프로그래밍 기초

13장. 구조체

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

- **구조체(structure)**
  - 정수, 문자, 실수나 포인터 그리고 이들의 배열 등을 묶어 하나의 자료형으로 사용하는 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 자료형의 변수들을 하나의 단위로 묶은 새로운 자료형
  - 유도 자료형(derived data types)
    - 기존 자료형에서 새로이 만들어진 자료형

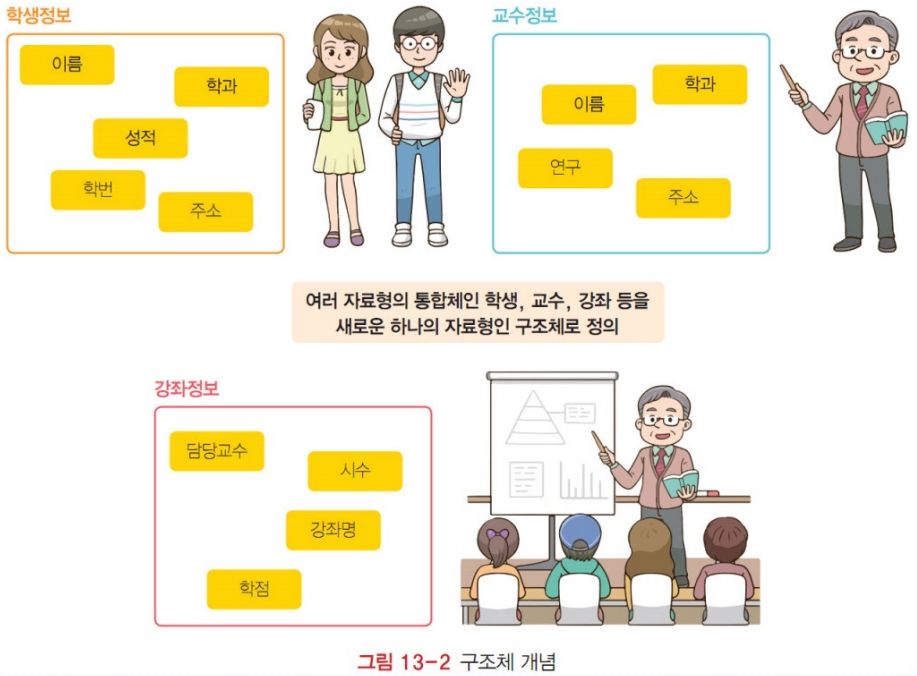
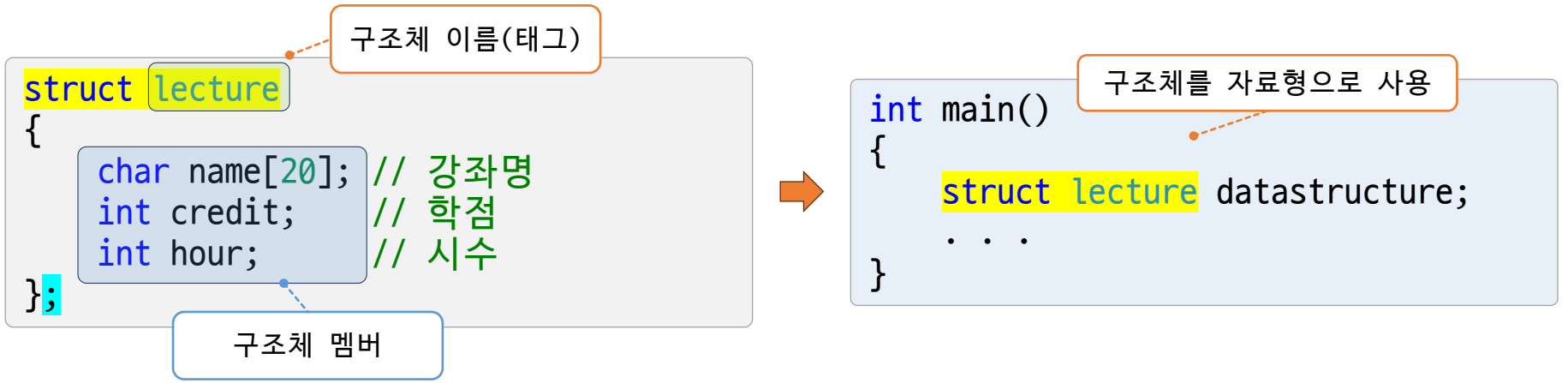


그림 13-2 구조체 개념

구조체 정의 개념

- **구조체를 자료형으로 사용하려면**
  - 먼저 구조체를 정의

- **구조체 정의 방법**
  - 키워드 struct 다음에 구조체 태그 이름을 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**
  - 반드시 세미콜론으로 종료: 모든 멤버 선언에 반드시 세미콜론 삽입
  - 각 구조체 멤버의 초기값 대입 불가능: 선언만 가능
    - `int credit; int hour;`
    - `int credit, hour;` 로도 가능하지만 권장하지 않음
  - 구조체 멤버의 이름은 모두 유일
    - 멤버로는 다양한 자료형, 다른 구조체 변수 및 구조체 포인터도 허용

```
struct 구조체태그이름
{
    자료형 변수명1;
    자료형 변수명2;
    . . .
};
```

세미콜론 추가

```
struct lecture
{
    char name[20]; // 강좌명
    int credit;    // 학점
    int hour;      // 시수
};
```

구조체 변수 선언

- **구조체 정의 후 구조체형 변수 선언이 가능**
  - 구조체 struct account가 새로운 자료형으로 사용 가능
  - 새로운 자료형 struct account 형 변수 yours을 선언 구문
    - struct account yours;

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

새로운 자료형

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

이름 없는 구조체

- **구조체 태그 이름이 없는 구조체 변수 선언 구문**
  - 이 구조체와 동일한 자료형의 변수를 더 이상 선언 불가능
    - ➢ 단 한번 이 구조체 형으로 변수를 선언하는 경우에만 이용
    - ➢ 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 초기값 지정이 가능**
  - 초기값은 중괄호 내부에서 각 멤버 정의 순서대로 초기값을 쉼표로 구분하여 기술
  - 초기값을 지정하지 않은 멤버 변수의 값은 자료형에 따라 기본값인 0, 0.0, '\0' 등으로 저장

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

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

구조체 변수 선언과 동시에 {}로 초기값 지정은 가능함

```
int main()
{
    struct account person1 = {"홍길동", 1001, 300000};

    person1 = {"김유신", 1002, 400000}; // Error 발생


    // double balance = 0.0이 저장
    struct account person2 = {"흥부", 1001};

    struct account p3 = {.name = "놀부", .actnum = 1003,
                         .balance = 50000}; // C99 버전 가능

    return 0;
}
```

구조체의 멤버 접근 연산자 dot(.)

- **구조체 변수에서 멤버 접근 방법**
  - 접근연산자(.) 또는 참조연산자를 사용하여 멤버에 접근
  - 구조체 변수의 값 수정 및 읽기
    - person1.actnum = 1002;
      - 구조체 변수 person1의 멤버 actnum에 1002를 저장
    - printf("%s", person1.name);
      - 구조체 변수 person1의 멤버 name을 출력

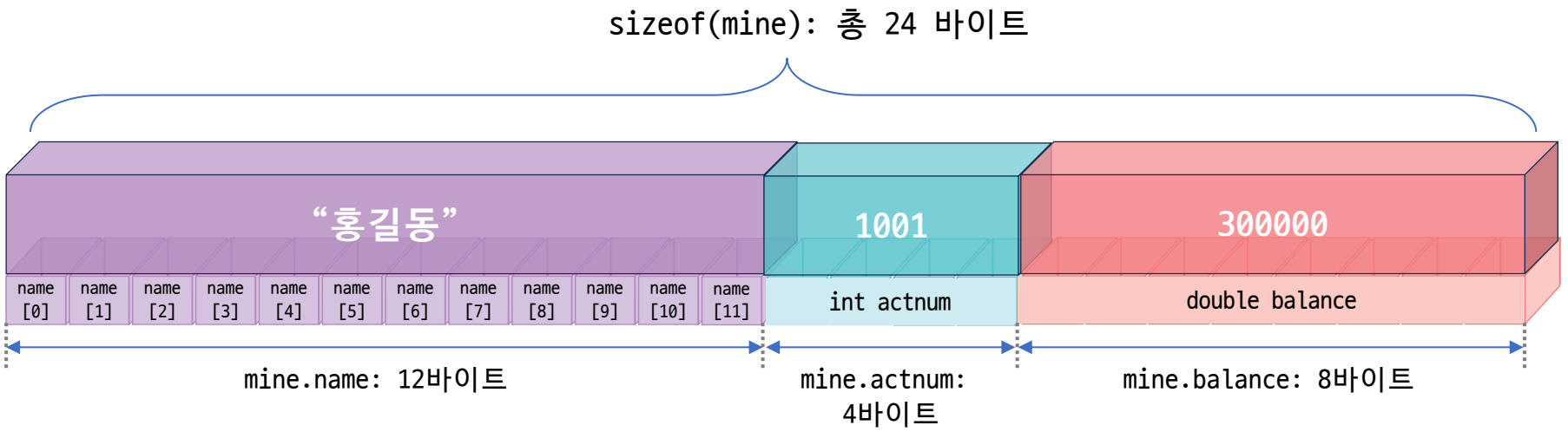
```
구조체변수이름.멤버 = 값;
```

```
person1.actnum = 1002;
person1.balance = 45000;

printf("%s, %d, %.2f\n", person1.name, person1.actnum, person1.balance);
```

접근연산자를 이용한
구조체 변수의 값 읽기

구조체 변수 크기

- **구조체 변수의 크기 계산: sizeof(구조체변수)**
  - struct account의 변수 mine은 다음 구조로 메모리에 할당
  - 변수 mine의 크기는 sizeof(mine)로 가능
    - ➢ 실제 구조체 변수의 크기는 멤버의 크기의 합보다 크거나 같을 수 있음



구조체 정의와 구조체 변수 선언 (실습)

<01structbasic.c>

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

// 은행 계좌를 위한 구조체 정의
struct account
{
    char name[12];  // 계좌주 이름(12bytes)
    int actnum;     // 계좌번호 (4bytes)
    double balance; // 잔고 (8bytes)
};
```

```c
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("구조체 크기: %zu\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);

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

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

    return 0;
}
```

C99 버전에 추가된 기능

실행 결과

```
구조체 크기: 24
홍길동, 1001, 300000.00
이동원, 1002, 500000.00
홍길동, 0, 50000.00
김파이, 0, 70000.00
```

구조체 멤버로 사용되는 구조체

- **구조체 멤버로 사용되는 구조체**
  - 이미 정의된 다른 구조체형 변수
  - 자기 자신을 포함한 구조체 포인터 변수

- **구조체 struct date**
  - 년, 월, 일 정보를 저장할 수 있는 구조체

- **구조체 struct account**
  - 계좌 개설일자를 저장할 멤버로 open을 추가
    - ➢ open의 자료형으로 위에서 정의한 struct date를 사용

  - struct account 변수 me의 메모리 구조

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

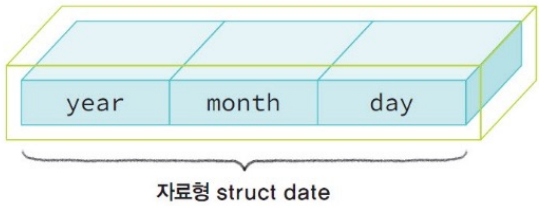
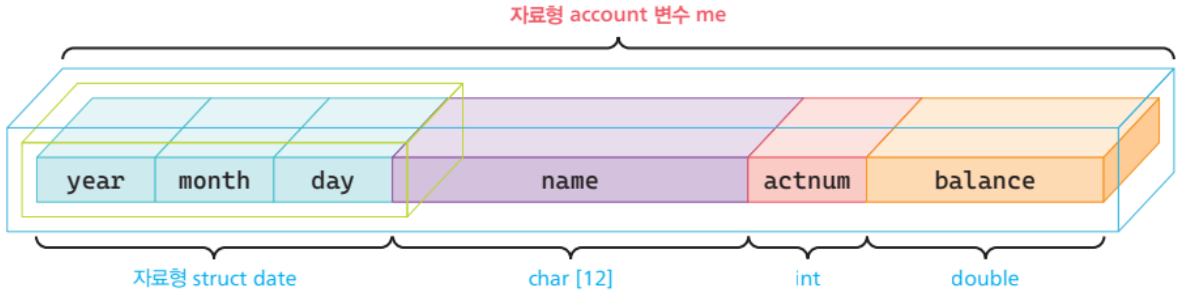


그림 13-12 자료형 struct date의 구조

```
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 포함 예제

<02nestedstruct.c>

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

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

//은행계좌를 위한 구조체
struct account
{
    struct date open;//계좌 개설일자
    char name[12]; //계좌주 이름
    int actnum; //계좌번호
    double balance; //잔고
};
```

date 멤버 초기화를 위해 중괄호 생략 가능

```c
int main(void)
{
    struct account me = {{2022, 3, 9}, "Hong", 1001, 2000};
    //struct account me = {2022, 3, 9, "Hong", 1001, 2000};

    printf("구조체 크기: %zu\n", sizeof(me));
    printf("[%d. %d. %d]\n", me.open.year, me.open.month, me.open.day);
    printf("%s %d %.2f\n", me.name, me.actnum, me.balance);
}
```

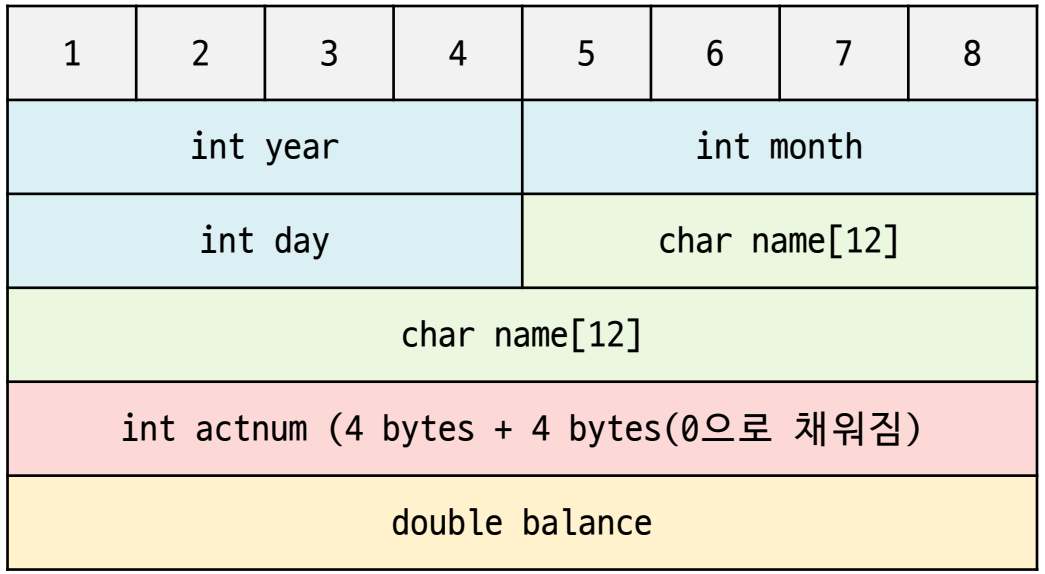
date 멤버에 접근하기 위해 2번 참조

실행 결과

```
구조체 크기: 40
[2022. 3. 9]
Hong 1001 2000.00
```

구조체의 크기 (참고)

- **구조체 정렬: 실제 구조체 크기 = 40 bytes**
  - 가장 큰 데이터 타입인 double을 기준으로 정렬




```
구조체 크기: 40
```

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

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

//은행계좌를 위한 구조체
struct account
{
    struct date open; //계좌 개설일자
    char name[12]; //계좌주 이름
    int actnum; //계좌번호
    double balance; //잔고
};

. . .
struct account me = {{2022, 3, 9}, "Hong",
                     1001, 2000};
printf("구조체 크기: %zu\n", sizeof(me));
```

12 bytes =
4 bytes x 3개

36 bytes =
12 bytes + 12 bytes
+ 4 bytes + 8 bytes

구조체 정렬 (참고)

- **구조체 정렬 방식**
  - 구조체 내부에서 가장 큰 데이터 형을 기준으로 데이터 필드 정렬

```
struct packet1 {
    char aa;  // 1 byte
    int bb;   // 4 bytes
    char cc;  // 1 bytes
};
```

int형(4bytes) 기준으로 구조체 정렬

1	2	3	4
aa(1)	Padding		
bb(1)	bb(2)	bb(3)	bb(4)
cc(1)	Padding		

sizeof(packet1) = 12 bytes

```
struct packet2 {
    char id;     // 1 byte
    double cnt;  // 8 bytes
};
```

double형(8bytes) 기준으로 구조체 정렬

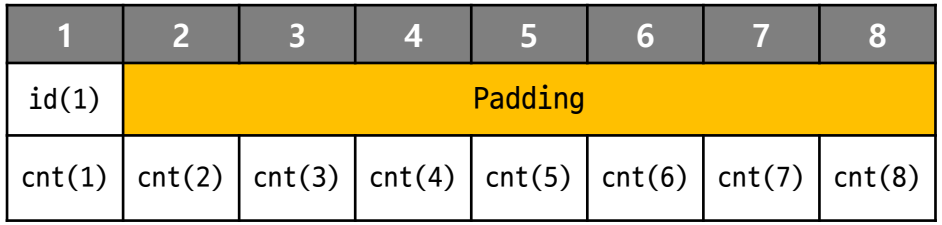
1	2	3	4	5	6	7	8
id(1)	Padding						
cnt(1)	cnt(2)	cnt(3)	cnt(4)	cnt(5)	cnt(6)	cnt(7)	cnt(8)

sizeof(packet2) = 16 bytes

구조체 정렬 (참고)

```c
#include <stdio.h>

struct packet1 {
    char aa; // 1 byte
    int bb; // 4 bytes
    char cc; // 1 bytes
};

struct packet2 {
    char id; // 1 byte
    double cnt; // 8 bytes
};

int main()
{
    struct packet1 p1;
    struct packet2 p2;

    printf("sizeof p1: %zu\n", sizeof(p1));
    printf("sizeof p2: %zu\n", sizeof(p2));

    return 0;
}
```

sizeof(packet1) = 12 bytes



sizeof(packet2) = 16 bytes

```
sizeof p1: 12
sizeof p2: 16
```

구조체 정의 위치

- **구조체 정의는 그 정의 위치에 따라 구조체의 유효 범위가 결정**
  - 구조체의 정의도 변수 선언처럼 유효 범위는 전역(global) 또는 지역(local)

  - 전역
    - ➢ main() 함수 외부 상단에서 정의된 구조체
    - ➢ 일반적인 구조체 정의 방법

  - 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 가능**하다. 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.

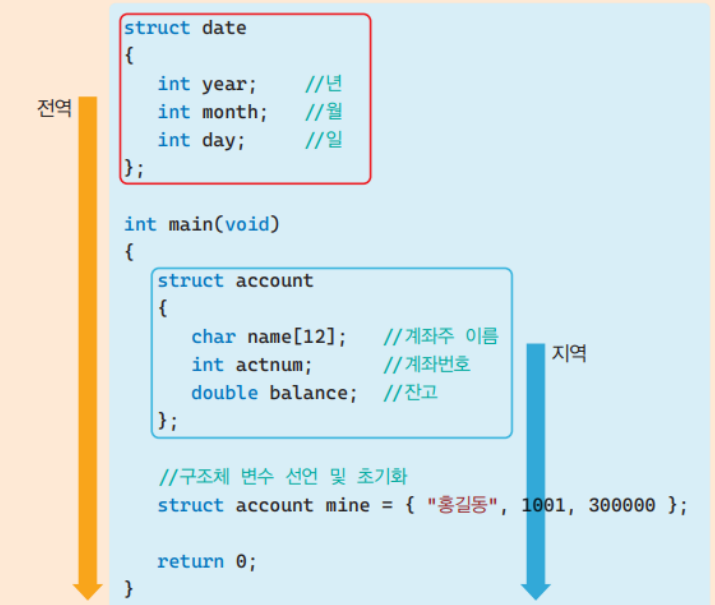


그림 13-14 구조체 정의의 유효 범위

구조체 변수의 대입과 동등 비교

- **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, "Computer Engineering", "HongGilDong"};
struct student one = hong;
```

동일한 구조체 변수인 경우, 대입 가능

- **구조체의 동등 비교: 비교는 불가능**
  - 구조체 비교: 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");
```


18

구조체 정의와 비교 예제 (실습)

<03structcmp.c>

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

// 학생을 위한 구조체
struct student
{
    int snum; // 학번
    char *dept; // 학과 이름
    char name[12]; // 학생 이름
};

int main(void)
{
    struct student hong = {202200001, "Computer Engineering",
                           "Hong"};
    struct student na = {202200002};
    struct student you = {202200003};

    // 학생이름 입력
    printf("Type a student's name: ");
    scanf("%s", na.name);
    // na.name = "Charlie"; //컴파일 오류
    // scanf("%s", na.dept); //실행 오류

    na.dept = "Electronic Engineering";
    you.dept = "Mechanical Engineering";
    memcpy(you.name, "Alice", 6);
    strcpy(you.name, "Alice");

    printf("[%d, %s, %s]\n", hong.snum, hong.dept, hong.name);
    printf("[%d, %s, %s]\n", na.snum, na.dept, na.name);
    printf("[%d, %s, %s]\n\n", you.snum, you.dept, you.name);

    struct student one;
    one = you; // 대입 연산자 가능
    if (one.snum == you.snum)
        printf("학번이 %d로 동일합니다.\n", one.snum);
    // if ( one == bae ) //컴파일 오류

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

    return 0;
}
```

char*로 문자열 상수의 주소 저장은 가능함
- scanf(), memcpy(), strcpy()로는 문자열 저장이 불가능

구조체 멤버 char 배열에 값 저장:
strcpy(), memcpy() 사용

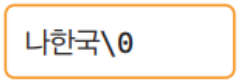
```
Type a student's name: Bob
[202200001, Computer Engineering, Hong]
[202200002, Electronic Engineering, Bob]
[202200003, Mechanical Engineering, Alice]

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 배열**
  - 문자열을 구성하는 모든 문자를 하나 하나를 저장
  - 마지막에 '\0' 문자를 저장하여 사용

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

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

정상 사용
`dept = "컴퓨터정보공학과";`
	정상 사용
	`strcpy(name, "배상문");`
	`scanf("%s", name);`

오류 발생
`strcpy(dept, "컴퓨터정보공학과"); //오류`
`scanf("%s", dept); //오류`
	오류 발생
	`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

구조체 멤버 변경 (실습)

<04struct_string.c>

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

struct book
{
    char title[100]; // 책 제목
    char author[50]; // 저자명
    int year;        // 출판년도
    int price;       // 가격
    char genre[30];  // 장르
};
```

```c
int main()
{
    struct book book1 = {"C Programming",
                         "Kernighan",
                         1978, 29000,
                         "Programming"};

    // 구조체 멤버 수정
    strcpy(book1.title, "The C Programming Language");
    strcpy(book1.author, "Kernighan & Richie");
    book1.price = 30000;
    strcpy(book1.genre, "Programming");

    // 출력
    printf("책 제목: %s\n", book1.title);
    printf("저자명: %s\n", book1.author);
    printf("출판년도: %d\n", book1.year);
    printf("가격: %d\n", book1.price);
    printf("장르: %s\n", book1.genre);

    return 0;
}
```

```
책 제목: The C Programming Language
저자명: Kernighan & Richie
출판년도: 1978
가격: 30000
장르: Programming
```

Questions?