

# 제 13 장 구조체와 공용체

---

- 01 구조체와 공용체
- 02 자료형 재정의
- 03 구조체와 공용체의 포인터와 배열



# 학습목표

- ▶ 구조체와 공용체를 이해하고 설명할 수 있다.
  - 구조체의 개념과 정의 방법
  - 필요한 구조체 변수의 선언 방법
  - 구조체 변수의 접근연산자 .의 사용 방법
  - 공용체 정의와 변수 선언 및 활용 방법
- ▶ 자료형 재정의를 위한 typedef를 사용할 수 있다.
  - 키워드 typedef를 사용한 자료형 재정의 방법과 필요성
  - 구조체 정의를 새로운 자료형으로 재정의
- ▶ 구조체 포인터와 배열을 활용할 수 있다.
  - 구조체의 주소를 저장하는 포인터의 선언과 활용
  - 구조체 포인터의 접근연산자 ->의 사용 방법
  - 구조체 배열의 선언과 활용 방법

# 구조체 개념

## • 선물세트

- 인기가 있거나 관련 있는 상품들을 묶어 하나의 구성제품으로 판매하는 것

## • 구조체

- 정수, 문자, 실수나 포인터 그리고 이들의 배열 등을 묶어 하나의 자료형으로 이용하는 것
- 서로 관련 있는 정보들을 하나로 묶어 처리하는 경우가 흔히 발생
  - 차에 대한 정보, 계좌에 대한 정보, 책에 대한 정보, 학생, 교수, 강좌에 관한 정보
- C 언어는 이러한 요구사항을 구조체(struct)로 지원
  - 연관된 멤버로 구성되는 통합 자료형으로 대표적인 유도 자료형
- 유도 자료형(derived data types)
  - 기존 자료형에서 새로이 만들어진 자료형



구조체

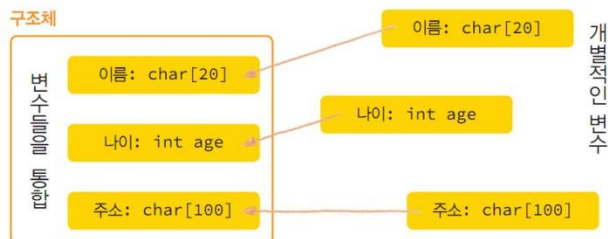


그림 13-1 선물세트와 구조체

학생정보



교수정보



여러 자료형의 통합체인 학생, 교수, 강좌 등을 새로운 하나의 자료형인 구조체로 정의

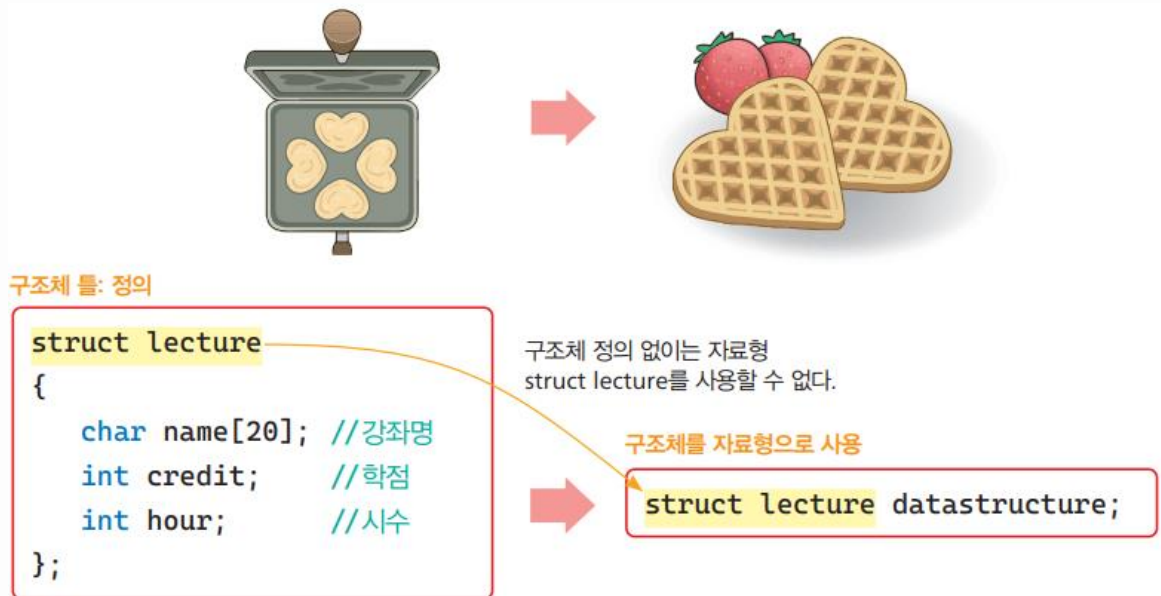
강좌정보



그림 13-2 구조체 개념

# 구조체 정의 개념

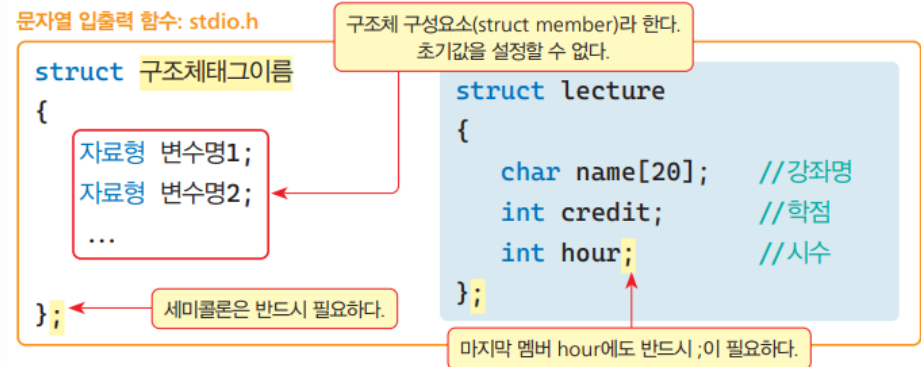
- 와플이나 봉어빵을 만들려면
  - 와플 기계나 봉어빵 기계가 필요하듯이
- 구조체를 자료형으로 사용하려면
  - 먼저 구조체를 정의
    - 구조체를 만들 구조체 틀(template)을 정의
- 구조체 틀을 만드는 구조체 정의 방법
  - 키워드 struct 다음에 구조체 태그이름을 기술
    - 중괄호를 이용하여 원하는 멤버를 여러 개의 변수로 선언하는 구조
  - 구조체 멤버(member) 또는 필드(field)
    - 구조체를 구성하는 하나 하나의 항목



# 구조체 정의 구문

## • 대학의 강좌정보를 처리하는 구조체의 한 예: struct lecture

- 구조체 정의는 변수의 선언과는 다름
  - 변수선언에서 이용될 새로운 구조체 자료형을 정의하는 구문
- 반드시 세미콜론으로 종료
  - 모든 멤버 선언에 반드시 세미콜론 삽입, 마지막 멤버도
- 각 구조체 멤버의 초기값 대입 불가능
  - `int credit; int hour;`
  - `Int credit, hour;`로도 가능
- 구조체 멤버의 이름은 모두 유일
  - 멤버로는 다양한 자료형, 다른 구조체 변수 및 구조체 포인터도 허용



## • 구조체 태그이름: account

- `struct account`
  - 계좌정보를 표현하는 구조체
- 계좌주이름, 계좌번호, 잔고 정보를 하나의 단위로 처리하는 자료형을 정의

```
struct account
{
    char name[10]; //계좌주이름
    int actnum; //계좌번호
    double balance; //잔고
};
```



# 구조체 변수 선언

- 구조체가 정의되었다면

- 구조체형 변수 선언이 가능
  - 구조체 `struct account`가 새로운 자료유형으로 사용 가능
- 새로운 자료형 `struct account` 형 변수 `mine`을 선언 구문
  - `struct account mine;`

```
struct 구조체태그이름 변수명;  
struct 구조체태그이름 변수명1, 변수명2, 변수명3, ... ;  
  
struct account yours;  
struct account act1, act2, act3;
```

여러 변수의 선언도 가능하다.

- 구조체 정의와 변수 선언을 함께하는 방법

- 이 문장 이후 `struct account`도 새로운 자료형으로 사용 가능

```
struct account  
{  
    char name[12];    //계좌주이름  
    int actnum;        //계좌번호  
    double balance;    //잔고  
} myaccount;  
  
struct account youraccount;
```

변수 `myaccount`는 `struct account`형 변수로 선언된다.

변수 `youraccount`도 `struct account`형 변수로 선언된다.

# 이름 없는 구조체

- 구조체 태그이름이 없는 구조체변수 선언 구문
  - 이 구조체와 동일한 자료형의 변수를 더 이상 선언 불가능
    - 단 한번 이 구조체 형으로 변수를 선언하는 경우에만 이용
    - 단 이러한 태그이름이 없는 구조체 정의
      - 바로 변수가 나오지 않는다면 아무 의미 없는 문장



TIP

이름 없는 구조체

구조체 변수 선언 구문에서 다음과 같이 구조체 태그이름을 생략할 수 있으나, 이러한 이름이 없는 변수 선언 방법으로는 이 구조체와 동일한 자료형의 변수를 더 이상 선언할 수 없다. 그러므로 단 한번 이 구조체 형으로 변수를 선언하는 경우에만 이용할 수 있는 방법이다. 단 이러한 태그이름이 없는 구조체 정의에서는 바로 변수가 나오지 않는다면 아무 의미 없는 문장이 된다.

```
struct
{
    char name[12];      //계좌주이름
    int actnum;         //계좌번호
    double balance;     //잔고
} youraccount;
```

변수 youraccount 이후에 이와 동일한 구조체의 변수 선언은 불가능하다.

그림 13-8 구조체 태그이름이 없는 변수 선언 방법

# 구조체 변수의 초기화

- 변수 선언 시 중괄호를 이용한 초기화 지정이 가능
  - 초기화 값은 중괄호 내부에서 각 멤버 정의 순서대로 초기값을 쉼표로 구분하여 기술
  - 기술되지 않은 멤버값은 자료형에 따라 기본값인 0, 0.0, '₩0' 등으로 저장

```
struct 구조체태그이름 변수명 = {초기값1, 초기값2, 초기값3, ... };
```

```
struct account
{
    char name[12];      //계좌주이름
    int actnum;         //계좌번호
    double balance;     //잔고
};
```

```
struct account mine = {"홍길동", 1001, 300000 };
```

```
struct account mine = {"한국인", 1001};
```

멤버 balance에는 double형  
기본값인 0.0이 저장

그림 13-9 구조체 변수의 초기화



## TIP 버전 C99 추가 기능

구조체 초기화에서 멤버의 순서와 관계없이 ".멤버이름 = 초기값"으로 지정된 멤버에 초기값을 저장(designated initializer)할 수 있다.

```
struct account me = { .name = "홍길동", .balance = 50000 };
printf("%s %d %.2f\n", me.name, me.actnum, me.balance);
```



# 구조체의 멤버 접근 연산자 . 와 변수 크기

- 선언된 구조체형 변수에서 멤버 접근 방법

- 접근연산자 .를 사용하여 멤버를 참조

- `yours.actnum=1002;`

- 변수 `yours`의 멤버 `actnum`에 1002를 저장하는 기능을 수행

- 접근연산자는 .는 참조연산자라고도 부름

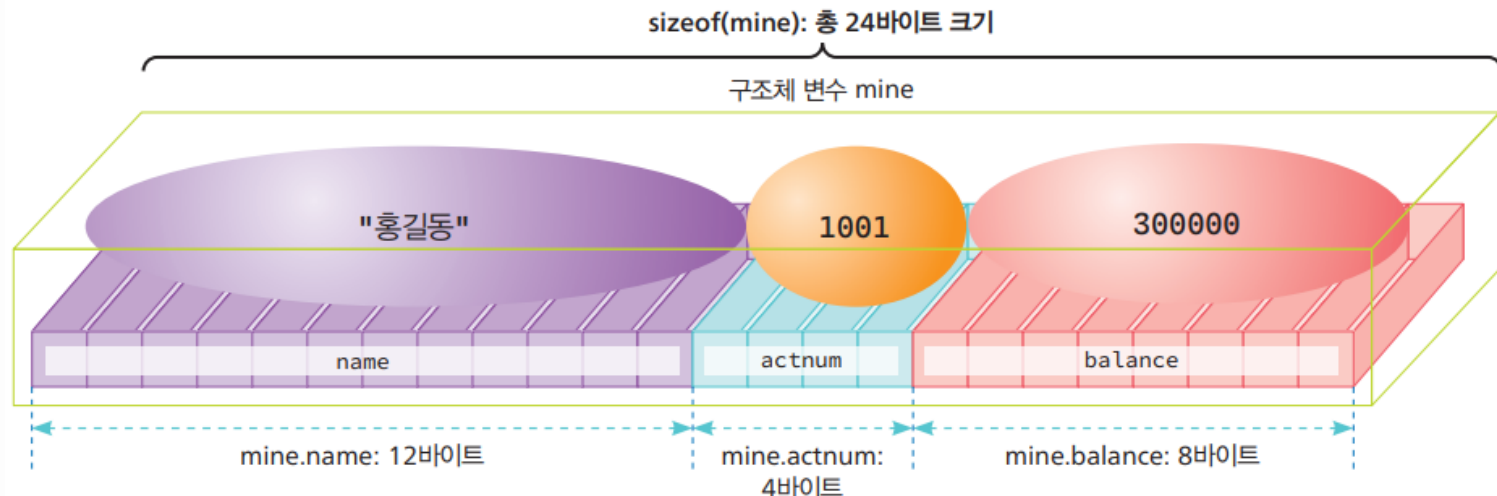
구조체변수이름.멤버

```
mine.actnum = 1002;   mine.balance = 300000;
```

- 구조체 `struct account` 의 변수 `mine`은 다음 구조로 메모리에 할당

- 변수 `mine`의 크기는 `sizeof(mine)`로 가능

- 실제 구조체의 크기는 멤버의 크기의 합보다 크거나 같을 수 있음



# 구조체 정의와 구조체 변수 선언

실습예제 13-1

Prj01

01structbasic.c

구조체 정의와 구조체 변수 선언

난이도: ★

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 #include <string.h>
04
05 //은행 계좌를 위한 구조체 정의
06 struct account
07 {
08     char name[12];    //계좌주 이름
09     int actnum;        //계좌번호
10     double balance;   //잔고
11 };
12
13 int main(void)
14 {
15     //구조체 변수 선언 및 초기화
16     struct account mine = { "홍길동", 1001, 3000000 };
17     struct account yours;
18
19     strcpy(yours.name, "이동원");
20     //strcpy_s(yours.name, 12, "이동원"); //가능
21     //yours.name = "이동원"; //오류
22     yours.actnum = 1002;
23     yours.balance = 5000000;
24
25     printf("구조체 크기: %zu\n", sizeof(mine));
26     printf("%s %d %.2f\n", mine.name, mine.actnum, mine.balance);
27     printf("%s %d %.2f\n", yours.name, yours.actnum, yours.balance);
28
29     return 0;
30 }
```



## TIP 버전 C99 추가 기능

이미 선언된 구조체 변수에도 쉽게 멤버 값을 지정하는 방법을 제공한다. 이미 선언된 구조체 `you`에 초기화 방법과 같이 멤버에 일괄적으로 값을 대입한 후 타입 변환을 사용하여 대입한다.

```
struct account you;
you = (struct account) { .name = "김파이", .balance = 70000 };
printf("%s %d %.2f\n", you.name, you.actnum, you.balance);
```

구조체 struct account 형인 mine을 선언하면서 초기화, 모든 멤버를 모두 초기화

연산자 sizeof(mine)로 변수 mine의 크기를 조회하여 출력

결과

```
구조체 크기: 24
홍길동 1001 3000000.00
이동원 1002 5000000.00
```

# 구조체 멤버로 사용되는 구조체

- 구조체 멤버로 가능

- 이미 정의된 다른 구조체 형 변수
- 자기 자신을 포함한 구조체 포인터 변수

- 구조체 **struct date**

- 년, 월, 일 정보를 저장할 수 있는 구조체

```
struct date
{
    int year;    //년
    int month;   //월
    int day;     //일
};
```

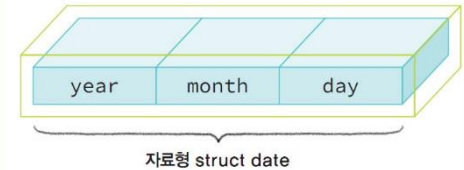


그림 13-12 자료형 struct date의 구조

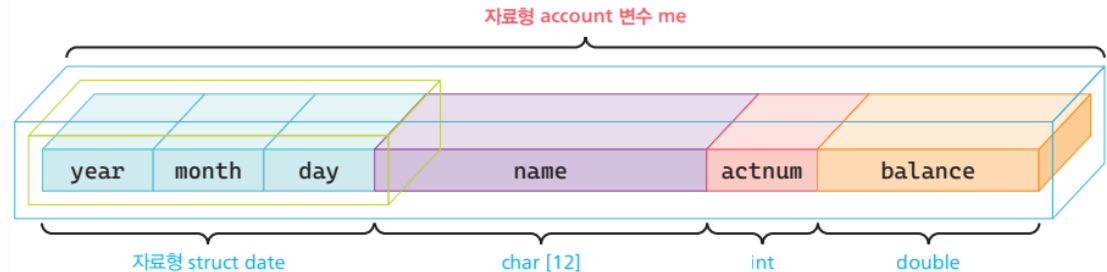
- 구조체 **struct account**

- 계좌 개설일자를 저장할 멤버로 open을 추가

- open의 자료형으로  
위에서 정의한  
**struct date**를 사용

- struct account 변수  
me의 메모리 구조

```
struct account
{
    struct date open;    //계좌 개설일자
    char name[12];       //계좌주 이름
    int actnum;          //계좌번호
    double balance;      //잔고
};
struct account me = {{2022, 3, 9}, "홍길동", 1001, 300000};
```



# 구조체 멤버로 다른 구조체 형 포함

- **account** 구조체를 사용한 프로그램
  - 멤버가 구조체 **date**인 초기화
    - {2012, 3, 9}
- 구조체 **account** 변수인 **me**로 년, 월, 일을 참조
  - 접근연산자를 2번 사용
    - **me.open.year**, **me.open.month**, **me.open.day**를 이용

```
Prj02      02nestedstruct.c      구조체 멤버로 다른 구조체 형 포함      난이도: ★

01  #include <stdio.h>
02  #include <string.h>
03
04  //날짜를 위한 구조체
05  struct date
06  {
07      int year;    //년
08      int month;  //월
09      int day;    //일
10  };
11
12  //은행계좌를 위한 구조체
13  struct account
14  {
15      struct date open;    //계좌 개설일자
16      char name[12];       //계좌주 이름
17      int actnum;          //계좌번호
18      double balance;      //잔고
19  };
20
21  int main(void)
22  {
23      struct account me = { { 2022, 3, 9 }, "홍길동", 1001, 300000 };
24
25      printf("구조체 크기: %zu\n", sizeof(me));
26      printf("[%d. %d. %d]\n", me.open.year, me.open.month, me.open.day);
27      printf("%s %d %.2f\n", me.name, me.actnum, me.balance);
28  }
```

구조체 멤버로 다른 구조체 변수를 포함

변수 open을 위한 0는 생략 가능

중첩된 구조체를 접근하려면 접근연산자를 2번 사용

구조체 크기: 40

산술적인 크기인 36보다 큼

[2022. 3. 9]

홍길동 1001 300000.00

# 구조체 정의 위치

- 구조체 정의는 그 정의 위치에 따라 구조체의 유효 범위가 결정
  - 구조체의 정의도 변수 선언처럼 유효범위는 전역(global) 또는 지역(local)
  - 전역
    - main() 함수 외부 상단에서 정의된 구조체
  - 지역
    - main() 함수 또는 다른 함수 내부에서 정의된 구조체



## TIP 구조체 정의의 위치

구조체 정의는 변수의 선언처럼 그 정의 위치에 따라 구조체 자료형의 유효 범위가 결정된다. 즉 구조체의 정의도 변수 선언처럼 유효범위는 전역(global) 또는 지역(local)으로 모두 가능하다. 다음과 같이 main() 함수 외부 상단에서 정의된 구조체 struct date는 전역으로 이 파일의 이 위치 이후 모든 함수에서 사용 가능하다. 그러나 main() 함수 내부에서 정의된 구조체 struct account는 지역으로 이 위치 이후 함수 main() 내부에서만 사용 가능하다.

```
struct date
{
    int year;    //년
    int month;   //월
    int day;     //일
};

int main(void)
{
    struct account
    {
        char name[12]; //계좌주 이름
        int actnum;    //계좌번호
        double balance; //잔고
    };

    //구조체 변수 선언 및 초기화
    struct account mine = { "홍길동", 1001, 300000 };

    return 0;
}
```

전역

지역

그림 13-14 구조체 정의의 유효 범위

# 구조체 변수의 대입과 동등비교

- 구조체 변수의 대입문이 가능

- 동일한 구조체형의 변수는 대입문이 가능
  - 변수 대입으로 한번에 모든 멤버의 대입이 가능

```
struct student
{
    int snum;           //학번
    char *dept;         //학과 이름
    char name[12];      //학생 이름
};
struct student hong = { 202200001, "컴퓨터정보공학과", "홍길동" };
struct student one;

one = hong;
```

- 구조체의 동등 비교

- struct student 형의 변수 hong과 one에서 (one == hong)
  - 동등 비교는 사용 불가능
- 만일 구조체를 비교하려면
  - 구조체 멤버, 하나 하나를 비교

```
if ( one == hong ) //오류
    printf("내용이 같은 구조체입니다.\n");
```

```
if (one.snum == hong.snum)
    printf("학번이 %d로 동일합니다.\n", one.snum);
```

```
if (one.snum == hong.snum && !strcmp(one.name, hong.name) && !strcmp(one.dept, hong.dept))
    printf("내용이 같은 구조체입니다.\n");
```

# 구조체 정의와 비교

Prj03

03structcmp.c

구조체 정의와 비교

난이도: ★★

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 #include <string.h>
04
05 int main(void)
06 {
07     //학생을 위한 구조체
08     struct student
09     {
10         int snum;           //학번
11         char* dept;         //학과 이름
12         char name[12];      //학생 이름
13     };
14     struct student hong = { 202200001, "컴퓨터정보공학과", "홍길동" };
15     struct student na = { 202200002 };
16     struct student you = { 202200003 };
17
18     //학생이름 입력
19     scanf("%s", na.name);
20     //na.name = "나한국"; //컴파일 오류 '식이 수정할 수 있는 lvalue여야 합니다.'
21     //scanf("%s", na.dept); //실행 오류
22
```

변수 dept는 char \*로 문자열 상수의 주소가 저장 가능하나, scanf()나 memcpy(), strcpy()로는 문자열 저장이 불가능

변수 name은 char 배열로 문자열 자체의 저장이 가능하므로 scanf()나 memcpy(), strcpy()의 사용도 가능

구조체 struct student 형인 na를 선언하면서 초기화, 첫 번째 멤버인 학번만 기술되었으므로 나머지는 각각 NULL과 NULL 문자로 초기화

```
23     na.dept = "컴퓨터정보공학과";
24     you.dept = "기계공학과";
25     memcpy(you.name, "홍길동", 7);
26     strcpy(you.name, "홍길동");
27     strcpy_s(you.name, 7, "홍길동");
28
29     printf("[%d, %s, %s]\n", hong.snum, hong.dept, hong.name);
30     printf("[%d, %s, %s]\n", na.snum, na.dept, na.name);
31     printf("[%d, %s, %s]\n\n", you.snum, you.dept, you.name);
32
33     struct student one;
34     one = you;
35     if (one.snum == you.snum)
36         printf("학번이 %d로 동일합니다.\n", one.snum);
37     //if ( one == bae ) //컴파일 오류
38     if (one.snum == you.snum && !strcmp(one.name, you.name) &&
39         !strcmp(one.dept, you.dept))
40         printf("내용이 같은 구조체입니다.\n");
41
42     return 0;
43 }
```

구조체 변수 one에 변수 you의 내용을 모든 비교하려면 각각의 멤버를 모두 비교

김현식

[202200001, 컴퓨터정보공학과, 홍길동]

[202200002, 컴퓨터정보공학과, 김현식]

[202200003, 기계공학과, 홍길동]

학번이 202200003로 동일합니다.

내용이 같은 구조체입니다.

# 문자열을 처리하기 위한 포인터 char \*와 배열 char []

- char 포인터

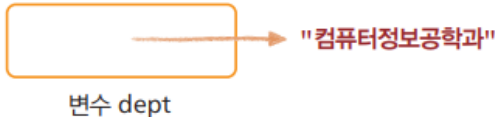
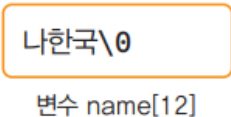
- 문자열의 첫 문자 주소를 저장하므로  
문자열 상수의 주소로 사용

```
char *dept;           //학과 이름
char name[12];        //학생 이름
```

- char 배열

- 문자열을 구성하는 모든 문자를 하나 하나  
저장하고 마지막에 '\0' 문자를 저장하여 사용

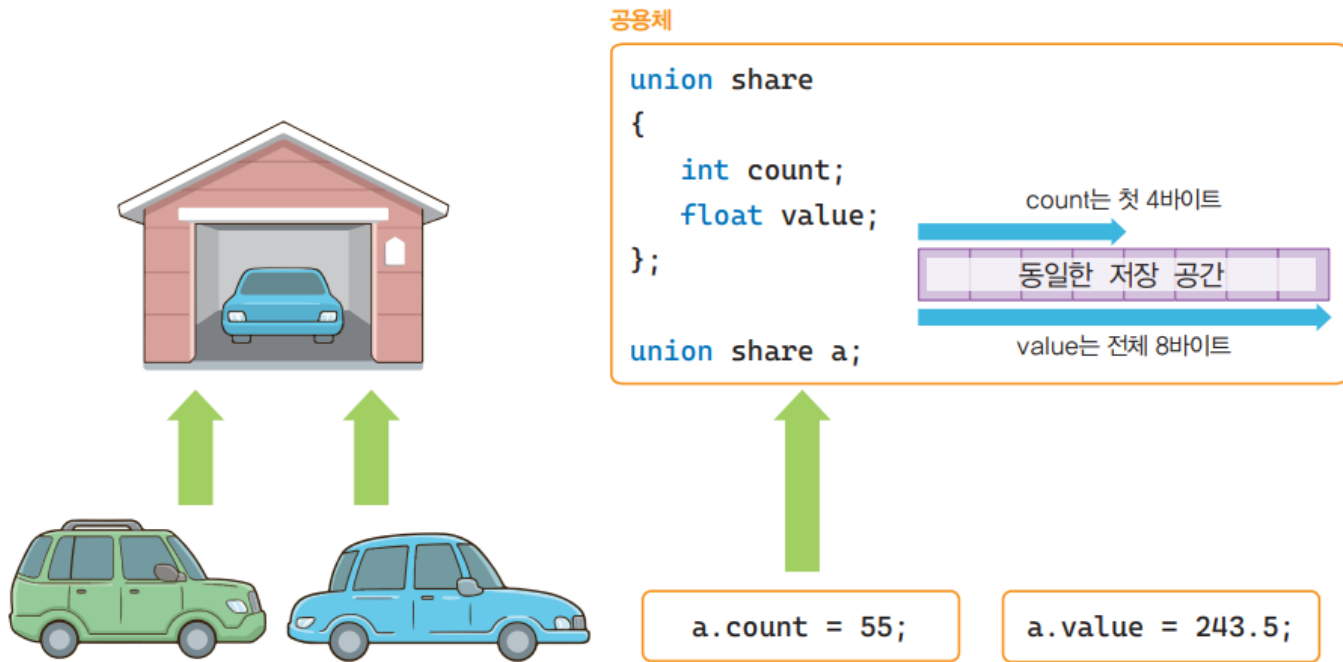
표 13-1 char 포인터와 char 배열의 비교

char 포인터	char 배열
<code>char *dept; //학과 이름</code>	<code>char name[12]; //학생 이름</code>
<code>char *dept = "컴퓨터정보공학과";</code> 	<code>char name[12] = "나한국";</code> 
변수 dept는 포인터로 단순히 문자열 상수를 다루는 경우 효과적	변수 name은 배열로 12바이트 공간을 가지며 문자열을 저장하고 수정 등이 필요한 경우 효과적
정상 사용 <code>dept = "컴퓨터정보공학과";</code>	정상 사용 <code>strcpy(name, "배상문");</code> <code>scanf("%s", name);</code>
오류 발생 <code>strcpy(dept, "컴퓨터정보공학과"); //오류</code> <code>scanf("%s", dept); //오류</code>	오류 발생 <code>name = "나한국"; //오류</code>
단지 문자열 상수의 첫 주소를 저장하므로 문자열 자체를 저장하거나 수정하는 것은 불가능하므로 다음 구문은 사용 불가능	문자열 자체를 저장하는 배열이므로 문자열의 저장 및 수정이 가능하고 문자열 자체를 저장하는 다음 구문 사용도 가능



# 공용체 개념

- 하나의 차고에 일반 세단과 SUV를 각각 주차한다고 생각
  - 공간이 하나이므로 어느 시간에 한 대의 주차는 가능
- 공용체
  - 동일한 저장 장소에 여러 자료형을 저장하는 방법
    - 이러한 겸용 주차장과 비슷한 개념
  - 멤버에 한번에 한 종류만 저장하고 참조 가능



# union을 사용한 공용체 정의 및 변수 선언

- 공용체(union)
  - 서로 다른 자료형의 값을 동일한 저장공간에 저장하는 자료형
- 공용체 선언 방법
  - union을 struct로 사용하는 것을 제외하면 구조체선언 방법과 동일

공용체 정의 및 변수 선언 구문

```
union 공용체태그이름
{
    자료형 멤버변수명1;
    자료형 멤버변수명2;
    ...
}
```

공용체 구성요소인 멤버(struct member)이다.

```
} [변수명1] [, 변수명2];
```

세미콜론은 반드시 필요하다.

```
union data
{
    char ch;        //문자형
    int cnt;         //정수형
    double real;     //실수형
} data1;
```

```
union udata
{
    char name[4];    //char형 배열
    int n;           //정수형
    double val;      //실수형
};
```

# 공용체 크기와 초기화

## • 공용체 변수의 크기

- union data의 변수 data1은 멤버 중 가장 큰 자료형의 크기로 정해짐
  - 가장 큰 크기인 **double** 형의 8바이트의 저장공간을 세 멤버가 함께 이용
- 동시에 여러 멤버의 값을 동시에 저장하여 이용할 수 없으며
  - **마지막에 저장된 단 하나의 멤버 자료값만을 저장**
- 구조체와 같이 typedef를 이용하여 새로운 자료형으로 정의 가능

## • 공용체의 초기화

- 처음 선언한 멤버의 초기값으로만 저장이 가능
- 만일 다른 멤버로 초기값을 지정하면
  - 컴파일 시 경고가 발생
  - 초기값으로 동일한 유형의 다른 변수의 대입도 가능

```
typedef union data uniondata;
```

```
uniondata data2 = {'A'};           //첫 멤버인 char형으로만 초기화 가능  
//uniondata data2 = {10.3};      //컴파일 시 경고 발생
```

warning C4244: '초기화중' : 'double'에서 'char'(으)로 변환하면서 데이터가 손실될 수 있습니다.

```
uniondata data3 = data2;           //다른 변수로 초기화 가능
```



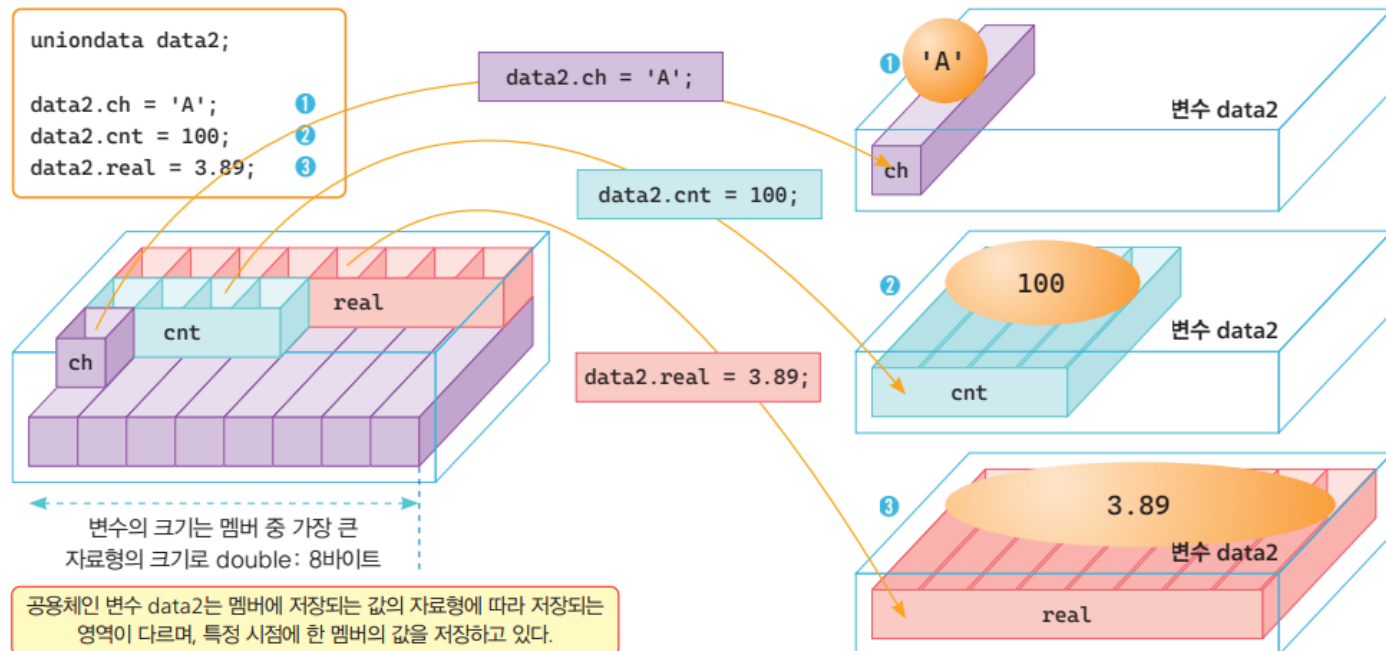
**TIP** 버전 C99 추가 가능

구조체처럼 공용체도 멤버의 순서와 관계없이 ".멤버이름 = 초기값"으로 지정된 멤버에 초기값을 저장(designated initializer)할 수 있다.

```
union data value = { .real = 3.98 };  
printf("%.2f\n", value.real);
```

# 공용체 멤버 접근

- 구조체와 같이 접근연산자 .를 사용: 문장 `data2.ch = 'A';`
  - 이 문장 이후에 멤버 `cnt`나 `real`의 출력은 가능하나 의미는 없음
  - 유형이 `char`인 `ch`를 접근하면 8바이트 중에서 첫 1바이트만 참조
    - `int`인 `cnt`를 접근하면 전체 공간의 첫 4바이트만 참조
    - `double`인 `real`을 접근하면 8바이트 공간을 모두 참조
  - 항상 마지막에 저장한 멤버로 접근해야 원하는 값을 얻을 수 있음
    - 공용체를 참조할 경우 정확한 멤버를 사용하는 것은 프로그래머의 책임



# 공용체 정의와 변수 선언 및 사용

실습예제 13-4

Prj04

04union.c

공용체 정의와 변수 선언 및 사용

난이도: ★

```
01 #include <stdio.h>
02
03 //유니온 구조체를 정의하면서 변수 data1도 선언한 문장
04 union data
05 {
06     char ch;        //문자형
07     int cnt;        //정수형
08     double real;    //실수형
09 } data1;            //data1은 전역변수
10
11 int main(void)
12 {
13     union data data2 = { 'A' }; //첫 멤버인 char형으로만 초기화 가능
14     union data data3 = { 97.78 }; //컴파일 시 경고 발생
15     union data data4 = data2;    //다른 변수로 초기화 가능
16     data4.real = 3.78;
17
18     printf("%zu %zu\n", sizeof(union data), sizeof(data3));
19     printf("%c %c %f\n", data2.ch, data3.ch, data4.real);
20
21     //멤버 ch에 저장
22     data1.ch = 'a';
23     printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
24     //멤버 cnt에 저장
25     data1.cnt = 100;
26     printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
27     //멤버 real에 저장
28     data1.real = 3.156759;
29     printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
30
31     return 0;
32 }
```

warning C4244: '초기화 중': 'double'에서 'char'(으)로 변환하면서 데이터가 손실될 수 있습니다.

초기화 값 97.78에서 정수인 97만 멤버 ch에 저장

공용체인 data1에서 자료형 int인 멤버 cnt에 정수 100을 저장, 이 이후로는 data1.cnt만 의미가 있음

결과

```
8 8
A a 3.780000
a 97 0.000000
d 100 0.000000
N -590162866 3.156759
```

# LAB 도시의 이름과 위치를 표현하는 구조체

- 도시의 이름과 위치를 표현하는 구조체 **struct city**
  - 멤버로 char \*와 struct position으로 구성
  - 멤버인 구조체 struct position
    - 위도(latitude)와 경도(longitude)를 표현하는 멤버
- 프로그램
  - 구조체 struct city의 변수를 선언
    - 서울과 뉴욕의 정보를 저장
    - 도시의 정보를 다시 출력
  - 구조체 struct position 변수
    - seoul과 newyork
- 결과
  - [서울] 위도= 37.3 경도= 126.6
  - [뉴욕] 위도= 40.8 경도= 73.9

lab1structcity.c

난이도 ★

```
01 #include <stdio.h>
02 #include <string.h>
03
04 //지구 위치 구조체
05 struct position
06 {
07     double latitude;    //위도
08     double longitude;   //경도
09 };
10
11 int main(void)
12 {
13     //도시 정보 구조체
14     struct city
15     {
16         char* name;      //이름
17         struct position place; //위치
18     };
19     struct city seoul, newyork;
20
21     struct position seoul_place;
22     seoul.place.latitude = 37.33;
23     seoul.place.longitude = 126.58;
24
25     newyork.name = "뉴욕";
26     newyork.place.latitude = 40.8;
27     struct position newyork_place;
28     newyork_place.longitude = 73.9;
29
30     printf("[%s] 위도= %.1f 경도= %.1f\n",
31            seoul.name, seoul.place.latitude, seoul.place.longitude);
32     printf("[%s] 위도= %.1f 경도= %.1f\n",
33            newyork.name, newyork.place.latitude, newyork.place.longitude);
34
35     return 0;
36 }
37
38 struct city seoul, newyork;
39 seoul.name = "서울";
40 newyork.place.longitude = 73.9;
```

# typedef 구문

- typedef 키워드

- 이미 사용되는 자료 유형을 다른 새로운 자료형 이름으로 재정의
- typedef int profit;
  - profit을 int와 같은 자료형으로 새롭게 정의하는 문장

자료형 재정의 typedef 구문

```
typedef 기존자료유형이름 새로운자료형1, 새로운자료형2, ... ;
```

```
typedef int profit;  
typedef unsigned int budget;  
typedef unsigned int size_t;  
typedef unsigned __int64 size_t;
```

여러 이름으로도 재정의할 수 있다.

새로운 이름 size\_t도 자료형 unsigned int와 동일하게 이용 가능하다.

- 프로그램의 시스템 간 호환성과 편의성을 위해 필요

- 터보 C++ 컴파일러에서 자료유형 int는 저장공간 크기가 2바이트
  - 비주얼 스튜디오에서는 4바이트
- 비주얼 스튜디오에서 작성한 프로그램은 터보 C++에서는 문제가 발생
  - 2 바이트로는 2000000을 저장할 수 없기 때문

Visual C++: 4바이트

```
int salary = 2000000;
```

Turbo C++: 2바이트

```
int salary = 2000000;
```

오버플로 발생(데이터 손실)

# 자료형 재정의

- 이 문제를 해결하는 방안

- Visual C++에서는 다음과 같이 int를 myint로 재정의
- 모든 int 형을 myint형으로 선언하여 이용
- 만일 이 소스를 터보 C++에서 컴파일 한다면
  - typedef 문장에서 int를 long으로 수정
  - 아무 문제 없이 다른 소스는 수정 없이 그대로 이용 가능

이 typedef 문장만 수정하면 터보 C++에서 이 소스를 그대로 이용 가능하다.

Visual C++ 소스

```
typedef int myint;  
...  
myint salary = 2000000;
```

Turbo C++ 소스

```
typedef long myint;  
...  
myint salary = 2000000;
```

- 자료형 int를 여러 개의 새로운 자료형 이름 integer와 word로 재정의

```
typedef int integer, word;  
  
integer myAge; //int myAge와 동일  
word yourAge; //int yourAge와 동일
```



# 자료형 재정의의 키워드 typedef 이용

- 문장 typedef에 의한 자료형 사용 범위를 제한
  - 함수 내부에서 재정의
    - 선언된 이후의 그 함수에서만 이용이 가능
  - 함수 외부에서 재정의
    - 재정의된 이후 그 파일에서 이용이 가능

Prj05      05typedef.c      자료형 재정의의 키워드 typedef 이용      난이도: ★

```
01  #include <stdio.h>
02
03  //함수 외부에서 정의된 자료형은 이후 파일에서 사용 가능
04  typedef unsigned int budget;
05
06  int main(void)
07  {
08      budget year = 24500000; //새로운 자료형 budget 사용
09
10      //함수 내부에서 정의된 자료형은 함수 내부에서만 사용 가능
11      typedef int profit;
12      profit month = 4600000; //새로운 자료형 profit 사용
13      printf("올 예산은 %d, 이달의 이익은 %d 입니다.\n", year, month);
14
15      return 0;
16  }
17
18  void test(void)
19  {
20      budget year = 24500000; //새로운 자료형 budget 사용
21
22      //profit은 이 함수에서는 사용 불가, 컴파일 오류 발생
23      //profit year;
24  }
```

budget은 int와 같은 자료형으로 변수 year를 budget으로 선언하면서 초기값 대입

함수 내부에서 정의된 자료형은 함수 내부에서만 사용 가능

# struct를 생략한 새로운 자료형

- 구조체 자료형은 struct date 처럼 항상 키워드 struct를 써야 하나?

- typedef 사용하여 구조체 struct date를 date로 재정의
  - 물론 date가 아닌 datatype 등 다른 이름으로도 재정의가 가능

```
struct date
{
    int year;    //년
    int month;   //월
    int day;     //일
};

typedef struct date date;
```

자료형인 date는 struct date와 함께 동일한 자료유형으로 이용이 가능하다.

- 구조체 정의 자체를 typedef와 함께 처리하는 방법

- typedef 구문에서 새로운 자료형으로 software 형 정의
  - 이 구문 이후에는 software를 구조체 자료형으로 변수 선언에 사용
- 구조체 태그이름은 생략 가능
- 구조체 software 형
  - 멤버로 구조체 date형 변수 release

```
typedef struct
{
    char title[30];    //제목
    char company[30];  //제작회사
    char kinds[30];    //종류
    date release;      //출시일
} software;
```

software는 변수가 아니라 새로운 자료형이다.

# 문장 typedef를 이용하여 구조체 자료형을 다른 이름으로 재정의해 사용

Prj06

06typedefstruct.c

문장 typedef를 이용하여 구조체 자료형을 다른 이름으로 재정의해 사용

난이도: ★

```
01 #include <stdio.h>
02
03 struct date
04 {
05     int year;    //년
06     int month;   //월
07     int day;     //일
08 };
09
10 //struct date 유형을 간단히 date 형으로 사용하기 위한 구문
11 typedef struct date date;
12
13 int main(void)
14 {
15     //구조체를 정의하면서 바로 자료형 software로 정의하기 위한 구문
16     typedef struct
17     {
18         char title[30];    //제목
19         char company[30];  //제작회사
20         char kinds[30];    //종류
21         date release;      //출시일
22     } software;
23     software vs = { "비주얼스튜디오 커뮤니티", "MS", "통합개발환경",
24                     { 2022, 8, 29 } };
25
26
27     printf("제품명: %s\n", vs.title);
28     printf("회사 : %s\n", vs.company);
29     printf("종류 : %s\n", vs.kinds);
```

```
30     printf("출시일: %d. %d. %d\n", vs.release.year,
31                                     vs.release.month, vs.release.day);
32
33     return 0;
34 }
```

제품명: 비주얼스튜디오 커뮤니티

회사 : MS

종류 : 통합개발환경

출시일: 2022. 8. 29

# LAB 영화 정보를 표현하는 구조체

- 구조체 struct movie
  - 멤버로 char \*
    - title은 영화 제목
  - long long
    - profit은 총수익
- 구조체 struct movie의 자료형을 다시 movie로 정의
  - 변수 parasite에 <기생충> 영화 정보를 저장한 후, 다시 출력

Lab 13-2

lab2structmovie.c

난이도: ★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     //영화 정보 구조체
06     typedef struct movie
07     {
08         char* title;        //영화제목
09         long long profit;   //총수익
10         
11
12          parasite;
13         parasite.title = "기생충";
14         parasite.profit = 310000000000; //전 세계에서 3,100억 수익
15
16         printf("[%s] 총수익: %lld\n", parasite.title, parasite.profit);
17
18         return 0;
19     }
```

정답

```
10     } movie;
12     movie parasite;
```

# 구조체 포인터 변수 선언

- 구조체 포인터는 구조체의 주소값을 저장하는 변수

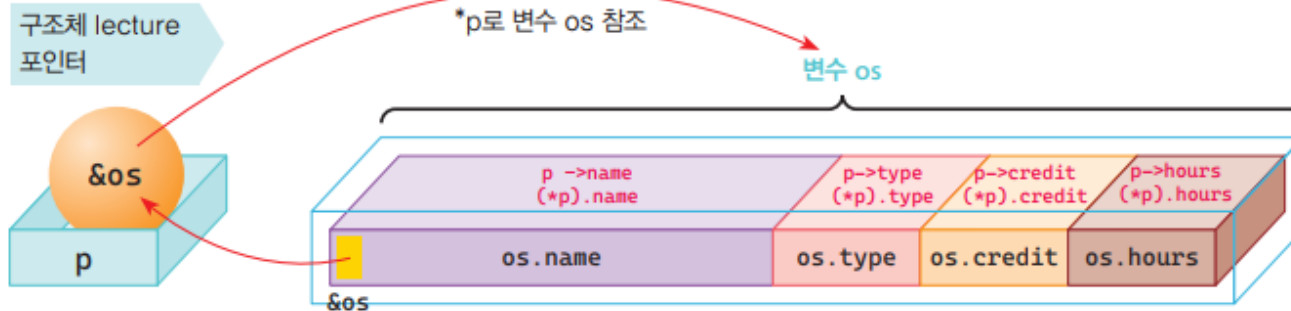
- 대학 강좌를 처리하는 구조체 자료형 lecture를 선언한 구문
- 구조체 포인터 변수 p는 lecture \*p로 선언

```
struct lecture
{
    char name[20];    //강좌명
    int type;         //강좌구분
    int credit;       //학점
    int hours;        //시수
};
typedef struct lecture lecture;
lecture *p;
```

- 변수 os를 선언한 후

- 문장 lecture \*p = &os ;
  - lecture 포인터 변수 p에 &os를 저장
  - 이로써 포인터 p로 구조체 변수 os 멤버 참조가 가능

```
lecture os = {"운영체제", 2, 3, 3};
lecture *p = &os;
```

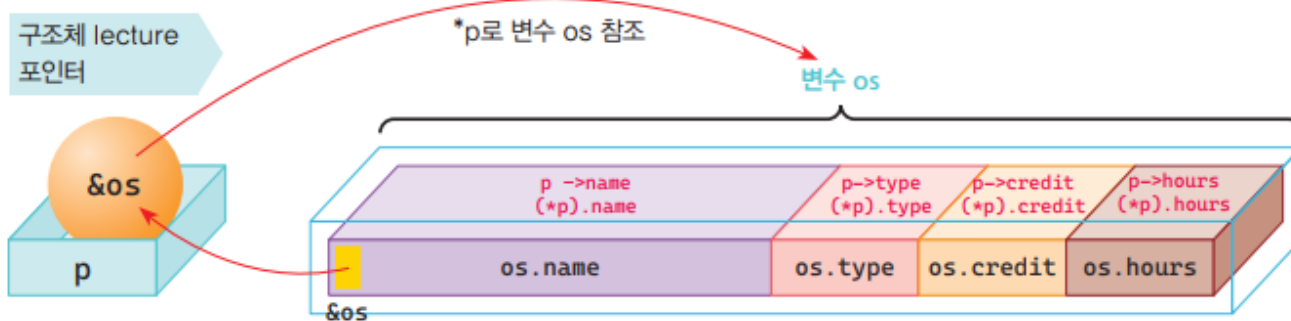


# 포인터 변수의 구조체 멤버 접근 연산자 ->

- **p->name**

- 포인터 p가 가리키는 구조체 변수의 멤버 name을 접근하는 연산식
- p->type, p->credit, p->hours
  - 각각 os.type, os.credit, os.hours를 참조
  - ->에서 -와 > 사이에 공백이 들어가는 것은 절대 안됨
- 연산식 (\*p).name으로도 사용 가능
  - (\*p).name은 \*p.name과는 다르다는 것에 주의
    - \*p.name은 \*(p.name)과 같은 연산식
    - p가 포인터이므로 p.name 는 문법오류가 발생

```
lecture os = {"운영체제", 2, 3, 3};  
lecture *p = &os;
```



# 구조체 변수와 구조체 포인터 변수를 이용한 멤버의 참조

- **멤버 접근연산자 ->, 구조체 멤버 접근연산자 .의 연산자 우선순위**
  - 다른 어떠한 연산자 우선순위보다 가장 높음
  - 연산자 ->와 .은 우선순위 1위
    - 결합성은 좌에서 우이며
  - 연산자 \*은 우선순위 2
    - 결합성은 우에서 좌

접근 연산식	구조체 변수 os와 구조체 포인터 변수 p인 경우의 의미
p->name	포인터 p가 가리키는 구조체의 멤버 name
(*p).name	
*p.name	*(p.name)이고 p가 포인터이므로 p.name 은 문법오류가 발생
*os.name	*(os.name)를 의미하며, 구조체 변수os의 멤버 포인터 name이 가리키는 변수로, 이 경우는 구조체 변수 os 멤버 강좌명의 첫 문자임, 다만 한글인 경우에는 실행 오류
*p->name	*(p->name)을 의미하며, 포인터 p이 가리키는 구조체의 멤버 name이 가리키는 변수로 이 경우는 구조체 포인터 p이 가리키는 구조체의 멤버 강좌명의 첫 문자임, 마찬가지로 한글인 경우에는 실행 오류

# 구조체 포인터의 선언과 사용

Prj07 07structptr.c 구조체 포인터의 선언과 사용

난이도: ★★

```
01 #include <stdio.h>
02
03 struct lecture
04 {
05     char name[20]; //강좌명
06     int type;      //강좌구분 0: 교양, 1: 일반선택, 2: 전공필수, 3: 전공선택
07     int credit;    //학점
08     int hours;     //시수
09 };
10 typedef struct lecture lecture;
11
```

```
12 //제목을 위한 문자열
13 char* head[] = { "강좌명", "강좌구분", "학점", "시수" };
14 //강좌 종류를 위한 문자열
15 char* lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
16
17 int main(void)
18 {
19     lecture os = { "운영체제", 2, 3, 3 };
20     lecture c = { "C프로그래밍", 3, 3, 4 };
21     lecture* p = &os;
22
23     printf("구조체 크기: %zu, 포인터 크기: %zu\n", sizeof(os), sizeof(p));
24     printf("%10s %12s %6s %6s\n", head[0], head[1], head[2], head[3]);
25     printf("%12s %10s %5d %5d\n", p->name, lectype[p->type],
26                                             p->credit, p->hours);
27
28     //포인터 변경
29     p = &c;
30     printf("%12s %10s %5d %5d\n", (*p).name, lectype[(*p).type],
31                                             (*p).credit, (*p).hours);
32     printf("%12c %10s %5d %5d\n", *c.name, lectype[c.type],
33                                             c.credit, c.hours);
34
35     return 0;
36 }
```

강좌구분이 2인 "전공필수"로 지정

문자열 "C프로그래밍"에서 첫 글자인 C만 참조하는 연산식

구조체 크기: 32, 포인터 크기: 8

강좌명	강좌구분	학점	시수
운영체제	전공필수	3	3
C프로그래밍	전공선택	3	4
C	전공선택	3	4



# 공용체 정의와 변수 선언 및 사용

- 포인터 변수로 멤버 접근
  - 접근연산자 ->를 이용
- 공용체 포인터 p를 선언
  - p->ch = 'a';
    - p가 가리키는 공용체 멤버 ch에 'a'를 저장
  - p->cnt, p->real
  - 각각 value.cnt, value.real을 참조

```
union data
{
    char ch;
    int cnt;
    double real;
} value, *p;
```

변수 value는 union data형이며 p는 union data 포인터형으로 선언

```
p = &value; //포인터 p에 value의 주소값을 저장
p->ch = 'a'; //value.ch = 'a';와 동일한 문장
```

Prj08 08 unionptr.c 공용체 정의와 변수 선언 및 사용 난이도: ★★

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     //공용체 union data 정의
06     union data
07     {
08         char ch;
09         int cnt;
10         double real;
11     };
12
13     //유니온 union data를 다시 자료형 udata로 정의
14     typedef union data udata;
15
16     //udata 형으로 value와 포인터 p 선언
17     udata value, *p;
18
19     p = &value;
20     p->ch = 'a';
21     printf("%c %c\n", p->ch, (*p).ch);
22     p->cnt = 100;
23     printf("%d ", p->cnt);
24     p->real = 3.14;
25
26
27     return 0;
28 }
```

char, int, double 자료형 중 하나를 동시에 저장할 수 있는 8바이트 공간의 공용체 union data를 정의하기 위한 문장 시작으로 6행에서 11행까지 정의, 이 공용체 정의로 이 위치 이후 모든 파일에서 공용체 union data 사용 가능

변수 value와 p는 모두 함수 main() 내부에서만 사용이 가능한 지역변수

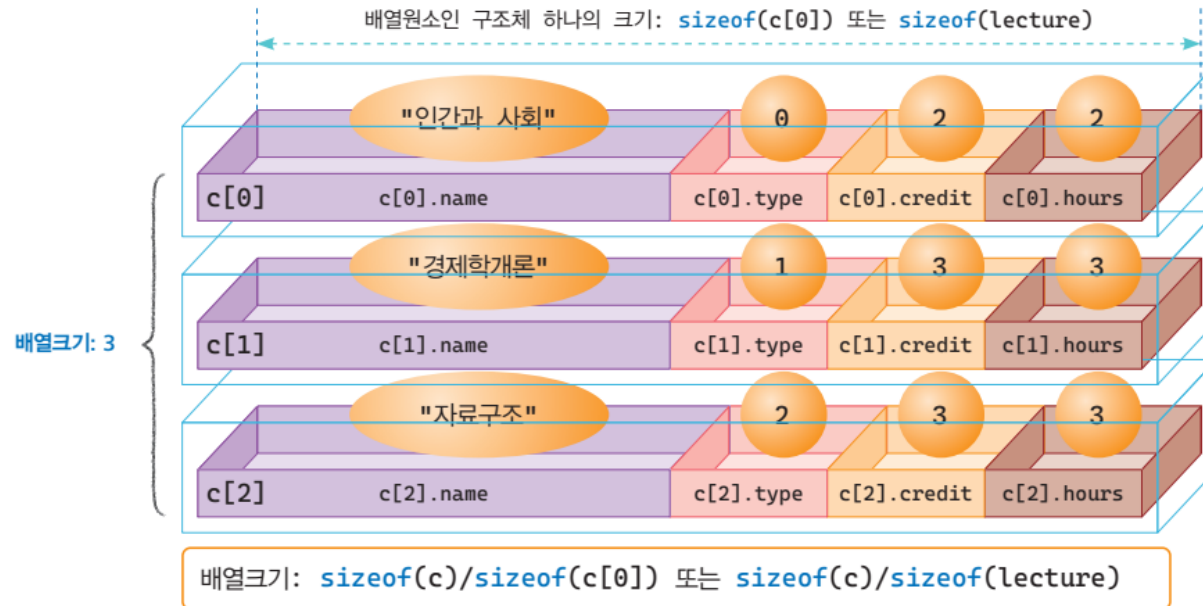
연산식 p->ch와 (\*p).ch는 모두 value.ch를 참조

a a  
100 3.14

# 구조체 배열 변수 선언

- 구조체 `lecture`의 배열크기 3인 `c`를 선언하고 초기값을 저장하는 구문
  - 구조체 배열의 초기값 지정 구문에서는 중괄호가 중첩되게 표시
    - 외부 중괄호
      - 배열 초기화의 중괄호
    - 내부 중괄호
      - 배열원소인 구조체 초기화를 위한 중괄호

```
lecture c[] = { {"인간과 사회", 0, 2, 2},  
               {"경제학개론", 1, 3, 3},  
               {"자료구조", 2, 3, 3}};
```



# 구조체 배열을 선언한 후 내용 출력

Prj09      09structary.c      구조체 배열을 선언한 후 내용 출력

```
01 #include <stdio.h>
02
03 struct lecture
04 {
05     char name[20];    //강좌명
06     int type;         //강좌구분
```

```
07     int credit;    //학점
08     int hours;    //시수
09 };
10 typedef struct lecture lecture;
11
12 char* lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
13 char* head[] = { "강좌명", "강좌구분", "학점", "시수" };
14
15 int main(void)    구조체 struct의 배열을 선언하면서 바로 초기값을 대입,
                  배열 크기는 지정하지 않고 초기값을 지정한 원소 수가 5
16 {
17     //구조체 lecture의 배열 선언 및 초기화
18     lecture course[] = { { "인간과 사회", 0, 2, 2 },
19                           { "경제학개론", 1, 3, 3 },
20                           { "자료구조", 2, 3, 3 },
21                           { "모바일프로그래밍", 2, 3, 4 },
22                           { "고급 C프로그래밍", 3, 3, 4 } };
23
24     int arysize = sizeof(course) / sizeof(course[0]);
25
26     printf("배열크기: %d\n\n", arysize);
27     printf("%12s %12s %6s %6s\n", head[0], head[1], head[2], head[3]);
28     printf("=====\n");
29
30     for (int i = 0; i < arysize; i++)
31         printf("%16s %10s %5d %5d\n", course[i].name,
32             lectype[course[i].type], course[i].credit, course[i].hours);
33
34     return 0;
35 }
```

배열크기:

강좌명	강좌구분	학점	시수
인간과 사회	교양	2	2
경제학개론	일반선택	3	3
자료구조	전공필수	3	3
모바일프로그래밍	전공필수	3	4
고급 C프로그래밍	전공선택	3	4

# LAB 영화 정보를 표현하는 구조체의 배열

- 구조체 **struct movie**
  - char \*title
    - 영화의 제목
  - char director[20]
    - 감독
  - int attendance
    - 관객수
- 구조체 **movie**의 배열 선언
  - 초기화로 영화 세 편의 정보를 저장
    - 세 번째 영화의 감독을 “김용화”로 저장
    - 모든 영화의 정보를 다시 출력

lab1boxoffice.c

난이도: ★★

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 #include <string.h>
04
05 int main(void)
06 {
07     //영화 정보 구조체
08     typedef struct movie
09     {
10         char* title;        //영화제목
11         int attendance;     //관객수
12         char director[20];  //감독
13     } movie;
14
15     movie box[] = { { "명량", 17613000, "김한민" },
16                     { "극한직업", 16261000, "윤세균" },
17                     { "신과함께-죄와벌", 14419000 } };
18
19     //세 번째 영화의 감독을 김용화로 저장
20     strcpy(box[2].director, "김용화");
21
22     printf("      제목      감독      관객수\n");
23     printf("=====");
24     for (int i = 0; i < 3; i++)
25         printf("[%15s] %6s %d\n",
26                box[i].title, box[i].director, box[i].attendance);
27
28     return 0;
29 }
```

감사합니다.