9장 구조체와 공용체

박 종 혁 교수

UCS Lab

Tel: 970-6702

Email: jhpark1@seoultech.ac.kr

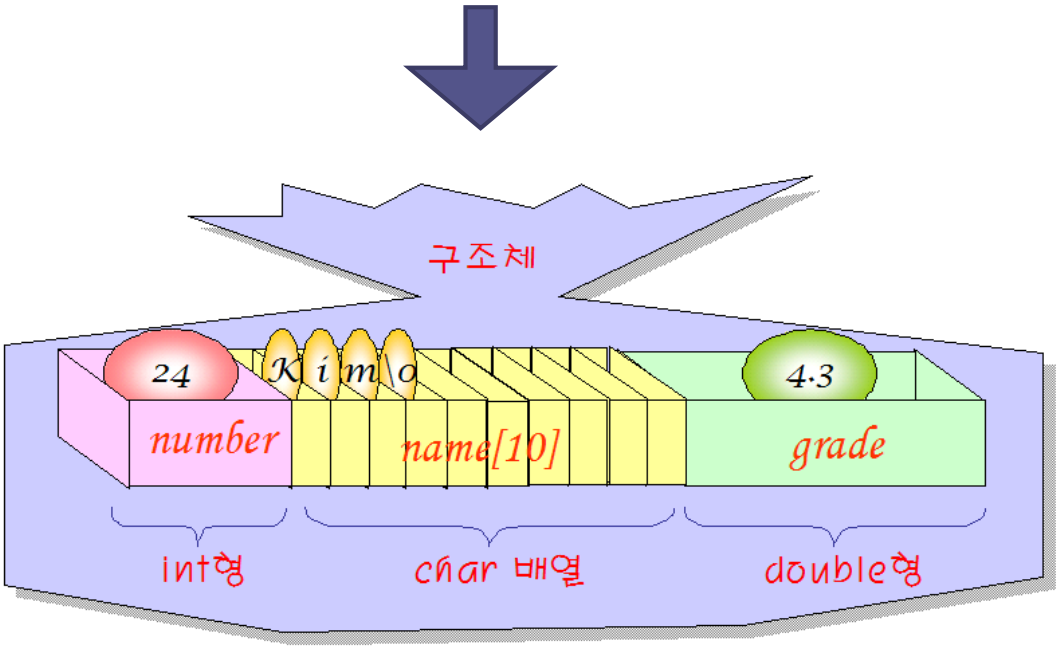
구조체와 공용체

- **C 언어의 확장 방법**
  - 매크로와 라이브러리
  - 사용자 정의 자료형 (배열, 구조체, 공용체)

    // 파생형 자료형으로 쓰기도함

구조체의 필요성

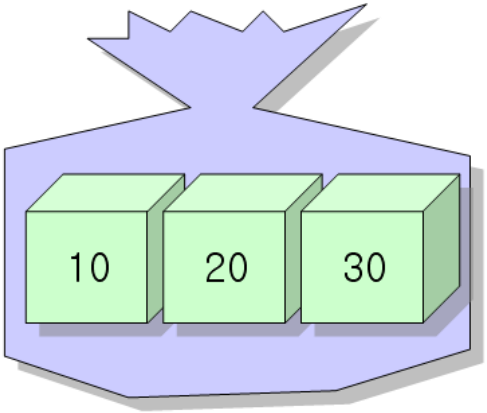
```
int number;
char name[10];
double grade;
```



구조체를 사용하면 변수들을 하나로 묶을 수 있습니다.

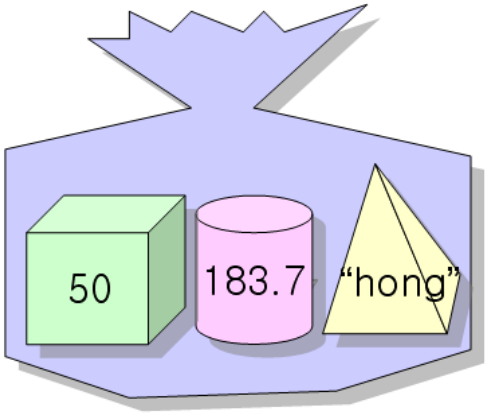
구조체와 배열

- 구조체 vs 배열



배열

같은 타입의 집합



구조체

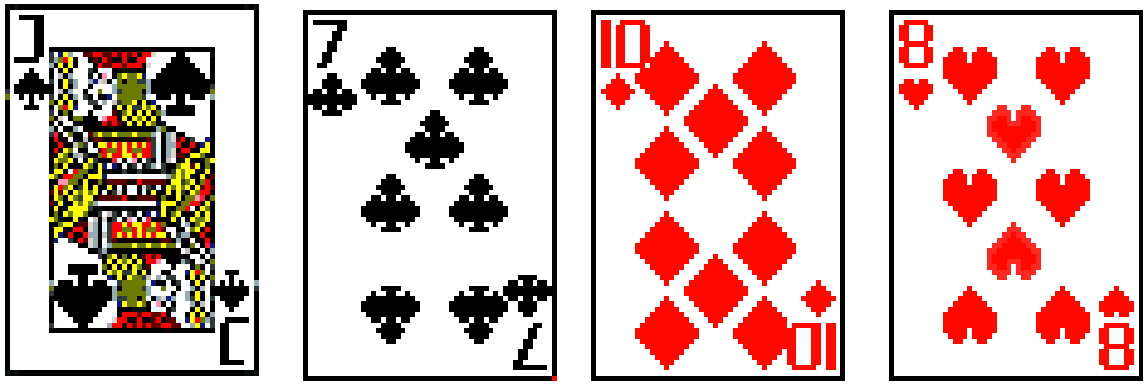
다른 타입의 집합

구조체의 장점

- 통계자료나 성적 등과 같이 관련 있는 서로 다른 자료형을 한 덩어리(집합)로 만들어 처리
- 연관 있는 데이터를 한 덩어리로 묶으면 프로그램의 관리와 구현이 용이
- 순서대로 정렬(sort)하는 경우 구조체 단위로 처리되므로 간단
- 네트워크 프로그램을 작성할 때 소켓이나 헤더(header)의 format(형식)을 구조체로 묶어서 처리
- 함수를 반환할 때 한 개의 데이터가 아닌 구조체 단위로 묶어서 전달할 수 있음

구조체

- 서로 다른 형의 변수들을 하나로 묶어 주는 방법
- 예제 - 카드



구조체

- **예제 - 카드**
  - 각 카드는 고유의 **무늬**와 **숫자**를 가짐
    →구조체를 사용하여 표현하면 효율적
  - 카드를 위한 구조체 선언

```
struct card {
    int   pips;
    char  suit;
};
```

구조체 선언

- **예제 - 카드**

```
struct card {
    int   pips;
    char  suit;
};
```

  - struct : 키워드
  - card : 구조체 태그 이름
  - pips, suit : 구조체 멤버

* 이것은 **struct card 형의 정의**이고, 변수 선언은 아님

구조체 변수 선언 방법 1

- **struct card 형 변수 c1, c2**

```
struct card {
    int   pips;
    char  suit;
};
struct  card    c1, c2;
```

구조체 변수 선언 방법 2

- **struct card 형 변수 c1, c2**

```
struct card {
    int   pips;
    char  suit;
} c1, c2;
```

구조체 변수 선언 방법 3

- **struct card 형 변수 c1, c2**

```
struct card {
    int    pips;
    char   suit;
};
typedef  struct card   card;
card     c1, c2;
```

구조체 변수 선언 방법 4

- **struct card 형 변수 c1, c2**

```
typedef struct {
    int     pips;
    char    suit;
} card;
card  c1, c2;
```

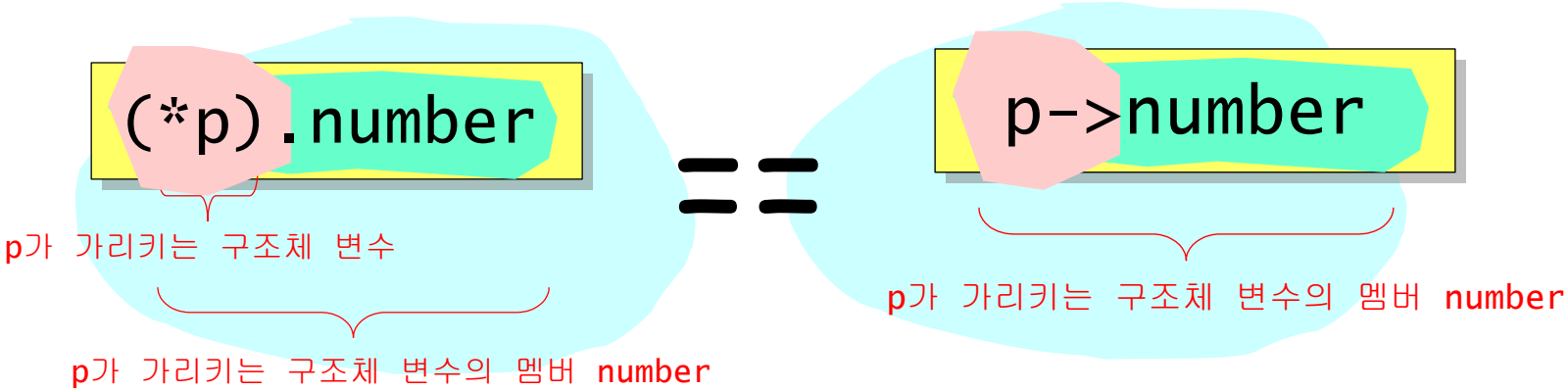
구조체 변수 선언 방법 5

- **struct 형 변수 선언**

```
struct {
    int   pips;
    char  suit;
} c1, c2;
```

  - 구조체 태그 이름이 없음에 주의
  - C1, c2와 같은 형의 변수는 다시는 선언할 수 없음

멤버 접근 연산자


(*p).number
==
p->number
p가 가리키는 구조체 변수
p가 가리키는 구조체 변수의 멤버 number
p가 가리키는 구조체 변수의 멤버 number

멤버 접근 연산자 .

- **일반 구조체 멤버 접근 연산자**

```
c1.pips = 3;
c1.suit = 's';
c2.pips = c1.pips;
c2.suit = c1.suit;
```

멤버 접근 연산자 ->

- **포인터를 통한 구조체 멤버 접근 연산자**

```
pointer_to_structure -> member_name
```

- **다른 방법**

```
(*pointer_to_structure).member_name
```

멤버 접근 연산자 -> 예제

```
struct complex {
   double    re;          /* real part */
   double    im;          /* imag part */
};

typedef   struct complex   complex;

void add(complex *a, complex *b, complex *c) {
   a -> re = b -> re + c -> re;
   a -> im = b -> im + c -> im;
}
```

멤버 접근 연산자

선언문과 배정문		

`struct student  tmp, *p = &tmp;`
`tmp.grade = 'A';`
`tmp.last_name = "Casanova";`
`tmp.student_id = 910017`		

수식	동등한 수식	개념적 값
`tmp.grade`	`p -> grade`	`A`
`tmp.last_name`	`p -> last_name`	`Casanova`
`(*p).student_id`	`p -> student_id`	`910017`
`*p -> last_name + 1`	`(*(p -> last_name)) + 1`	`D`
`*(p -> last_name + 2)`	`(p -> last_name)[2]`	`s`

구조체 멤버

- 한 구조체내에서 멤버 이름은 유일해야 하나, 서로 다른 구조체에서는 같은 멤버 이름을 사용할 수 있음

```
struct fruit {
    char     *name;
    int      calories;
};
struct vegetable {
    char     *name;
    int      calories;
};
struct fruit        a;
struct vegetable    b;
```

구조체 멤버

- 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
```

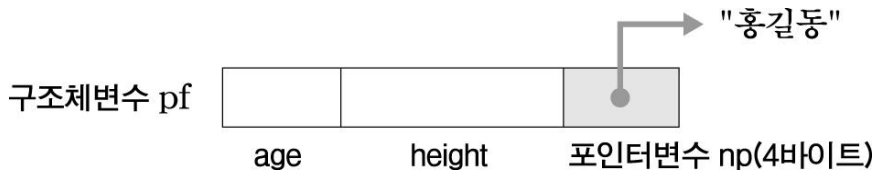
name멤버의 사용 ⇨

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

구조체 멤버

- 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
```

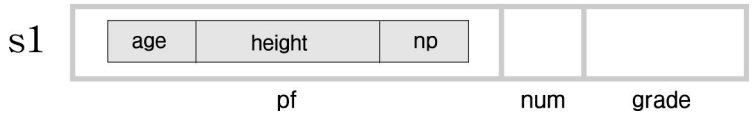


- 포인터변수를 멤버로 사용하는 경우 키보드로부터 문자열 입력은 불가능하다(문자열을 저장할 기억공간이 없다!).

구조체 멤버

- 구조체의 멤버로 다른 구조체의 변수를 사용하는 예

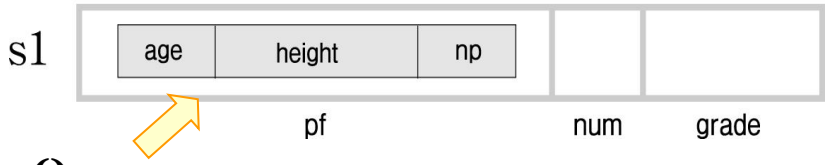
```
struct student{
    struct profile pf;    // 이미 선언된 구조체를 멤버로 사용
    int num;
    double grade;
};

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



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

구조체 멤버

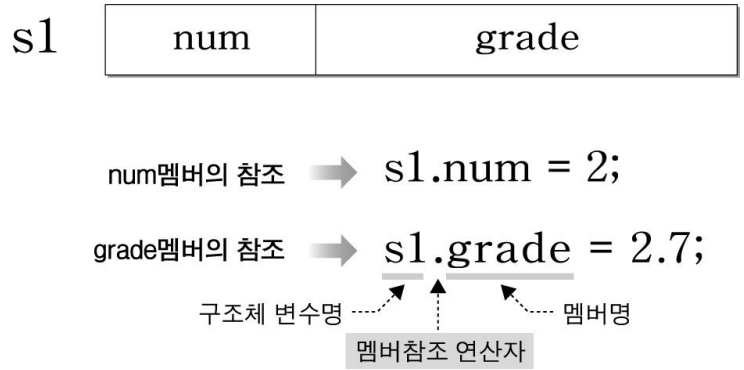
```
#include <stdio.h>

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

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




```
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;
}
```

멤버 접근 연산자 .

- 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;
}
```

멤버 접근 연산자 ->

- **포인터변수로 멤버를 참조하기 전에 먼저 구조체변수를 참조한다.**

  - **멤버참조연산자(.)가 참조연산자(*)보다 우선순위가 높으므로 괄호가 필요하다.**

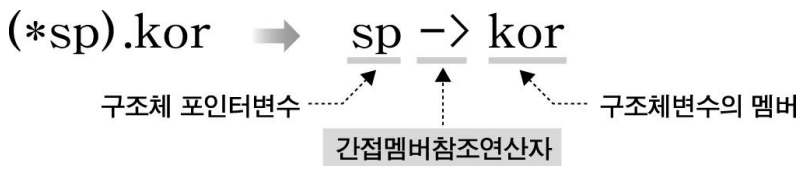
(*sp) . kor

구조체변수 참조 ① ② 구조체변수의 kor멤버 참조

```
#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;
}
```

멤버 접근 연산자 ->

- 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.
```

멤버 접근 연산자 ->

- 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.

함수에서 구조체

- 구조체는 함수의 인자로써 함수에 전달될 수 있고, 함수로부터 리턴될 수도 있음
- 함수의 인자로서 구조체가 전달될 때 구조체는 값으로 전달됨
- 구조체가 많은 멤버를 가지거나, 큰 배열을 멤버로 가질 경우, 함수의 인자로 구조체를 전달하는 것은 상대적으로 비효율적임
- 따라서 <u>대부분의 응용 프로그램에서는 함수의 인자로 구조체의 주소를 사용함</u>

함수에서 구조체 예제

```
typedef struct {
    char                    name[25];
    int                     employee_id;
    struct dept             department;
    struct home_address     *a_ptr;
    double                  salary;
    ......
} employee_data;
```

* department 멤버는 그 자체가 구조체이고, 컴파일러는 각 멤버의 크기를 미리 알아야 하므로 struct dept에 대한 선언이 먼저 와야 함

함수에서 구조체 예제

- **함수 선언 방법**

```
employee_data update(employee_data e){
    .....
    printf("Input the department number: ");
    scanf("%d", &n);
    e.department.dept_no = n;
    .....
    return e;
}
```

* 함수 호출

```
employee_data e;
e = update(e);
```

함수에서 구조체 예제

- **함수 선언 방법**

```
Void update(employee_data *p) {
    .....
    printf("Input the department number: ");
    scanf("%d", &n);
    p->department.dept_no = n;
    .....
}
```

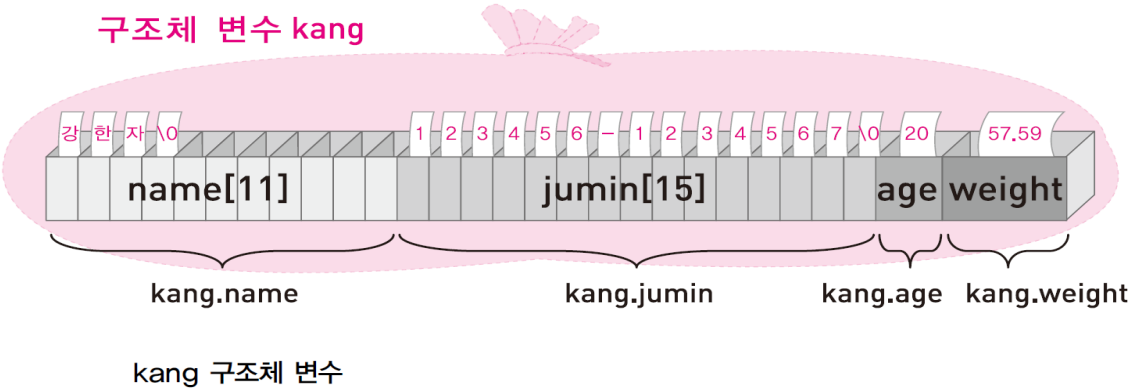
* 함수 호출

```
employee_data e;
update(&e);
```

구조체의 초기화

```
struct 태그이름 구조체 변수 =
{
   데이터1, 데이터2, ....
};
```

```
struct Person
{
   char   name[11];   /* 이름 */
   char   jumin[15];   /* 주민등록번호 */
   int    age;                    /* 나이 */
   double   weight;   /* 몸무게 */
};

struct Person kang = {"강한자", "123456-1234567", 20, 57.59};
```



- **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.

구조체의 초기화 - 예제

```
card     c = {13, 'h'};      /* the king of hearts */
complex  a[3][3] = {
   {{1.0, -0.1}, {2.0, 0.2}, {3.0, 0.3}},
   {{4.0, -0.4}, {5.0, 0.5}, {6.0, 0.6}},
};     /* a[2][] is assigned zeroes */
struct fruit  frt = {"plum", 150};
struct  home_address {
   char   *street;
   char   *city_and_state;
   long    zip_code;
} address = {"87 West Street", "Aspen, Colorado", 80526};
struct home_address    previous_address = {0};
```

구조체 예제

/*세 명의 데이터 중에서 학점이 가장 높은 학생의 학번, 이름, 학점을 출력*/

```
#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;
}
```

구조체 사용

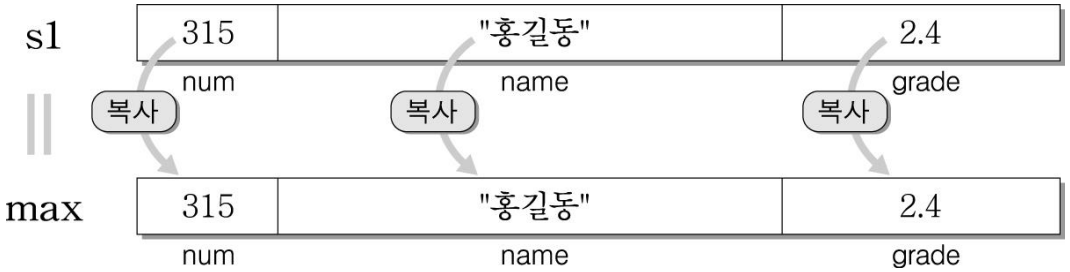
- 배열은 대입연산이 불가능하다.

```
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;
```



구조체 사용

- 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);
}
```

구조체 사용

- 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 사용**

  - **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

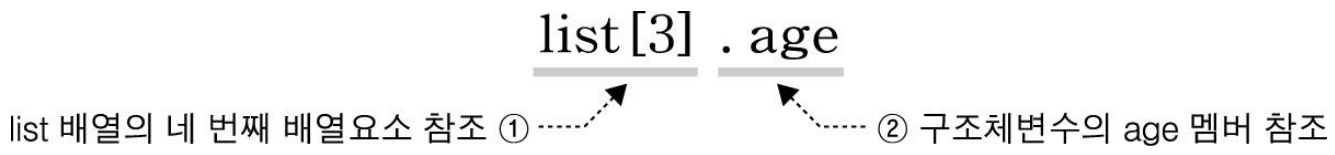
list[0]	name	age	tel	addr
list[1]	name	age	tel	addr
list[2]	name	age	tel	addr
list[3]	name	age	tel	addr
list[4]	name	age	tel	addr

배열요소는 구조체변수

구조체 배열

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

  - 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 배열

- 배열의 초기화 방법을 그대로 적용

  단, 배열의 요소가 구조체이므로 각각의 초기값은 구조체 초기화 형식을 사용

  - 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”} };
```

```
struct address list[5] = { { "홍길동", 23, "012-345-6789", "울릉도 독도" }, ...
```

배열 초기화 괄호

구조체 초기화 괄호

첫번째 구조체변수의 초기화 값

구조체 배열 예제

/* 배열요소의 값을 반복문으로 출력 */

```
#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;
}
```

공용체

- **union**
- 공용체는 구조체와 비슷한 구문 형식을 가지지만 각 멤버들은 같은 기억장소를 공유함
- 공용체형은 메모리의 같은 위치에 저장될 여러 값의 집합을 정의
- 저장된 값을 올바르게 해석하는 것은 프로그래머의 책임

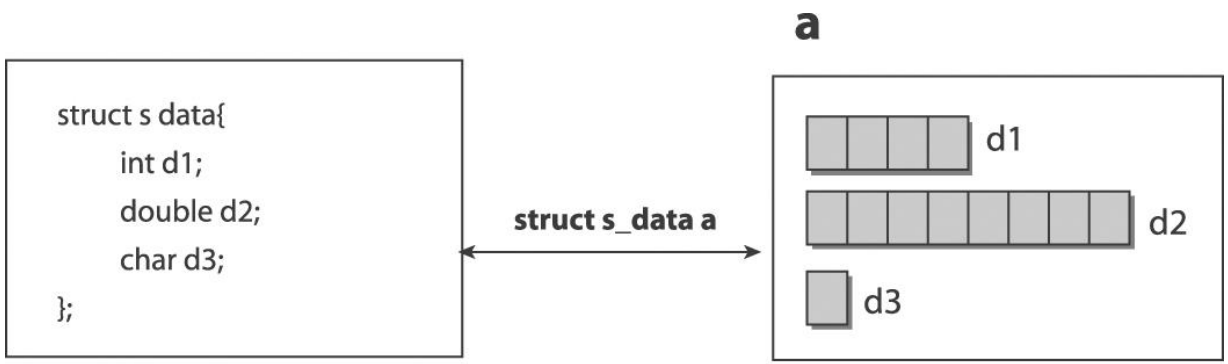
공용체 변수 선언

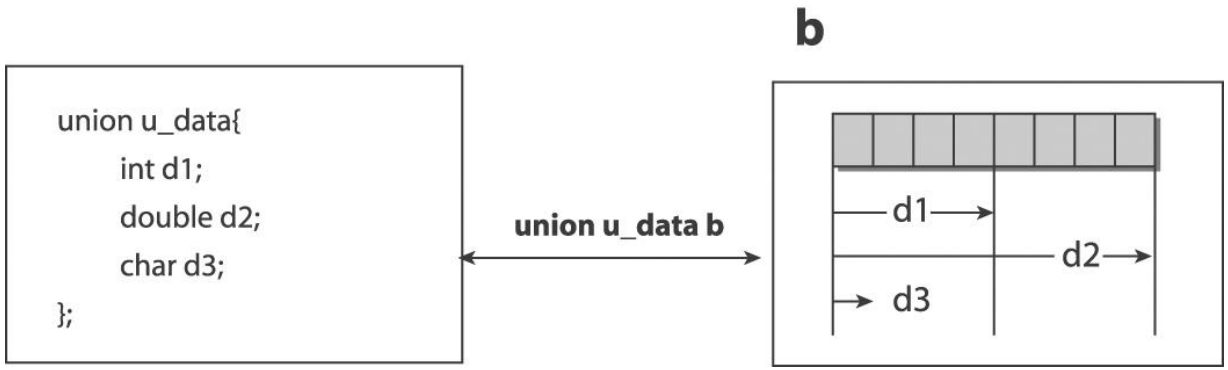
- **공용체 변수 선언**

```
union int_or_float      a, b, c;
```

  - 이 선언으로 식별자 a, b, c에 대한 기억장소가 할당
- **<u>컴파일러는 공용체의 멤버 중에서 가장 큰 기억장소를 요구하는 멤버의 요구만큼 기억장소를 할당</u>**
- **공용체의 멤버 접근 방법은 구조체의 멤버 접근 방법과 동일**
- **첫번째 멤버만 초기화 가능함**

- **공용체의 특성**
  - 하나의 메모리 공간을 둘 이상의 변수가 공유하는 형태





- **형식**

```
union 태그이름
{
   자료형 멤버;
};
```

```
union data
{
   char ch;
   int num;
   double y;
};
```

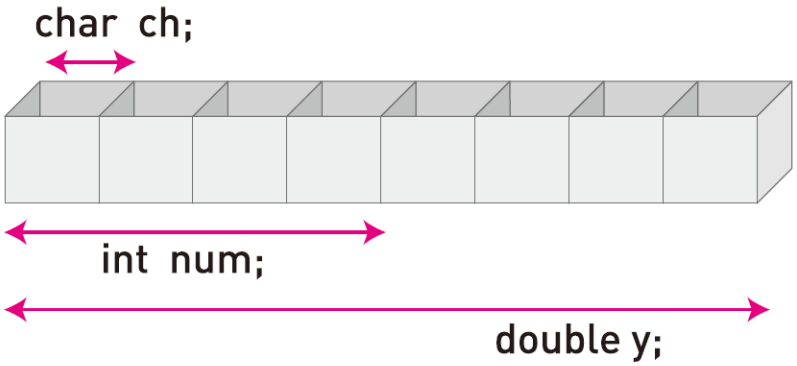
```c
#include <stdio.h>

union data { /* 공용체 정의 */
    char ch;
    int num;
    double y;
};

int main()
{
    union data value; /* 공용체 변수 선언 */

    value.ch = 'A';
    printf("ch = %d\n", value.ch);

    value.num = 300;
    printf("num = %d\n", value.num);
    value.y = 45.234;
    printf("y = %.3f\n", value.y);

    return 0;
}
```

실행 결과

```
ch = 65
num = 300
y = 45.234
```

```c
union data {
        char ch;
        int  num;
        double y;
};
```

공용체이므로 8바이트만 메모리 영역이 확보



공용체

공용체 예제

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

```
#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;
}
```

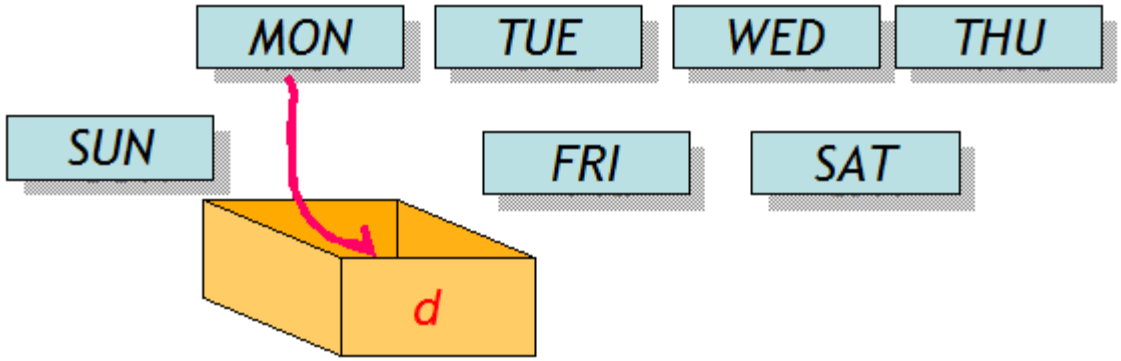
출력 결과

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

학번의 초기값이 학점 멤버에 의해서 변질되었다.

열거형

- *열거형(enumeration)*이란 변수가 가질 수 있는 값들을 미리 열거해놓은 자료형
- (예) 요일을 저장하고 있는 변수는 { 일요일, 월요일, 화요일, 수요일, 목요일, 금요일, 토요일 } 중의 하나의 값만 가질 수 있다.



열거형의 선언

```
enum days { SUN, MON, TUE, WED, THU, FRI, SAT };
```

태그 이름

값들을 나열

열거형 변수 선언

```
enum days today;
today = SUN;      // OK!
```

열거형이 필요한 이유

- 다음과 같이 프로그램을 작성할 수 있다.
  - int today;
  - today = 0; // 일요일
  - today = 1; // 월요일

- 되도록 오류를 줄이고 가독성을 높여야 된다.
- 0보다는 SUN라는 기호상수가 더 바람직하다. 의미를 쉽게 알 수 있기 때문이다.
- today에 9와 같은 의미없는 값이 대입되지 않도록 미리 차단하는 것도 필요하다.

열거형 초기화

```
enum days { SUN, MON, TUE, WED, THU, FRI, SAT };          // SUN=0, MON=1, ...
enum days { SUN=1, MON, TUE, WED, THU, FRI, SAT };        // SUN=1, MON=2, ...
enum days { SUN=7, MON=1, TUE, WED, THU, FRI, SAT=6 };// SUN=7, MON=1, ...
```

- 값을 지정하지 않으면 0부터 할당

열거형의 예

```
enum colors { white, red, blue, green, black };
enum boolean { false, true };
enum levels { low, medium, high };
enum car_types { sedan, suv, sports_car, van, pickup, convertible };
```

예제

```c
#include <stdio.h>
enum days { SUN, MON, TUE, WED, THU, FRI, SAT };
char *days_name[] = {
"sunday", "monday", "tuesday", "wednesday", "thursday", "friday",
"saturday" };

int main(void)
{
  enum days d;
  d = WED;
  printf("%d번째 요일은 %s입니다\n", d, days_name[d]);
  return 0;
}
```

```
3번째 요일은 wednesday입니다
```

열거형과 다른 방법과의 비교

정수 사용	기호 상수	열거형

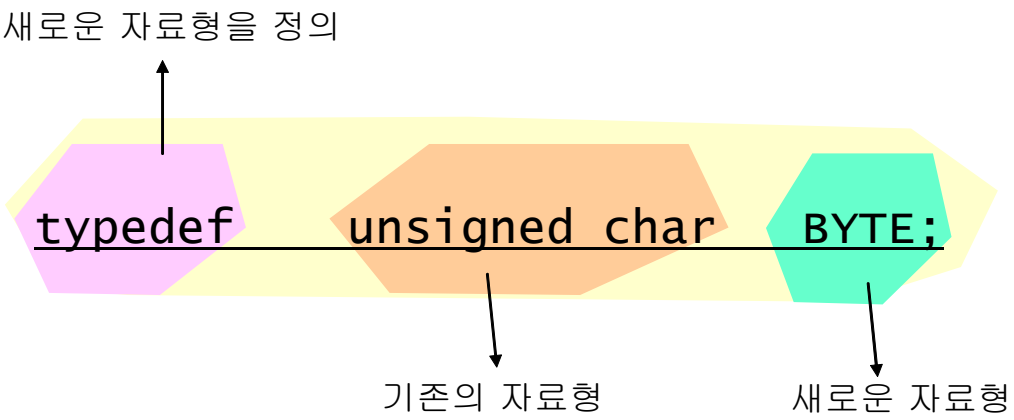
switch(code) {
  case 1:
  printf("LCD TV\n");
  break;
  case 2:
  printf("PDP TV\n");
  break;
}
	#define LCD 1
	#define PDP 2
	
	switch(code) {
	  case LCD:
	  printf("LCD TV\n");
	  break;
	  case PDP:
	  printf("PDP TV\n");
	  break;
	}
		enum tvtype { LCD, PDP };
		enum tvtype code;
		
		switch(code) {
		  case LCD:
		  printf("LCD TV\n");
		  break;
		  case PDP:
		  printf("PDP TV\n");
		  break;
		}

컴퓨터는 알기 쉬우나 사람은 기억하기 어렵다.	기호 상수를 작성할 때 오류를 저지를 수 있다.	컴파일러가 중복이 일어나지 않도록 체크한다.

typedef

- typedef은 새로운 자료형(type)을 정의(define)
- C의 기본 자료형을 확장시키는 역할

```
typedef    old_type    new_type;
```



- **typedef를 사용하여 응용자료형을 간단하게 사용**

```
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.
```

typedef 예제

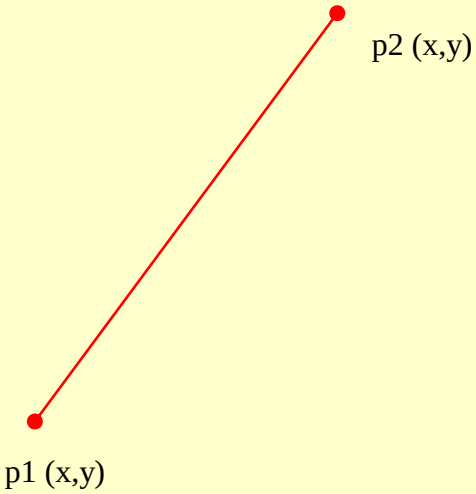
```
#include <stdio.h>

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);
}
```

이론 실습- 예제

예제 #1

```c
#include <math.h>
struct point {
	int x;
	int y;
};

int main(void)
{
	struct point p1, p2;
	int xdiff, ydiff;
	double dist;

	printf("점의 좌표를 입력하시오(x  y): ");
	scanf("%d %d", &p1.x, &p1.y);

	printf("점의 좌표를 입력하시오(x  y): ");
	scanf("%d %d", &p2.x, &p2.y);

	xdiff = p1.x - p2.x;
	ydiff = p1.y - p2.y;

	dist = sqrt(xdiff * xdiff + ydiff * ydiff);

	printf("두 점사이의 거리는 %f입니다.\n", dist);
	return 0;
}
```

```
점의 좌표를 입력하시오(x y): 10 10
점의 좌표를 입력하시오(x y): 20 20
두 점사이의 거리는 14.142136입니다.
```



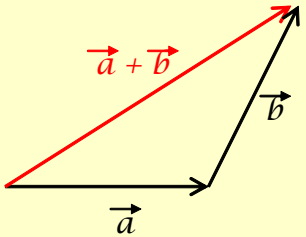
예제 #2 – 구조체 배열

```
#define SIZE 3

struct student {
	int number;
	char name[20];
	double grade;
};
int main(void)
{
	struct student list[SIZE];
	int i;

	for(i = 0; i < SIZE; i++)
	{
		printf("학번을 입력하시오: ");
		scanf("%d", &list[i].number);
		printf("이름을 입력하시오: ");
		scanf("%s", list[i].name);
		printf("학점을 입력하시오(실수): ");
		scanf("%lf", &list[i].grade);
	}

	for(i = 0; i< SIZE; i++)
		printf("학번: %d, 이름: %s,  학점: %f\n", list[i].number, list[i].name, list[i].grade);
	return 0;
}
```

```
학번을 입력하시오: 20070001
이름을 입력하시오: 홍길동
학점을 입력하시오(실수): 4.3
학번을 입력하시오: 20070002
이름을 입력하시오: 김유신
학점을 입력하시오(실수): 3.92
학번을 입력하시오: 20070003
이름을 입력하시오: 이성계
학점을 입력하시오(실수): 2.87
학번: 20070001, 이름: 홍길동, 학점: 4.300000
학번: 20070002, 이름: 김유신, 학점: 3.920000
학번: 20070003, 이름: 이성계, 학점: 2.870000
```

예제 #2 – 구조체 배열

```
#include <stdio.h>

struct vector {
	float x;
	float y;
};
struct vector get_vector_sum(struct vector a, struct vector b);

int main(void)
{
	struct vector a = { 2.0, 3.0 };
	struct vector b = { 5.0, 6.0 };
	struct vector sum;

	sum = get_vector_sum(a, b);
	printf("벡터의 합은 (%f, %f)입니다.\n", sum.x, sum.y);

	return 0;
}
```



예제 #3 – 구조체와 배열 그리고 포인터 (1)

```
struct person {
      char name[20];
      char phone[20];
};

int main()
{
    struct person man={"Thomas", "354-00xx"};
    struct person * pMan;
    pMan=&man;

    // 구조체 변수를 이용한 출력.
    printf("name : %s\n", man.name);
    printf("phone : %s\n", man.phone);

    // 구조체 포인터를 이용한 출력1.
    printf("name : %s\n", (*pMan).name);
    printf("phone : %s\n", (*pMan).phone);

    // 구조체 포인터를 이용한 출력2.
    printf("name : %s\n", pMan->name);
    printf("phone : %s\n", pMan->phone);
    return 0;
}
```

예제 #3 – 구조체와 배열 그리고 포인터(2)

```
#include <stdio.h>

struct perInfo {
   char addr[30];
   char tel[20];
};
struct person {
   char name[20];
   char pID[20];
   struct perInfo* info;
};

int main()
{
   struct perInfo info={"Korea Seoul", "333-4444"};
   struct person man={"Mr. Lee", "820204-xxxx512"};

   man.info=&info;

   printf("name : %s\n", man.name);
   printf("pID  : %s\n", man.pID);
   printf("addr : %s\n", man.info->addr);
   printf("tel  : %s\n", man.info->tel);
   return 0;
}
```

예제 #4 – Typedef 활용

```
#include <stdio.h>

typedef struct point {
	int x;
	int y;
} POINT;

POINT translate(POINT p, POINT delta);

int main(void)
{
	POINT p = { 2, 3 };
	POINT delta = { 10, 10 };
	POINT result;

	result = translate(p, delta);
	printf("새로운 점의 좌표는(%d, %d)입니다.\n", result.x, result.y);

	return 0;
}
```

참고문헌

- 열혈 C 프로그래밍, 윤성우, 오렌지미디어
- 쉽게 풀어쓴 C언어 Express, 천인국, 생능출판사
- 뇌를 자극하는 C 프로그래밍, 서현우, 한빛미디어
- 쾌도난마C프로그래밍, 강성수, 북스홀릭