프로그래밍 기초

13장. 구조체와 공용체

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**

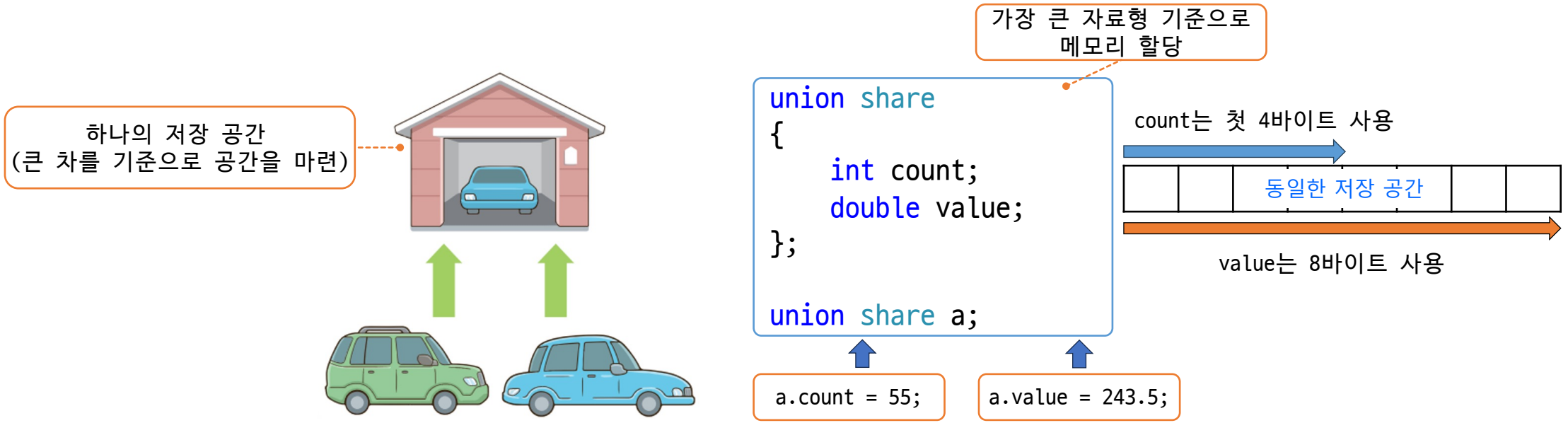
- **자료형 재정의를 위한 typedef를 사용할 수 있다.**
  - 키워드 typedef를 사용한 자료형 재정의 방법과 필요성
  - 구조체 정의를 새로운 자료형으로 재정의

- **구조체 포인터와 배열을 활용할 수 있다.**
  - 구조체의 주소를 저장하는 포인터의 선언과 활용
  - 구조체 포인터의 접근연산자 ->의 사용 방법
  - 구조체 배열의 선언과 활용 방법

공용체(union) 개념

- **하나의 차고에 일반 세단과 SUV를 각각 주차한다고 생각**
  - 공간이 하나이므로 어느 시간에 한 대의 주차는 가능

- **공용체**
  - 하나의 저장 공간에 여러 자료형을 저장하는 방법
  - 공용체 멤버에 한번에 한 종류만 저장하고 참조 가능



union을 사용한 공용체 정의 및 변수 선언

- **공용체(union)**
  - 서로 다른 자료형의 값을 하나의(동일한) 저장공간에 저장하는 자료형

- **공용체 선언 방법**
  - struct 대신에 union사용하는 것을 제외하면 구조체 선언 방법과 동일

```
union 공용체태그이름
{
    자료형 변수명1;
    자료형 변수명2;
    . . .
};
```

```
union data
{
    char ch;
    int cnt;
    double real;
} data1;
```

함수외부에 선언된 경우, data1은 전역 변수

```
union udata
{
    char name[4];
    int n;
    double val;
};
```

공용체 크기와 초기화

- **공용체 변수의 크기**
  - union data의 변수는 멤버 중 가장 큰 자료형의 크기로 정해짐
    - ➢ 가장 큰 크기인 double형의 8바이트의 저장공간을 세 멤버가 함께 이용
  - 여러 멤버의 값을 동시에 저장하여 이용할 수 없음
    - ➢ 마지막에 저장된 단 하나의 멤버 자료값만을 저장
  - 구조체와 같이 typedef를 이용하여 새로운 자료형으로 정의 가능

- **공용체의 초기화**
  - 중괄호를 이용한 초기화: 공용체의 첫 번째 멤버만 초기화
  - 공용체이름.멤버변수 = 값; 형태로 초기화
  - 초기값으로 동일한 유형의 다른 변수의 대입도 가능

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

```c
union data data1;
data1.cnt = 10;

typedef union data uniondata;
uniondata data2 = {'A'}; // 첫 번째 멤버인 char ch가 초기화
uniondata data3 = data2; // 다른 변수로 초기화 가능
```

C99 버전 추가 기능

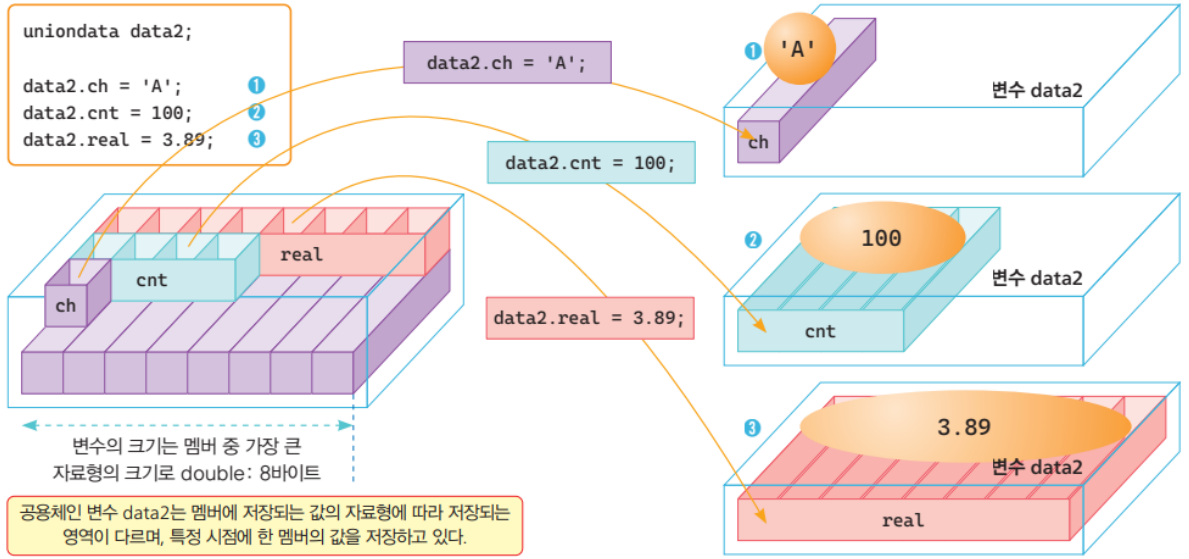
```c
union data value = {.real = 3.98};
printf("%.2f\n", value.real);

value.real = 4.0;
```

구조체 형식의 초기화

공용체 멤버 접근

- **구조체와 같이 접근연산자(.)를 사용:**
  - 문장 data2.ch = 'A';
    - 이 문장 이후에 멤버 cnt나 real의 출력은 가능하나 의미는 없음
  - 유형이 char인 ch를 접근하면 8바이트 중에서 첫 1바이트만 참조
    - int인 cnt를 접근: 전체 공간의 첫 4바이트만 참조
    - double인 real을 접근: 8바이트 공간을 모두 참조
  - 항상 마지막에 저장한 멤버로 접근해야 원하는 값을 얻을 수 있음
    - 공용체를 참조할 경우 정확한 멤버를 사용하는 것은 프로그래머의 책임

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



공용체 초기화 예제 (실습)

<union_init.c>

```c
#include <stdio.h>
#include <string.h>

union udata
{
    int i;
    float f;
    char str[20];
};
```

```c
int main()
{
    // 첫 번째 멤버 int를 초기화
    union udata data1 = {10};

    data1.i = 20;
    printf("data1.i = %d\n", data1.i);

    printf("sizeof(udata): %zd, sizeof(data1): %zd\n",
            sizeof(union udata), sizeof(data1));

    // 명시적 초기화 사용: float 멤버 초기화
    union udata data2 = {.f = 3.14f}; // C99 버전
    //union udata data2.f = 3.14f; // 오류 발생
    printf("data2.f = %f\n", data2.f);

    // 명시적 초기화는 문자 배열에도 가능
    union udata data3;
    strcpy(data3.str, "Hello Union");
    printf("data3.str = %s\n", data3.str);

    return 0;
}
```

가장 일반적인 초기화 방법

```
data1.i = 20
sizeof(udata): 20, sizeof(data1): 20
data2.f = 3.140000
data3.str = Hello Union
```

구조체 및 공용체 예제 #1 (실습)

<04union2.c>

```c
#include <stdio.h>
#include <string.h>

#define STU_NUMBER 1
#define REG_NUMBER 2

struct student
{
    int type;
    union
    {
        int stu_number; // 학번
        char reg_number[15]; // 주민등록번호
    } id;
    char name[20];
};
```

type변수: 공용체가 어떤 변수를 사용하는지 저장

stu_number 또는 reg_number[15] 둘 중에 하나만 사용

```c
void print(struct student s)
{
    switch (s.type)
    {
    case STU_NUMBER:
        printf("학번: %d\n", s.id.stu_number);
        printf("이름: %s\n", s.name);
        break;
    case REG_NUMBER:
        printf("주민등록번호: %s\n", s.id.reg_number);
        printf("이름: %s\n", s.name);
        break;
    default:
        printf("타입오류\n");
        break;
    }
}
```

구조체 및 공용체 예제 #2 (실습)

<04union2.c>

```c
int main(void)
{
    struct student s1, s2;

    s1.type = STU_NUMBER;
    s1.id.stu_number = 20190001;
    strcpy(s1.name, "홍길동");

    s2.type = REG_NUMBER;
    strcpy(s2.id.reg_number, "860101-1058031");
    strcpy(s2.name, "김철수");

    print(s1);
    print(s2);

    return 0;
}
```

union의 stu_number 사용

union의 reg_number 사용

```
학번: 20190001
이름: 홍길동
주민등록번호: 860101-1058031
이름: 김철수
```

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

Lab. 도시의 이름과 위치를 표현하는 구조체 예제

<lab1structcity.c>

```c
#include <stdio.h>
#include <string.h>


//지구 위치 구조체
struct position
{
    double latitude; //위도
    double longitude; //경도
};


//도시 정보 구조체
struct city
{
    char* name; //이름
    struct position place; // 지구 위치 구조체 포함
};
```

```c
int main(void)
{
    struct city seoul, newyork;

    seoul.name = "서울";
    seoul.place.latitude = 37.33;
    seoul.place.longitude = 126.58;

    newyork.name = "뉴욕";
    newyork.place.latitude = 40.8;
    newyork.place.longitude = 73.9;

    printf("[%s] 위도= %.1f 경도= %.1f\n",
                seoul.name, seoul.place.latitude,
                seoul.place.longitude);

    printf("[%s] 위도= %.1f 경도= %.1f\n",
                newyork.name, newyork.place.latitude,
                newyork.place.longitude);

    return 0;
}
```

```
[서울] 위도= 37.3 경도= 126.6
[뉴욕] 위도= 40.8 경도= 73.9
```

자료형 재정의 typedef 구문

- **typedef 키워드**
  - 이미 사용되는 자료 유형을 다른 새로운 자료형 이름으로 재정의
  - typedef 기존자료형 새로운자료형;

```
typedef unsigned char u8;
typedef unsigned short u16;
typedef unsigned int u32;
```

- **자료형 재정의 typedef 구문**
  - typedef int profit;
    - ➢ profit을 int와 같은 자료형으로 새롭게 정의하는 문장

```
typedef 기존자료유형이름 새로운자료형1, 새로운자료형2, …;
```

```
typedef int profit;
typedef unsigned int budget;
typedef unsigned int size_t;
typedef unsigned int __int64, uint64;
```

자료형을 재정의하는 이유

- **프로그램의 시스템 간 호환성과 편의성을 위해 필요**
  - 터보 C++ 컴파일러에서 int 타입의 저장공간 크기는 2바이트
    - 비주얼 스튜디오에서는 4바이트
  - 비주얼 스튜디오에서 작성한 프로그램은 터보 C++에서는 문제가 발생
    - 2 바이트로는 2000000을 저장할 수 없기 때문

GCC, Visual C++: 4 바이트

```
int salary = 2000000;
```

Turbo C++: 2 바이트

```
int salary = 2000000;
```

Overflow 발생(데이터 손실)

자료형 재정의: 호환성 문제 해결

- **자료형 재정의를 통해 호환성 문제 해결**
  - GCC 나 Visual C++에서 int를 myint로 재정의 후 모든 int 형을 myint형으로 선언하여 이용
  - 해당 소스 코드를 터보 C++에서 컴파일하는 경우,
    - ➢ typedef 문장에서 int를 long으로 수정: 다른 소스는 수정 없이 그대로 이용 가능

GCC, 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 재정의 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

<05typedef.c>

```c
#include <stdio.h>

//함수 외부에서 정의된 자료형은 이후 파일에서 사용 가능
typedef unsigned int budget;

int main(void)
{
    budget year = 24500000; //새로운 자료형 budget 사용

    //함수 내부에서 정의된 자료형은 이후 함수내부에서만 사용 가능
    typedef int profit;
    profit month = 4600000; //새로운 자료형 profit 사용
    printf("올 예산은 %d, 이달의 이익은 %d 입니다.\n",
            year, month);
    return 0;
}

void test(void)
{
    budget year = 24500000; //새로운 자료형 budget 사용

    //profit은 이 함수에서는 사용 불가, 컴파일 오류 발생
    //profit year;
}
```

구조체 자료형 재정의 : struct를 생략한 새로운 자료형

- **구조체 자료형은 struct date처럼 항상 키워드 struct를 써야 하나?**
  - struct date data1; 형태
  - typedef 사용하여 구조체 struct date를 datetype으로 재정의
    - ➢ datetype 등 다른 이름으로 재정의가 가능

```
struct date
{
    int year;
    int month;
    int day;
};
typedef struct date datetype;
```

새로운 자료형 datetype 정의

- **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 타입
    - ➢ 멤버로 char 배열 및 구조체 date형 변수 release를 포함

태그이름 생략 가능

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

software: 새로운 자료형 (구조체 별칭 이름)

구조체 자료형 재정의 예제: typedef struct 사용 (실습)

<06tdefstruct.c>

```c
#include <stdio.h>

struct date
{
    int year; // 년
    int month; // 월
    int day; // 일
};
// struct date 유형을 date 형으로 사용
typedef struct date date;


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

태그이름(software)은 생략 가능

일반적으로 태그이름(software)과 별칭이름(software)을 동일하게 사용

```c
int main(void)
{
    software vs = {"Visual Studio", "MS", "IDE", 2022, 8, 29};
    //software vs = {"Visual Studio", "MS", "IDE", {2022, 8, 29}};

    printf("제품명: %s\n", vs.title);
    printf("회사 : %s\n", vs.company);
    printf("종류 : %s\n", vs.kinds);
    printf("출시일: %d. %d. %d\n",
                vs.release.year, vs.release.month, vs.release.day);

    return 0;
}
```

구조체 타입 software를 자료형처럼 사용

date 구조체 값 초기화

실행 결과

```
제품명: Visual Studio
회사 : MS
종류 : IDE
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 영화 정보를 저장한 후, 다시 출력

<lab2structmovie.c>

```c
#include <stdio.h>

int main(void)
{
    //영화 정보 구조체
    typedef struct movie
    {
        char* title; //영화제목
        long long profit; //흥행수익
    } movie;

    movie parasite;
    parasite.title = "기생충";
    parasite.profit = 310000000000; //전 세계에서 3,100억 수익

    printf("[%s] 총수익: %lld\n",
                parasite.title, parasite.profit);

    return 0;
}
```

태그 이름(movie) 생략 가능

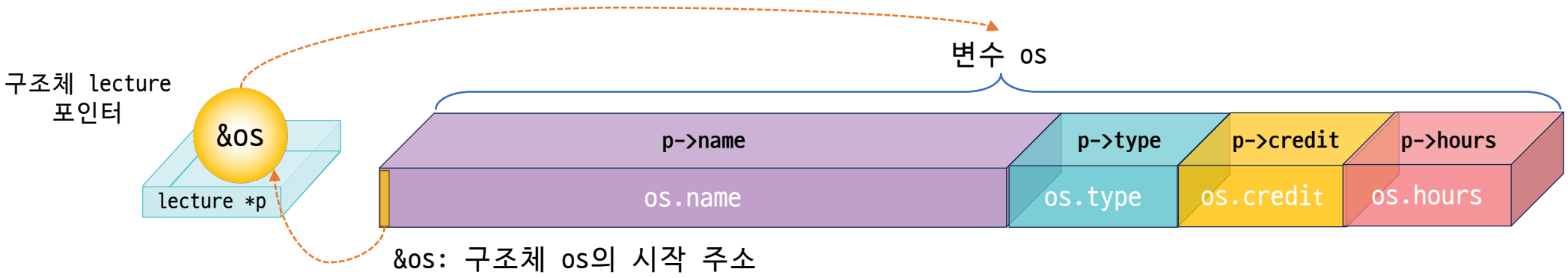
구조체 포인터 변수 선언

- **구조체 포인터**는 **구조체의 주소를 저장하는 변수**
  - 대학 강좌를 처리하는 구조체 자료형 lecture를 선언
  - 구조체 포인터 변수 p는 lecture *p로 선언
    - 일반 자료형 포인터처럼 사용

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

- **변수 os를 선언한 후**
  - 문장 lecture *p = &os ;
    - lecture 포인터 변수 p에 lecture os의 주소(&os)를 저장
    - 포인터 p로 구조체 변수 os 멤버 참조가 가능

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



포인터 변수의 구조체 멤버 접근 연산자 ->

- **구조체 포인터 변수(lecture *p)의 멤버 접근 방법: p->name**
  - 포인터 p가 가리키는 구조체 변수의 멤버 name에 접근하는 연산식(->)
  - p->type, p->credit, p->hours은 각각 os.type, os.credit, os.hours를 참조
    - ->에서 –와 > 사이에 공백이 들어가서는 절대 안됨
  - 연산식 p->name은 (*p).name으로도 사용 가능
    - (*p).name은 *p.name과는 다름
      - ***p.name은 *(p.name)과 같은 연산식**

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

printf("%12s %10s %5d %5d\n", os.name, lectype[os.type], os.credit, os.hours);
printf("%12s %d %5d %5d\n", p->name, p->type, p->credit, p->hours);
```

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 (실습)

<07structptr.c>

```c
#include <stdio.h>

typedef struct
{
    char name[20]; // 강좌명
     // 0: 교양, 1: 일반선택
     // 2: 전공필수, 3: 전공선택
    int type;      // 강좌구분(0, 1, 2, 3)
    int credit;    // 학점
    int hours;     // 시수
}lecture;

// 제목을 위한 문자열
char *head[] = {"강좌명", "강좌구분", "학점", "시수"};

// 강좌 종류를 위한 문자열
char *lectype[] = {"교양", "일반선택",
                   "전공필수", "전공선택"};
```

```c
int main(void)
{
    lecture os = {"운영체제", 2, 3, 3};
    lecture c = {"C프로그래밍", 3, 3, 4};
    lecture *p = &os;

    printf("구조체 크기: %zu, 포인터 크기: %zu\n\n",
                sizeof(os), sizeof(p));
    printf("%10s %12s %8s %8s\n", head[0], head[1], head[2], head[3]);
    printf("%12s %10s %5d %5d\n", p->name, lectype[p->type],
                                  p->credit, p->hours);
    printf("\n");

    // 포인터 변경
    p = &c;
    printf("%12s %10s %5d %5d\n", (*p).name, lectype[(*p).type],
                                  (*p).credit, (*p).hours);
    printf("%11c %10s %5d %5d\n", *c.name, lectype[c.type],
                                  c.credit, c.hours);

    return 0;
}
```

구조체 포인터 접근 권장 방법

첫 글자인 'C'만 참조

실행 결과

```
구조체 크기: 32, 포인터 크기: 8

 강좌명 강좌구분   학점   시수
운영체제 전공필수     3     3

C프로그래밍 전공선택     3     4
          C 전공선택     3     4
```

공용체 포인터

- **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 ;

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

변수 value는 union data형이며 p는 union data 포인터형으로 선언

```
a a
100 3.14
```

<08unionptr.c>

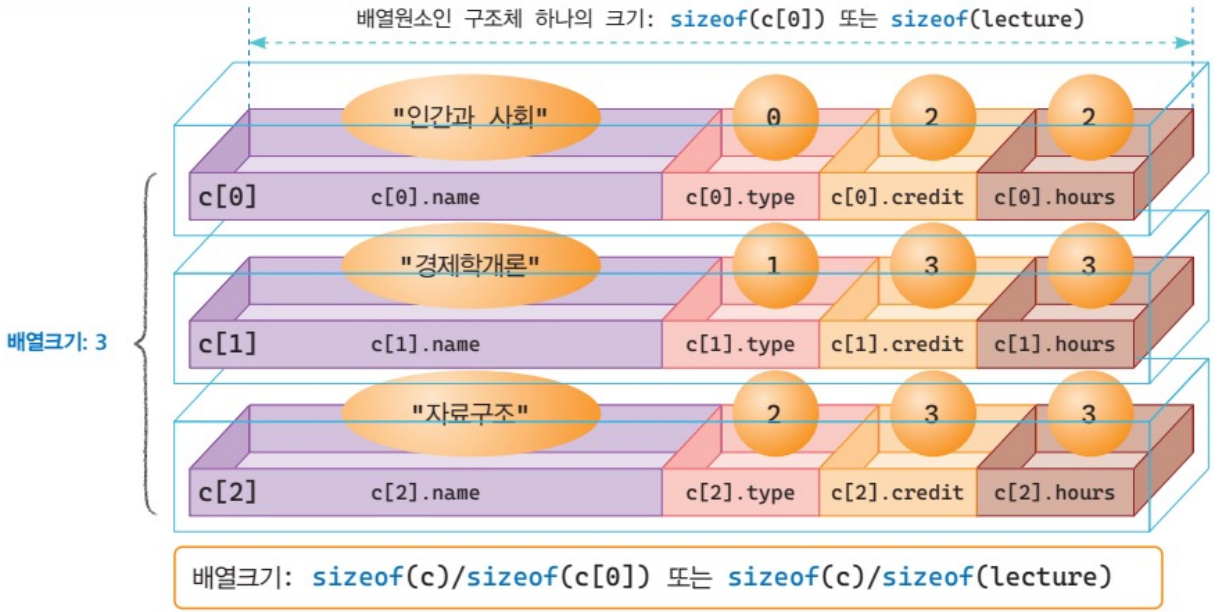
```c
#include <stdio.h>

int main(void)
{
    //유니온 union data 정의
    union data
    {
        char ch;
        int cnt;
        double real;
    };
    //유니온 union data를 다시 자료형 udata로 정의
    typedef union data udata;

    //udata 형으로 value와 포인터 p 선언
    udata value, *p;
    p = &value;

    p->ch = 'a';
    printf("%c %c\n", p->ch, (*p).ch);
    p->cnt = 100;
    printf("%d ", p->cnt);
    p->real = 3.14;
    printf("%.2f \n", p->real);

    return 0;
}
```

공용체 포인터(*p)에 udata value의 메모리 주소 전달

구조체 배열 변수 선언

- **구조체 lecture의 배열 크기 3인 c를 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 배열

- **구조체 배열 활용**
  - 구조체 배열 (lecture course[])의 이름(course)은 구조체 배열 포인터 변수에 대입이 가능
  - 문장 lecture *p = course;
    - p[i]로 배열 원소 접근 가능

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

```
lecture *p = course;
int arysize = sizeof(course) / sizeof(course[0]);

for (int i = 0; i < arysize; i++)
{
    printf("%20s %12s %5d %5d\n",
           p[i].name, lectype[p[i].type], p[i].credit, p[i].hours);
}
```

포인터 변수 p를 사용해서 배열로 접근 가능

구조체 배열을 선언한 후 내용 출력 (실습)

<09structary.c>

```c
#include <stdio.h>

struct lecture
{
    char name[24]; // 강좌명
    int type; // 강좌구분
    int credit; // 학점
    int hours; // 시수
};
typedef struct lecture lecture;

char *lectype[] = {"교양", "일반선택", "전공필수", "전공선택"};
char *head[] = {"강좌명", "강좌구분", "학점", "시수"};
```

```c
int main(void)
{
    // 구조체 lecture의 배열 선언 및 초기화
    lecture course[] = {{"인간과 사회", 0, 2, 2},
                        {"경제학개론", 1, 3, 3},
                        {"자료구조", 2, 3, 3},
                        {"모바일프로그래밍", 2, 3, 4},
                        {"고급 C프로그래밍", 3, 3, 4}};

    int arysize = sizeof(course) / sizeof(course[0]);

    printf("배열크기: %d\n\n", arysize);
    printf("%18s %12s %8s %8s\n", head[0], head[1], head[2], head[3]);
    printf("==============================================\n");

    for (int i = 0; i < arysize; i++)
        printf("%20s %14s %5d %5d\n", course[i].name,
                                      lectype[course[i].type],
                                      course[i].credit,
                                      course[i].hours);

     return 0;
}
```

```
배열크기: 5

      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 movie의 배열을 선언**
  - 초기화로 영화 세 편의 정보를 저장
    - 세 번째 영화의 감독을 "김용화"로 저장
    - 모든 영화의 정보를 다시 출력

lab1boxoffice.c 난이도: ★★

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main(void)
{
   //영화 정보 구조체
   typedef struct movie
   {
      char* title;         //영화제목
      int attendance;      //관객수
      char director[20];   //감독
   } movie;

   [          ] = { { "명량", 17613000, "김한민" },
                    { "극한직업", 16261000, "윤제균" },
                    { "신과함께-죄와벌", 14419000 } };

   //세 번째 영화의 감독을 김용화로 저장
   strcpy([                    ]);

   printf("       제목        감독        관객수\n");
   printf("=================================\n");
   for (int i = 0; i < 3; i++)
      printf("[%15s] %6s %d\n",
         [                                  ]

   return 0;
}
```

```c
   movie box[] = {{ "명량", 17613000, "김한민" },
   strcpy(box[2].director, "김용화");
         box[i].title, box[i].director, box[i].attendance);
```

Questions?