강06. 구조체와 사용자 정의 자료형

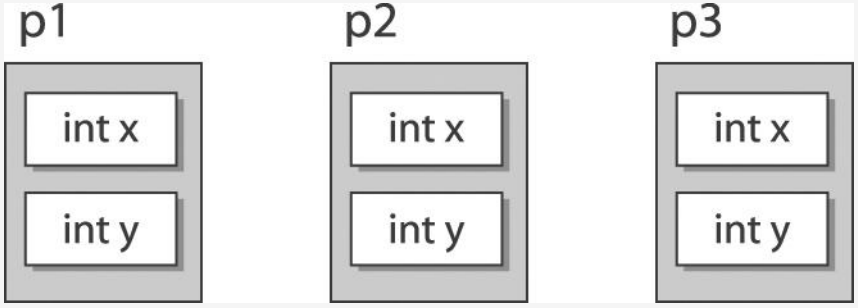
22장. 구조체와 사용자 정의 자료형 1

- 구조체의 정의
  - 하나 이상의 기본 자료형을 기반으로 사용자 정의 자료형을 만들 수 있는 문법 요소

```
struct point                // point라는 이름의 구조체 선언
{
    int x;                  // 구조체 멤버 int x
    int y;                  // 구조체 멤버 int y
};
```

- 구조체 변수의 선언: case 1

```
struct point {
   int x;
   int y;
} p1, p2, p3;

int main(void)
{
   . . . . .
}
```



- 구조체 변수의 선언: case 2

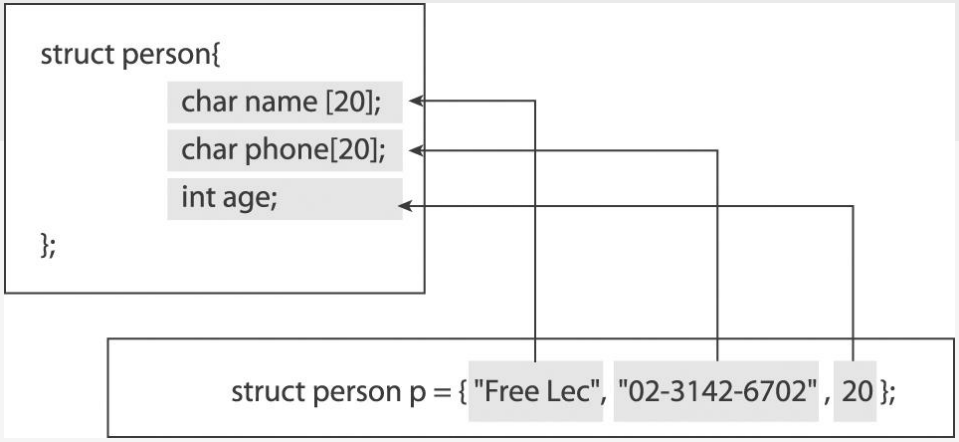
```
struct point {
    int x;
    int y;
};

int main(void)
{
   struct point p1, p2, p3;
   …
   return 0;
}
```

- 구조체 변수의 접근

```
struct point {
    int x;
    int y;
};

int main(void)
{
   struct point p1;
   p1.x=10;                          // p1의 멤버 x에 10을 대입
   p1.y=20;                          // p1의 멤버 y에 20을 대입
   …
   return 0;
}
```

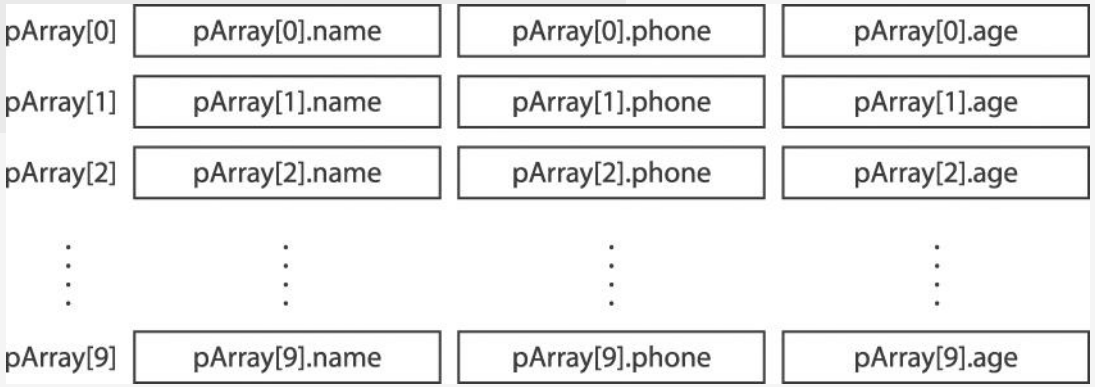
- 구조체 변수의 초기화
  - 배열 초기화 문법과 일치

```
struct person {
    char name[20];
    char phone[20];
    int age;
};
int main (void)
{
    struct person p={"Free Lec", "02-3142-6702", 20};
    ...
    return 0;
}
```



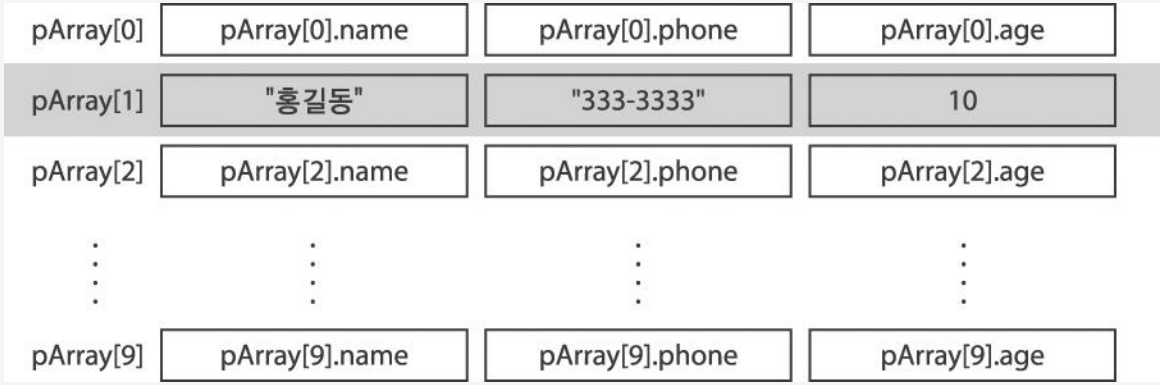
- 구조체 배열의 선언

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

int main (void)
{
    struct person pArray[10];
    ...
    return 0;
}
```



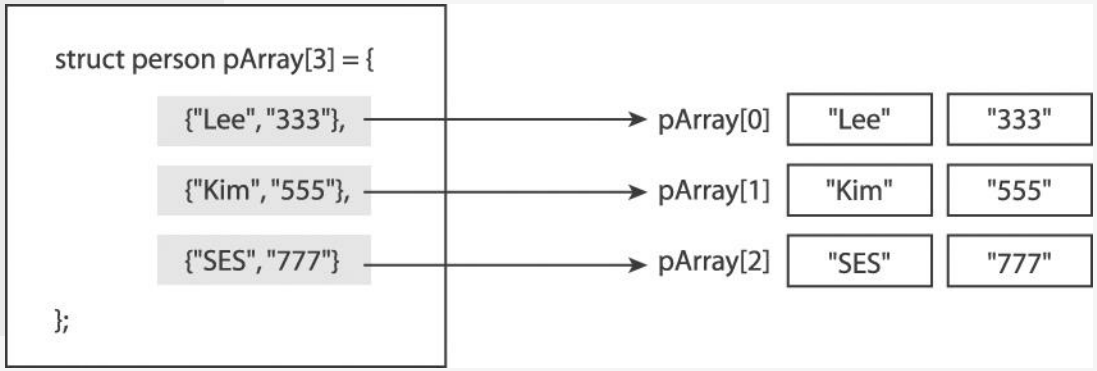
- 구조체 배열 요소의 접근

```
pArray[1].age=10;                                    // 두 번째 요소의 age에 접근
strcpy(pArray[1].name, “홍길동”);                    // 두 번째 요소의 name에 접근
strcpy(pArray[1].phone, “333-3333”);                 // 두 번째 요소의 phone에 접근
```



- 구조체 배열의 초기화

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

int main (void)
{
   struct person pArray[3]={
          {"Lee", "333"},
          {"Kim", "555"},
          {"SES", "777"}
    };
   .....
   return 0;
}
```



- 구조체와 포인터
  - 첫째 : 구조체 포인터를 선언하여 구조체 변수를 가리키는 경우
  - 둘째 : 구조체의 멤버로 포인터 변수가 선언되는 경우

구조체와 배열 그리고 포인터 사용례

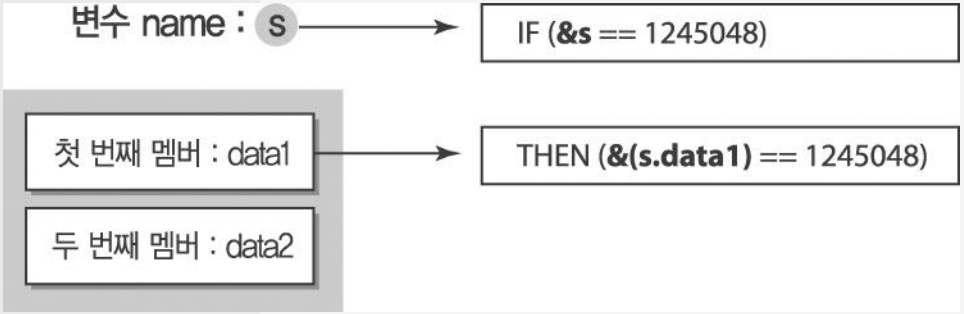
```
struct person {
      char name[20];
      char phone[20];
};

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;
}
```

```
#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;
}
```

```
struct person {
    char name[20];
    char pID[20];
    struct person* frnd;
};

int main()
{
    struct person man1={"Mr. Lee", "820204-0000512"};
    struct person man2={"Mr. Lee's Friend", "820000-0000101"};

    man1.frnd=&man2;

    printf("[Mr. Lee]₩n");
    printf("name : %s₩n", man1.name);
    printf("pID  : %s₩n", man1.pID);

    printf("[His Friend]₩n");
    printf("name : %s₩n", man1.frnd->name);
    printf("pID  : %s₩n", man1.frnd->pID);
    return 0;
}
```

- 구조체 변수와 주소 값의 관계

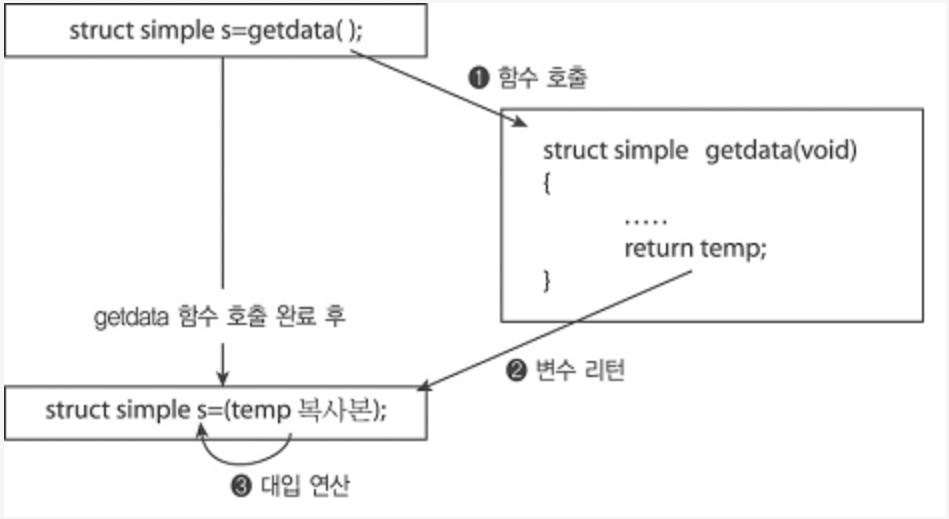
```
/* pointer_pointer.c */
#include <stdio.h>

struct simple {
      int data1;
      int data2;
};
int main()
{
      struct simple s={1, 2};

      printf("address1 : %d₩n", &s);
      printf("address2 : %d₩n", &(s.data1));
      return 0;
}
```



23장. 구조체와 사용자 정의 자료형 2

- 함수의 인자로 전달되는 구조체 변수
  - 구조체 변수의 인자 전달 방식은 기본 자료형 변수의 인자 전달 방식과 동일
- 구조체 변수의 연산
  - 허용되는 대표적인 연산은 대입 연산(=)이며, 이외의 사칙 연산들은 적용 불가능

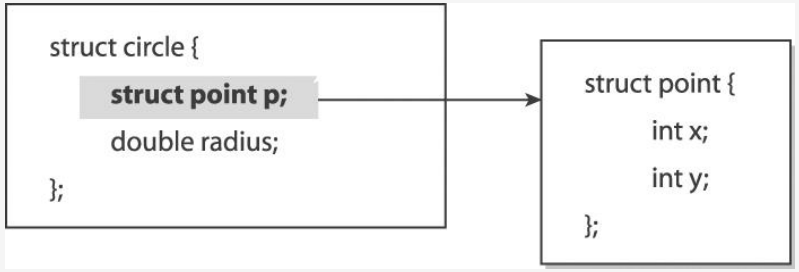
- 구조체 변수의 리턴 방식
  - 기본 자료형 변수의 리턴 방식과 동일

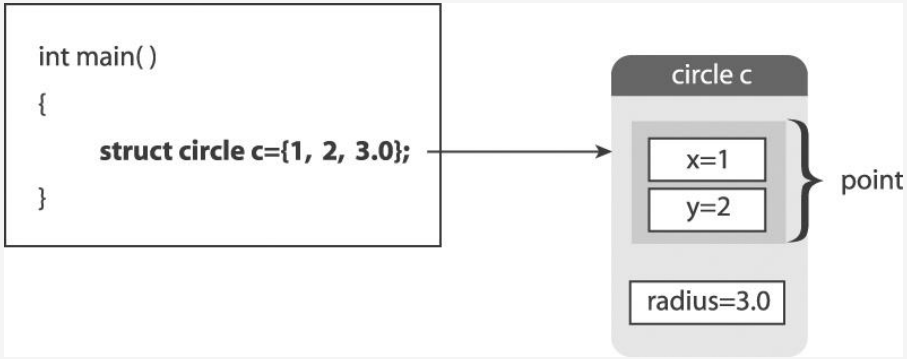


- 잘 구현된 프로그램은 처리되어야 할 데이터의 부류가 적절히 나뉘어진다.

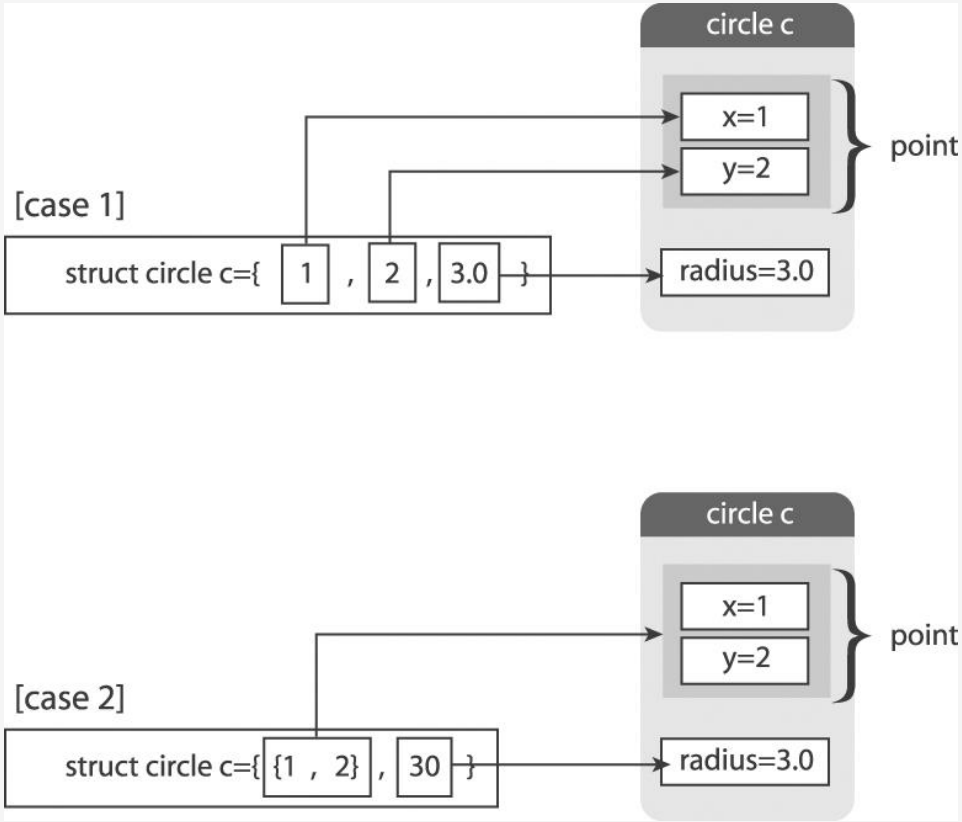
- 부류를 적절히 나누면 데이터를 처리하는 과정이 수월해진다.

- 중첩된 구조체
  - 구조체의 멤버로 구조체 변수가 오는 경우

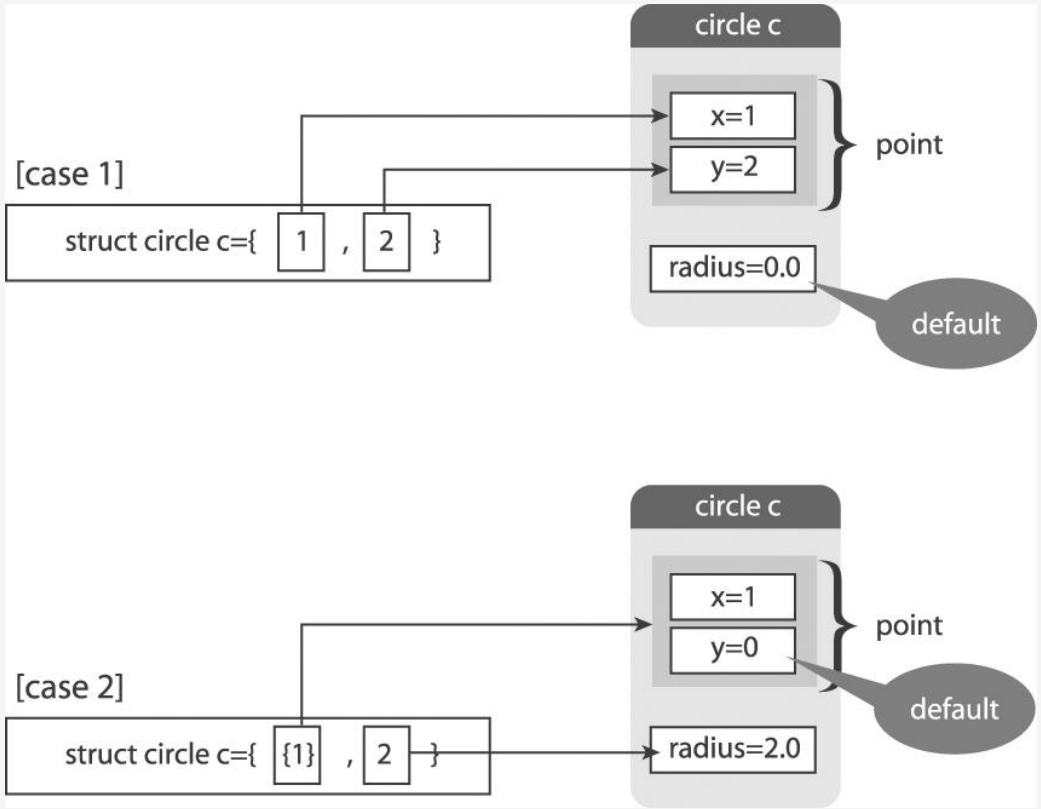




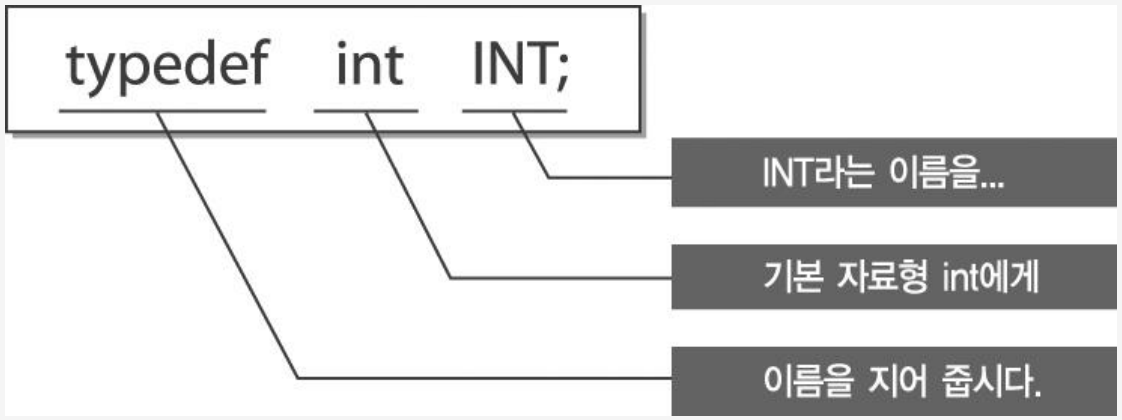
- 중첩된 구조체 변수의 초기화 방식
  - case 1



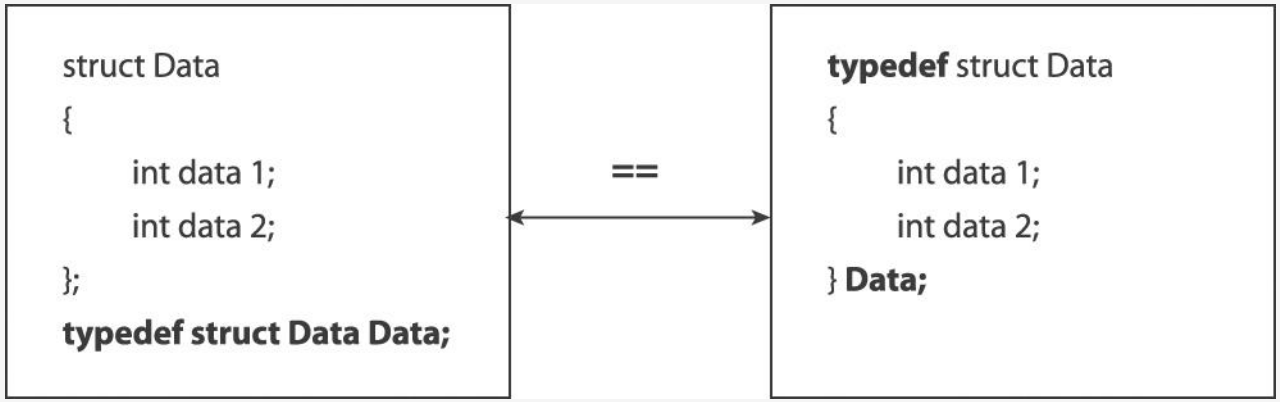
- 중첩된 구조체 변수의 초기화 방식
  - case 2



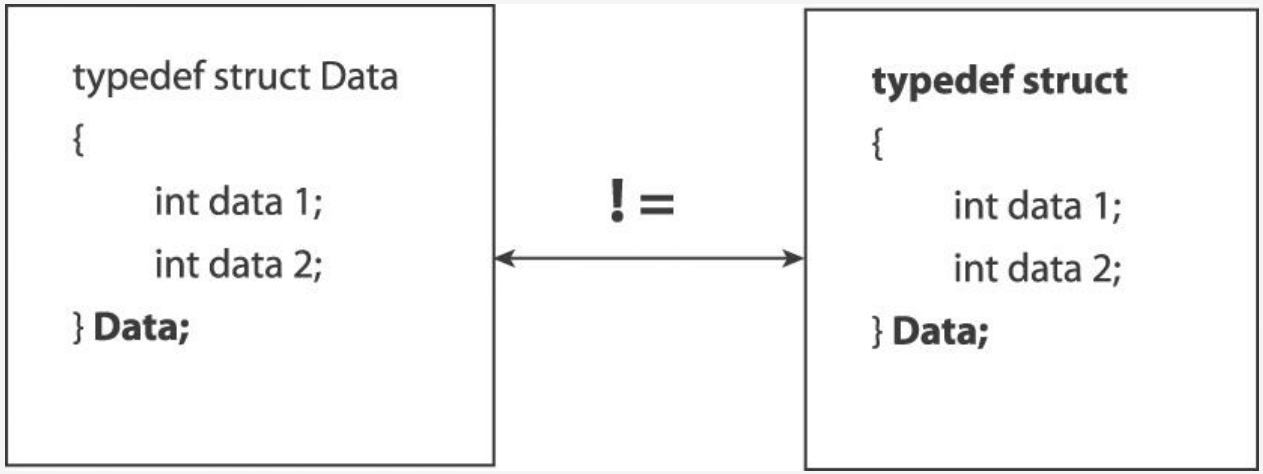
- typedef 키워드의 이해



- typedef의 적용

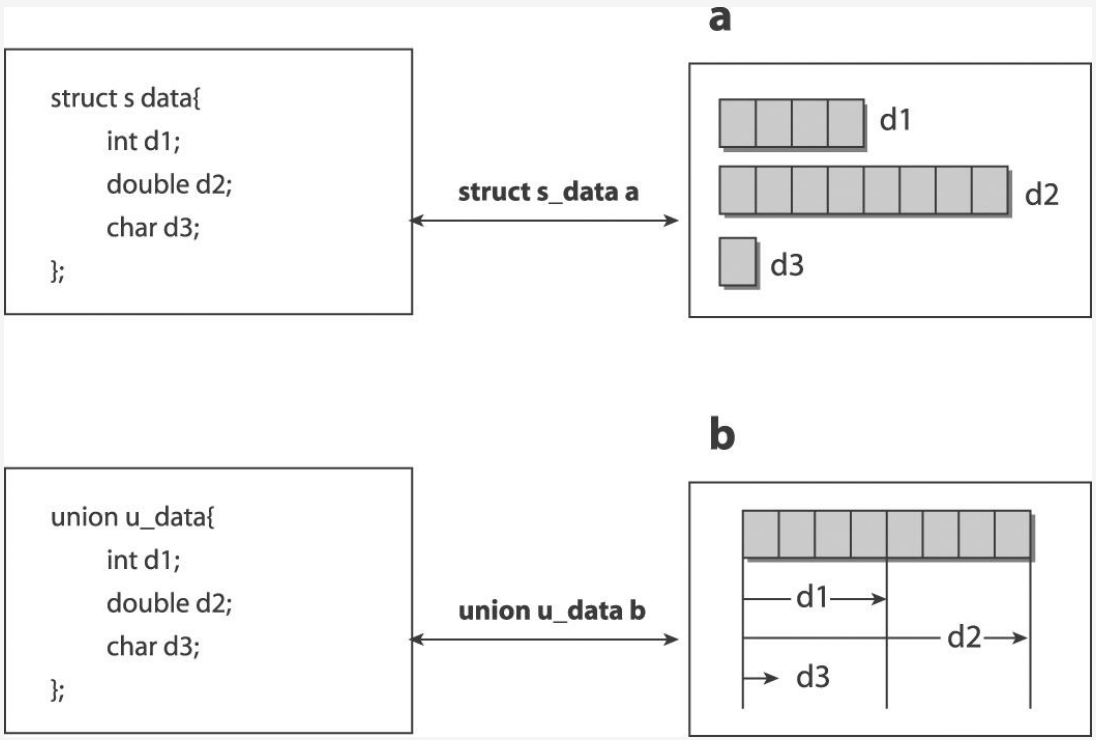

struct Data
{
int data 1;
int data 2;
};
typedef struct Data Data;
==
typedef struct Data
{
int data 1;
int data 2;
} Data;

- 구조체 이름의 생략


typedef struct Data
{
int data 1;
int data 2;
} Data;
!=
typedef struct
{
int data 1;
int data 2;
} Data;

UNION – 공용체

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



Enum 열거형

- 열거형의 정의와 의미


```
enum color c          // 열거형 color의 변수 c를 선언
    c=RED;            // c에 RED 대입
  c=GREEN;            // c에 GREEN 대입
   c=BLUE;            // c에 BLUE 대입
```

- 할당되는 상수의 값

```
enum color {RED, GREEN, BLUE};
enum color {RED, GREEN=100, BLUE};
```

- 열거형을 사용하는 이유
  - 특정 정수 값에 의미를 부여할 수 있다.
  - 따라서 프로그램의 가독성을 높이는데 한몫을 한다.