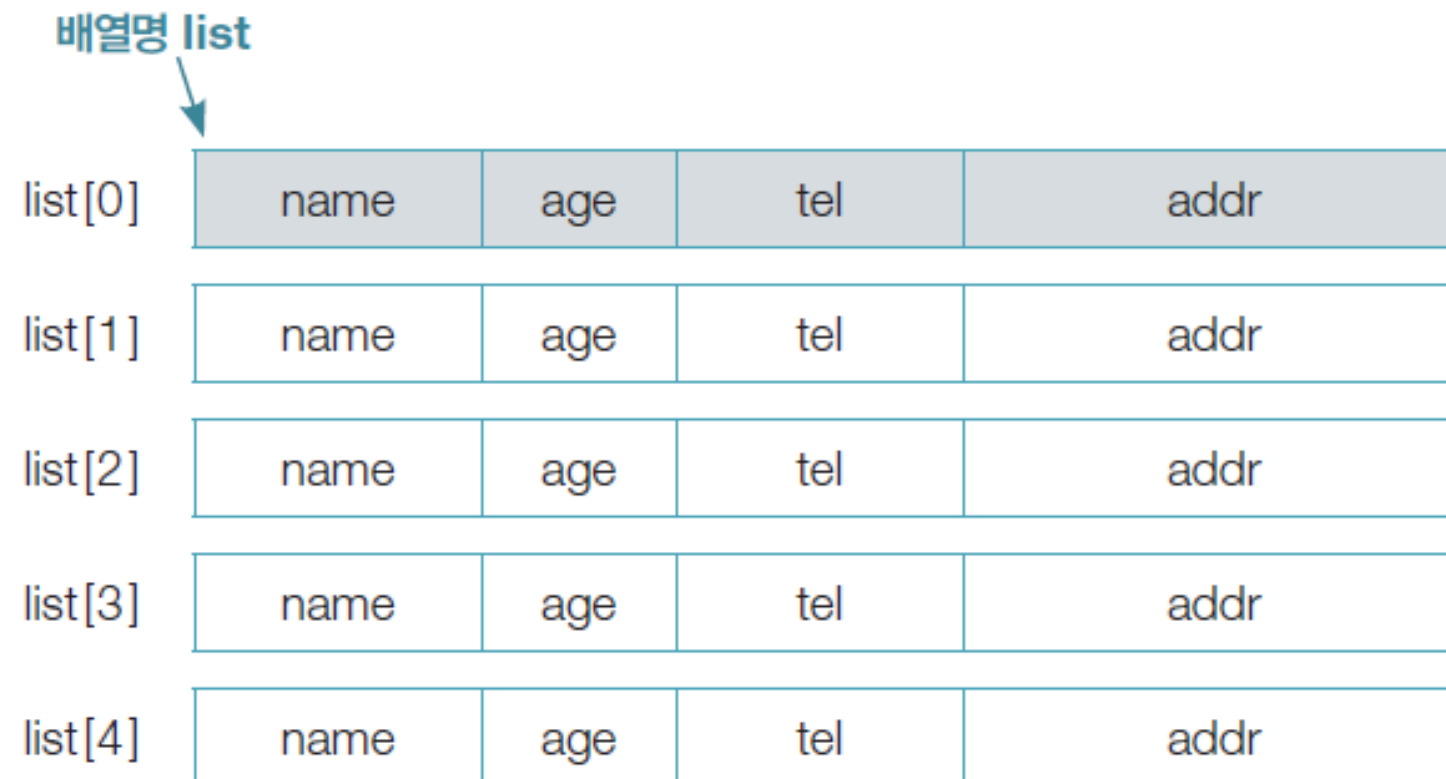
실행 결과

홍길동	23	111-1111	울릉도 독도
이순신	35	222-2222	서울 건천동
장보고	19	333-3333	완도 청해진
유관순	15	444-4444	충남 천안
안중근	45	555-5555	황해도 해주

구조체 배열을 처리하는 함수

- ▶ print_list 함수 호출
 - ▶ 배열명 list를 인수로 줌

▶ 배열명 list는 첫 번째 요소의 주소로 struct address 구조체 변수 가리킴



구조체 배열을 처리하는 함수

- ▶ print_list 함수의 매개변수로 struct address 구조체 가리키는 포인터 선언
 - ▶ 포인터가 배열명을 저장하면 배열명처럼 사용
 - ▶ 매개변수 lp로 각 배열요소 참조하고 멤버들 출력
- ▶ lp에 포인터 연산만 사용
 - ▶ 다음과 같이 배열 표현식도 쓸 수 있음

```
printf("%10s%5d%15s%20s\n", lp[i].name, lp[i].age, lp[i].tel, lp[i].addr);
```

구조체 배열을 처리하는 함수

▶ * 연산자 사용하는 표현

▶ 배열 요소를 참조하는 연산자인 대괄호([])

▶ . 연산자와 우선순위가 같고 연산 방향이 왼쪽에서 오른쪽이라 배열의 요소에 먼저 접근하고 해당 배열 요소의 name 멤버에 접근

▶ 배열 표현을 포인터 표현식으로 바꾸면 * 연산자가 . 연산자보다 우선순위 낮으므로 반드시 괄호 추가 사용

`lp [i] . name` → `(* (lp + i)) . name` → `(lp + i) -> name`

배열 표현 포인터 표현 → 연산자 사용

자기 참조 구조체

▶ 자기 참조 구조체란?

- ▶ 자신의 구조체 가리키는 포인터를 멤버로 가짐

- ▶ 개별적으로 할당된 구조체 변수들을 포인터로 연결

 - ▶ 관련된 데이터 하나로 묶어 관리

 - ▶ 자기 참조 구조체 사용

예제 17-10 자기 참조 구조체로 list 만들기

```
1. #include <stdio.h>
2.
3. struct list                // 자기 참조 구조체
4. {
5.     int num;                // 데이터를 저장하는 멤버
6.     struct list *next;      // 구조체 자신을 가리키는 포인터 멤버
7. };
8.
9. int main(void)
10. {
11.     struct list a = {10, 0}, b = {20, 0}, c = {30, 0}; // 구조체 변수 초기화
12.     struct list *head = &a, *current;                  // 헤드 포인터 초기화
13.
14.     a.next = &b;                                         // a의 포인터 멤버가 b를 가리킴
15.     b.next = &c;                                         // b의 포인터 멤버가 c를 가리킴
16.
17.     printf("head->num : %d\n", head->num); // head가 가리키는 a의 num 멤버 사용
18.     printf("head->next->num : %d\n", head->next->num); // head로 b의 num 멤버 사용
19.
20.     printf("list all : ");
21.     current = head;                                     // 최초 temp 포인터가 a를 가리킴
22.     while(current != NULL)                               // 마지막 구조체 변수까지 출력하면 반복 종료
23.     {
24.         printf("%d ", current->num); // temp가 가리키는 구조체 변수의 num 출력
25.         current = current->next;      // temp가 다음 구조체 변수를 가리키도록 함
26.     }
27.     printf("\n");
28.
29.     return 0;
30. }
```

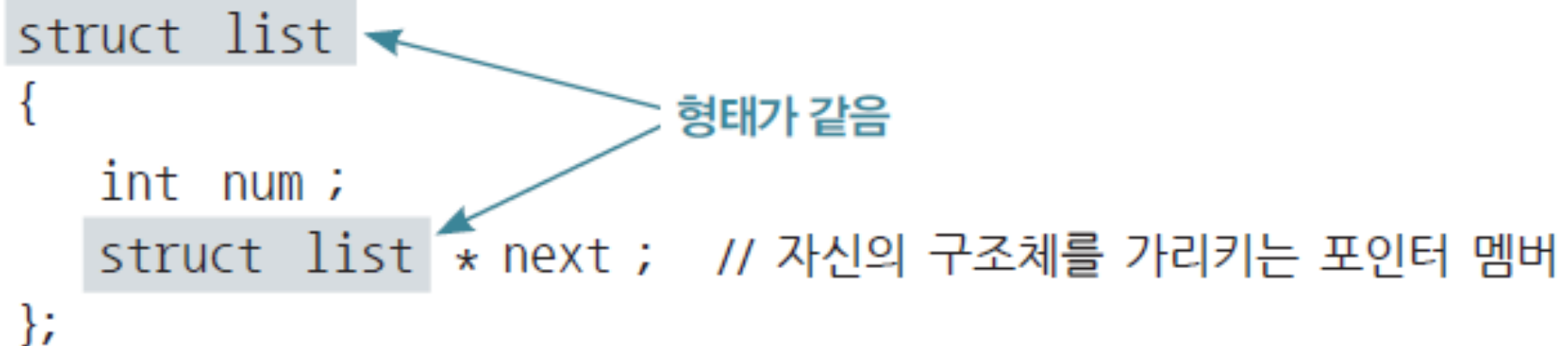
실행
결과

```
head-> num : 10
head-> next-> num : 20
list all : 10  20  30
.....
```

자기 참조 구조체

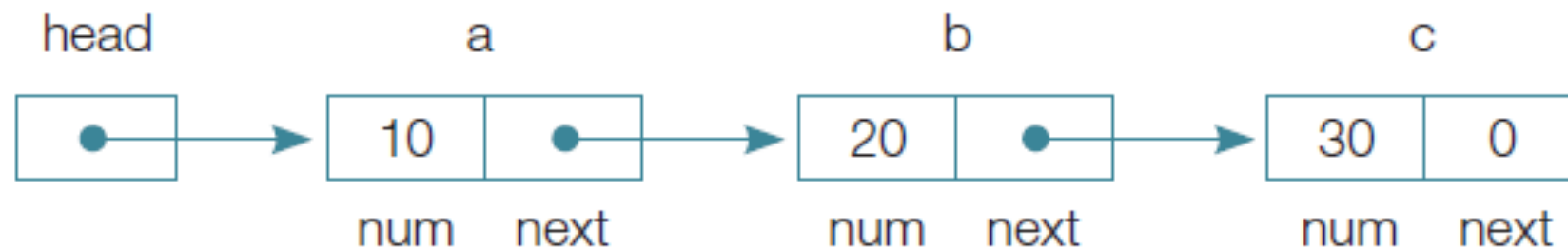
- ▶ 자신의 구조체 가리키는 포인터 멤버 포함
- ▶ struct list 구조체 변수
 - ▶ next 멤버로 다른 변수 가리킬 수 있음

```
struct list
{
    int num ;
    struct list * next ; // 자신의 구조체를 가리키는 포인터 멤버
};
```



자기 참조 구조체

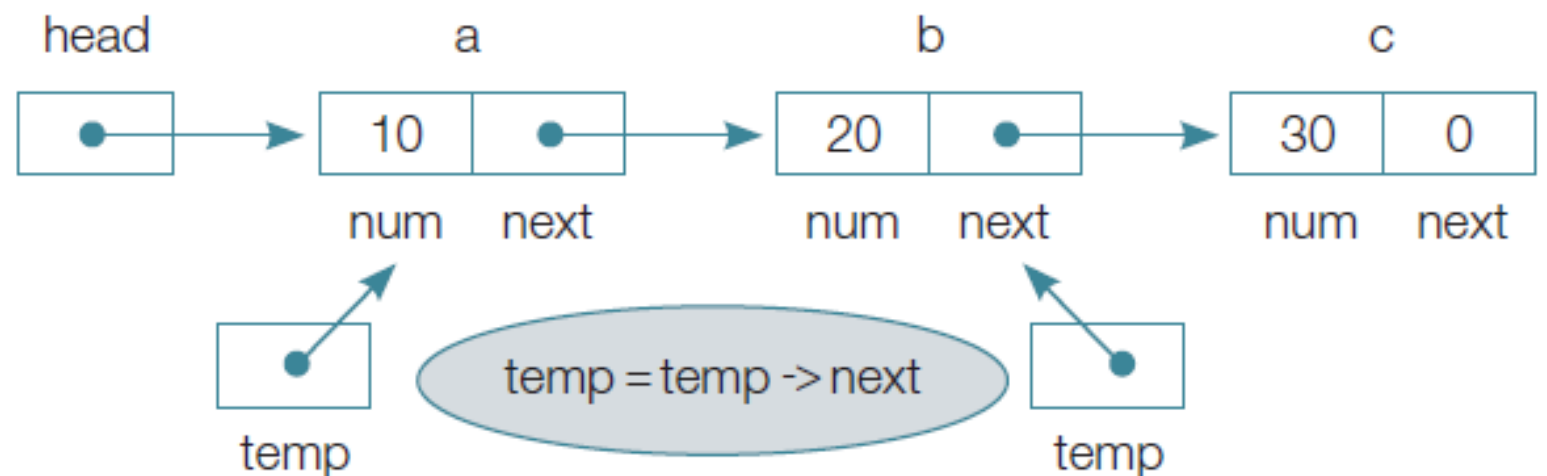
- ▶ 연결 리스트(linked list)
 - ▶ 구조체 변수를 포인터로 연결한 것
 - ▶ 첫 번째 변수의 위치만 알면 나머지 변수는 포인터 따라 모두 사용 가능
 - ▶ 첫 번째 변수의 위치 head 포인터에 저장해사용



자기 참조 구조체

- ▶ temp - 최초 a
 - ▶ 반복문 안에서 수행될 때마다 다음 변수 가리키며 모든 num 값 출력
- ▶ 최초 temp -> next는 a의 next
 - ▶ 그 값을 temp에 다시 저장하면 temp는 b를 가리키게
 - ▶ 다음 반복에서 temp -> next는 b의 next
 - ▶ 그 값을 temp에 저장하면 결국 temp는 c 가리킴

```
21. current = head;
22. while(current != NULL)
23. {
24.     printf("%d ", current->num);
25.     current = current ->next;
26. }
```



예제 17-11 공용체를 사용한 학번과 학점 데이터 처리

```
1. #include <stdio.h>
2.
3. union student                // 공용체 선언
4. {
5.     int num;                 // 학번을 저장할 멤버
6.     double grade;           // 학점을 저장할 멤버
7. };
8.
9. int main(void)
10. {
11.     union student s1 = {315}; // 공용체 변수의 선언과 초기화
12.
13.     printf("학번 : %d\n", s1.num); // 학번 멤버 출력
14.     s1.grade = 4.4;               // 학점 멤버에 값 대입
15.     printf("학점 : %.1lf\n", s1.grade);
16.     printf("학번 : %d\n", s1.num); // 학번 다시 출력
17.
18.     return 0;
19. }
```

.....

실행
결과

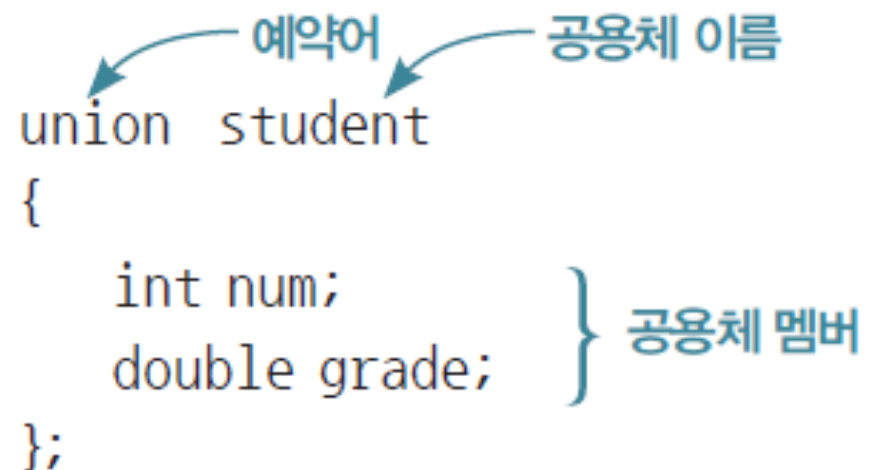
학번 : 315

학점 : 4.4

학번 : -1717986918 // 학번의 초깃값이 학점 멤버에 의해서 바뀜

공용체

- ▶ 공용체 선언
 - ▶ 공용체도 구조체와 마찬가지로 자료형 선언 후 사용
 - ▶ 공용체는 예약어 union 사용
 - ▶ 그 외의 다른 부분들은 구조체를 선언하는 형식과 같음

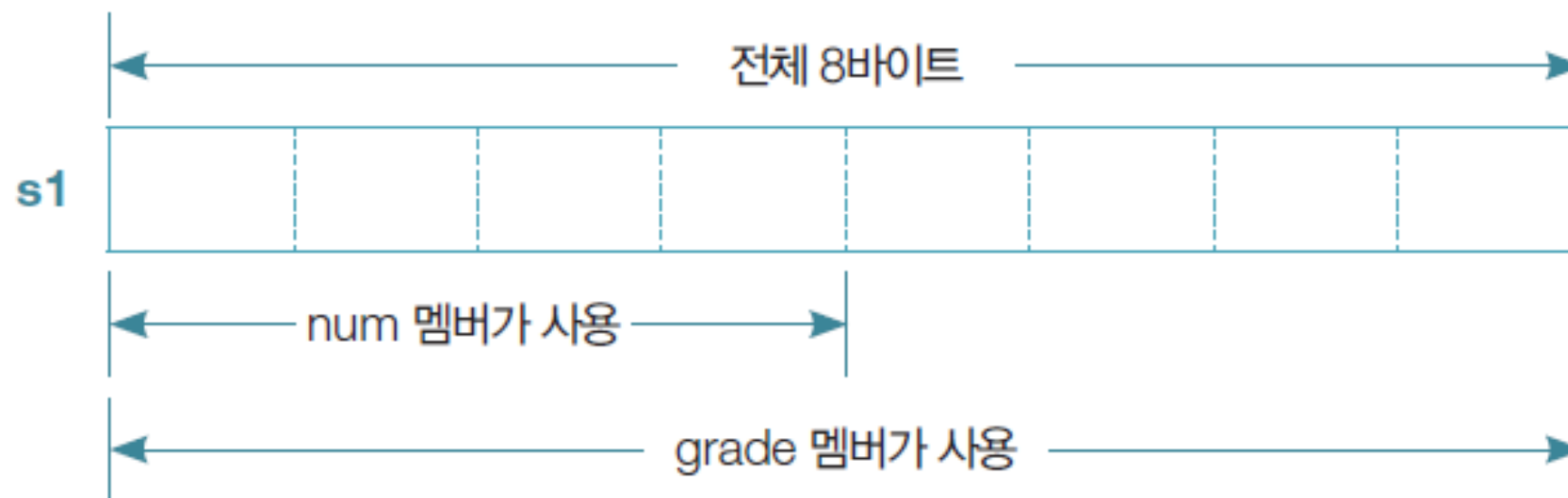


```
union student
{
    int num;
    double grade;
};
```

The diagram shows a C union declaration. The word 'union' is annotated with '예약어' (keyword) via a curved arrow. The word 'student' is annotated with '공용체 이름' (union name) via a curved arrow. The closing brace '}' is annotated with '공용체 멤버' (union member) via a curved arrow.

공용체

- ▶ 공용체 형으로 변수 선언
 - ▶ 할당되는 저장 공간의 크기의 규칙
 - ▶ 멤버 중에서 크기가 가장 큰 멤버로 결정
- ▶ union student의 변수 선언
 - ▶ double형 멤버의 크기인 8바이트의 저장 공간 할당
 - ▶ num과 grade 멤버가 하나의 공간 공유



공용체

- ▶ 공용체는 저장 공간 공유하는 점 외에 구조체와 다르지 않음
 - ▶ 멤버 참조, 배열, 포인터 사용하는 것은 구조체와 같음
- ▶ 저장 공간이 하나이므로 초기화하는 방법 다름
 - ▶ 공용체 변수의 초기화는 중괄호를 사용하여 첫 번째 멤버만 초기화
 - ▶ 첫 번째 멤버가 아닌 멤버 초기화할 때는 . 연산자로 멤버 직접 지정

```
union student a = {.grade = 3.4}; // grade 멤버를 3.4로 초기화
```

공용체

- ▶ 공용체 멤버 사용의 단점

- ▶ 공용체 멤버는 언제든지 다른 멤버에 의해 값이 변할 수 있으므로 항상 각 멤버의 값 확인

- ▶ 공용체 멤버 사용의 장점

- ▶ 여러 멤버가 하나의 저장 공간 공유하므로 메모리 절약

- ▶ 같은 공간에 저장된 값을 여러 가지 형태로 사용할 수 있는 장점

```

1. #include <stdio.h>
2.
3. enum season {SPRING, SUMMER, FALL, WINTER}; // 열거형 선언
4.
5. int main(void)
6. {
7.     enum season ss; // 열거형 변수 선언
8.     char *pc; // 문자열을 저장할 포인터
9.
10.    ss = SPRING; // 열거 멤버의 값 대입
11.    switch(ss) // 열거 멤버 판단
12.    {
13.        case SPRING: // 봄이면
14.            pc = "inline"; break; // 인라인 문자열 선택
15.        case SUMMER: // 여름이면
16.            pc = "swimming"; break; // 수영 문자열 선택
17.        case FALL: // 가을이면
18.            pc = "trip"; break; // 여행 문자열 선택
19.        case WINTER: // 겨울이면
20.            pc = "skiing"; break; // 스키 문자열 선택
21.    }
22.    printf("나의 레저 활동 => %s\n", pc); // 선택된 문자열 출력
23.
24.    return 0;
25. }

```

열거형

- ▶ 열거형 선언

- ▶ 예약어 enum과 열거형 이름을 짓고 괄호 안에 멤버를 콤마로 나열

```
enum season { SPRING, SUMMER, FALL, WINTER } ;
```

예약어

열거형 변수에 저장할 수 있는 기호화된 정수값들

열거형 이름

- ▶ 컴파일러는 멤버들을 0부터 차례로 하나씩 큰 정수로 바꿈

- ▶ Ex) SPRING은 0, SUMMER는 1, FALL은 2, WINTER는 3
 - ▶ 초깃값은 원하는 값으로 다시 설정할 수 있음

```
enum season { SPRING = 5, SUMMER, FALL = 10, WINTER } ;
```

TYPDEF를 사용한 형 재정의

- ▶ 구조체, 공용체, 열거형의 이름
 - ▶ 항상 struct와 함께 써야 하므로 불편
- ▶ 함수의 매개변수나 반환값의 형태에 쓰면 ?
 - ▶ 함수 원형이 복잡해짐
 - ▶ typedef 사용하면 자료형 이름에서 struct 생략
 - ▶ 짧고 쉬운 이름 사용

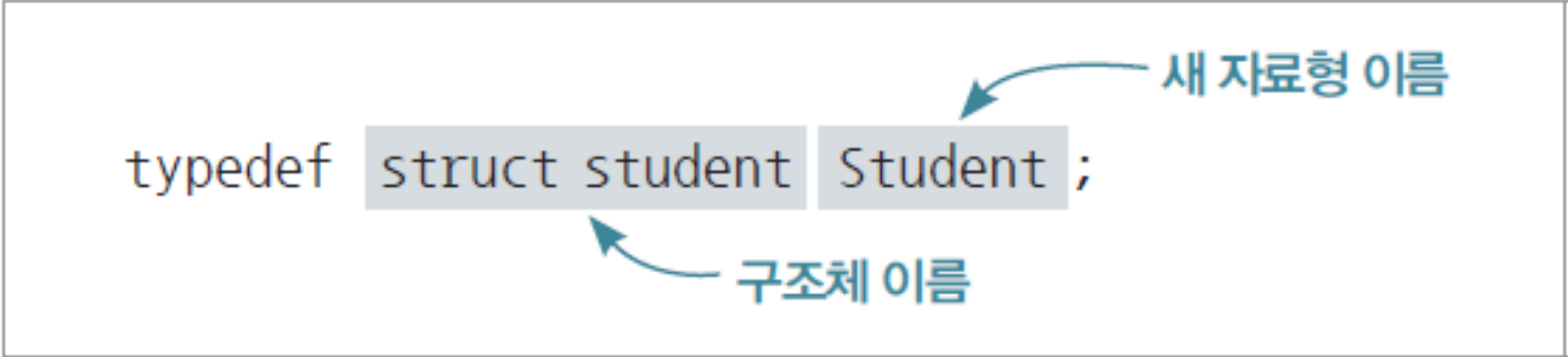
예제 17-13 typedef를 사용한 자료형 재정의

```
1. #include <stdio.h>
2.
3. struct student
4. {
5.     int num;
6.     double grade;
7. };
8. typedef struct student Student;    // Student형으로 재정의
9. void print_data(Student *ps);      // 매개변수는 Student형의 포인터
10.
11. int main(void)
12. {
13.     Student s1 = {315, 4.2};        // Student형의 변수 선언과 초기화
14.
15.     print_data(&s1);                // Student형 변수의 주소 전달
16.
17.     return 0;
18. }
19.
20. void print_data(Student *ps)
21. {
22.     printf("학번 : %d\n", ps -> num); // Student 포인터로 멤버 접근
23.     printf("학점 : %.1lf\n", ps -> grade);
24. }
```

실행
결과 학번 : 315
학점 : 4.2

typedef를 사용한 형 재정의

- ▶ 함수를 사용하여 구조체 변수의 값 출력
 - ▶ 구조체를 선언한 후 typedef로 자료형 재정의
 - ▶ typedef 뒤에 재정의할 자료형의 이름을 적고 뒤이어서 새로운 이름을 적음
 - ▶ 맨 뒤에 세미콜론(;)



```
typedef struct student Student ;
```

The diagram shows the code `typedef struct student Student ;` with two annotations. A blue arrow points from the text "새 자료형 이름" (New data type name) to the word `Student`. Another blue arrow points from the text "구조체 이름" (Structure name) to the words `struct student`.

TYPDEF를 사용한 형 재정의

- ▶ 재정의 이후 구조체 struct student를 Student로 간단히 쓸 수 있음
- ▶ 재정의하기 전 이름도 함께 사용 가능
 - ▶ 일반 변수 이름과 구분하기 위해 재정의된 자료형 이름을 대문자로 쓰기도 함
 - ▶ 재정의하기 전 자료형을 굳이 사용할 필요가 없다면 다음과 같이 형 선언과 동시에 재정의하기도 함

```
typedef struct                // 재정의될 것이므로 구조체 이름 생략
{
    int num;
    double grade;
} Student ;                  // 재정의된 자료형 이름
```

구조체, 공용체, 열거형을 사용한 프로그램

.....

예제 17-14 경품 당첨자 목록 관리 프로그램

```
1. #include <stdio.h>
2.
3. typedef union
4. {
5.     int ea;                // 개수
6.     double kg;            // 키로그램
7.     double liter;        // 리터
8. } Unit;                   // Unit 공용체형 재정의
9.
10. typedef struct
11. {
12.     char name[20];         // 당첨자 이름
13.     enum { EGG = 1, MILK, MEAT } kind; // 상품 종류, 열거형 멤버
14.     Unit amount;          // 지급양, 공용체 멤버
15. } Gift;                   // Gift 구조체 재정의
```

```
16.  
17. void print_list(Gift a);  
18.  
19. int main(void)  
20. {  
21.     Gift list[5];                // 5명 경품 지급 명단  
22.     int i;  
23.  
24.     for(i = 0; i < 5; i++)  
25.     {  
26.         printf("이름 입력 : ");  
27.         scanf("%s", list[i].name);  
28.         printf("품목 선택(1.계란, 2.우유, 3.고기) : ");  
29.         scanf("%d", &list[i].kind);    // 품목 선택  
30.  
31.         switch (list[i].kind)          // 선택 품목에 따라 지급 단위 결정  
32.         {  
33.             case EGG: list[i].amount.ea = 30; break;  
34.             case MILK: list[i].amount.liter = 4.5; break;  
35.             case MEAT: list[i].amount.kg = 0.6; break;  
36.         }  
37.     }  
38.  
39.     printf("# 세 번째 경품 당첨자...\n");  
40.     print_list(list[2]);  
41.  
42.     return 0;  
43. }  
44.
```

.....

```

45. void print_list(Gift a)
46. {
47.     printf(" 이름 : %s, 선택 품목 : ", a.name);
48.
49.     switch(a.kind)                // 선택 품목에 따라 출력
50.     {
51.         case EGG: printf("계란 %d개\n", a.amount.ea); break;
52.         case MILK: printf("우유 %.1lf리터\n", a.amount.liter); break;
53.         case MEAT: printf("고기 %.1lfkg\n", a.amount.kg); break;
54.     }
55. }

```

.....

실행 결과

```

이름 입력 : 홍길동
품목 선택(1.계란, 2.우유, 3.고기) : 1
이름 입력 : 이순신
품목 선택(1.계란, 2.우유, 3.고기) : 2
이름 입력 : 유관순
품목 선택(1.계란, 2.우유, 3.고기) : 2
이름 입력 : 조예경
품목 선택(1.계란, 2.우유, 3.고기) : 3
이름 입력 : 서하운
품목 선택(1.계란, 2.우유, 3.고기) : 2
# 세 번째 경품 당첨자...
이름 : 유관순, 선택 품목 : 우유 4.5리터

```

구조체, 공용체, 열거형을 사용한 프로그램

- ▶ 공용체를 선언하면서 바로 형 재정의
 - ▶ 공용체 이름은 생략할 수 있음
- ▶ 이 경우 재정의된 이름만 자료형으로 가능

```
union unit
{
    int ea;
    double kg;
    double liter;
};
typedef union unit Unit; // 형 재정의
```

공용체 선언



```
typedef union unit
{
    int ea;
    double kg;
    double liter;
} Unit;
```

생략 가능

