참고자료: chapter 9-1.

구조체

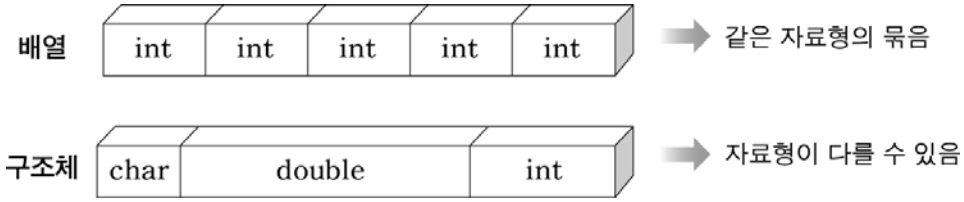
박 종 혁

Tel: 970-6702
Email: jhpark1@snut.ac.kr

출처: 뇌를 자극하는 C프로그래밍, 한빛미디어

❑ 구조체의 형 선언과 멤버 참조

- 구조체는 배열과 달리 다른 형태의 자료형도 묶어서 처리할 수 있다.



- 한 학생과 관련된 여러 형태의 데이터를 묶어서 처리할 수 있으므로 배열보다 효율적이다.

  – 5명의 학생에 대한 학번과 학점을 처리하는 예

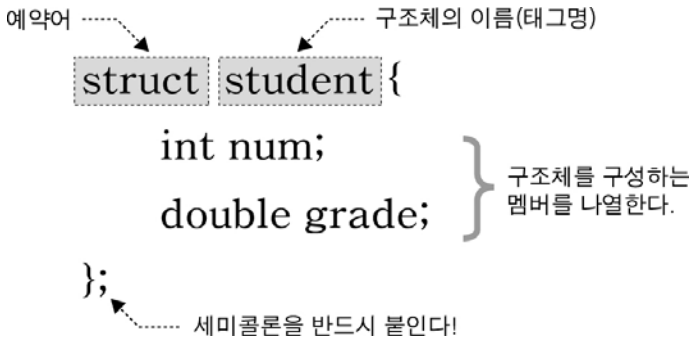
int형 배열(학번 저장)	1	2	3	4	5

double형 배열(학점 저장)	3.2	2.7	4.4	3.0	3.9

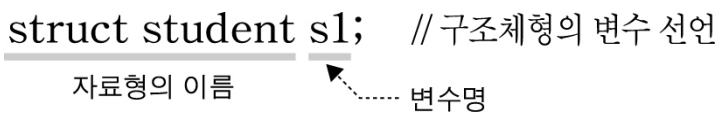
⇨ 학점순서로 정렬하면 학번 배열도 정렬해야 한다.

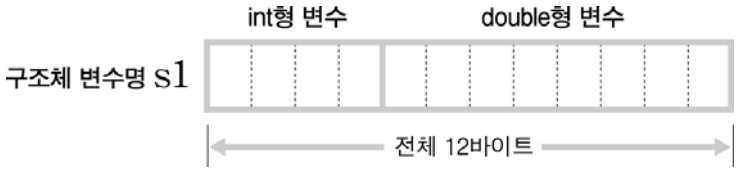
▶ 구조체는 형틀을 먼저 선언한다.

- 구조체를 사용하기 위해서는 먼저 원하는 구조체의 형태를 컴파일러에게 알려줘야 한다.
  - 학생 구조체를 선언하는 예



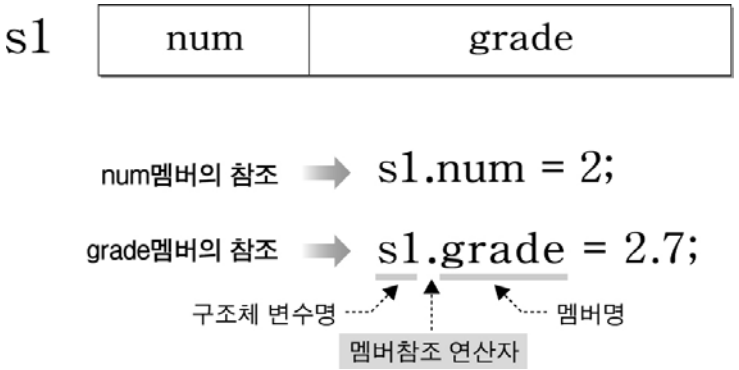
- 새로 만든 구조체 형으로 변수를 선언한다.





▶ 구조체의 변수는 멤버를 참조하여 사용한다.

- 배열은 배열요소의 형태가 같으므로 주소계산에 의해 각 멤버의 참조가 가능하지만 구조체는 각 멤버의 형태가 다르므로 멤버참조연산자(.)로 직접 멤버를 참조해야 한다.

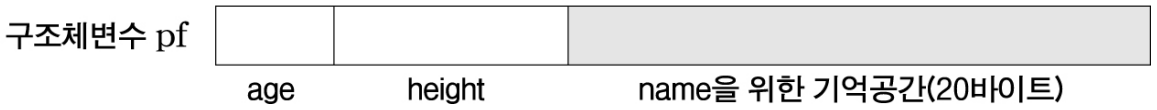



```
#include <stdio.h>

struct student{
    int num;
    double grade;
};

int main()
{
    struct student s1;  // 구조체 변수 선언

    s1.num=2;           // 구조체 멤버 참조
    s1.grade=2.7;
    printf("학번 : %d₩n", s1.num);
    printf("학점 : %.1lf₩n", s1.grade);

    return 0;
}
```

▶ 구조체에는 다양한 멤버를 사용할 수 있다.

- 구조체의 멤버로는 배열, 포인터변수, 이미 정의된 다른 구조체의 변수 등 모든 응용자료형을 사용할 수 있다.
- 멤버로 배열을 사용하는 예

```
struct profile{
    int age;              // 나이를 저장할 멤버
    double height;        // 키를 저장할 멤버
    char name[20];        // 이름을 저장할 멤버
};

struct profile pf;        // 구조체 변수의 선언
```



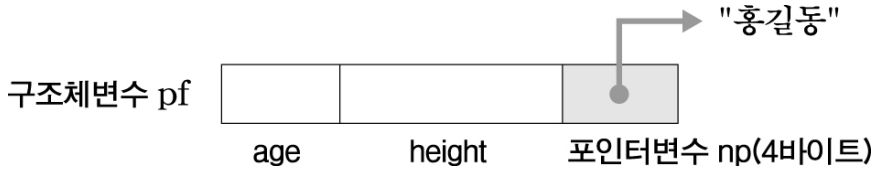
name멤버의 사용

```
strcpy(pt.name, "홍길동");
printf("%s₩n", pf.name);
```

▶ 구조체에는 다양한 멤버를 사용할 수 있다.

- 멤버로 포인터변수를 사용하는 예

```
struct profile{
    int age;                // 나이를 저장할 멤버
    double height;          // 키를 저장할 멤버
    char *np;               // 이름을 연결할 포인터변수 멤버
};

struct profile pf;          // 구조체 변수의 선언
pf.np="홍길동";             // 포인터변수 멤버에 문자열 연결
```

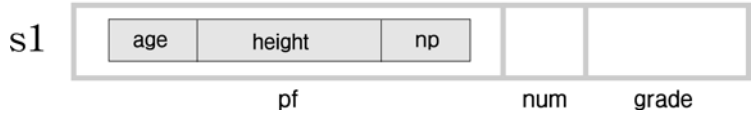


– 포인터변수를 멤버로 사용하는 경우 키보드로부터 문자열 입력은 불가능하다(문자열을 저장할 기억공간이 없다!).

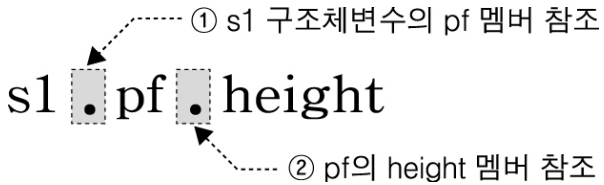
▶ 구조체에는 다양한 멤버를 사용할 수 있다.

- 구조체의 멤버로 다른 구조체의 변수를 사용하는 예

```
struct student{
    struct profile pf;      // 이미 선언된 구조체를 멤버로 사용
    int num;
    double grade;
};

struct student s1;          // 구조체 변수의 선언
```

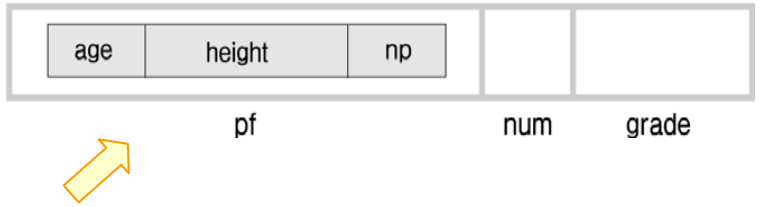


– 구조체의 멤버로 구조체를 사용한 경우 멤버참조연산자를 두 번 사용하여 멤버를 참조해야 한다.



▶ 다양한 멤버를 가진 구조체를 사용한 프로그램

```c
#include <stdio.h>

struct profile{
    int age;
    double height;
    char *np;
};

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




```c
int main()
{
    struct student s1;

    s1.pf.age=23;
    s1.pf.height=187.5;
    s1.pf.np="홍길동";

    s1.num=5;
    s1.grade=4.4;

    printf("이름 : %s₩n", s1.pf.np);
    printf("나이 : %d₩n", s1.pf.age);
    printf(" 키  : %.1lf₩n", s1.pf.height);
    printf("학번 : %d₩n", s1.num);
    printf("학점 : %.1lf₩n", s1.grade);
    return 0;
}
```

▶ 구조체변수의 초기화

- 구조체변수도 배열과 같이 중괄호를 사용하여 초기화 한다.

  – profile구조체 변수를 초기화하는 예

```
struct profile{
    int age;              // 나이를 저장할 멤버
    double height;        // 키를 저장할 멤버
    char name[20];        // 이름을 저장할 멤버
};

struct profile pf = { 23, 187.5, "홍길동" };
```

각 멤버의 형태에 맞는 데이터로 초기화한다.

- 구조체의 형틀선언, 변수선언, 초기화를 동시에 할 수 있다.

```
struct profile{
    int age;
    double height;
    char name[20];
} pf = { 23, 187.5, "홍길동" };
```

구조체의 형 선언, 변수 선언
초기화를 동시에 할 수 있다.

▶ 학생 데이터를 구조체로 처리하는 프로그램 예

- 세 명의 데이터 중에서 학점이 가장 높은 학생의 학번, 이름, 학점을 출력한다.

```
#include <stdio.h>

struct student{             // 학생 데이터에 대한 구조체 선언
    int num;                // 학번을 저장할 멤버
    char name[20];          // 이름을 저장할 멤버
    double grade;           // 학점을 저장할 멤버
};

int main()
{
    struct student s1={315, "홍길동", 2.4},      // 구조체 변수의 선언과 초기화
                   s2={247, "이순신", 3.7},
                   s3={330, "세종대왕", 4.4};
    struct student max;     // 학점이 가장 높은 학생의 데이터를 저장할 구조체 변수

    max=s1;                 // 처음에 홍길동의 학점이 가장 높다고 가정한다.
    if(s2.grade > max.grade) max=s2;          // 각 학생의 학점을 비교하여 학점이 가장 높은
    if(s3.grade > max.grade) max=s3;          // 학생의 데이터가 max에 저장되도록 한다.

    printf("학번 : %d₩n", max.num);           // 학점이 가장 높은 학생의 각 데이터를 출력한다.
    printf("이름 : %s₩n", max.name);
    printf("학점 : %.1lf₩n", max.grade);

    return 0;
}
```

▶ 구조체는 대입연산이 가능하다.

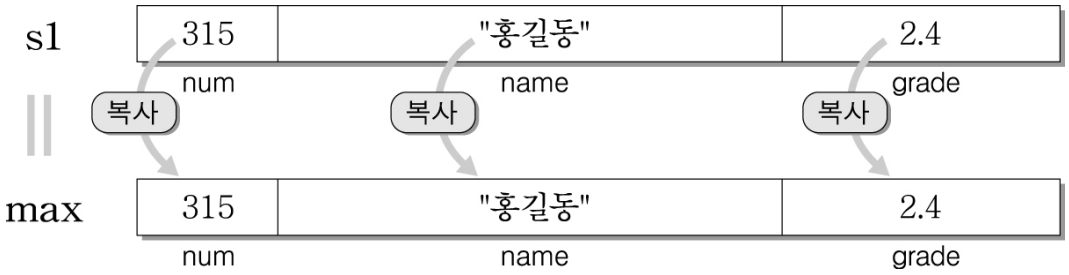
- 배열은 대입연산이 불가능하다.

```
int ary1[5]={10,20,30,40,50};
int ary2[5];

ary2 = ary1;   // 불가능하다.
```

⇨ 각 배열요소를 일일이 대입해야 한다!

- 구조체변수는 대입연산으로 모든 멤버들을 복사할 수 있다.

```
struct student s1={315, “홍길동”, 2.4};
struct student max;
max=s1;
```



▶ 구조체변수를 함수의 전달인자로 사용하자.

- 구조체는 대입연산이 가능하므로 함수의 전달인자로 줄 수 있다.

  – 최고학점의 학생 데이터를 함수로 출력해 보자.

함수의 호출

```
max_prn(max);   // 구조체변수를 전달인자로 주고 호출한다.
```

함수의 정의

```
void max_prn(struct student max)     // 매개변수는 구조체변수를 선언한다.
{
    printf("학번 : %d₩n", max.num);
    printf("이름 : %d₩n", max.name);
    printf("학점 : %d₩n", max.grade);
}
```

▶ 구조체를 리턴하는 함수

- 구조체를 사용하면 포인터 없이도 두 변수의 값을 바꿀 수 있다.

  – 로보트의 양쪽 시력을 바꾸는 프로그램 예

```
#include <stdio.h>

struct vision{
    double left;
    double right;
};

struct vision exchange(struct vision);

int main()
{
    struct vision robot;

    printf("로보트의 시력을 입력하세요(좌, 우) : ");
    scanf("%lf%lf", &robot.left, &robot.right);
    robot=exchange(robot);
    printf("바뀐 로보트의 시력(좌, 우) : %.1lf, %.1lf₩n", robot.left, robot.right);

    return 0;
}
```

```
struct vision exchange(struct vision robot)
{
    double temp;

    temp=robot.left;
    robot.left=robot.right;
    robot.right=temp;

    return robot;
}
```

❑ 구조체 배열

- 구조체 변수가 많이 필요하면 배열로 선언하여 사용한다.
  - 주소록을 만드는 프로그램의 예(5명의 주소를 저장할 경우).

```
struct address{
    char name[20];          // 이름을 저장할 멤버
    int age;                // 나이를 저장할 멤버
    char tel[20];           // 전화번호를 저장할 멤버
    char addr[80];          // 주소를 저장할 멤버
};
```

```
struct address list[5];    // 구조체배열의 선언
```

자료형 / 배열명 / 배열요소의 개수

list 배열

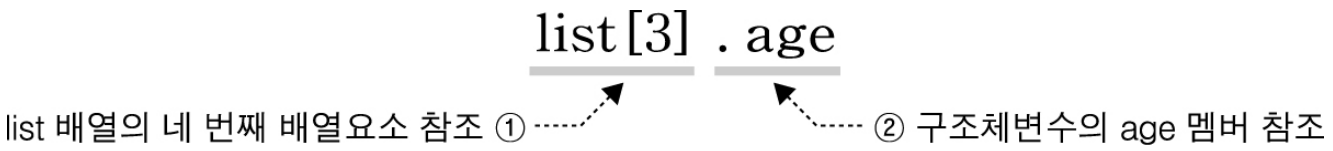
list[0]	name	age	tel	addr
list[1]	name	age	tel	addr
list[2]	name	age	tel	addr
list[3]	name	age	tel	addr
list[4]	name	age	tel	addr

배열요소는 구조체변수

▶ 구조체 배열의 배열요소 참조

- 구조체 배열의 참조된 배열요소는 구조체변수이므로 실제 데이터를 저장하기 위해서는 다시 멤버를 참조해야 한다.

  - list배열의 네 번째 배열요소의 age멤버를 참조할 때



  - list배열의 네 번째 배열요소의 모든 멤버에 값을 저장

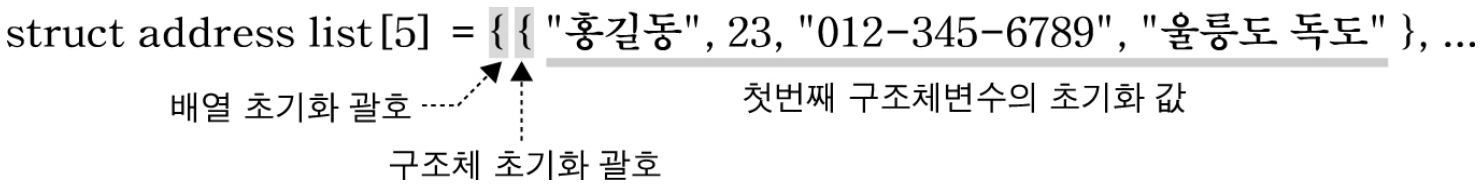
```
strcpy(list[3].name, “홍길동”);
list[3].age=23;
strcpy(list[3].tel, “012-345-6789”);
strcpy(list[3].addr, “울릉도 동남쪽 외로운 섬 독도”);
```

▶ 구조체 배열의 초기화

- 배열의 초기화 방법을 그대로 적용한다. 단, 배열의 요소가 구조체이므로 각각의 초기값은 구조체 초기화 형식을 사용한다.

  - list배열의 초기화

```
struct address list[5] = { { “홍길동”, 23, “012-345-6789”, “울릉도 독도”},
                           { “이순신”, 35, “111-222-3333”, “서울 건천동”},
                           { “장보고”, 19, “222-333-4444”, “완도 청해진”},
                           { “유관순”, 15, “333-444-5555”, “충남 천안”},
                           { “안중근”, 45, “444-555-6666”, “황해도 해주”} };
```



▶ 구조체 배열을 초기화하고 출력하는 예

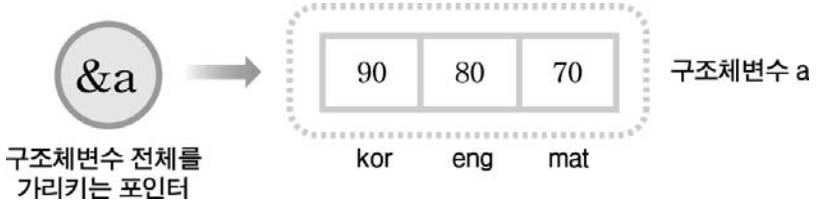
– 배열요소의 값을 반복문으로 출력한다.

```
#include <stdio.h>

struct address {
    char name[20];
    int age;
    char tel[20];
    char addr[80];
};

int main()
{
    struct address list[5]={{"홍길동", 23, "012-345-6789", "울릉도 독도"},
                            {"이순신", 35, "111-222-3333", "서울 건천동"},
                            {"장보고", 19, "222-333-4444", "완도 청해진"},
                            {"유관순", 15, "333-444-5555", "충남 천안"},
                            {"안중근", 45, "444-555-6666", "황해도 해주"}};
    int i;

    for(i=0; i<5; i++){        // 배열요소가 5개이므로 5번 반복
        printf("%10s%5d%15s%20s₩n", list[i].name, list[i].age, list[i].tel, list[i].addr);
    }
    return 0;
}
```

❑ 구조체 포인터

- 구조체 변수도 하나의 변수이므로 포인터를 구하여 사용할 수 있다.

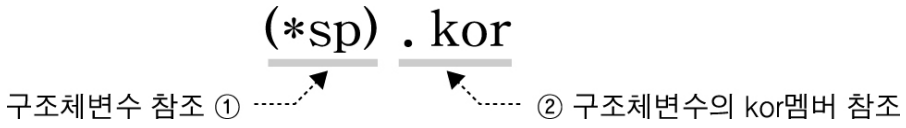
```
struct score{
    int kor, eng, mat;     // 같은 자료형의 멤버는 함께 선언할 수 있다.
};

struct score a={90, 80, 70};     // 구조체변수의 선언과 초기화
```



- 구조체 포인터변수를 선언하고 포인터를 저장하자.

▶ 구조체 포인터변수를 사용한 멤버의 참조

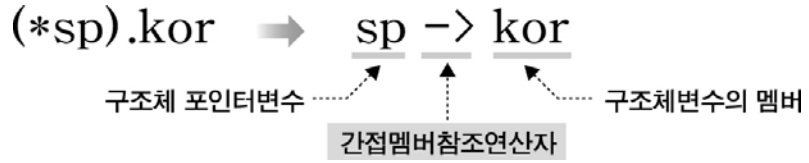
- 포인터변수로 멤버를 참조하기 전에 먼저 구조체변수를 참조한다.
  - 멤버참조연산자(.)가 참조연산자(*)보다 우선순위가 높으므로 괄호가 필요하다.




```
#include <stdio.h>
struct score{                    // 구조체의 형틀 선언
    int kor, eng, mat;
};
int main()
{
    struct score a={90, 80, 70};          // 구조체변수의 선언과 초기화
    struct score *sp=&a;                  // 포인터변수에 포인터 저장
    printf("국어 : %d₩n", (*sp).kor);      // 포인터변수로 구조체변수의
    printf("영어 : %d₩n", (*sp).eng);      // 각 멤버를 참조하여 출력한다.
    printf("수학 : %d₩n", (*sp).mat);
    return 0;
}
```

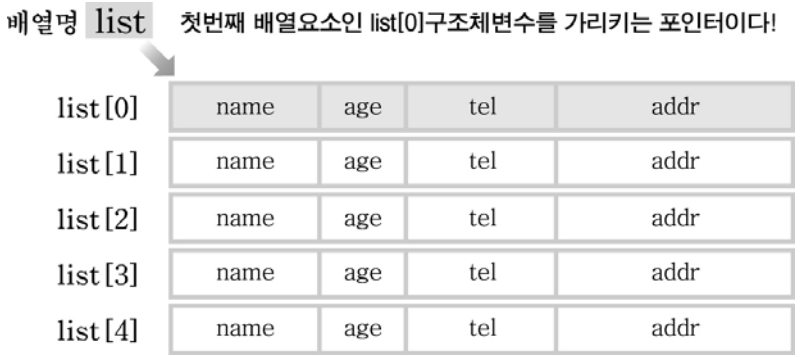
▶ 간접멤버참조연산자(->)

- 포인터변수가 가리키는 구조체변수의 멤버를 간단히 참조할 때 간접멤버참조연산자(->)를 사용한다.



- list 구조체배열의 데이터를 출력하는 함수를 만들자.

```
struct address list[5] = { ... };      // list는 구조체배열의 배열명
list_prn(list);                        // 배열명을 주고 함수를 호출한다.
```



▶ 간접멤버참조연산자(->)

- 구조체배열의 배열명은 첫 번째 구조체를 가리키는 포인터이므로 배열명을 받는 매개변수는 구조체 포인터변수이어야 한다.

```
void list_prn(struct address *lp)
{
    int i;
    for(i=0; i<5; i++){
        printf("%10s%5d%15s%20s₩n",
            lp[i].name, lp[i].age, lp[i].tel, lp[i].addr);
    }
}
```

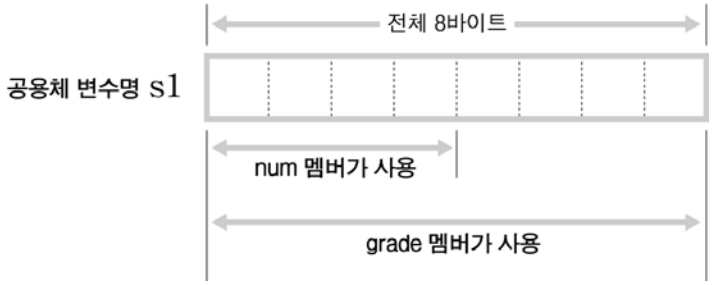
- 매개변수 lp가 구조체 포인터변수이므로 간접멤버참조연산자로 멤버를 참조할 수 있다.

```
lp[i].name      ➡  (*(lp+i)).name      // 배열표현을 포인터표현으로 바꾼다.
(*(lp+i)).name  ➡  (lp+i)->name        // 간접멤버참조연산자를 사용한다.
```

❑ 공용체

- 공용체는 모든 멤버가 하나의 기억공간을 공유하며 기억공간의 크기는 멤버 중에서 크기가 가장 큰 멤버로 결정된다.

```
union student{        // union예약어를 사용하여 선언한다.
    int num;
    double grade;
};

union student s1;    // 공용체 변수의 선언
```



- 공용체의 멤버참조, 배열, 포인터의 사용은 구조체와 같다. 단, 초기화 할 때는 첫 번째 멤버만 초기화가 가능하다.

```
union student s1 = {315};    // 첫 번째 멤버인 학번만 초기화 가능하다.
```

▶ 공용체의 장단점

- 공용체는 모든 멤버가 하나의 기억공간을 공유하므로 메모리를 절약할 수 있지만 다른 멤버에 의해서 데이터가 변질될 위험이 있다.

```
#include <stdio.h>

union student{
    int num;
    double grade;
};

int main()
{
    union student s1={315};

    printf("학번 : %d₩n", s1.num);
    s1.grade=4.4;
    printf("학점 : %.1lf₩n", s1.grade);
    printf("학번 : %d₩n", s1.num);

    return 0;
}
```

출력 결과

```
학번 : 315
학점 : 4.4
학번 : -1717986918
```

학번의 초기값이 학점 멤버에 의해서 변질되었다.

❑ 열거형

- 열거형은 기억공간에 저장될 데이터의 집합을 정의한다.

```
enum  season  { spring, summer, fall, winter } ;
```

예약어 — 열거형의 이름 — 열거형 변수에 저장될 데이터의 종류를 나열한다.

- 열거형은 읽기 쉬운 프로그램을 작성하는데 도움이 된다.

```
#include <stdio.h>

enum season {spring, summer, fall, winter};

int main()
{
    enum season ss;      // 열겨형 변수 선언
    char *cp;            // 문자열을 저장할 포인터변수
    ss=spring;           // 열거멤버의 값을 변수에 대입

    switch(ss){
    case spring:
        cp="inline"; break;
    case summer:
        cp="swimming"; break;
    case fall:
        cp="trip"; break;
    case winter:
        cp="skiing"; break;
    }
    printf("나의 레저활동 => %s₩n", cp);
    return 0;
}
```

❑ typedef를 사용한 형 재정의

- typedef를 사용하여 응용자료형을 간단하게 사용할 수 있다.

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

구조체의 형 선언

```
typedef  sturct student  Student ;     // 자료형의 재정의
```

구조체의 자료형 — 새로운 자료형의 이름

```
Student s1;      // 재정의된 자료형으로 간단하게 구조체변수 선언
```

- 형 선언과 동시에 재정의하는 방법도 가능하다.

```
typedef struct {          // 재정의될 것이므로 자료형의 이름이 필요 없다.
    int num;
    double grade;
} Student;                // 새로운 자료형의 이름을 바로 적어준다.
```

▶ typedef를 사용한 프로그램 예

```
#include <stdio.h>

typedef struct {                    // 구조체의 선언과 동시에 자료형의 재정의한다.
    int num;
    double grade;
} Student;

void data_prn(Student *);           // 함수의 선언, 매개변수는 Student형의 포인터변수

int main()
{
    Student s1={315, 4.2};          // Student형의 변수 선언과 초기화
    data_prn(&s1);                  // Student 변수의 포인터를 전달한다.
    return 0;
}

void data_prn(Student *sp)          // Student형을 가리키는 포인터변수
{
    printf("학번 : %d₩n", sp->num);      // 구조체포인터변수로 멤버 참조하기
    printf("학점 : %.1lf₩n", sp->grade);
}
```