제13장

구조체와 공용체

Perfect C

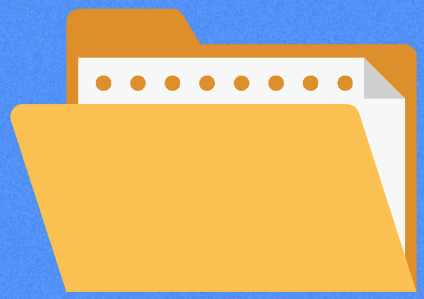
C언어로 배우는 프로그래밍 기초

단원 목표

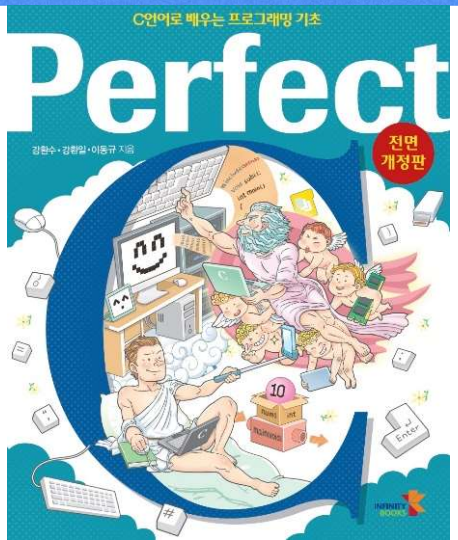
학습목표

- **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 활용방법

학습목차



01. 구조체와 공용체



구조체 개념

- **선물셋트**
  - 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
    - 연관성이 있는 서로 다른 개별적인 자료형의 변수들을 하나의 단위로 묶은 새로운 자료형
    - 연관된 멤버로 구성되는 통합 자료형으로 대표적인 유도 자료형
      - 기존 자료형으로 새로이 만들어진 자료형을 유도 자료형(derived data types)

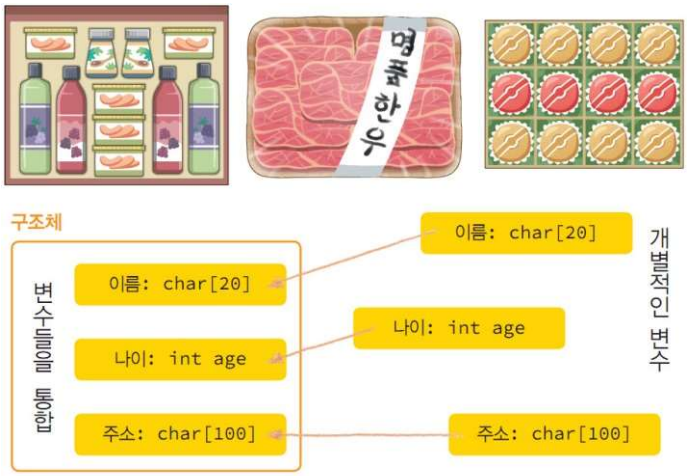


그림 13-1 선물셋트와 구조체

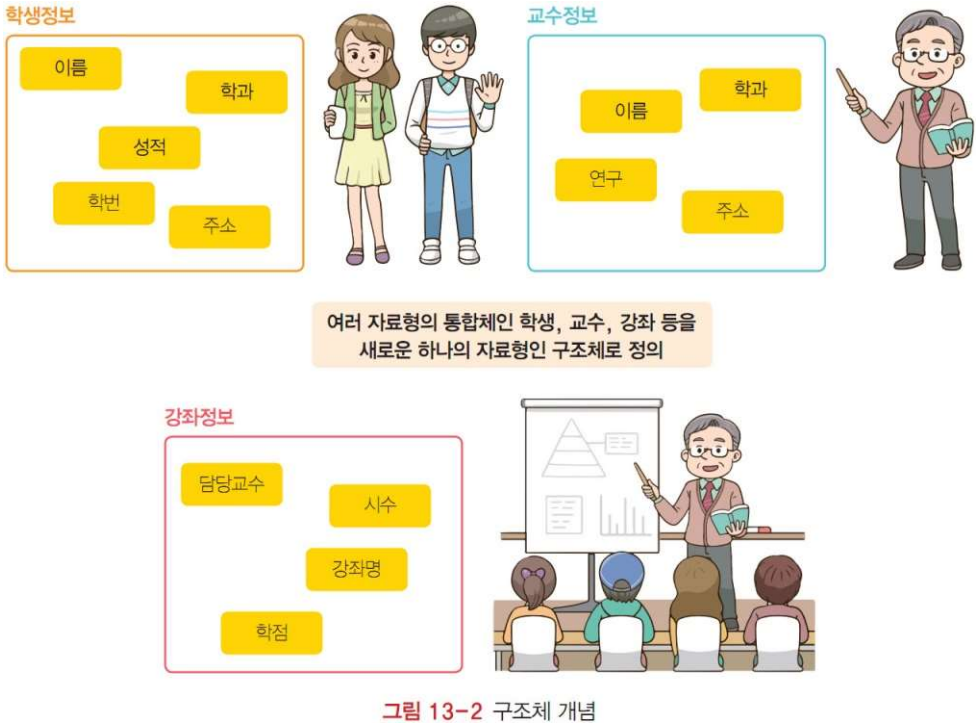


그림 13-2 구조체 개념

구조체 정의 개념

- **와플이나 붕어빵을 만들려면**
  - 와플 기계나 붕어빵 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

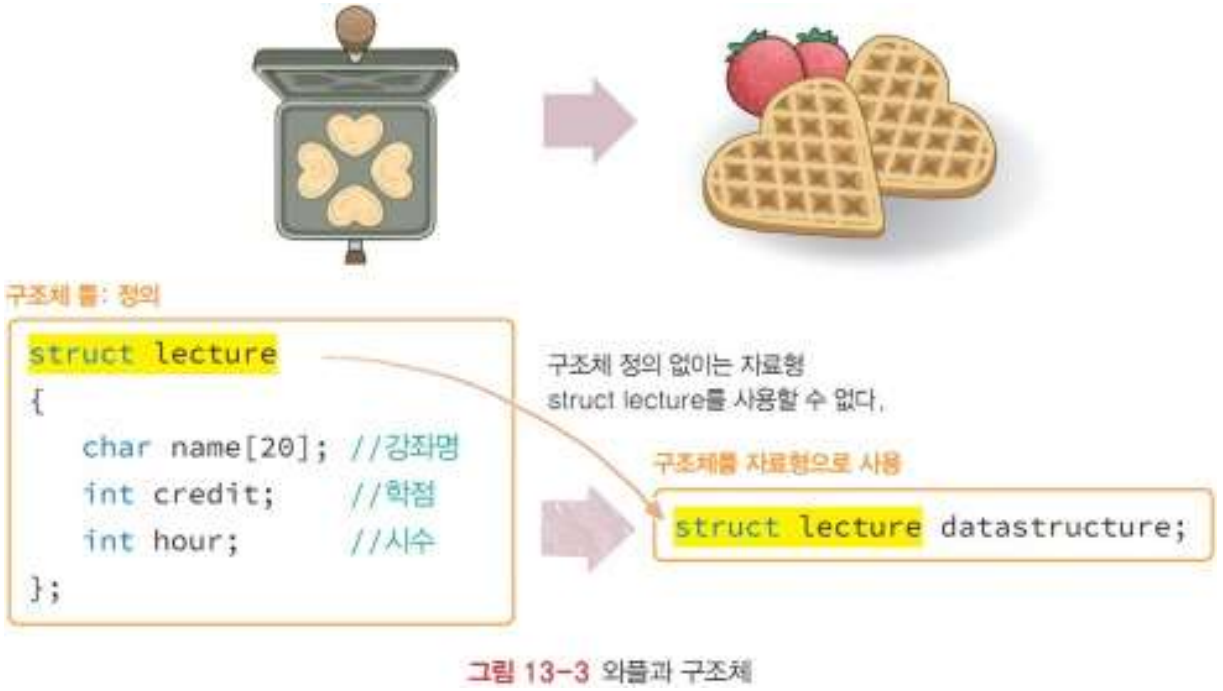


그림 13-3 와플과 구조체

구조체 정의 구문

- **대학의 강좌정보를 처리하는 구조체의 한 예: struct lecture**
  - 구조체 정의는 변수의 선언과는 다름
    - 변수선언에서 이용될 새로운 구조체 자료형을 정의하는 구문
  - 모두 하나의 문장이므로 반드시 세미콜론으로 종료
  - 각 구조체 멤버의 초기값 대입 불가능
  - 모든 멤버 선언에 반드시 세미콜론 삽입, 마지막 멤버도
  - 구조체 멤버의 이름은 모두 유일
  - 멤버로는 다양한 자료형, 다른 구조체 변수 및 구조체 포인터 도 허용
- **구조체 태그이름: account**
  - struct account
    - 계좌정보를 표현하는 구조체
  - 계좌주이름, 계좌번호, 잔고 정보를

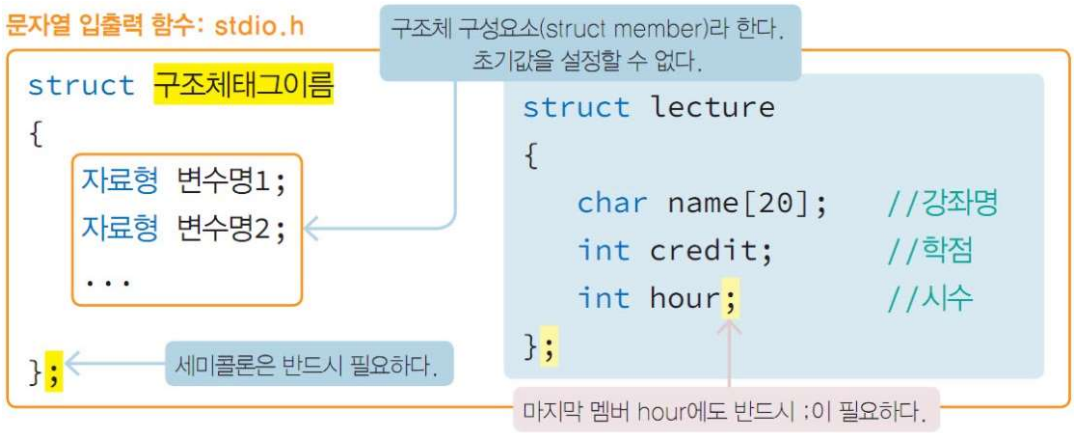


그림 13-4 구조체 정의 구문과 예문

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

그림 13-5 은행의 계좌를 표현하는 구조체 struct account 정의

구조체 변수 선언

- **구조체가 정의되었다면**
  - 구조체형 변수 선언이 가능
    - 구조체 struct account가 새로운 자료유형으로 사용 가능
  - 새로운 자료형 struct account 형 변수 mine을 선언 구문
    - struct account mine;

구조체 자료형 변수 선언 및 초기화 구문

```
struct   구조체태그이름    변수명;
struct   구조체태그이름    변수명1, 변수명2, 변수명3, ...;
```

여러 변수의 선언도 가능하다.

```
struct account yours;
struct account act1, act2, act3;
```

그림 13-6 구조체 자료형 변수 선언 및 초기화 구문

- **구조체 정의와 변수 선언을 함께하는 방법**
  - 이 문장 이후 struct account도 새로운 자료형으로 사용 가능

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

struct account youraccount;
```

변수 myaccount는 struct account형 변수로 선언된다.

변수 youraccount도 struct account형 변수로 선언된다.

그림 13-7 구조체 정의와 변수 선언을 함께하는 문장

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

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

그림 13-9 구조체 변수의 초기화

- **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

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

변수 youraccount이후에 이와 동일한 구조체의 변수선언은 불가능하다.

그림 13-8 구조체 태그이름이 없는 변수 선언 방법

구조체의 멤버 접근 연산자 . 와 변수 크기

- **선언된 구조체형 변수에서 멤버 접근 방법**
  - 접근연산자 .를 사용하여 멤버를 참조
  - 문장 yours.actnum=1002;
    - 변수 yours의 멤버 actnum에 1002를 저장하는 기능을 수행
    - 접근연산자는 .는 참조연산자라고도 부름

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

그림 13-10 구조체 멤버 접근 연산자

- **구조체 struct account 의 변수 mine은 다음 구조로 메모리에 할당**
  - 변수 mine의 크기는 sizeof(mine)로 가능
  - 실제 구조체의 크기는 멤버의 크기의 합보다 크거나 같을 수 있음

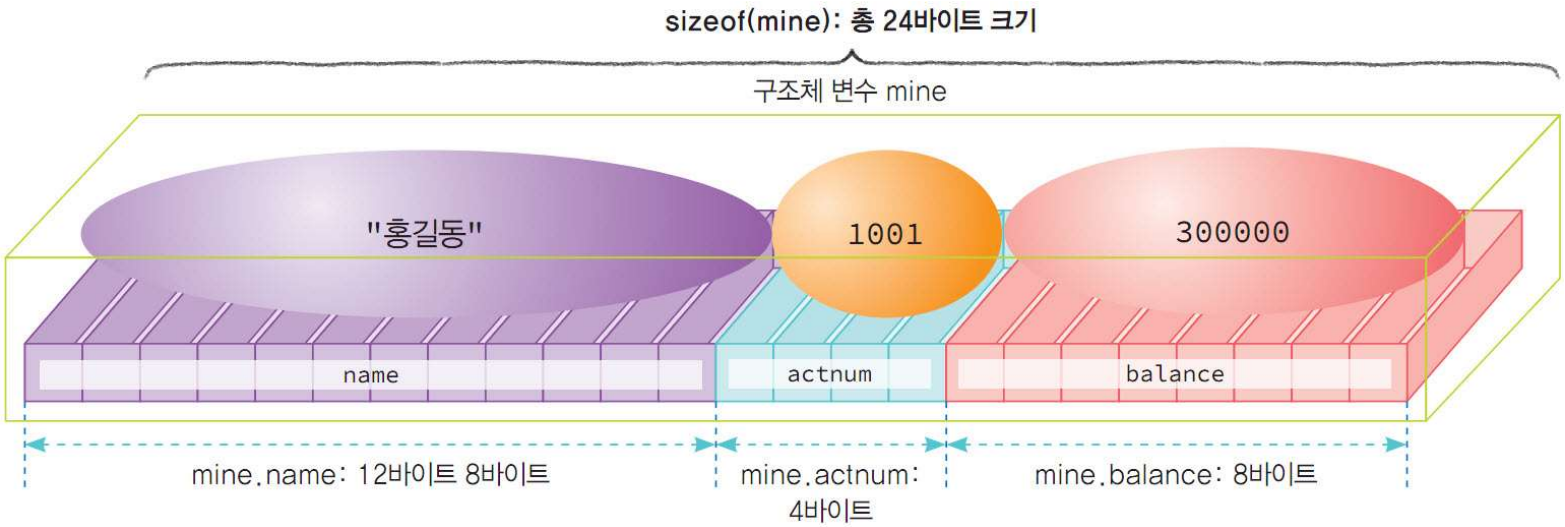


그림 13-11 구조체 struct account 변수 mine의 메모리 구조와 접근 연산자

구조체 정의와 선언

예제 structbasic.c

- 은행계좌를 위한 구조체 사용

실습예제 13-1 struct.c

구조체 정의와 구조체 변수 선언

```c
// file: structbasic.c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

//은행 계좌를 위한 구조체 정의
struct account
{
   char name[12];   //계좌주 이름
   int actnum;   //계좌번호
   double balance;   //잔고
};

int main(void)
{
   //구조체 변수 선언 및 초기화
   struct account mine = { "홍길동", 1001, 300000 };
   struct account yours;

   strcpy(yours.name, "이동원");
   //strcpy_s(yours.name, 12, "이동원"); //가능
   //yours.name = "이동원"; //오류
   yours.actnum = 1002;
   yours.balance = 500000;

   printf("구조체크기: %d\n", sizeof(mine));
   printf("%s %d %.2f\n", mine.name, mine.actnum, mine.balance);
   printf("%s %d %.2f\n", yours.name, yours.actnum, yours.balance);

   return 0;
}
```

접근 연산자 .를 사용한 멤버의 참조

구조체 변수의 크기도 알 수 있다.

설명

```
구조체 변수 yours의 멤버로 int 형인 actnum에는 1002를 저장
구조체 변수 yours의 멤버로 double 형인 ablance에는 500000를 저장
연산자 sizeof(mine)로 변수 mine의 크기를 조회하여 출력
참조연산자 .를 사용하여 구조체 변수 mine에서 모든 멤버를 참조하여 출력
참조연산자 .를 사용하여 구조체 변수 yours에서 모든 멤버를 참조하여 출력
```

실행결과

```
구조체크기: 24
홍길동 1001 300000.00
이동원 1002 500000.00
```

```
은행계좌를 위한 구조체 struct account를 정의하기 위한 문장 시작으로 7 행에서 12 행까지 정의
구조체 struct account 멤버인 계좌주 이름 name을 위한 선언 문으로 마지막에 세미콜론 ;이 필요
구조체 struct account 멤버인 계좌번호 actnum을 위한 선언 문으로 마지막에 세미콜론 ;이 필요
구조체 struct account 멤버인 잔고 balance를 위한 선언 문으로 마지막에 세미콜론 ;이 필요
반드시 마지막에 세미콜론 ;이 필요
구조체 struct account 형인 mine을 선언하면서 초기화, 모든 멤버를 모두 초기화
구조체 struct account 형인 yours를 선언
구조체 struct account 형인 yours의 이름을 "이동원"으로 저장
char 배열인 name으로는 바로 문자열 상수 "이동원"으로는 대입이 불가능
```

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
};
```

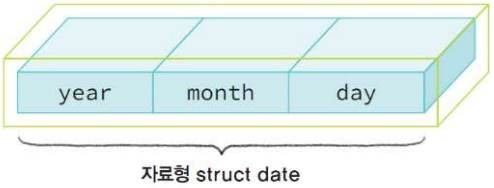


그림 13-12 자료형 struct date의 구조

- **구조체 struct account**
  - 계좌 개설일자를 저장할 멤버로 open을 추가
  - open의 자료형으로 위에서 정의한 struct date를 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 = {{2012, 3, 9}, "홍길동", 1001, 300000 };
```

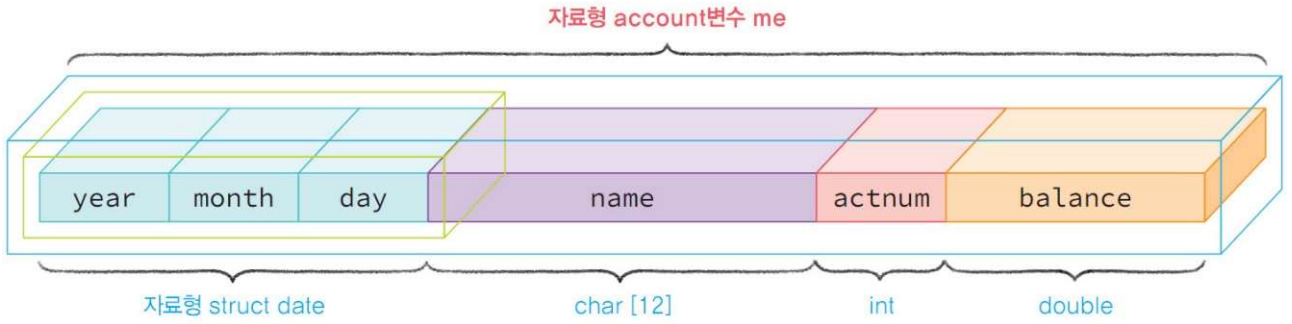


그림 13-13 변수 me의 구조

구조체 멤버의 구조체

예제 nestedstruct.c

- account 구조체를 사용한 프로그램
- 멤버가 구조체 date인 초기화
  - {2012, 3, 9}
- 구조체 account 변수인 me로 년, 월, 일을 참조
  - 접근연산자를 2번 사용
  - me.open.year, me.open.month, me.open.day를 이용

실습예제 13-2 nestedstruct.c

구조체 멤버로 다른 구조체 허용

```
01  // file: nestedstruct.c
02  #include <stdio.h>
03  #include <string.h>
04
05  //날짜를 위한 구조체
06  struct date
07  {
08     int year;    //년
09     int month;   //월
10     int day;     //일
11  };
12
13  //은행계좌를 위한 구조체
14  struct account
15  {
16     struct date open;     //계좌 개설일자
17     char name[12];        //계좌주 이름
18     int actnum;           //계좌번호
19     double balance;       //잔고
20  };
21
22  int main(void)
23  {
24     struct account me = { { 2018, 3, 9 }, "홍길동", 1001, 300000 };
25
26     printf("구조체크기: %d\n", sizeof(me));
27     printf("[%d, %d, %d]\n", me.open.year, me.open.month, me.open.day);
28     printf("%s %d %.2f\n", me.name, me.actnum, me.balance);
29  }
```

구조체 멤버로 다른 구조체 변수를 허용한다.

변수 opendate를 위한 {}는 생략 가능하다.

중첩된 구조체를 접근하려면 접근연산자를 2번 사용한다.

설명

06 날짜를 위한 구조체 struct date를 정의하기 위한 문장 시작으로 6 행에서 11 행까지 정의

08 구조체 date 멤버인 년 year를 위한 선언 문으로 마지막에 세미콜론 ;이 필요

09 구조체 date 멤버인 월 month를 위한 선언 문으로 마지막에 세미콜론 ;이 필요

10 구조체 date 멤버인 일 day를 위한 선언 문으로 마지막에 세미콜론 ;이 필요

11 반드시 마지막에 세미콜론 ;이 필요

14 은행계좌를 위한 구조체 struct account를 정의하기 위한 문장 시작으로 14 행에서 20 행까지 정의

16 구조체 account 내부 멤버로 구조체 struct date 형으로 변수 open을 선언하며, 이 변수는 계좌 개설 날짜를 의미

24 구조체 struct account 형인 me를 선언하면서 초기화, 모든 멤버를 모두 초기화, 첫 멤버는 다시 구조체이므로 {2016, 3, 9}처럼 초기화하는 편이 좋으나 중괄호는 없어도 상관 없음

26 연산자 sizeof(me)로 변수 me의 크기를 조회하여 출력

27 참조연산자 .를 사용하여 구조체 변수 me에서 첫 멤버인 구조체 형인 open의 모든 멤버를 참조하여 출력하려면 me.open.year와 같이 두 번의 참조가 필요

28 참조연산자 .를 사용하여 구조체 변수 me의 나머지 모든 멤버를 참조하여 출력

실행결과

```
구조체크기: 40
[2018, 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

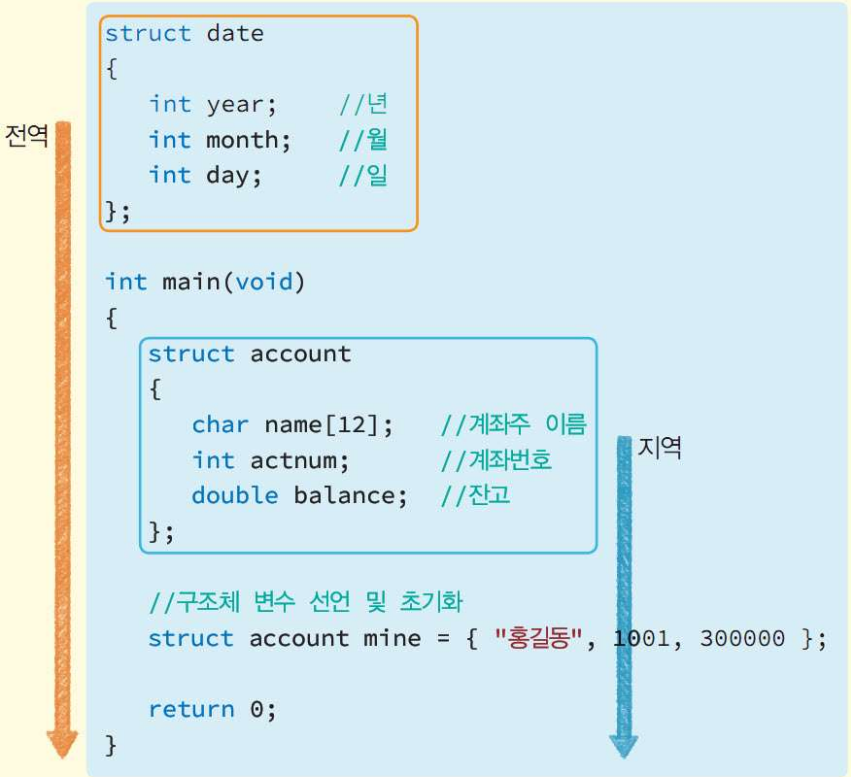


그림 13-14 구조체 정의의 유효 범위

구조체 변수의 대입과 동등비교

- **구조체 변수의 대입문이 가능**
  - 동일한 구조체형의 변수는 대입문이 가능
  - 변수 대입으로 한번에 모든 맴버의 대입이 가능

```c
struct student
{
   int snum;    //학번
   char *dept;    //학과 이름
   char name[12];    //학생 이름
};
struct student hong = { 201800001, "컴퓨터정보공학과", "홍길동" };
struct student one;

one = hong;
```

그림 13-15 구조체 변수의 대입

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

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

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

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

그림 13-16 구조체의 동등 비교

구조체 대입과 비교

예제 structstudent.c

실습예제 13-3 structstudent .c

```
01  // file: structstudent.c
02  #define _CRT_SECURE_NO_WARNINGS
03  #include <stdio.h>
04  #include <string.h>
05
06  int main(void)
07  {
08      //학생을 위한 구조체
09      struct student
10      {
11          int snum;       //학번
12          char *dept;       //학과 이름
13          char name[12];    //학생 이름
14      };
15      struct student hong = { 201800001, "컴퓨터정보공학과", "홍길동" };
16      struct student na = { 201800002 };
17      struct student bae = { 201800003 };
18
19      //학생이름 입력
20      scanf("%s", na.name);
21      //na.name = "나한국"; //오류
22      //scanf("%s", na.dept); //오류
23
24      na.dept = "컴퓨터정보공학과";
25      bae.dept = "기계공학과";
26      memcpy(bae.name, "배상문", 7);
27      strcpy(bae.name, "배상문");
28      strcpy_s(bae.name, 7, "배상문");
29
30      printf("[%d, %s, %s]\n", hong.snum, hong.dept, hong.name);
31      printf("[%d, %s, %s]\n", na.snum, na.dept, na.name);
32      printf("[%d, %s, %s]\n", bae.snum, bae.dept, bae.name);
33
34      struct student one;
35      one = bae;
36      if (one.snum == bae.snum)
37          printf("학번이 %d으로 동일합니다.\n", one.snum);
38      //if ( one == bae ) //오류
39      if (one.snum == bae.snum && !strcmp(one.name, bae.name) &&
            !strcmp(one.dept, bae.dept))
40          printf("내용이 같은 구조체입니다.\n");
41
42      return 0;
43  }
```

13 구조체 student 멤버인 학생 이름 name을 위한 선언 문으로 마지막에 세미콜론 ;이 필요, 변수 name은 char 배열로 문자열 자체의 저장이 가능하므로 scanf()나 memcpy(), strcpy()의 사용도 가능
14 반드시 마지막에 세미콜론 ;이 필요
15 구조체 struct student 형인 hong을 선언하면서 초기화, 모든 멤버를 모두 초기화
16 구조체 struct student 형인 na을 선언하면서 초기화, 첫 번째 멤버인 학번만 기술되었으므로 나머지는 각각 NULL과 NULL 문자로 초기화
17 구조체 struct student 형인 bae을 선언하면서 초기화
20 구조체 변수 na의 멤버 name에 문자열 이름을 표준입력으로 저장
21 char 배열인 name으로는 바로 문자열 상수 "나한국"으로는 대입이 불가능
22 char 포인터인 dept로는 scanf()로 표준입력의 문자열 저장이 불가능
24 char 포인터인 na.dept에 문자열 상수 "컴퓨터정보공학과" 대입
25 char 포인터인 bae.dept에 문자열 상수 "기계공학과" 대입
26 라이브러리 memcpy()로 bae.name에 "배상문"을 저장
27~28 라이브러리 strcpy()와 strcpy_s()로도 가능
30~32 구조체 변수 hong, na, bae의 모든 멤버를 출력
34 구조체 student 변수 one을 선언
35 구조체 student 변수 one에 변수 bae를 대입
36 one의 학번과 bae의 학번을 비교
38 구조체 자체로는 비교 연산 one == bae은 불가능
39 구조체 변수 one에 변수 bae의 내용을 모든 비교하려면 각각의 멤버를 모두 비교
40 내용이 같으면 출력

실행결과

```
나한국
[201800001, 컴퓨터정보공학과, 홍길동]
[201800002, 컴퓨터정보공학과, 나한국]
[201800003, 기계공학과, 배상문]
학번이 201800003(으)로 동일합니다.
내용이 같은 구조체입니다.
```

09 학생을 위한 구조체 student를 정의하기 위한 문장 시작으로 10 행에서 14 행까지 정의, 이 정의가 있는 위치가 main() 함수 내부이므로 구조체 student는 main() 함수 내부에서만 사용 가능
11 구조체 student 멤버인 학번 snum을 위한 선언 문으로 마지막에 세미콜론 ;이 필요
12 구조체 student 멤버인 학과 이름 dept을 위한 선언 문으로 마지막에 세미콜론 ;이 필요, 변수 dept는 char *로 문자열 상수의 주소가 저장 가능하나, scanf()나 memcpy(), strcpy()로는 문자열 저장이 불가능

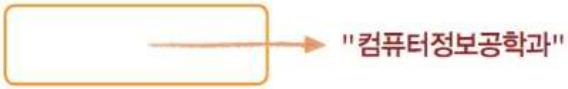
문자열을 처리하기 위한 포인터 char *와 배열 char []

- **char 포인터**
  - 문자열의 첫 문자 주소를 저장하므로 문자열 상수의 주소로 사용
- **char 배열**
  - 문자열을 구성하는 모든 문자를 하나 하나 저장하고 마지막에 '₩0' 문자를 저장하여 사용

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

그림 13-17 char 포인터와 char 배열의 선언

표 13-1 char 포인터와 char 배열의 비교

char 포인터	char 배열
`char *dept;  //학과 이름`	`char name[12];  //학생 이름`
`char *dept = "컴퓨터정보공학과";`  변수 dept	`char name[12] = "나한국";`  변수 name[12]
변수 dept는 포인터로 단순히 문자열 상수를 다루는 경우 효과적	변수 name은 배열로 12바이트 공간을 가지며 문자열을 저장하고 수정 등이 필요한 경우 효과적
`dept = "컴퓨터정보공학과";`	`name = "나한국"; //오류`
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 가능이 가능
`strcpy(dept, "컴퓨터정보공학과"); //오류`	`strcpy(name, "배상문");`
`scanf("%s", dept); //오류`	`scanf("%s", name);`

공용체 개념

- **하나의 차고에 일반 세단과 SUV를 각각 주차한다고 생각**
  - 주차 공간이 하나이므로 세단과 SUV를 동시에 주차시킬 수는 없으나 한 대의 주차는 가능
- **공용체**
  - 이러한 겸용 주차장과 비슷한 개념
  - 동일한 저장 장소에 여러 자료형을 저장하는 방법
  - 공용체를 구성하는 멤버에 한번에 한 종류만 저장하고 참조 가능

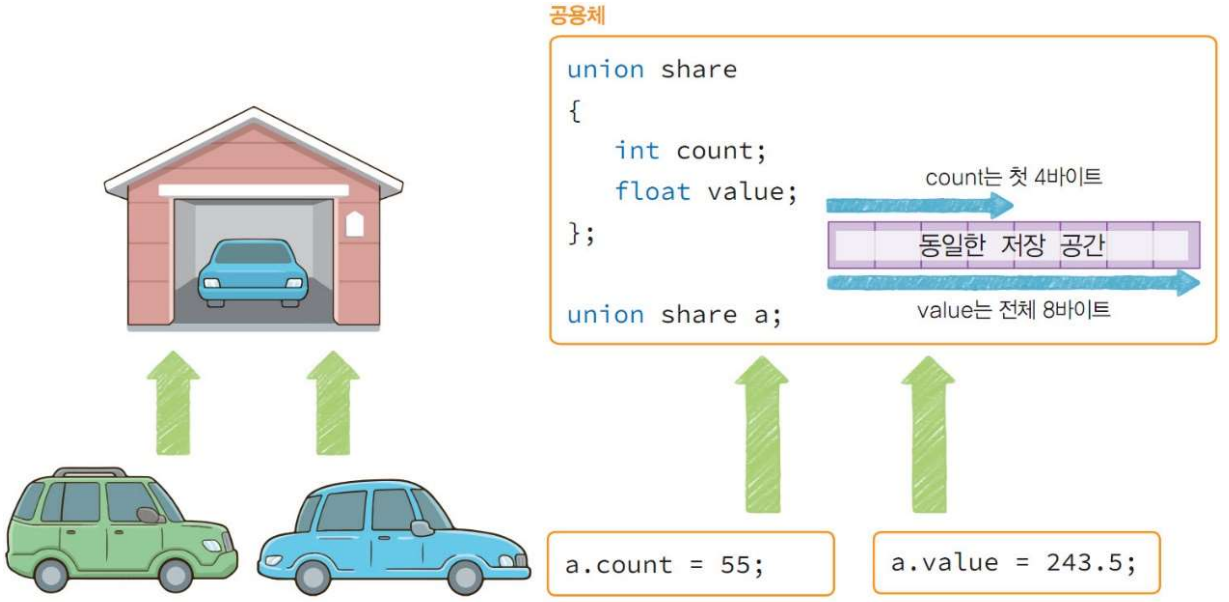


그림 13-18 동일한 공간을 함께 사용하는 공용체 개념

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 변수선언 구문

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

} [변수명1] [,변수명2] ;
```

공용체 구성요소인 멤버(struct member)이다.

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
};
```

그림 13-19 구조체 정의 구문

공용체 크기와 초기화

- **공용체 변수의 크기**
  - 멤버 중 가장 큰 자료형의 크기로 정해짐
  - union data의 변수 data1은 멤버 중 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
  - 공용체도 구조체와 같이 typedef를 이용하여 새로운 자료형으로 정의 가능
- **공용체의 초기화**
  - 공용체 정의 시 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
```

그림 13-20 공용체 초기화 및 경고

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 접근하면 전체 공간의 첫 4바이트만 참조
    - double인 real을 접근하면 8바이트 공간을 모두 참조
  - 항상 마지막에 저장한 멤버로 접근해야 원하는 값을 얻을 수 있음
    - 공용체를 참조할 경우 정확한 멤버를 사용하는 것은 프로그래머의 책임

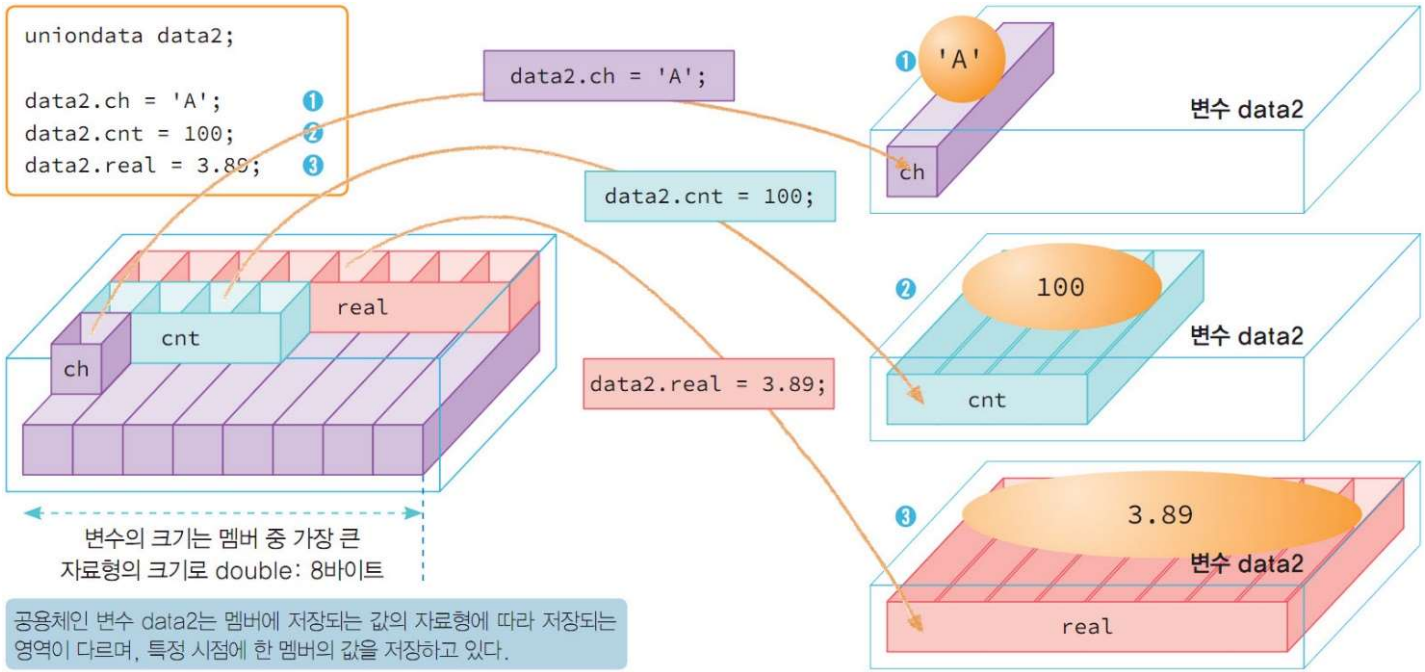


그림 13-21 공용체 변수의 저장 공간과 참조

공용체 정의와 선언

예제 union.c

- 문자와 정수와 실수를 각각 하나씩 저장할 수 있는 공용체의 정의와 활용

실습예제 13-4 union.c

공용체 정의와 변수 선언 및 사용

```
01  // file: union.c
02  #include <stdio.h>
03
04  //유니온 구조체를 정의하면서 변수 data1도 선언한 문장
05  union data
06  {
07      char ch;    //문자형
08      int cnt;    //정수형
09      double real;    //실수형
10  } data1; //data1은 전역변수
11
12  int main(void)
13  {
14      union data data2 = { 'A' }; //첫 멤버인 char형으로만 초기화 가능
15      //union data data2 = {10.3};//컴파일 시 경고 발생
16      union data data3 = data2;   //다른 변수로 초기화 가능
17
18      printf("%d %d\n", sizeof(union data), sizeof(data3));
19
20      //멤버 ch에 저장
21      data1.ch = 'a';
22      printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
23      //멤버 cnt에 저장
24      data1.cnt = 100;
25      printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
26      //멤버 real에 저장
27      data1.real = 3.156759;
28      printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
29
30      return 0;
31  }
```

설명

```
05  char, int, double 자료형 하나를 동시에 저장할 수 있는 8바이트 공간의 공용체 data를
    정의하기 위한 문장 시작으로 5 행에서 10 행까지 정의하면서, 동시에 변수 data1을 선언, 이 변수
    data1은 전역 변수로 이 선언이 있는 위치 이후 모든 파일에서 사용 가능
14  공용체 union data 형으로 변수 data2를 선언하면서 초기값으로 문자 'A'를 저장, 공용체의
    첫 멤버인 char형으로만 초기화 가능
15  공용체의 세 번째 멤버인 double 형으로는 초기화 불가능
16  공용체 union data 형으로 변수 data3를 선언하면서 초기값으로 같은 유형의 변수를 대입
18  공용체 union data 자료형과 변수의 크기인 8을 출력
21  공용체인 data1에서 자료형 char인 멤버 ch에 문자 'a'를 저장, 이 이후로는 data1.ch만
    의미가 있음
22  공용체인 data1에서 모든 멤버를 출력해 보나, data1.ch만 정확히 출력
24  공용체인 data1에서 자료형 int인 멤버 cnt에 정수 100을 저장, 이 이후로는 data1.cnt만
    의미가 있음
25  공용체인 data1에서 모든 멤버를 출력해 보나, data1.cnt만 정확히 출력
27  공용체인 data1에서 자료형 double인 멤버 real에 실수 3.156759를 저장, 이 이후로는
    data1.real만 의미가 있음
28  공용체인 data1에서 모든 멤버를 출력해 보나, data1.real만 정확히 출력
```

실행결과

```
08   8
a 97 0.000000
d 100 0.000000
N -590162866 3.156759
```

LAB 도시의 이름과 위치를 표현하는 구조체

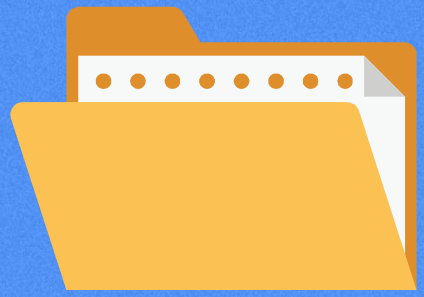
- **도시의 이름과 위치를 표현하는 구조체 struct city**
  - 멤버로 char *와 struct position으로 구성
  - 멤버인 구조체 struct position
    - 위도(latitude)와 경도(longitude)를 표현하는 멤버
- **프로그램**
  - 구조체 struct city의 변수를 선언해 서울과 뉴욕의 정보를 저장하고 이들 도시의 정보를 다시 출력
  - 구조체 struct position 변수 seoul과 newyork
- **결과**
  - [서울] 위도= 37.3 경도= 126.6
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

Lab 13-1 structcity.c

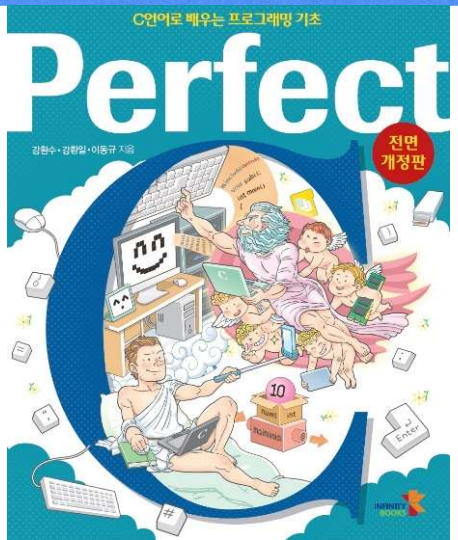
```c
// file: structcity.c
#include <stdio.h>
#include <string.h>

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

int main(void)
{
   //도시 정보 구조체
   struct city
   {
      char *name;      //이름
      struct position place;   //위치
   };
   ____________________ seoul, newyork;


   ___________________________________;
   ___________________________________;
   ___________________________________;

   ___________________________________;
   ___________________________________;
   ___________________________________;

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

   return 0;
}
```

정답

```c
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;
```



02. 자료형 재정의



자료형 재정의 typedef(1)

- **typedef 구문**
  - typedef는 이미 사용되는 자료 유형을 다른 새로운 자료형 이름으로 재정의할 수 있도록 하는 키워드
- **typedef int profit;**
  - profit을 int와 같은 자료형으로 새롭게 정의하는 문장

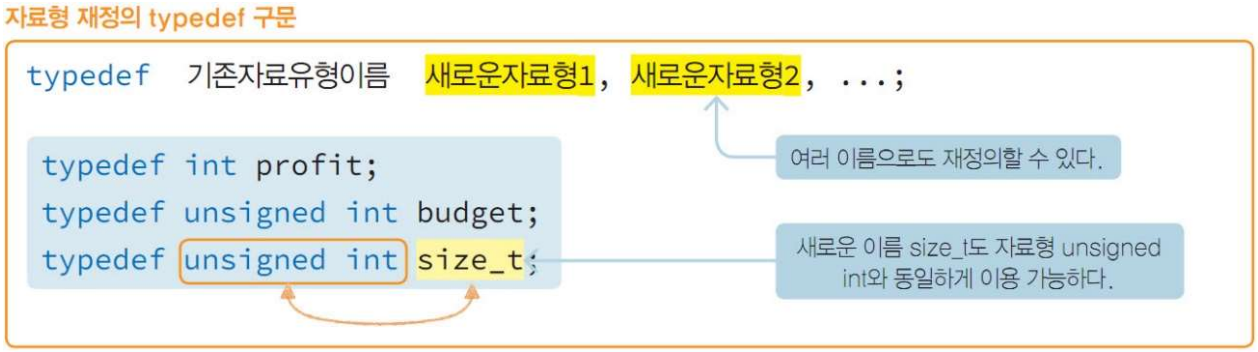


그림 13-22 자료형 재정의 typedef 구문

자료형을 재정의하는 이유

- **프로그램의 시스템 간 호환성과 편의성을 위해 필요**
  - 터보 C++ 컴파일러에서 자료유형 int는 저장공간 크기가 2바이트
  - Visual C++에서는 4바이트
  - Visual C++에서 작성한 프로그램은 터보 C++에서는 문제가 발생
    - 2 바이트로는 2000000을 저장할 수 없기 때문

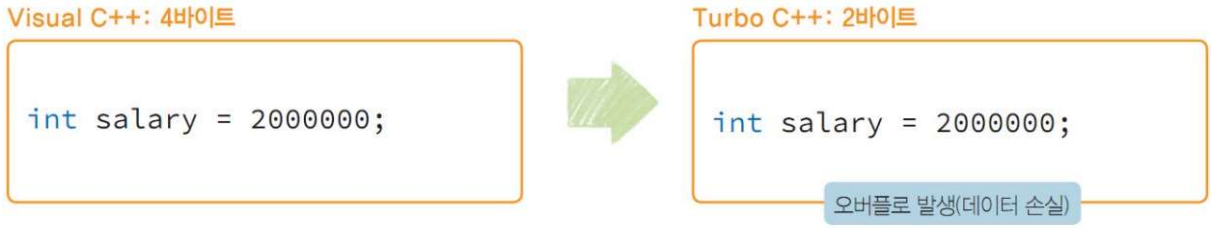


그림 13-23 개발환경의 변화에 따른 프로그램의 문제

- **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

그림 13-24 서로 다른 개발환경에서 호환이 되도록 자료유형 재정의

자료형 재정의

예제 typedef.c

- 다양한 자료형 키워드 생성

제어변수

- 자료형 int를 여러 개의 새로운 자료형 이름 integer와 word로 재정의하는 것도 가능
- 문장 typedef도 일반 변수와 같이 그 사용 범위를 제한
  - 함수 내부에서 재정의되면 선언된 이후의 그 함수에서만 이용이 가능
  - 함수 외부에서 재정의된다면 재정의된 이후 그 파일에서 이용이 가능

```
typedef int integer, word;

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

실습예제 13-5 typedef.c

자료형 재정의 이용

```
01  // file: typedef.c
02  #include <stdio.h>
03
01  //함수 외부에서 정의된 자료형은 이후 파일에서 사용 가능
02  typedef unsigned int budget;
03
04  int main(void)
05  {
06     //새로운 자료형 budget 사용
07     budget year = 24500000;
08
09     //함수 내부에서 정의된 자료형은 이후 함수내부에서만 사용 가능
10     typedef int profit;
11     //새로운 자료형 profit 사용
12     profit month = 4600000;
13
14     printf("올 예산은 %d, 이달의 이익은 %d 입니다.\n", year, month);
15
16     return 0;
17  }
18
19  void test(void)
20  {
21     //새로운 자료형 budget 사용
22     budget year = 24500000;
23
24     //profit은 이 함수에서는 사용 불가, 오류 발생
25     //profit year;
26  }
```

설명

```
05  이 문장으로 budget은 int와 같은 자료형, 이 위치가 함수 외부이므로 자료형 budget은 이 위치
    이후 파일에서 사용 가능
10  budget은 int와 같은 자료형으로 변수 year를 budget으로 선언하면서 초기값 대입
13  이 문장으로 profit은 int와 같은 자료형, 이 위치가 함수 내부이므로 자료형 profit은 이 위치
    이후 함수 main() 내부에서 사용 가능
15  profit은 int와 같은 자료형으로 변수 month를 profit으로 선언하면서 초기값 대입
25  자료형 budget은 함수 test()에서도 사용 가능
28  자료형 profit은 함수 test()에서는 사용 불가
```

실행결과

```
올 예산은 24500000, 이달의 이익은 4600000 입니다.
```

그림 13-25 하나의 자료형을 여러 자료형으로 재정의

struct를 생략한 새로운 자료형

- **구조체 자료형은 struct date 처럼 항상 키워드 struct를 써야 하나?**
  - typedef 사용하여 구조체 struct date를 date로 재정의
  - 물론 date가 아닌 datetype 등 다른 이름으로도 재정의가 가능

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

typedef struct date date;
```

자료유형인 date는 struct date와 함께 동일한 자료유형으로 이용이 가능하다.

그림 13-26 새로운 자료유형 date의 재정의와 이용

- **구조체 정의 자체를 typedef와 함께 처리하는 방법**
  - typedef 구문에서 새로운 자료형으로 software 형이 정의
  - 이 구문 이후에는 software를 구조체 자료형으로 변수 선언에 사용
  - 구조체 태그이름은 생략 가능
  - 구조체 software 형은 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.

그림 13-27 구조체 정의와 typedef를 함께 이용한 자료형의 정의

구조체 재정의

예제 typedefstruct.c

- 자료형 struct data를 간단히 date 자료형으로 재정의

실습예제 13-6 typedefstruct.c

문장 typedef를 이용하여 구조체의 자료유형

```c
// file: typedefstruct.c
#include <stdio.h>

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

//struct date 유형을 간단히 date 형으로 사용하기 위한 구문
typedef struct date date;

int main(void)
{
    //구조체를 정의하면서 바로 자료형 software 로 정의하기 위한 구문
    typedef struct
    {
        char title[30];     //제목
        char company[30];   //제작회사
        char kinds[30];     //종류
        date release;       //출시일
    } software;

    software vs = { "비주얼스튜디오 커뮤니티", "MS", "통합개발환경", { 2018, 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;
}
```

date는 구조체 struct date의 새로운 자료형이다.

software는 변수가 아니라 구조체의 새로운 자료형이다.

설명

- 04~09 구조체 struct date 정의
- 12 struct date 형을 간단히 date 형으로 다시 정의, 자료형 date는 이 구문 이후 파일 어디에서나 사용 가능
- 17 typedef로 시작하는 struct 정의 구문의 시작으로 태그 이름은 생략 가능
- 22 구조체 software의 멤버인 출시일 release를 위한 선언 문으로, 자료형 struct date 대신에 간단히 date 사용 가능
- 23 software는 변수 이름이 아니라 새로운 자료형
- 25 software 자료형으로 vs를 선언하면서 초기값 저장

실행결과

```
제품명: 비주얼스튜디오 커뮤니티
회사 : MS
종류 : 통합개발환경
출시일: 2018. 8. 29
```

LAB 영화 정보를 표현하는 구조체

- **영화의 제목과 관객수를 표현하는 구조체 struct movie**
  - 멤버로 char *와 int로 구성
    - 멤버인 title은 영화 제목을 표현
    - attendance는 관객수를 저장
  - 구조체 struct movie의 자료형
    - 다시 movie로 정의
  - 자료형 movie 변수 assassination에 하나의 영화 정보를 저장한 후 다시 출력
  - 구조체 struct movie를 정의하면서 동시에 자료형 movie 도 정의

```
struct movie
{
   char *title;       //영화제목
   int attendance;    //관객수
};
```

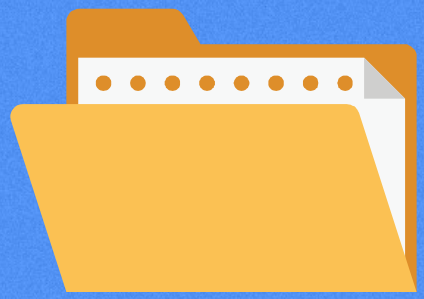
- **결과**
  - [암살] 관객수: 12700000

Lab 13-2 structcity.c

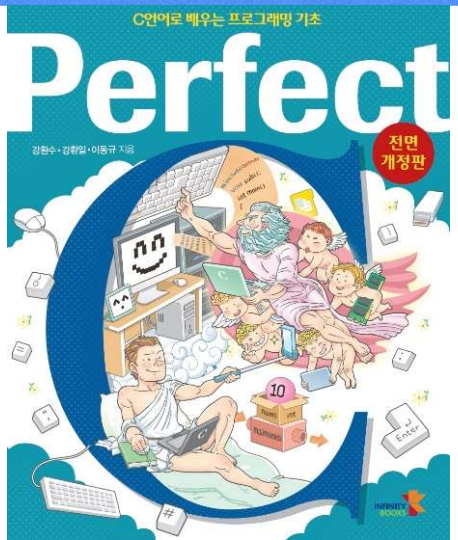
```
01  // file: typemovie.c
02  #include <stdio.h>
03
04  int main(void)
05  {
06     //영화 정보 구조체
07     typedef struct movie
08     {
09        char *title;    //영화제목
10        int attendance;    //관객수
11     ---------------- ;
12
13     -------------------------;
14
15     assassination.title = "암살";
16     assassination.attendance = 12700000;
17
18     printf("[%s] 관객수: %d\n", assassination.title, assassination.attendance);
19
20     return 0;
21  }
```

정답

```
11  } movie;
13  movie assassination;
```



03. 구조체와 공용체의 포인터와 배열



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;
```

그림 13-28 구조체 포인터 변수 선언

- **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;
```

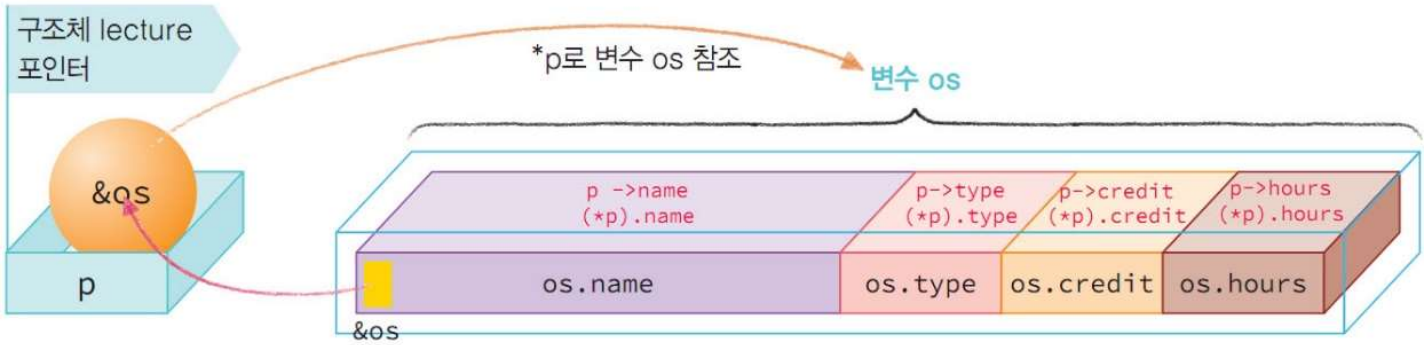


그림 13-29 구조체 변수 os와 포인터 변수 p의 관계 및 구조체 멤버 참조 방법

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 들어가서는 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
- **접근연산자 ->와 .의 연산자 우선순위**
  - 간접연산자 *를 포함한 다른 어떠한 연산자 우선순위보다 가장 높음
    - 연산자 ->와 .은 우선순위 1위이고 결합성은 좌에서 우이며,
    - 연산자 *은 우선순위 2이고 결합성은 우에서 좌

표 13-2 구조체 변수와 구조체 포인터 변수를 이용한 멤버의 참조

접근 연산식	구조체 변수 os와 구조체 포인터변수 p인 경우의 의미
p->name	포인터 p가 가리키는 구조체의 멤버 name
(*p).name	포인터 p가 가리키는 구조체의 멤버 name
*p.name	*(p.name)이고 p가 포인터이므로 p.name은 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 포인터

예제 structpointer.c

- 구조체 강좌와 포인터 활용

실습예제 13-7 structpointer.c

구조체 포인터의 선언과 사용

```
01  // file: structpointer.c
02  #include <stdio.h>
03
04  struct lecture
05  {
06      char name[20];   //강좌명
07      int type;   //강좌구분 0: 교양, 1: 일반선택, 2: 전공필수, 3: 전공선택
08      int credit;   //학점
09      int hours;   //시수
10  };
11  typedef struct lecture lecture;
12
13  //제목을 위한 문자열
14  char *head[] = { "강좌명", "강좌구분", "학점", "시수" };
15  //강좌 종류를 위한 문자열
16  char *lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
17
18  int main(void)
19  {
20      lecture os = { "운영체제", 2, 3, 3 };
21      lecture c = { "C프로그래밍", 3, 3, 4 };
22      lecture *p = &os;
23
24      printf("구조체크기: %d, 포인터크기: %d\n\n", sizeof(os), sizeof(p));
25      printf("%10s %12s %6s %6s\n", head[0], head[1], head[2], head[3]);
26      printf("%12s %10s %5d %5d\n", p->name, lectype[p->type], p->credit,
            p->hours);
27
28      //포인터 변경
29      p = &c;
30      printf("%12s %10s %5d %5d\n", (*p).name, lectype[(*p).type], (*p).
            credit, (*p).hours);
31      printf("%12c %10s %5d %5d\n", *c.name, lectype[c.type], c.credit,
            c.hours);
32
33      return 0;
34  }
```

강좌구분 변수 type에 저장된 값으로 문자 포인터 배열의 첨자를 사용하면 바로 '전공필수' 등이 출력

문자열 'C프로그래밍'에서 첫 글자만 참조하는 연산식

```
10 구조체 struct lecture 정의
   struct lecture를 자료형 lecture로 정의
   제목 출력인 "강좌명", "강좌구분", "학점", "시수"를 위한 문자 포인터 배열
   강좌 종류인 "교양", "일반선택", "전공필수", "전공선택"을 위한 문자 포인터 배열
   구조체 자료형 lecture 변수인 os를 선언하면서 초기화, 두 번째 맴버 초기값 2에서 2는 강좌구분
   "전공필수"를 의미하고, 16 행 배열 lectype의 첨자인 2
   구조체 자료형 lecture 변수인 c를 선언하면서 초기화, 두 번째 맴버 초기값 3에서 3은 강좌구분
   "전공선택"을 의미하고, 16 행 배열 lectype의 첨자인 3
   구조체 자료형 lecture 포인터 p를 선언하면서 os의 주소를 저장
24 os와 p의 저장공간 크기를 출력
25 제목 행 출력
26 구조체 포인터 p와 참조연산자 ->를 이용하여 멤버를 참조하여 출력
29 구조체 자료형 lecture 포인터 p에 c의 주소로 변경, 이제 p는 c를 가리킴
30 구조체 포인터 p와 간접연산자 (*p)와 참조연산자 .를 이용하여 멤버를 참조하여 출력
31 구조체 자체인 c와 참조연산자 .를 이용하여 멤버를 참조하여 출력, 첫 번째 출력인 *c.name은
   *(c.name)을 의미하므로 첫 글자인 C가 출력
```

실행결과

```
구조체크기: 32, 포인터크기: 4

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

공용체 포인터

예제 unionpointer.c

- 공용체와 포인터의 활용

공용체 data

- 공용체 변수도 포인터 변수 사용이 가능하며, 공용체 포인터 변수로 멤버를 접근하려면
  - 접근연산자 ->를 이용
- 공용체 변수 value를 가리키는 포인터 p를 선언
  - p가 가리키는 공용체 멤버 ch에 'a'를 저장
  - 연산식 p->ch는 포인터가 가리키는 공용체 변수의 멤버 ch를 접근하는 연산식
- 마찬가지로 p->cnt, p->real
  - 각각 value.cnt, value.real을 참조하는 연산식

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

변수 value는 union data형이며 p는 union data 포인터 형으로 선언

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

그림 13-30 공용체 포인터의 선언과 사용

실습예제 13-8 unionpointer.c

공용체 정의와 변수 선언 및 사용

```
01 // file: unionpointer.c
02 #include <stdio.h>
03
04 int main(void)
05 {
06    //공용체 union data 정의
07    union data
08    {
09       char ch;
10       int cnt;
11       double real;
12    };
13
14    //유니온 union data를 다시 자료형 udata로 정의
15    typedef union data udata;
16
17    //udata 형으로 value와 포인터 p 선언
18    udata value, *p;
19
20    p = &value;
21    p->ch = 'a';
22    printf("%c %c\n", p->ch, (*p).ch);
23    p->cnt = 100;
24    printf("%d ", p->cnt);
25    p->real = 3.14;
26    printf("%.2f \n", p->real);
27
28    return 0;
29 }
```

변수 value와 p는 모두 함수 main() 내부에서만 사용이 가능한 지역변수이다.

연산식 p->ch와 (*p).ch는 모두 value.ch를 참조한다.

설명

```
07 char, int, double 자료형 하나를 동시에 저장할 수 있는 8바이트 공간의 공용체 unin data를 정의하기 위한 문장 시작으로 7 행에서 12 행까지 정의, 이 공용체 정의로 이 위치 이후 모든 파일에서 공용체 unin data 사용 가능
15 유니온 union data를 다시 자료형 udata로 정의
   udata 형으로 value와 포인터 p 선언
20 포인터 변수 p에 공용체 변수 value의 주소를 저장
21 p가 가리키는 공용체 변수 value의 멤버 ch를 연산자 ->를 사용하여 p->ch로 참조, 멤버 ch에 문자 'a' 저장
22 p가 가리키는 공용체 변수 value의 멤버 ch를 p->ch 와 (*p).ch 로 참조하여 출력
23 p가 가리키는 공용체 변수 value의 멤버 cnt를 연산자 ->를 사용하여 p->cnt로 참조, 멤버 cnt에 정수 100 저장
24 p가 가리키는 공용체 변수 value의 멤버 cnt를 p->cnt 로 참조하여 출력, (*p).cnt도 가능
25 p가 가리키는 공용체 변수 value의 멤버 real을 연산자 ->를 사용하여 p->real로 참조, 멤버 real에 실수 3.14 저장
26 p가 가리키는 공용체 변수 value의 멤버 real을 p->real 로 참조하여 출력, (*p).real도 가능
```

실행결과

```
a a
100 3.14
```

구조체 배열 변수 선언

- **구조체 lecture의 배열크기 3인 c를 선언하고 초기값을 저장하는 구문**
  - 구조체 배열의 초기값 지정 구문에서는 중괄호가 중첩되게 표시
    - 외부 중괄호는 배열 초기화의 중괄호이며
    - 내부 중괄호는 배열원소인 구조체 초기화를 위한 중괄호

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

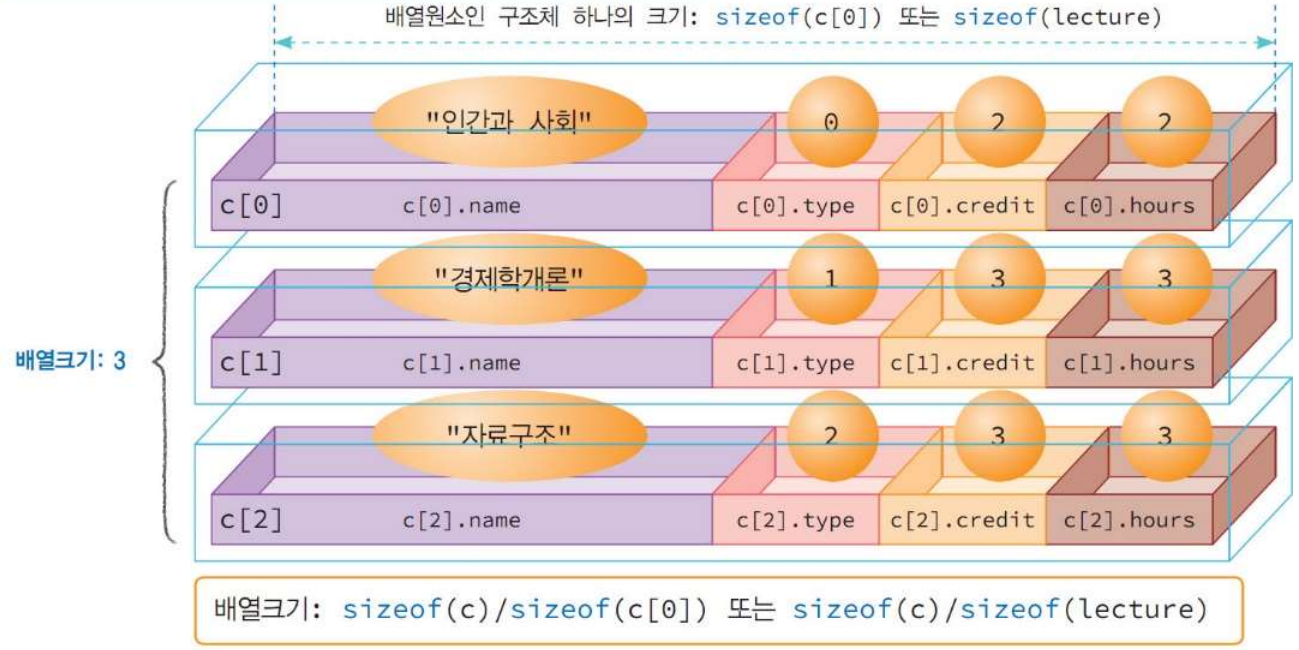


그림 13-31 구조체 lecture의 배열 변수 c

구조체 배열

예제 structarray.c

- 구조체 배열 활용
- 문장 lecture *p = c;
  - 구조체 배열이름은 구조체 포인터 변수에 대입이 가능
  - p[i]로 배열원소 접근이 가능

실습예제 13-9 structarray.c

구조체 배열을 선언한 후 출력 처리

```
01 // file: structarray.c
02 #include <stdio.h>
03
04 struct lecture
05 {
06    char name[20];   //강좌명
07    int type;   //강좌구분
08    int credit;   //학점
09    int hours;   //시수
10 };
11 typedef struct lecture lecture;
12
13 char *lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
14 char *head[] = { "강좌명", "강좌구분", "학점", "시수" };
15
16 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);
```

```
28    printf("%12s   %12s %6s %6s\n", head[0], head[1], head[2], head[3]);
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 }
```

설명

19 구조체 struct의 배열을 선언하면서 바로 초기값을 대입, 배열 크기는 지정하지 않고 초기값을 지정한 원소 수가 5

25 배열 크기를 계산하여 변수 arysize에 저장

27,28 배열 크기와 제목 출력

30 첨자 i는 0에서 arysize-1까지 실행

31,32 구조체 struct의 모든 배열 원소를 각각 출력

실행결과

```
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
```

```
lecture *p = c;

//p로도 참조 가능
for (i = 0; i < arysize; i++)
   printf("%16s %10s %5d %5d\n", p[i].name, lectype[p[i].type],p[i].credit, p[i].hours);
```

그림 13-32 배열의 주소를 저장

LAB 영화 정보를 표현하는 구조체의 배열

- **구조체 struct movie**
  - 영화의 제목과 감독, 관객수를 표현
  - 멤버로 char *title, int attendance, char director[20]로 구성
  - 구조체 movie의 배열을 선언하고 초기화로 영화세 편의 정보를 저장
  - 세 번째 영화의 감독을 "류승완"을 저장하고 모든 영화의 정보를 다시 출력
- **결과**
  - 제목 감독 관객수
  - ==========
  - [명량] 김한민 17613000
  - [국제시장] 윤제균 14257000
  - [베테랑] 류승완 13383000

Lab 13-3 structmovie.c

```c
// file: structmovie.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;

   movie ____________________________
   { "명량", 17613000, "김한민" },
   { "국제시장", 14257000, "윤제균" },
   { "베테랑", 13383000} };

   //영화 베테랑의 감독을 류승완으로 저장
   ________________________________

   printf("   제목      감독    관객수\n");
   printf("==========================\n");
   for (int i = 0; i < 3; i++)
      printf("[%8s] %6s %d\n",
         ____________________________________________________

   return 0;
}
```

정답

```c
box[] = {
strcpy(box[2].director, "류승완");
   box[i].title, box[i].director, box[i].attendance);
```

Thank you