10장. 구조체

박 종 혁 교수
서울과학기술대학교 컴퓨터공학과
UCS Lab
Tel: 970-6702
Email: jhpark1@seoultech.ac.kr

목차

구조체의 정의

- 서로 다른 데이터형의 변수들을 하나로 묶어서 사용하는 기능
- 사용자 정의형을 만드는 방법을 제공

```
char name[20];
char phone[20];
int ringtone;
```

함께 사용되는 변수들을
묶어서 구조체를 정의한다.

```
struct contact
{
   char name[20];   // 이름
   char phone[20]; // 전화번호
   int ringtone;    // 벨 소리
};
```

태그 이름

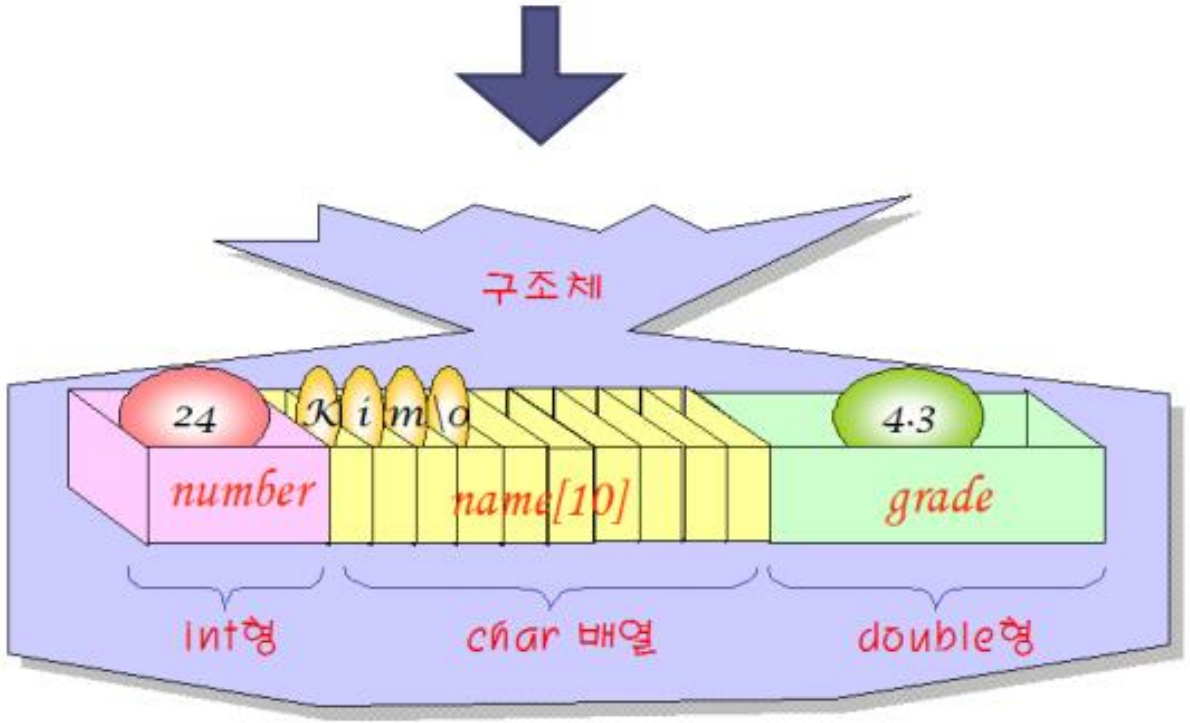
구조체의 멤버

세미콜론이 필요하다

struct contact이
새로운 데이터형이 된다.

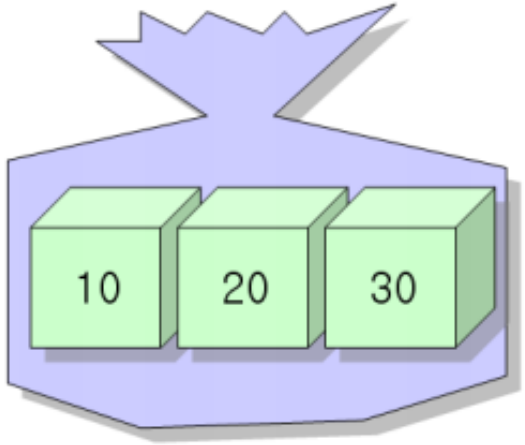
구조체의 필요성

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



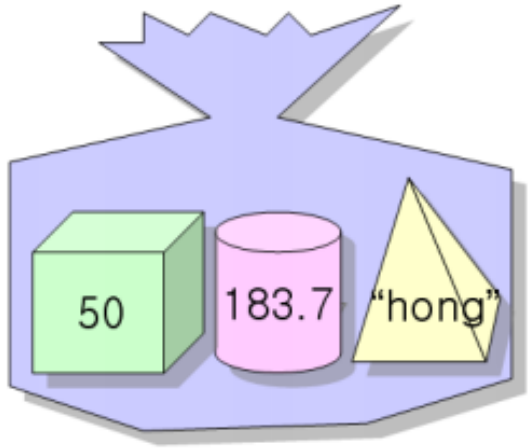
구조체와 배열

- 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 정의문의 위치

함수 안에 정의하는 경우

```
int main(void)
{
  struct contact
  {
    char name[20];
    char phone[20];
    int ringtone;
  };
  struct contact c1;

  return 0;
}
void test()
{
  struct contact c1;
}
```

정의된 함수에서만 사용할 수 있다.

컴파일 에러

함수 밖에 정의하는 경우

```
struct contact
{
  char name[20];
  char phone[20];
  int ringtone;
};

int main(void)
{
  struct contact c1;

  return 0;
}
void test()
{
  struct contact c1;
}
```

소스 파일 시작 부분에 정의해준다.

소스 파일 전체에서 사용할 수 있다.

사용자 정의형으로서의 구조체

❖구조체를 정의하면 새로운 데이터형이 만들어짐
- 데이터형만 정의할 뿐 구조체 변수가 생성되는 것은 아님
- 구조체의 멤버는 구조체 변수를 선언해야 메모리에 할당됨

❖sizeof 연산자로 구조체의 바이트 크기를 구할 수 있음

```
printf("contact 구조체의 크기 = %d\n", sizeof(struct contact));
```

구조체의 크기는 멤버들의 크기의 합보다 크거나 같다.

❖구조체형은 struct 키워드와 태그 이름을 함께 사용해야 함

```
printf("contact 구조체의 크기 = %d\n", sizeof(contact));    // 컴파일 에러
```

태그명만 사용하면 안된다.

예제 : 구조체의 정의

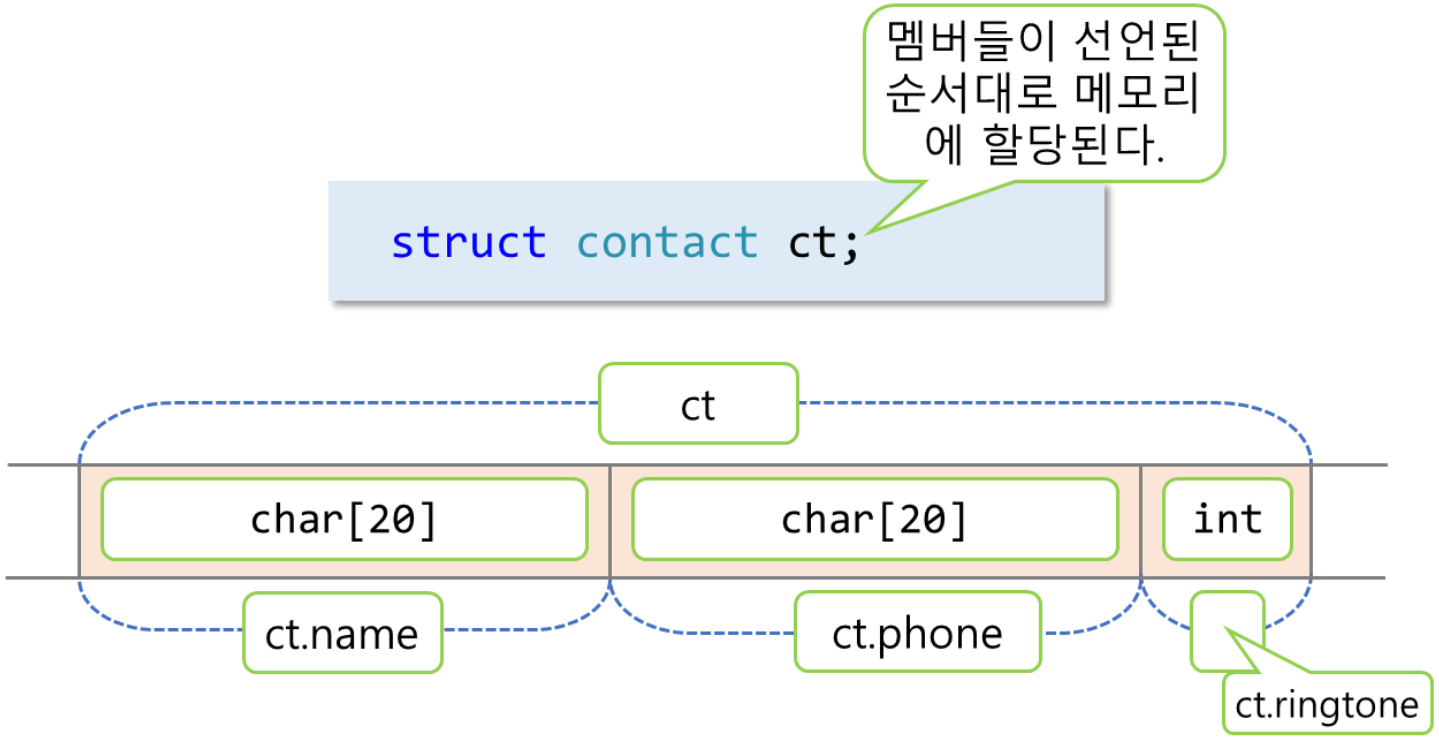
```
03  // 구조체의 정의는 보통 함수 밖에 써준다.
04  struct contact      // 연락처
05  {
06      char name[20];  // 이름
07      char phone[20]; // 전화번호(01012345678 형식의 문자열로 저장)
08      int ringtone;   // 벨 소리(0~9 선택)
09  };
10
11  int main(void)
12  {
13      printf("contact 구조체의 크기 = %d\n", sizeof(struct contact));
14      //printf("contact 구조체의 크기 = %d\n", sizeof(contact));        // 컴파일 에러
15
16      return 0;
17  }
19  void test()
20  {
21      struct contact c1;        // 여러 함수에서 구조체를 사용할 수 있다.
22  }
```

실행결과

```
contact 구조체의 크기 = 44
```

구조체 변수의 선언 (1/2)

- 구조체 변수가 메모리에 할당될 때 구조체의 멤버들이 선언된 순서대로 메모리에 할당됨



구조체 변수의 선언 (2/2)

- 구조체를 정의하면서 구조체 변수를 함께 선언할 수 있음

```
struct app_info {
    char name[128];
    char path[128];
    int version;
} this_app;
```

구조체 변수를 함께 선언한다.

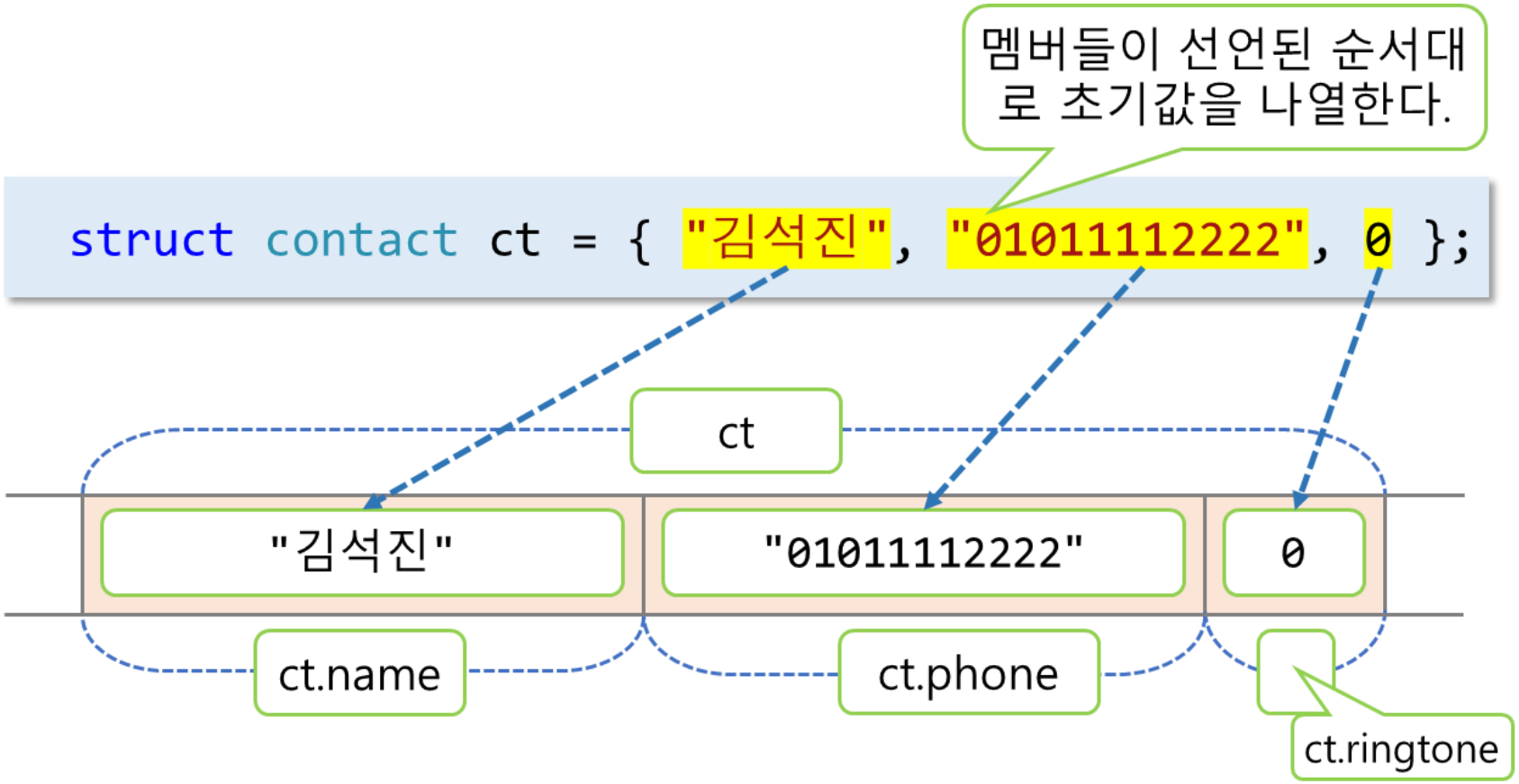
- 구조체를 정의하면서 변수를 함께 선언할 때는 구조체의 태그 이름을 생략할 수 있음

이름이 없으므로 나중에 다시 사용할 수 없다.

```
struct {
    char name[128];
    char path[128];
    int version;
} this_app;
```

구조체 변수의 초기화 (1/2)

- { } 안에 멤버들의 초기값을 멤버가 선언된 순서대로 나열



구조체 변수의 초기화 (2/2)

- 초기값이 멤버의 개수보다 부족하면 나머지 멤버들은 0으로 초기화

```
struct contact ct = { "김석진", "01011112222" }; // ringtone은 0으로 초기화
```

- 멤버의 개수보다 초기값을 더 많이 지정하면 안됨

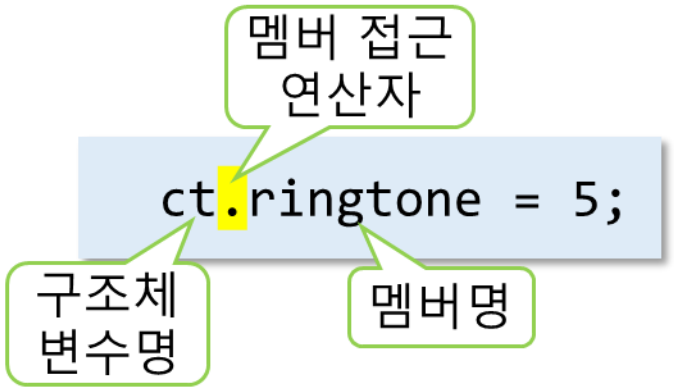
```
struct contact ct = { "김석진", "01011112222", 0, 1 };      // 컴파일 에러
```

- 초기값으로 { 0 }을 지정하면 모든 멤버가 0으로 초기화

```
struct contact ct = { 0 };    // 모든 멤버가 0으로 초기화
```

구조체 변수의 사용

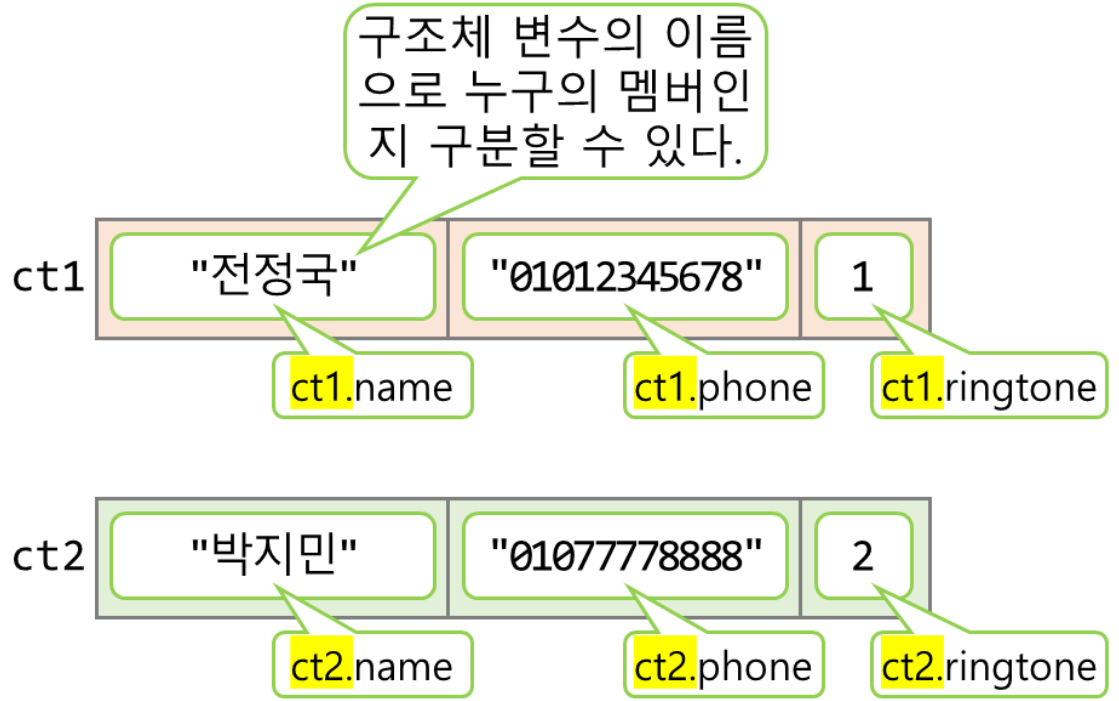
- 구조체의 멤버에 접근하려면 멤버 접근 연산자(.)를 이용
- 구조체 변수를 여러 개 선언하면, 각각의 구조체 변수는 서로 다른 메모리에 할당됨




```
struct contact ct1 = { 0 }, ct2 = { 0 };

strcpy(ct1.name, "전정국");
strcpy(ct1.phone, "01012345678");
ct1.ringtone = 1;


strcpy(ct2.name, "박지민");
strcpy(ct2.phone, "01077778888");
ct2.ringtone = 2;
```



예제 : 구조체 변수의 선언 및 사용

```
04  struct contact      // 연락처
05  {
06      char name[20];  // 이름
07      char phone[20]; // 전화번호(01012345678 형식의 문자열로 저장)
08      int ringtone;   // 벨 소리(0~9 선택)
09  };
10
11  int main(void)
12  {
13      struct contact ct = { "김석진", "01011112222", 0 };
14      struct contact ct1 = { 0 }, ct2 = { 0 };
15
16      ct.ringtone = 5;
17      strcpy(ct.phone, "01011112223");
18      printf("이     름: %s\n", ct.name);
19      printf("전화번호: %s\n", ct.phone);
20      printf("벨 소 리: %d\n", ct.ringtone);
22      strcpy(ct1.name, "전정국");
23      strcpy(ct1.phone, "01012345678");
24      ct1.ringtone = 1;
25      printf("이     름: %s\n", ct1.name);
26      printf("전화번호: %s\n", ct1.phone);
27      printf("벨 소 리: %d\n", ct1.ringtone);
28
29      // ct2로 연락처 정보를 입력받는다.
30      printf("이름? ");
31      scanf("%s", ct2.name);
32      printf("전화번호? ");
33      scanf("%s", ct2.phone);
34      printf("벨 소리(0~9)? ");
35      scanf("%d", &ct2.ringtone);
36      printf("이     름: %s\n", ct2.name);
37      printf("전화번호: %s\n", ct2.phone);
38      printf("벨 소 리: %d\n", ct2.ringtone);
```

실행결과

```
이     름: 김석진
전화번호: 01011112223
벨 소 리: 5
이     름: 전정국
전화번호: 01012345678
벨 소 리: 1
이름? 박지민
전화번호? 01077778888
벨 소리(0~9)? 2
이     름: 박지민
전화번호: 01077778888
벨 소 리: 2
```

멤버 접근 연산자

- **일반 구조체 멤버 접근 연산자**

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

- **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

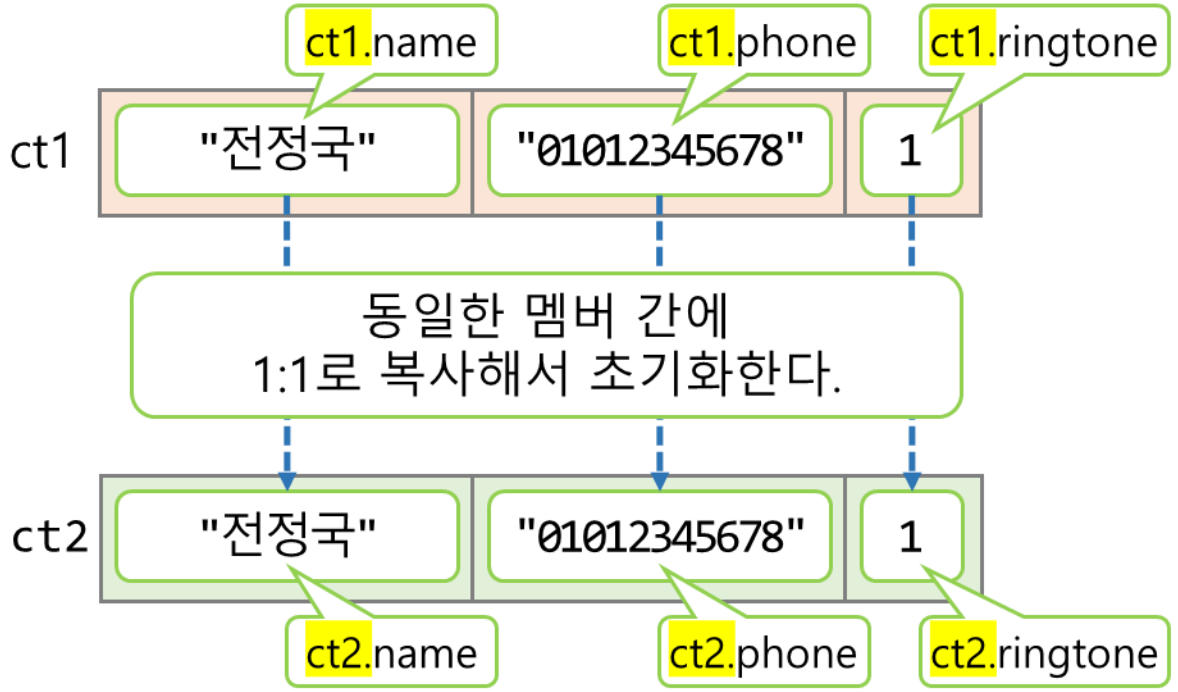
구조체 변수 간의 초기화

- 구조체 변수를 다른 구조체 변수로 초기화하면, 동일한 멤버 간에 1:1로 복사해서 초기화함

```
struct contact ct1 = {
    "전정국", "01012345678", 1
};

struct contact ct2 = ct1;
```

구조체 변수를 다른 구조체 변수로 초기화할 수 있다.



구조체 변수 간의 대입

- 같은 구조체형의 변수를 대입할 때도 동일한 멤버 간에 1:1로 대입함

```
ct2 = ct; // ct의 모든 멤버를 ct2의 멤버로 대입한다.
```

- 구조체 변수를 초기화할 때처럼 { } 안에 나열된 값을 대입할 수는 없음

```
ct2 = { "김석진", "01011112222" };      // { }는 초기화할 때만 사용할 수 있으므로 컴파일 에러
```

- 구조체의 멤버인 배열끼리 대입하는 것은 컴파일 에러

```
ct2.name = ct.name; // 배열에 다른 배열을 대입하면 컴파일 에러
```

예제 : 구조체 변수 간의 초기화와 대입

```
11  int main(void)
12  {
13      struct contact ct = { "김석진", "01011112222", 0 };
14      struct contact ct1 = { "전정국", "01012345678", 1 };
15      struct contact ct2 = ct1;// 구조체 변수로 초기화할 수 있다.
16      printf("ct1으로 초기화 후의 ct2 = %s, %s, %d\n", ct2.name, ct2.phone, ct2.ringtone);
17
18      ct2 = ct;       // 구조체 변수를 대입할 수 있다.
19      printf("ct를 대입한 후의 ct2 = %s, %s, %d\n", ct2.name, ct2.phone, ct2.ringtone);
20
21      return 0;
22  }
```

실행결과

```
ct1으로 초기화 후의 ct2 = 전정국, 01012345678, 1
ct를 대입한 후의 ct2 = 김석진, 01011112222, 0
```

구조체 변수의 비교

- 구조체 변수끼리 직접 관계 연산자로 비교해서는 안됨

```
if (ct1 == ct2)     // 구조체 변수에 관계연산자를 사용할 수 없으므로 컴파일 에러
    printf("ct1과 ct2의 값이 같습니다\n");
```

- 두 구조체 변수의 값이 같은지 비교하려면 구조체 변수끼리 비교하는 대신 멤버 대 멤버로 비교해야 함

```
if (strcmp(ct1.name, ct2.name) == 0 && strcmp(ct1.phone, ct2.phone) == 0
    && ct1.ringtone == ct2.ringtone)
    printf("ct1과 ct2의 값이 같습니다\n");
```

예제 : 구조체 변수의 비교

```
11  int main(void)
12  {
13      struct contact ct1 = { "전정국", "01012345678", 1 };
14      struct contact ct2 = ct1;
15
16      if (strcmp(ct1.name, ct2.name) == 0 && strcmp(ct1.phone, ct2.phone) == 0
17          && ct1.ringtone == ct2.ringtone)
18          printf("ct1과 ct2의 값이 같습니다.\n");
19      else
20          printf("ct1과 ct2의 값이 같지 않습니다.\n");
21      return 0;
22  }
```

실행결과

```
ct1과 ct2의 값이 같습니다.
```

typedef

- 기존의 데이터형에 대한 별명(alias)

형식	typedef 기존데이터형 새이름;
사용예	typedef struct contact CONTACT; typedef unsigned int UINT;

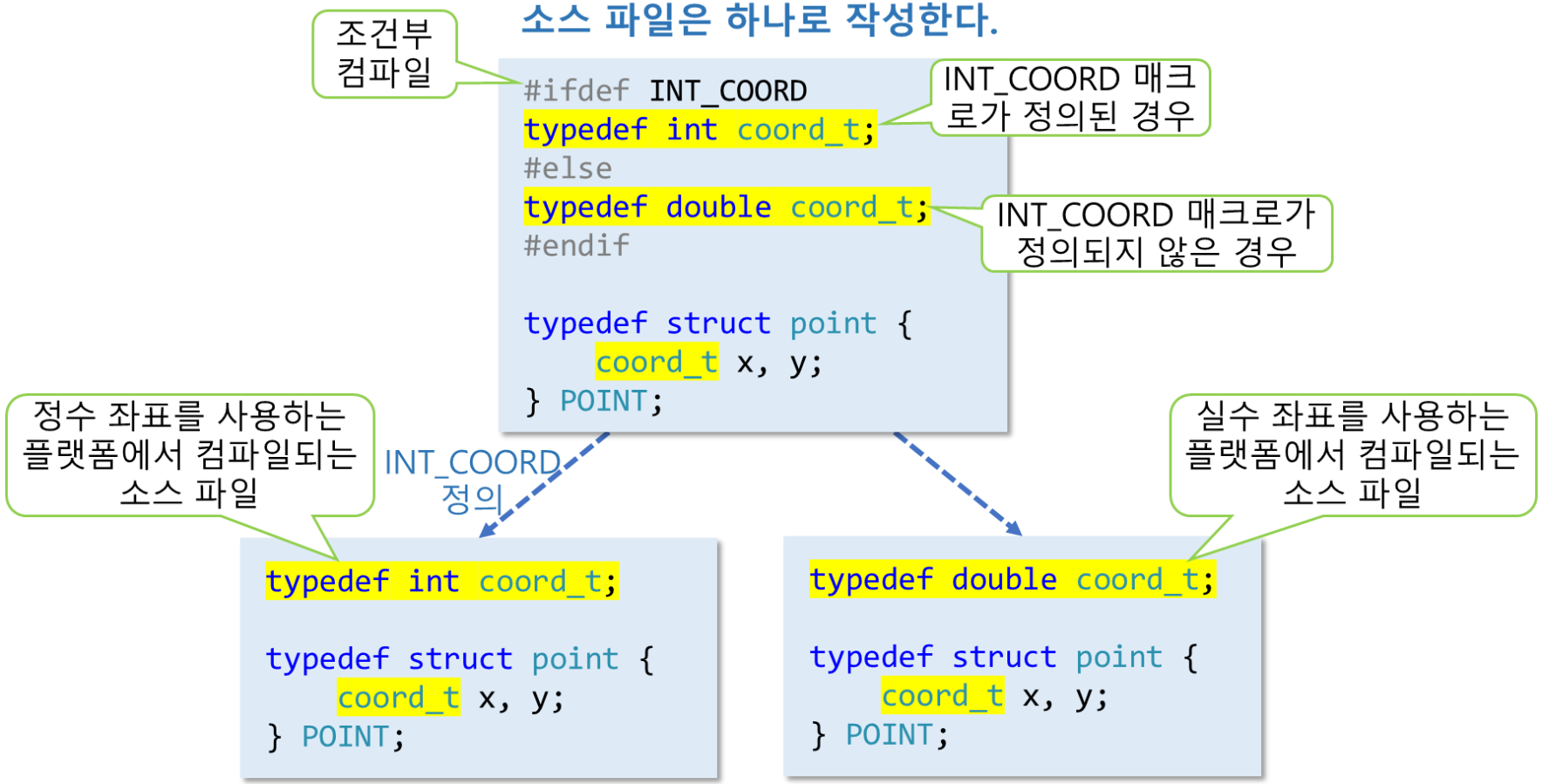
```
typedef struct contact CONTACT;
```

```
CONTACT ct = { "김석진", "01011112222", 0 };
```

```
typedef struct contact
{
    char name[20];
    char phone[20];
    int ringtone;
} CONTACT;          // 구조체를 정의하면서 typedef도 함께 정의한다.
```

typedef의 사용 목적 : 이식성

- 이식성 : 하나의 소스 파일로 여러 플랫폼에서 수정 없이 컴파일 되고 실행될 수 있는 특성



typedef의 사용 목적 : 가독성

❖1바이트 크기의 2진 데이터를 저장하는 변수 선언

```
unsigned char binary_data;    // unsigned char는 부호 없는 문자형이라는 의미
```

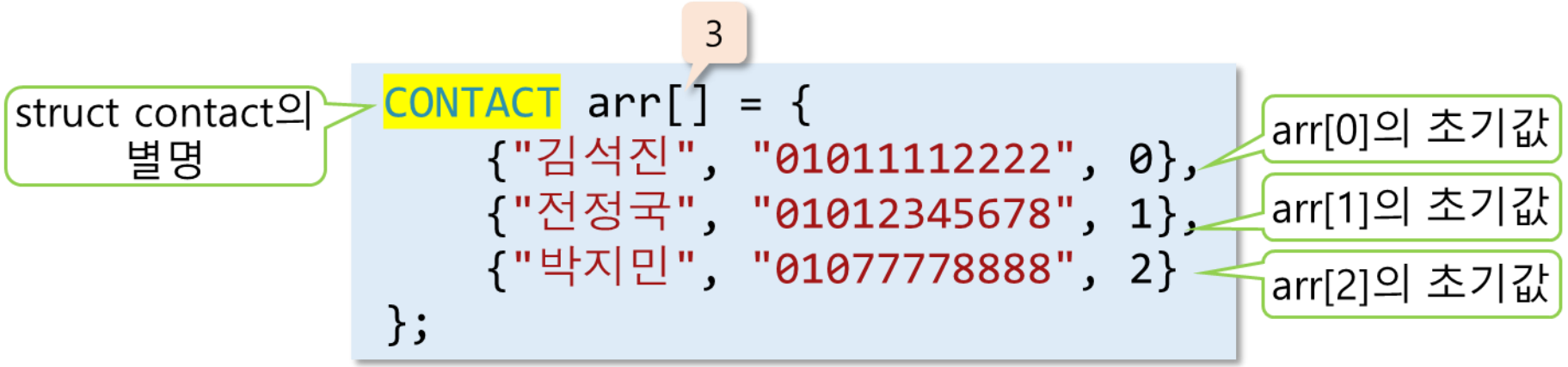
❖unsigned char형을 byte라는 이름으로 정의

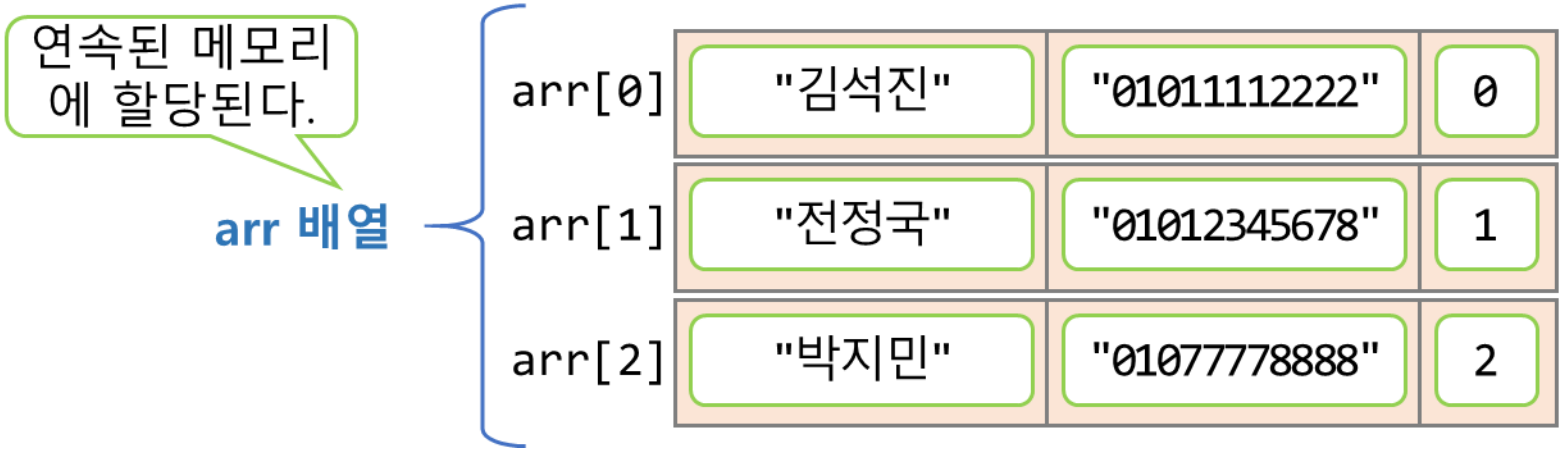
- 2진 데이터 값을 저장하는 용도로 사용하는 변수라는 의미

```
typedef unsigned char byte;  // 1바이트 크기의 데이터형
byte binary_data;   // 문자 코드를 저장하는 것이 아니라 바이트 데이터를 저장한다는 의미
```

구조체 배열 (1/2)

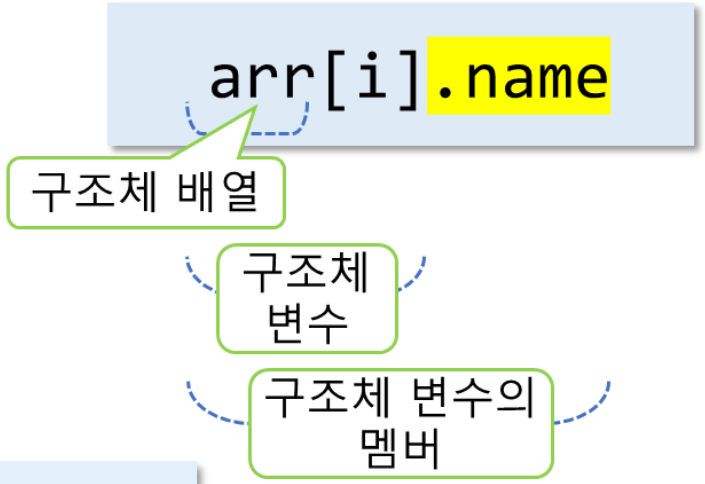
- 같은 구조체형의 변수를 여러 개 묶어서 사용





구조체 배열 (2/2)

- 구조체의 멤버에 접근하려면 arr[i].*member* 형식으로 접근




```
for (i = 0; i < size; i++)
{   // arr[i]가 CONTACT 변수이다.
    printf("%6s %11s %d\n", arr[i].name, arr[i].phone, arr[i].ringtone);
}
```

예제 : 구조체 배열

```
04  typedef struct contact       // 연락
05  {
06      char name[20];
07      char phone[20];
08      int ringtone;
09  } CONTACT;
10
11  int main(void)
12  {
13      CONTACT arr[] = {        // 배열의 크기를 생략할 수 있다.
14          {"김석진", "01011112222", 0},
15          {"전정국", "01012345678", 1},
16          {"박지민", "01077778888", 2}
17      };
18      int size = sizeof(arr) / sizeof(arr[0]);
19      int i;
21      printf(" 이름   전화번호    벨\n");
22      for (i = 0; i < size; i++)
23      {
24          printf("%6s %11s %d\n", arr[i].name, arr[i].phone, arr[i].ringtone);
25      }
26      return 0;
27  }
```

실행결과

```
이름   전화번호    벨
김석진 01011112222 0
전정국 01012345678 1
박지민 01077778888 2
```

예제 : 구조체 배열의 검색

```c
#define STR_SIZE 20

typedef struct contact
{
    char name[STR_SIZE];
    char phone[STR_SIZE];
    int ringtone;
} CONTACT;
int main(void)
{
    CONTACT arr[] = {        // 초기화된 배열
        {"김석진", "01011112222", 0},
        {"전정국", "01012345678", 1},
        {"박지민", "01077778888", 2},
        {"김남준", "01098765432", 9},
        {"민윤기", "01011335577", 5},
        {"정호석", "01024682468", 7},
        {"김태형", "01099991111", 3}
    };
    int size = sizeof(arr) / sizeof(arr[0]);      // 배열의 크기
    int i;
    char name[STR_SIZE];     // 입력받은 이름을 저장할 문자 배열
    int index;

    printf("이름? ");
    scanf("%s", name);

    index = -1;     // 이름을 찾을 수 없으면 -1
    for (i = 0; i < size; i++)
    {
        if (strcmp(arr[i].name, name) == 0)       // 문자열 비교
        {
            index = i;
            break;
        }
    }
    if (index >= 0)   // 검색 성공
    {
       printf("%s의 전화번호: %s\n", arr[index].name, arr[index].phone);
    }
    else   // 검색 실패
    {
        printf("연락처를 찾을 수 없습니다.\n");
    }

    return 0;
}
```

실행결과

```
이름? 김태형
김태형의 전화번호: 01099991111
```

구조체 포인터

- 구조체 변수의 주소를 저장하는 포인터
- 구조체 포인터로 구조체 변수의 멤버에 접근하려면 간접 멤버 접근 연산자인 -> 연산자를 사용

```
struct contact ct = {
  "김석진", "01011112222", 0
};
struct contact *p = &ct;
```


```
p->ringtone = 5;
strcpy(p->phone, "01011112223");
```

```
(*p).ringtone = 5;
strcpy((*p).phone, "01011112223");
```

예제 : 구조체 포인터

```
13  int main(void)
14  {
15      CONTACT ct = { "김석진", "01011112222", 0 };
16      CONTACT *p = &ct;
17
18      p->ringtone = 5;
19      strcpy(p->phone, "01011112223");
20      printf("이    름: %s\n", p->name);
21      printf("전화번호: %s\n", p->phone);
22      printf("벨 소 리: %d\n", p->ringtone);
23
24      return 0;
25  }
```

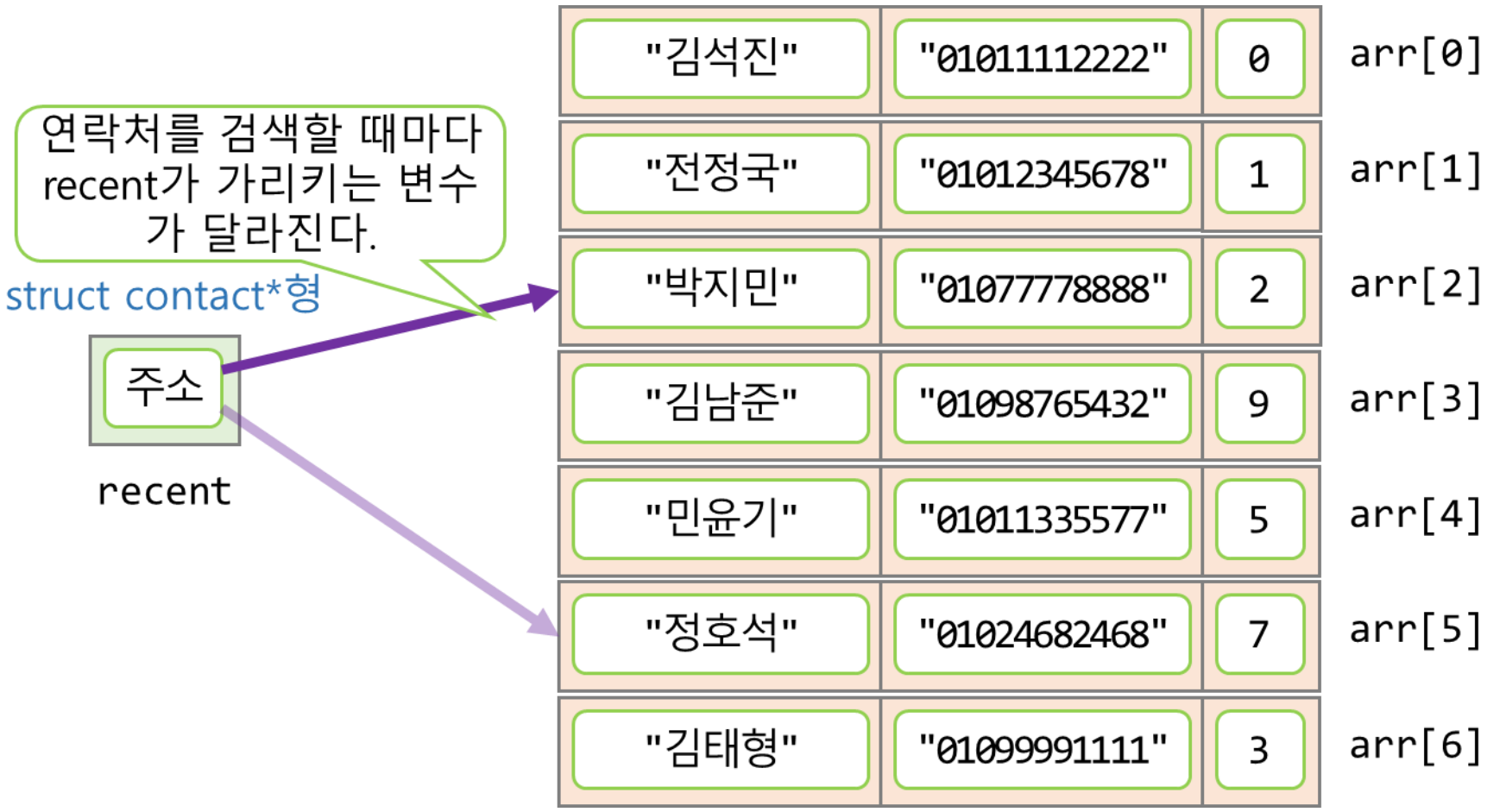
실행결과

```
이    름: 김석진
전화번호: 01011112223
벨 소 리: 5
```

구조체 포인터의 활용 (1/2)

```
CONTACT *recent = NULL;   // 마지막으로 검색한 연락처를 가리키는 포인터
while (1)
{
    printf("이름(. 입력 시 종료)? ");
    scanf("%s", name);
    if (strcmp(name, ".") == 0)  // name이 "."이면 while 탈출
        break;
    index = -1;    // 이름을 찾을 수 없으면 -1
    for (i = 0; i < size; i++) {
        ⋮
    }
    if (index >= 0) { // 검색 성공
        printf("%s의 전화번호: %s\n", arr[index].name, arr[index].phone);
        recent = &arr[index];    // recent는 찾은 원소를 가리킨다.
    }
    else                  // 검색 실패
        printf("연락처를 찾을 수 없습니다.\n");
}
```

구조체 포인터의 활용 (2/2)



예제 : 구조체 포인터의 활용

```c
int main(void)
{
    CONTACT arr[] = {        // 초기화된 배열
        {"김석진", "01011112222", 0},
        {"전정국", "01012345678", 1},
        {"박지민", "01077778888", 2},
        {"김남준", "01098765432", 9},
        {"민윤기", "01011335577", 5},
        {"정호석", "01024682468", 7},
        {"김태형", "01099991111", 3}
    };
    int size = sizeof(arr) / sizeof(arr[0]);      // 배열의 크기
    int i;
    char name[STR_SIZE];     // 입력받은 이름을 저장할 문자 배열
    int index;
    CONTACT *recent = NULL;  // 마지막으로 검색한 연락처를 가리키는 포인터
    while (1)
    {
        printf("이름(. 입력 시 종료)? ");
        scanf("%s", name);
        if (strcmp(name, ".") == 0)     // name이 "."이면 while 탈출
            break;

        index = -1;     // 이름을 찾을 수 없으면 -1
        for (i = 0; i < size; i++)
        {
            if (strcmp(arr[i].name, name) == 0)  // 문자열 비교
            {
                index = i;
                break;
            }
        }
        if (index >= 0)      // 검색 성공
        {
            printf("%s의 전화번호 %s로 전화를 겁니다....\n",
                arr[index].name, arr[index].phone);
            recent = &arr[index];      // recent는 찾은 원소를 가리킨다.
        }
        else       // 검색 실패
        {
            printf("연락처를 찾을 수 없습니다.\n");
        }
    }
    if (recent)     // recent가 NULL이 아니면 최근 통화 연락처를 출력한다.
        printf("최근 통화: %s %s\n", recent->name, recent->phone);

    return 0;
}
```

실행결과

```
이름(. 입력 시 종료)? 전정국
전정국의 전화번호 01012345678로 전화를 겁니다....
이름(. 입력 시 종료)? 김석진
김석진의 전화번호 01011112222로 전화를 겁니다....
이름(. 입력 시 종료)? .
최근 통화: 김석진 01011112222
```

.이 입력될 때까지 연락처를 반복적으로 검색한다.

recent는 마지막으로 찾은 CONTACT 배열의 원소를 가리킨다.

함수의 인자로 구조체 전달하기 (1/3)

❖값에 의한 전달

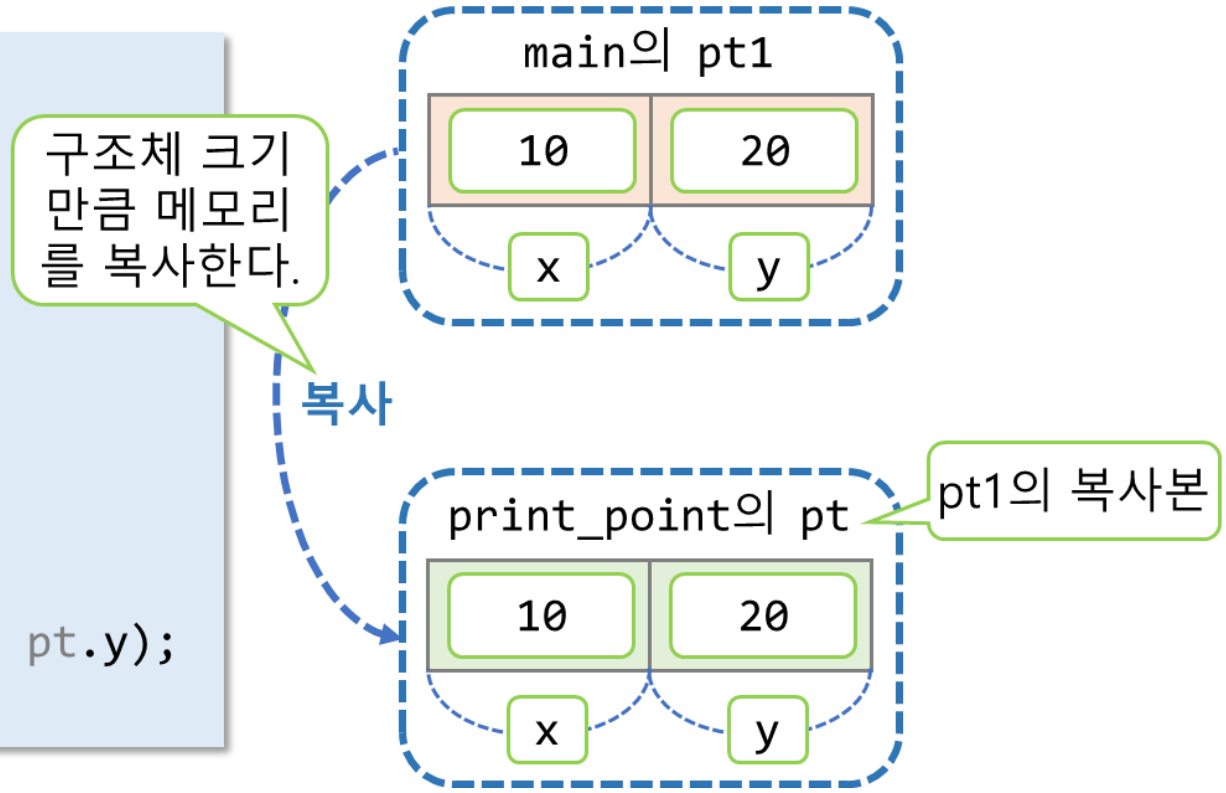
- 구조체를 복사해서 전달 → 시간적·공간적 성능 저하

값에 의한 전달

```
int main(void)
{
    POINT pt1 = { 10,20 };
    print_point(pt1);
     ⋮
}

void print_point(POINT pt)
{
    printf("(%d, %d)", pt.x, pt.y);
}
```

POINT pt = pt1;



예제 : 구조체를 값으로 전달하는 경우

```
03  typedef struct point
04  {
05      int x, y;    // 점의 좌표
06  } POINT;
07
08  void print_point(POINT pt);
09
10  int main(void)
11  {
12      POINT arr[] = {
13          {0, 0}, {10, 10}, {20, 20}, {30, 30}, {40, 40},
14      };
15      int sz = sizeof(arr) / sizeof(arr[0]);
16      int i;
17
18      for (i = 0; i < sz; i++)
19      {
20          print_point(arr[i]);  // arr[i]를 pt로 복사해서 전달한다.
21          printf(" ");
22      }
23      printf("\n");
24
25      return 0;
26  }
27
28  void print_point(POINT pt)    // 값에 의한 전달
29  {
30      printf("(%d, %d)", pt.x, pt.y);
31  }
```

실행결과

```
(0, 0) (10, 10) (20, 20) (30, 30) (40, 40)
```

함수의 인자로 구조체 전달하기 (2/3)

❖포인터에 의한 전달

- 구조체를 복사하지 않고 전달하려면 포인터로 전달

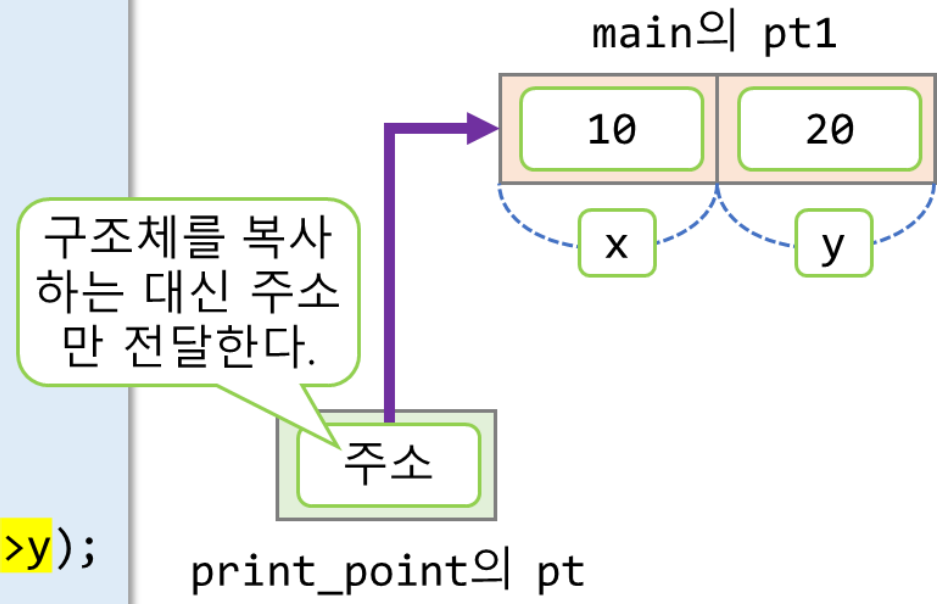
포인터에 의한 전달

```
int main(void)
{
    POINT pt1 = { 10,20 };
    print_point(&pt1);
    ⋮
}

POINT *pt = &pt1;

void print_point(POINT *pt)
{
    printf("(%d, %d)", pt->x, pt->y);
}
```

구조체 포인터로 멤버에 접근한다.



예제 : 구조체를 포인터로 전달하는 경우

```c
typedef struct point
{
    int x, y;       // 점의 좌표
} POINT;

void print_point(POINT *pt);

int main(void)
{
    POINT arr[] = {
        {0, 0}, {10, 10}, {20, 20}, {30, 30}, {40, 40},
    };
    int sz = sizeof(arr) / sizeof(arr[0]);
    int i;

    for (i = 0; i < sz; i++)
    {
        print_point(&arr[i]); // 구조체 변수의 주소를 전달한다.
        printf(" ");
    }
    printf("\n");

    return 0;
}

void print_point(POINT *pt)  // 포인터에 의한 전달
{
    printf("(%d, %d)", pt->x, pt->y);
}
```

실행결과

```
(0, 0) (10, 10) (20, 20) (30, 30) (40, 40)
```

예제 : 구조체형의 출력 매개변수를 가진 함수의 정의

```c
#include <stdlib.h>
#include <time.h>

typedef struct point
{
    int x, y;       // 점의 좌표
} POINT;

void print_point(const POINT *pt);
void set_point(POINT *pt, int x, int y);

int main(void)
{
    POINT arr[5] = { 0 };
    int sz = sizeof(arr) / sizeof(arr[0]);
    int i;

    srand((unsigned int)time(NULL));    // 난수의 시드를 지정한다.
    for (i = 0; i < sz; i++)
    {
        int x = rand() % 100;           // 0~99사이의 임의의 정수를 생성한다.
        int y = rand() % 100;
        set_point(&arr[i], x, y);
    }
    for (i = 0; i < sz; i++)
    {
        print_point(&arr[i]);
        printf(" ");
    }
    printf("\n");

    return 0;
}
```

```c
// 점의 좌표를 출력하는 함수 (pt는 입력 매개변수)
void print_point(const POINT *pt)      // 포인터에 의한 전달
{
    printf("(%d, %d)", pt->x, pt->y);
}

// 점의 좌표를 변경하는 함수 (pt는 출력 매개변수)
void set_point(POINT *pt, int x, int y)
{
    pt->x = x;      // pt가 가리키는 POINT 변수의 멤버 x를 변경한다.
    pt->y = y;      // pt가 가리키는 POINT 변수의 멤버 y를 변경한다.
}
```

실행결과

(49, 47) (6, 89) (12, 4) (33, 32) (99, 52)

난수를 생성해서 사용하므로 실행할 때마다 좌표가 달라진다.

함수의 인자로 구조체 전달하기 (3/3)

- 구조체 변수를 함수의 매개변수로 전달하는 방법
  1. 함수의 매개변수는 구조체 포인터형을 선언
  2. 구조체 변수가 입력 매개변수일 때는 const 키워드를 지정
  3. 구조체를 매개변수로 전달받는 함수를 호출할 때는 구조체 변수의 주소를 인자로 전달
  4. 함수를 정의할 때는 매개변수인 포인터로 구조체의 멤버에 접근

```
void print_point(POINT *pt);
```

```
void print_point(const POINT *pt);
```

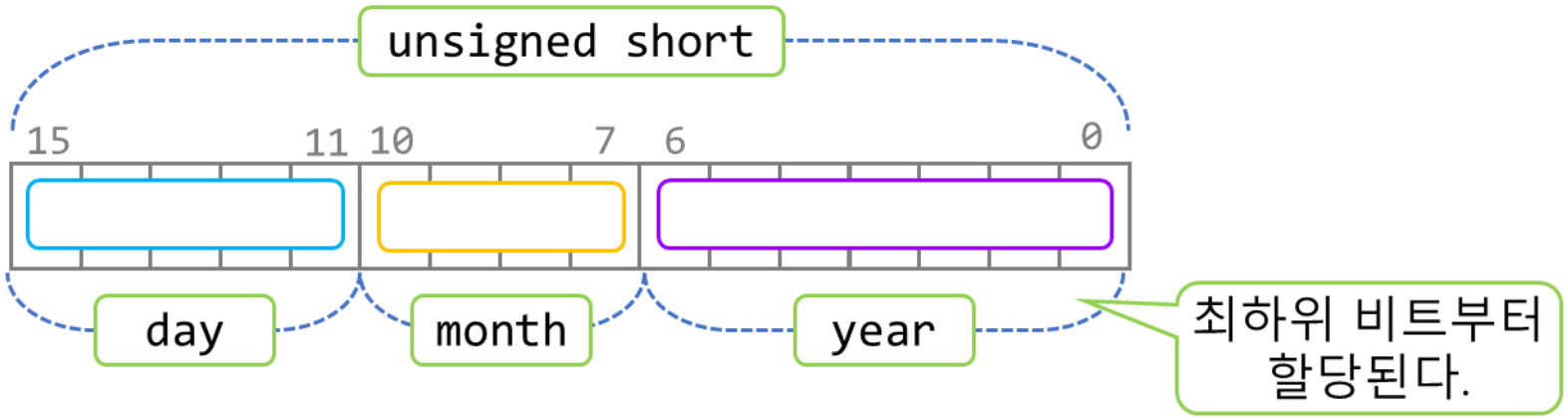
```
POINT pt = { 10, 20 };
print_point(&pt);
```

```
void print_point(const POINT *pt)
{
    printf("(%d, %d)", pt->x, pt->y);
}
```

비트필드 (1/3)

- 구조체의 멤버를 비트 단위로 사용하도록 설정할 수 있음

```c
typedef struct date {
    unsigned short year:7;    // 7비트에 연도를 저장한다. (0~99사이의 값)
    unsigned short month:4;   // 4비트에 월을 저장한다. (1~12 사이의 값)
    unsigned short day:5;     // 5비트에 일을 저장한다. (1~31 사이의 값)
} DATE;
```



비트필드 (2/3)

- 비트필드 멤버에 주어진 비트로 표현할 수 있는 범위를 넘어서는 값을 저장하면 오버플로우가 발생

```
DATE dday;
dday.year = 18;
dday.month = 11;
dday.day = 40;
```

오버플로우가 발생해서 8이 된다.

- 구조체 안에 일반 멤버와 비트필드를 함께 정의할 수도 있음

```
typedef struct date {
    unsigned short year : 7;
    unsigned short month : 4;
    unsigned short day : 5;
    char the_day_of_week[4];
} DATE;
```

일반 멤버를 함께 정의할 수 있다.

예제 : 비트필드의 정의 및 사용

```
03  typedef struct date {
04      unsigned short year:7;
05      unsigned short month:4;
06      unsigned short day:5;
07      //unsigned short the_day_of_week : 3;
08  } DATE;
09
10  int main(void)
11  {
12      DATE dday;
13      dday.year = 18; // 연도를 0~99사이의 값으로 저장한다.
14      dday.month = 11;
15      dday.day = 30;
17      printf("DATE의 크기 = %d\n", sizeof(DATE));
18      printf("%d/%d/%d\n", dday.year+2000, dday.month, dday.day);
19
20      return 0;
21  }
```

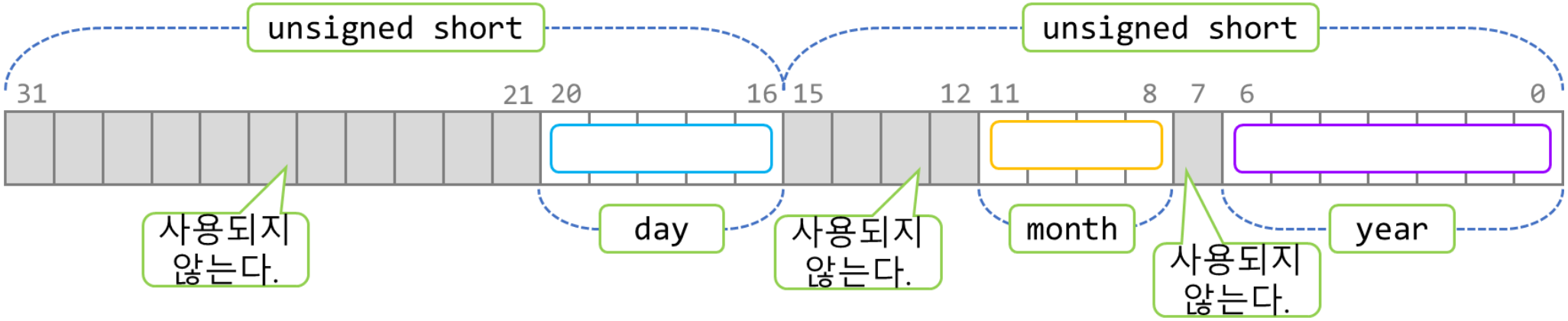
실행결과

```
DATE의 크기 = 2
2018/11/30
```

비트필드 (3/3)

- 중간에 일부 비트를 비워 두고 멤버를 특정 비트에 할당할 수 있음

```
typedef struct date {
    unsigned short year : 7;
    unsigned short : 1;       // 사용하지 않는다.
    unsigned short month : 4;
    unsigned short : 4;       // 사용하지 않는다.
    unsigned short day : 5;
} DATE;
```

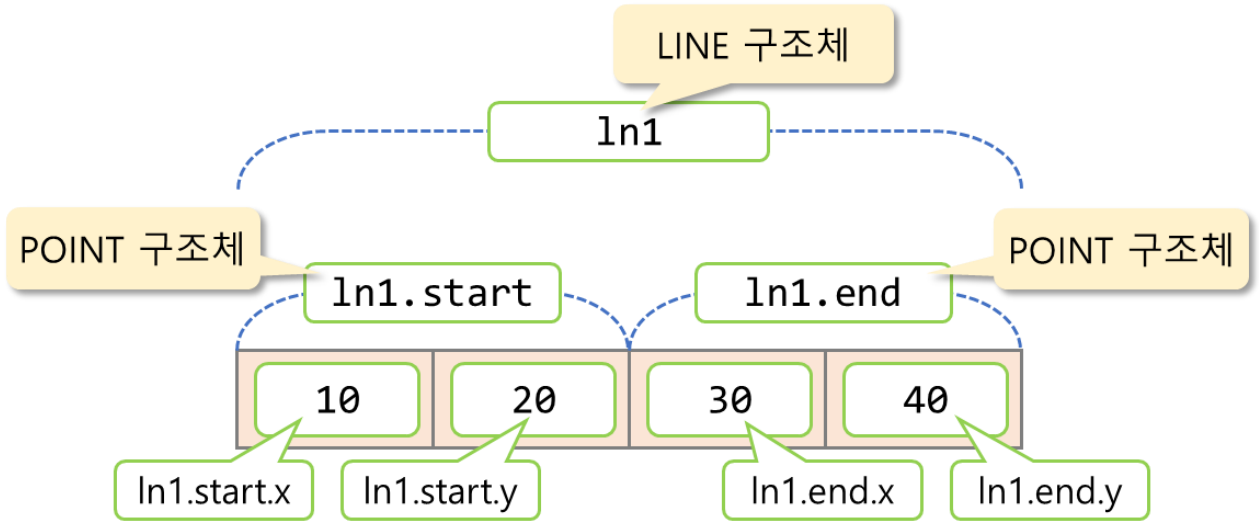


구조체의 멤버로 다른 구조체 변수 사용하기

- 구조체 변수 안에 다른 구조체 변수가 멤버로 포함될 수 있음

```
typedef struct point {
    int x, y;
} POINT;
typedef struct line {
    POINT start, end;
} LINE;

LINE ln1 = { {10, 20}, {30, 40} };
```



입력 매개변수이므로
const 포인터로 전달

```
double get_length(const LINE *ln) {
    int dx = ln->end.x - ln->start.x;
    int dy = ln->end.y - ln->start.y;
    return sqrt(dx*dx + dy * dy);
}
```

예제 : LINE 구조체의 정의 및 사용

```c
#include <math.h>

typedef struct point          // 점의 좌표
{
    int x, y;
} POINT;

typedef struct line// 직선
{
    POINT start, end;         // 멤버로 다른 구조체의 변수를 선언한다.
} LINE;

double get_length(const LINE *ln);

int main(void)
{
    LINE ln1 = { {10,20}, {30,40} };

    printf("직선의 시작점: ((%d, %d)\n", ln1.start.x, ln1.start.y);
    printf("직선의 끝점: (%d, %d)\n", ln1.end.x, ln1.end.y);

    printf("직선의 길이: %f\n", get_length(&ln1));
    return 0;
}

double get_length(const LINE *ln)       // 직선의 길이 구하는 함수
{
    int dx = ln->end.x - ln->start.x;
    int dy = ln->end.y - ln->start.y;
    return sqrt(dx*dx + dy * dy);
}
```

실행결과

```
직선의 시작점:  (10, 20)
직선의 끝점: (30, 40)
직선의 길이: 28.284271
```

열거체

- 정수형 변수가 특정 값들 중 한가지 값을 가질 때 유용하게 사용
- 정수형 변수가 가질 수 있는 값들을 열거 상수로 정의

정수형 변수와 매크로 상수를 사용하는 경우

```
#define NORTH     0
#define SOUTH     1
#define EAST      2
#define WEST      3
#define NORTHEAST     4


int main(void)
{
     int d1 = NORTH;
     d1 = EAST;
      ⋮
}
```

값을 직접 정의해야 한다.

매크로 상수를 추가하려면 값을 직접 지정해야 한다.

열거체와 열거 상수를 사용하는 경우

```
enum direction {
   NORTH, SOUTH, EAST, WEST,
   NORTHEAST
};

int main(void)
{
     enum direction d1 = NORTH;
     d1 = EAST;
   ⋮
}
```

자동으로 값이 지정된다.

열거 상수를 추가하기 쉽다.

열거체의 정의

- ❖열거체는 C 컴파일러에 의해 int형으로 처리됨
- ❖열거 상수는 정수형 상수가 됨
  - 따로 지정하지 않으면 열거 상수는 0부터 1씩 증가되는 값으로 정의됨
- ❖열거 상수를 특정 값으로 정의할 수도 있음

형식 enum 태그명 {열거상수1, 열거상수2, ...};

사용예

```
enum color {RED, GREEN, BLUE};
enum direction {
    NORTH, SOUTH, EAST, WEST
};
```

```
enum direction {NORTH=1, SOUTH=-1, EAST=10, WEST=-10};
```

```
enum direction {NORTH=1, SOUTH, EAST, WEST};
```

2 3 4

예제 : 열거체와 열거 상수의 정의 및 사용

```
03  enum direction {NORTH, SOUTH, EAST, WEST};
04
05  int main(void)
06  {
07      enum direction d1 = NORTH;          // 열거체 변수 선언
08
09      d1 = EAST;                          // 열거체 변수에 열거 상수를 대입한다.
10      printf("d1 = %d\n", d1);            // 2가 출력된다.
11
12      switch (d1)
13      {
14      case NORTH:                         // 열거 상수는 case문에 사용할 수 있다.
15          printf("북쪽으로 이동합니다.\n");
16          break;
17      case SOUTH:
18          printf("남쪽으로 이동합니다.\n");
19          break;
20      case EAST:
21          printf("동쪽으로 이동합니다.\n");
22          break;
23      case WEST:
24          printf("서쪽으로 이동합니다.\n");
25          break;
26      }
27      return 0;
28  }
```

실행결과

```
d1 = 2
동쪽으로 이동합니다.
```

열거형과 다른 방법과의 비교

정수 사용	기호 상수	열거형

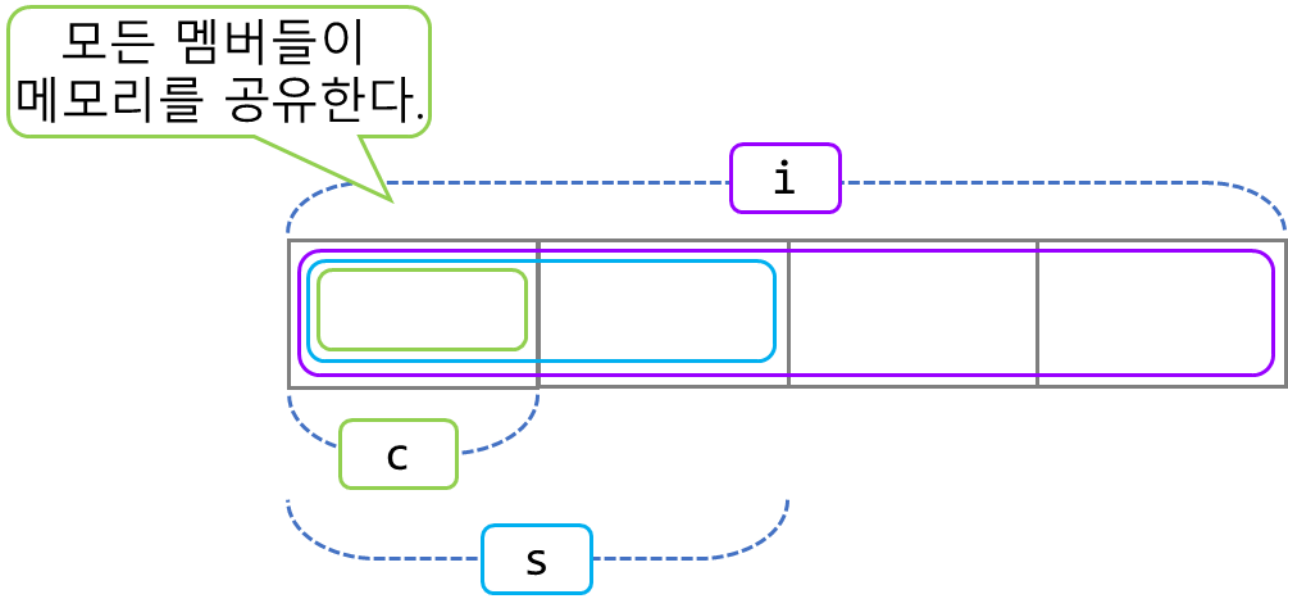
switch(code) {
  case 1:
  printf(“LCD TV\n”);
  break;
  case 2:
  printf(“PDP TV\n”);
  break;
}
	#define LCD 1
	#define PDP 2
	
	switch(code) {
	  case LCD:
	  printf(“LCD TV\n”);
	  break;
	  case PDP:
	  printf(“PDP TV\n”);
	  break;
	}
		enum tvtype { LCD, PDP };
		enum tvtype code;
		
		switch(code)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

공용체 (1/2)

- 공용체의 멤버들은 같은 주소에 할당되고, 서로 메모리를 공유함

```
union test {    // 공용체의 모든 멤버가 같은 주소에 할당된다.
    int i;
    char c;
    short s;
};
```



공용체 (2/2)

- 공용체의 크기는 멤버 중 가장 큰 멤버의 크기와 같음

```
printf("sizeof(union test) = %d\n", sizeof(union test));    // 4바이트 크기
```

- 공용체의 모든 멤버는 메모리를 공유하기 때문에 주소가 같음

```
printf("t1.i의 주소 = %p\n", &t1.i);    // 멤버들의 주소가 모두 같다.
printf("t1.c의 주소 = %p\n", &t1.c);
printf("t1.s의 주소 = %p\n", &t1.s);
```

- 공용체를 초기화할 때는 { } 안에 첫 번째 멤버의 초기값만 지정

```
union test t1 = { 0x12345678 };        // t1.i를 초기화한다.
```

예제 : 공용체의 정의 및 사용

```
03  union test {
04      int i;                          // 모든 멤버가 같은 주소에 할당된다.
05      char c;
06      short s;
07  };
08
09  int main(void)
10  {
11      union test t1 = { 0x12345678 };    // t1.i를 초기화한다.
12
13      printf("sizeof(union test) = %d\n", sizeof(union test));
14
15      printf("t1.i의 주소 = %p\n", &t1.i);// 멤버들의 주소가 모두 같다.
16      printf("t1.c의 주소 = %p\n", &t1.c);
17      printf("t1.s의 주소 = %p\n", &t1.s);
19      printf("t1.i = %x\n", t1.i);          // 12345678 출력
20      printf("t1.c = %x\n", t1.c);          // 78 출력
21      printf("t1.s = %x\n", t1.s);          // 5678 출력
22
23      return 0;
24  }
```

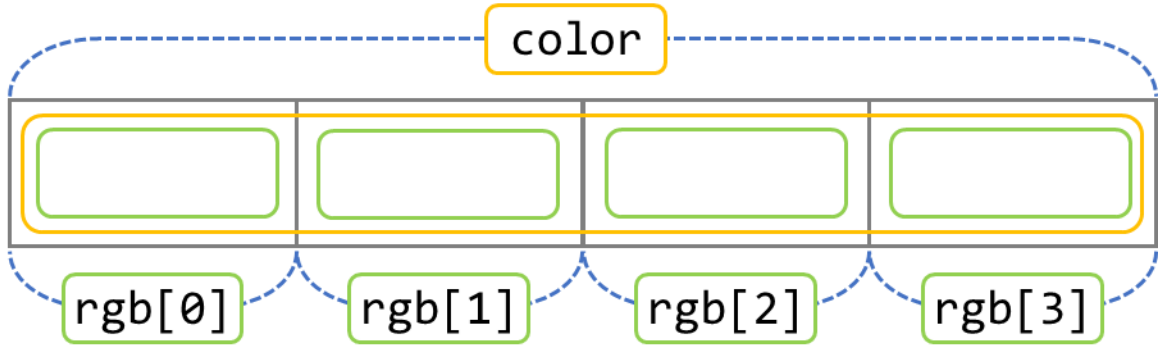
실행결과

```
sizeof(union test) = 4
t1.i의 주소 = 00EFFD6C
t1.c의 주소 = 00EFFD6C
t1.s의 주소 = 00EFFD6C
t1.i = 12345678
t1.c = 78
t1.s = 5678
```

모든 멤버의 주소가 같다.

공용체의 사용 예

```
typedef union color_t {
    unsigned int color;
    unsigned char rgb[4];
} COLOR_T;
```

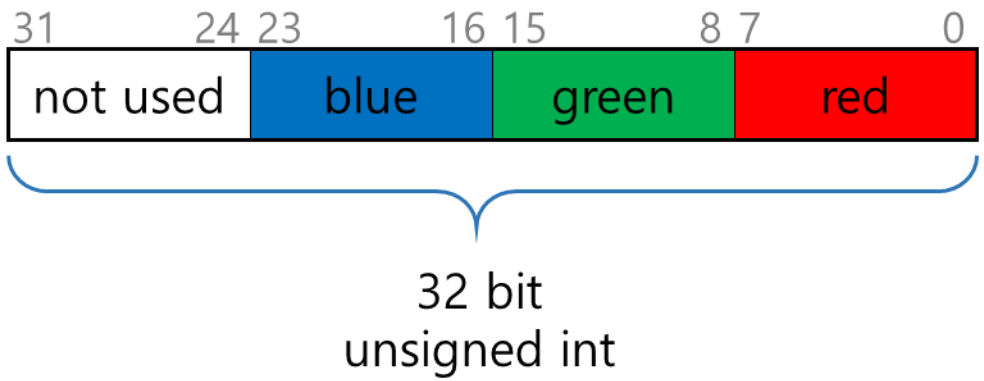



```
COLOR_T c1;
c1.rgb[0] = 0xFF; // red
c1.rgb[1] = 0xAB; // green
c1.rgb[2] = 0x1F; // blue
c1.rgb[3] = 0x0;  // not used

printf("rgb color = %08X\n", c1.color);
```

001FABFF
출력

24비트 트루 컬러



예제 : 공용체를 이용한 RGB 색상 표현

```
03  typedef union color_t {
04      unsigned int color;
05      unsigned char rgb[4];     // rgb[0]은 red, rgb[1]은 blue, rgb[2]는 green,
06                                // rgb[3]은 not used
07  } COLOR_T;
08  int main(void)
09  {
10      COLOR_T c1;
11
12      c1.rgb[0] = 0xFF;         // red
13      c1.rgb[1] = 0xab;         // green
14      c1.rgb[2] = 0x1F;         // blue
15      c1.rgb[3] = 0x0;          // not used
16
17      printf("rgb color = %08X\n", c1.color);       // 32비트 데이터로 사용한다.
18
19      return 0;
20  }
```

실행결과

```
rgb color = 001FABFF
```

질의 및 응답

참고문헌

- 천정아,『Core C Programming』, 연두에디션(2019)
- C가 보이는 그림책, ANK Co., Ltd. , 성안당 (2018)
- Greg Perry, Dean Miller『어서와 C언어는 처음이지』, 천인국 옮김, 인피니티북스(2015)
- KELLEY (역 : 김명호 외),『A Book on C』, 홍릉과학출판사(2003)
- 윤성우,『 열혈 C 프로그래밍』, 오렌지미디어
- 천인국,『 쉽게 풀어쓴 C언어 Express』, 생능출판사
- 서현우,『 뇌를 자극하는 C 프로그래밍』, 한빛미디어
- 강성수,『 쾌도난마 C프로그래밍』, 북스홀릭
- 고응남,『 C프로그래밍 기초와 응용실습』, 정익사