구조체

- Chapter 10 -

학습목차

I. 구조체 개요

II. 구조체 포인터

III. 구조체 멤버 정렬

IV. 구조체 배열

구조체 개요

구조체의 필요성

- 지금까지는 변수를 자료형 별로 하나씩 선언해서 사용했음
- 인적 정보를 처리한다면 한 사람의 이름, 나이, 주소, 등을 저장할 변수가 필요

```
char name[20];     // 이름
int age;           // 나이
char address[100]; // 주소
```

- 위 변수에는 한 사람의 정보만 저장할 수 있으며 여러 명의 정보를 저장하려면 변수를 계속 추가해야 함
  - 인원이 늘어날 수록 복잡하고 비효율적

```
char name1[20];
char name2[20];
...
char name100[20];

int age1;
int age2;
...
int age100;
```

```
char address1[100];
char address2[100];
...
char address100[100];
```

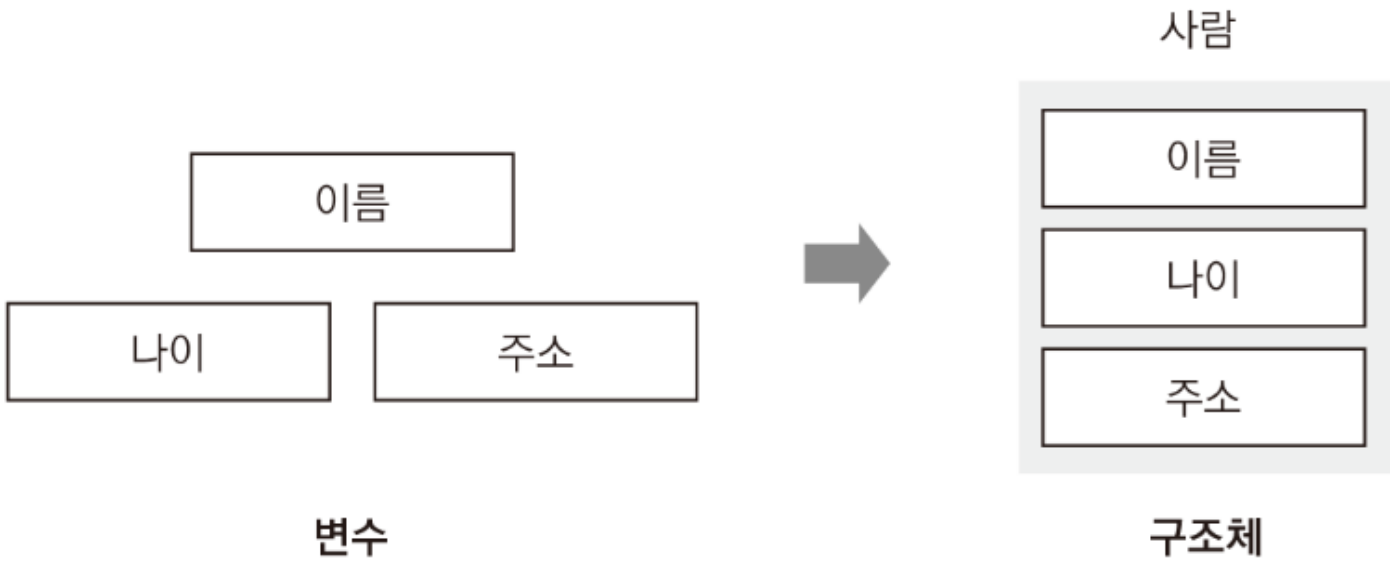
구조체 개요

- 구조체의 필요성
  - C언어는 자료를 체계적으로 관리하기 위해 구조체를 제공
  - 구조체는 struct로 정의
    - data structure(자료 구조)의 약어
    - 인적 정보를 구조체로 표현하면 효율적
      - Person이라는 구조체 내부에 이름, 나이, 주소 정보를 저장

```
struct Person {
    char name[20];          // 이름
    int age;                // 나이
    char address[100];      // 주소
};
```

구조체 개요

- 변수와 구조체
  - 구조체도 배열로 만들 수 있음
    - Person personArr[100];
  - 구조체는 관련 정보를 하나의 의미로 묶을 때 사용



구조체 개요

- 구조체 사용 방법(1)
  - 구조체는 struct 키워드로 정의
    - struct 키워드 뒤에 구조체 이름을 지정해주고 { } (중괄호) 안에 변수를 선언
    - 구조체 안에 들어있는 변수를 멤버라고 함
    - 구조체를 정의할 때 } (닫는 중괄호) 뒤에는 반드시 세미콜론;을 붙임
  - 구조체를 정의한 후 변수로 선언해서 사용

```
struct 구조체이름 {
    자료형 멤버이름;
};
struct 구조체이름 변수이름;
```

```
struct Person {    // 구조체 정의
    char name[20];        // 구조체 멤버 1
    int age;              // 구조체 멤버 2
    char address[100];    // 구조체 멤버 3
};
struct Person p1;    // 구조체 변수 선언
```

구조체 개요

구조체 사용 방법(1)

구조체 정의 후 선언하는 예제

```
#define _CRT_SECURE_NO_WARNINGS    // strcpy 보안 경고로 인한 컴파일 에러 방지
#include <stdio.h>
#include <string.h>    // strcpy 함수가 선언된 헤더 파일
struct Person {        // 구조체 정의
    char name[20];         // 구조체 멤버 1
    int age;               // 구조체 멤버 2
    char address[100];     // 구조체 멤버 3
};

int main(){
    struct Person p1;      // 구조체 변수 선언
    // 점으로 구조체 멤버에 접근하여 값 할당
    strcpy(p1.name, "홍길동");
    p1.age = 30;
    strcpy(p1.address, "서울시 용산구 한남동"); //문자열 멤버는 = (할당 연산자)로 저장할 수 없으므로 strcpy 함수를 사용
    // 점으로 구조체 멤버에 접근하여 값 출력
    printf("이름: %s\n", p1.name);        // 이름: 홍길동
    printf("나이: %d\n", p1.age);         // 나이: 30
    printf("주소: %s\n", p1.address);     // 주소: 서울시 용산구 한남동
    return 0;
}
```

```
이름: 홍길동
나이: 30
주소: 서울시 용산구 한남동
```

구조체 개요

- 구조체 사용 방법(2)
  - 앞서 설명한 방식은 구조체의 정의와 선언을 분리하여 사용하는 경우
  - 세미콜론 사이에 변수를 지정해주면 구조체를 정의하는 동시에 변수를 선언할 수 있음

```
struct 구조체이름 {
    자료형 멤버이름;
} 변수;
```

구조체 개요

▶구조체 사용 방법(2)

```
#define _CRT_SECURE_NO_WARNINGS    // strcpy 보안 경고로 인한 컴파일 에러 방지
#include <stdio.h>
#include <string.h>    // strcpy 함수가 선언된 헤더 파일
struct Person {    // 구조체 정의
    char name[20];        // 구조체 멤버 1
    int age;              // 구조체 멤버 2
    char address[100];    // 구조체 멤버 3
} p1;                // 구조체를 정의하는 동시에 변수 p1 선언 / 전역변수
int main(){
    // 점으로 구조체 멤버에 접근하여 값 할당
    strcpy(p1.name, "홍길동");
    p1.age = 30;
    strcpy(p1.address, "서울시 용산구 한남동");
    // 점으로 구조체 멤버에 접근하여 값 출력
    printf("이름: %s\n", p1.name);       // 이름: 홍길동
    printf("나이: %d\n", p1.age);        // 나이: 30
    printf("주소: %s\n", p1.address);    // 주소: 서울시 용산구 한남동
    return 0;
}
```

```
이름: 홍길동
나이: 30
주소: 서울시 용산구 한남동
```

구조체 개요

- 구조체 변수를 선언하는 동시에 초기화
  - 구조체 변수를 선언하는 동시에 값을 초기화하려면 중괄호 안에 .(점)과 멤버 이름을 적고 값을 할당

```
struct 구조체이름 변수이름 = { .멤버이름1 = 값1, .멤버이름2 = 값2, … };
```

  - 멤버 이름과 할당 연산자 없이 값만 콤마로 구분하여 나열하여 초기화
    - 구조체 멤버가 선언된 순서대로 입력
    - 각 멤버의 자료형에 맞게 데이터를 입력
      - 처음부터 순서대로 값을 채워 넣어야 하며 중간에 있는 멤버만 값을 할당하거나, 중간에 있는 멤버만 생략할 수 없음

```
struct 구조체이름 변수이름 = { 값1, 값2, … };
```

구조체 개요

구조체 변수를 선언하는 동시에 초기화

예제

```
#include <stdio.h>
struct Person {
    char name[20];
    int age;
    char address[100];
};
int main(){
    // name에는 "홍길동", age에는 30, address에는 "서울시 용산구 한남동"
    struct Person p1 = { .name = "홍길동", .age = 30, .address = "서울시 용산구 한남동" };
    printf("이름: %s\n", p1.name);        // 이름: 홍길동
    printf("나이: %d\n", p1.age);         // 나이: 30
    printf("주소: %s\n", p1.address);     // 주소: 서울시 용산구 한남동
    // name에는 "고길동", age에는 40, address에는 "서울시 서초구 반포동"
    struct Person p2 = { "고길동", 40, "서울시 서초구 반포동" };
    printf("이름: %s\n", p2.name);        // 이름: 고길동
    printf("나이: %d\n", p2.age);         // 나이: 40
    printf("주소: %s\n", p2.address);     // 주소: 서울시 서초구 반포동
    return 0;
}
```

```
이름: 홍길동
나이: 30
주소: 서울시 용산구 한남동
이름: 고길동
나이: 40
주소: 서울시 서초구 반포동
```

구조체 개요

- typedef로 struct 키워드 없이 구조체 선언
  - 구조체 변수를 선언할 때 struct 키워드를 생략가능

```
typedef struct 구조체이름 {
    자료형 멤버이름;
} 구조체별칭;
```

  - typedef로 구조체를 정의하면서 별칭(alias)을 지정
  - 관례상 구조체 이름은 _구조체이름처럼 앞에 _를 붙임
  - 구조체별칭 변수이름;

구조체 개요

- typedef로 struct 키워드를 생략하고 구조체를 선언하는 방법
  - 구조체 변수를 선언할 때 일일이 struct 키워드를 사용하면 번거로움
  - typedef로 구조체를 정의하면서 별칭(alias)을 지정 가능
    - 구조체 이름과 구조체 별칭은 겹쳐도 상관없음

```
typedef struct 구조체이름 {
    자료형 멤버이름;
} 구조체별칭;
구조체별칭 변수이름;
```

구조체 개요

- typedef로 struct 키워드를 생략하고 구조체를 선언하는 방법
  - 예제

```
#define _CRT_SECURE_NO_WARNINGS    // strcpy 보안 경고로 인한 컴파일 에러 방지
#include <stdio.h>
#include <string.h>    // strcpy 함수가 선언된 헤더 파일
typedef struct _Person {   // 구조체 이름은 _Person / 별명과 동일하게 Person으로 지정하여도 가능
    char name[20];             // 구조체 멤버 1
    int age;                   // 구조체 멤버 2
    char address[100];         // 구조체 멤버 3
} Person;                  // typedef로 구조체 별칭을 Person으로 정의 / 구조체 명과 동일하게 _Person으로 지정 가능

int main(){
    Person p1;    // 구조체 별칭 Person으로 변수 선언 / struct _Person p1;과 Person p1;은 완전히 같음
    // 점으로 구조체 멤버에 접근하여 값 할당
    strcpy(p1.name, "홍길동");
    p1.age = 30;
    strcpy(p1.address, "서울시 용산구 한남동");
    // 점으로 구조체 멤버에 접근하여 값 출력
    printf("이름: %s\n", p1.name);       // 이름: 홍길동
    printf("나이: %d\n", p1.age);        // 나이: 30
    printf("주소: %s\n", p1.address);    // 주소: 서울시 용산구 한남동
    return 0;
}
```

```
이름: 홍길동
나이: 30
주소: 서울시 용산구 한남동
```

구조체 개요

- typedef 활용
  - typedef는 자료형의 별칭을 만드는 기능
    - 구조체뿐만 아니라 모든 자료형의 별칭을 만들 수 있음

```
typedef 자료형 별칭
typedef 자료형* 별칭
```

구조체 개요

- int를 별칭 MYINT, int 포인터를 별칭 PMYINT로 정의하는 예제
  - 보통 포인터 별칭은 포인터라는 의미로 앞에 P를 붙임

```
typedef int MYINT;      // int를 별칭 MYINT로 정의
typedef int* PMYINT;    // int 포인터를 별칭 PMYINT로 정의

MYINT num1;         // MYINT로 변수 선언
PMYINT numPtr1;     // PMYINT로 포인터 변수 선언

numPtr1 = &num1;    // 포인터에 변수의 주소 저장
                    // 사용 방법은 일반 변수, 포인터와 같음
```

  - typedef로 정의한 별칭을 사용자 정의 자료형, 사용자 정의 타입이라고 지칭
  - PMYINT는 안에 *가 이미 포함되어 있으므로 포인터 변수를 선언할 때 *를 붙여버리면 이중 포인터가 되므로 사용에 주의

```
PMYINT *numPtr1;    // PMYINT에는 *가 이미 포함되어 있어서 이중 포인터가 선언됨
int* *numPtr2;      // PMYINT *와 같은 의미. 이중 포인터
```

구조체 개요

구조체 태그

- struct 뒤에 붙는 구조체 이름을 구조체 태그(tag)라고 부름
- typedef로 정의한 구조체 별칭은 사용자 정의 타입 이름

```
struct 태그 {
    자료형 멤버이름;
};

typedef struct 태그 {
    자료형 멤버이름;
} 타입이름;
```

- 구조체 태그와 타입 이름을 구분하기 위해 관례상 태그 앞에 _, tag_, tag를 붙임
  - 코드에 따라서 태그 뒤에 _t를 붙이는 경우도 있음
  - 예) _Person, tag_Person, tagPerson, Person_t
- 구조체 태그와 타입 이름은 사실상 같은 구조체를 지칭하므로 이름을 완전히 다르게 명시할 필요는 없음
  - 구조체 태그와 타입 이름을 똑같이 지정할 수 있음

구조체 개요

- 익명 구조체 사용
  - typedef 구조체 별칭을 정의할 때 매번 구조체 이름을 지정하는 경우 번거로움
  - 익명 구조체(anonymous structure)를 사용하면 구조체 이름을 생략 가능
    - typedef로 구조체를 정의하면서 이름(태그)를 지정하지 않음

```
typedef struct {
    자료형 멤버이름
} 구조체별칭;
구조체별칭 변수이름;
```

구조체 개요

▶ 익명 구조체 사용 예제

```
#define _CRT_SECURE_NO_WARNINGS    // strcpy 보안 경고로 인한 컴파일 에러 방지
#include <stdio.h>
#include <string.h>    // strcpy 함수가 선언된 헤더 파일

typedef struct {   // 구조체 이름이 없는 익명 구조체
    char name[20];        // 구조체 멤버 1
    int age;              // 구조체 멤버 2
    char address[100];    // 구조체 멤버 3
} Person;          // typedef를 사용하여 구조체 별칭을 Person으로 정의 / 별칭은 반드시 정의

int main(){
    Person p1;    // 구조체 별칭 Person으로 변수 선언
    // 점으로 구조체 멤버에 접근하여 값 할당
    strcpy(p1.name, "홍길동");
    p1.age = 30;
    strcpy(p1.address, "서울시 용산구 한남동");
    // 점으로 구조체 멤버에 접근하여 값 출력
    printf("이름: %s\n", p1.name);       // 이름: 홍길동
    printf("나이: %d\n", p1.age);        // 나이: 30
    printf("주소: %s\n", p1.address);    // 주소: 서울시 용산구 한남동
    return 0;
}
```

```
이름: 홍길동
나이: 30
주소: 서울시 용산구 한남동
```

확인문제

▶다음 중 구조체를 정의할 때 사용하는 키워드를 고르시오.

1. structure
2. for
3. struct
4. include
5. typedef

확인문제

▶ 다음 코드에서 구조체를 정의할 때 필요한 코드의 조합을 순서에 맞게 고르시오.

```
____ Person {
    char name[20];
    int age;
    char address[100];
____
```

1. struct, }
2. structure, }
3. struct, };
4. struct,);
5. typedef, ;

확인문제

- 다음과 같이 구조체 이름 없이 정의하는 구조체는 무엇인가?

```
typedef struct {
    char name[20];
    int age;
    char address[100];
} Person;
```

확인문제

▶ 다음 중 구조체 별칭을 정의할 때 사용하는 키워드는?

```
____ Person {
    char name[20];
    int age;
    char address[100];
____
```

1. include
2. case
3. struct
4. typedef
5. switch

실습문제 01

좌표 구조체 정의하기

- 다음 소스 코드를 완성하여 2차원 좌표 x, y를 표현하는 구조체 Point2D를 정의하고, 10 20이 출력되도록 완성하시오.
- 좌표의 자료형은 int로 설정

```
#include <stdio.h>

①
struct_____________
_____________________
_____________________
_____________________

int main()
{
    ②_____________ p1;

    ③_____________
    p1.y = 20;

    printf("%d %d\n", p1.x, p1.y);

    return 0;
}
```

```
10 20
```

실습문제 02

- typedef로 좌표 구조체 정의하기
  - 다음 소스 코드를 완성하여 2차원 좌표 x, y를 표현하는 구조체 Point2D를 정의하고, 10 20이 출력되도록 완성하시오.
  - 좌표의 자료형은 int로 설정

```
#include <stdio.h>

①
typedef_______________
_____________________
_____________________
_____________________

int main()
{
    Point2D ②_____;

    p1.x = 10;
    ③_______________

    printf("%d %d\n", p1.x, p1.y);

    return 0;
}
```

```
10 20
```

실습문제 03

익명 구조체로 좌표 구조체 정의하기

다음 소스 코드를 완성하여 10 20이 출력되도록 하시오.

```
#include <stdio.h>

typedef struct {
    int x;
    int y;
_____________

int main()
{
    Point2D p1;

    p1.x = 10;
    p1.y = 20;

    printf("%d %d\n", p1.x, p1.y);

    return 0;
}
```

```
10 20
```

실습문제 04

계기판 정보 출력

- 자동차에서 속도, 연료, 주행거리, 엔진 온도, 회전수를 표현하는 계기판 구조체가 정의되어 있다.
- 다음 소스 코드를 완성하여 계기판 정보가 출력되도록 하시오.

```
#include <stdio.h>

struct Dashboard {
    int speed;
    char fuel;
    float mileage;
    int engineTemp;
    int rpm;
};
int main(){
    ________________

    printf("Speed: %dkm/h\n", d1.speed);
    printf("Fuel: %c\n", d1.fuel);
    printf("Mileage: %fkm\n", d1.mileage);
    printf("Engine temp: %d℃\n", d1.engineTemp);
    printf("RPM: %d\n", d1.rpm);
    return 0;
}
```

```
Speed: 80km/h
Fuel: F
Mileage: 5821.442871km
Engine temp: 200℃
RPM: 1830
```

구조체 포인터

- 구조체 포인터의 필요성
  - 구조체는 여러 멤버 변수를 포함하므로 일반적인 변수보다 크기가 큼
  - 다른 자료형과 마찬가지로 구조체도 포인터를 선언할 수 있음
  - malloc 함수를 사용하여 동적 메모리를 할당 가능

```
struct 구조체이름 *포인터이름 = malloc(sizeof(struct 구조체이름));
```

구조체 포인터

구조체 포인터를 선언하고 메모리를 할당하는 예제

구조체 포인터의 멤버에 접근할 때는 -> (화살표 연산자)를 사용

```
#define _CRT_SECURE_NO_WARNINGS    // strcpy 보안 경고로 인한 컴파일 에러 방지
#include <stdio.h>
#include <string.h>    // strcpy 함수가 선언된 헤더 파일
#include <stdlib.h>    // malloc, free 함수가 선언된 헤더 파일
struct Person {    // 구조체 정의
    char name[20];        // 구조체 멤버 1
    int age;              // 구조체 멤버 2
    char address[100];    // 구조체 멤버 3
};
int main(){
    struct Person *p1 = malloc(sizeof(struct Person));    // 구조체 포인터 선언, 메모리 할당
    // 화살표 연산자로 구조체 멤버에 접근하여 값 할당
    strcpy(p1->name, "홍길동"); //문자열 멤버는 = (할당 연산자)로 저장할 수 없으므로 strcpy 함수를 사용
    p1->age = 30;
    strcpy(p1->address, "서울시 용산구 한남동");
    // 화살표 연산자로 구조체 멤버에 접근하여 값 출력
    printf("이름: %s\n", p1->name);       // 홍길동 / == (*p1).age;
    printf("나이: %d\n", p1->age);        // 30
    printf("주소: %s\n", p1->address);    // 서울시 용산구 한남동
    free(p1);    // 동적 메모리 해제
    return 0;
}
```

```
이름: 홍길동
나이: 30
주소: 서울시 용산구 한남동
```

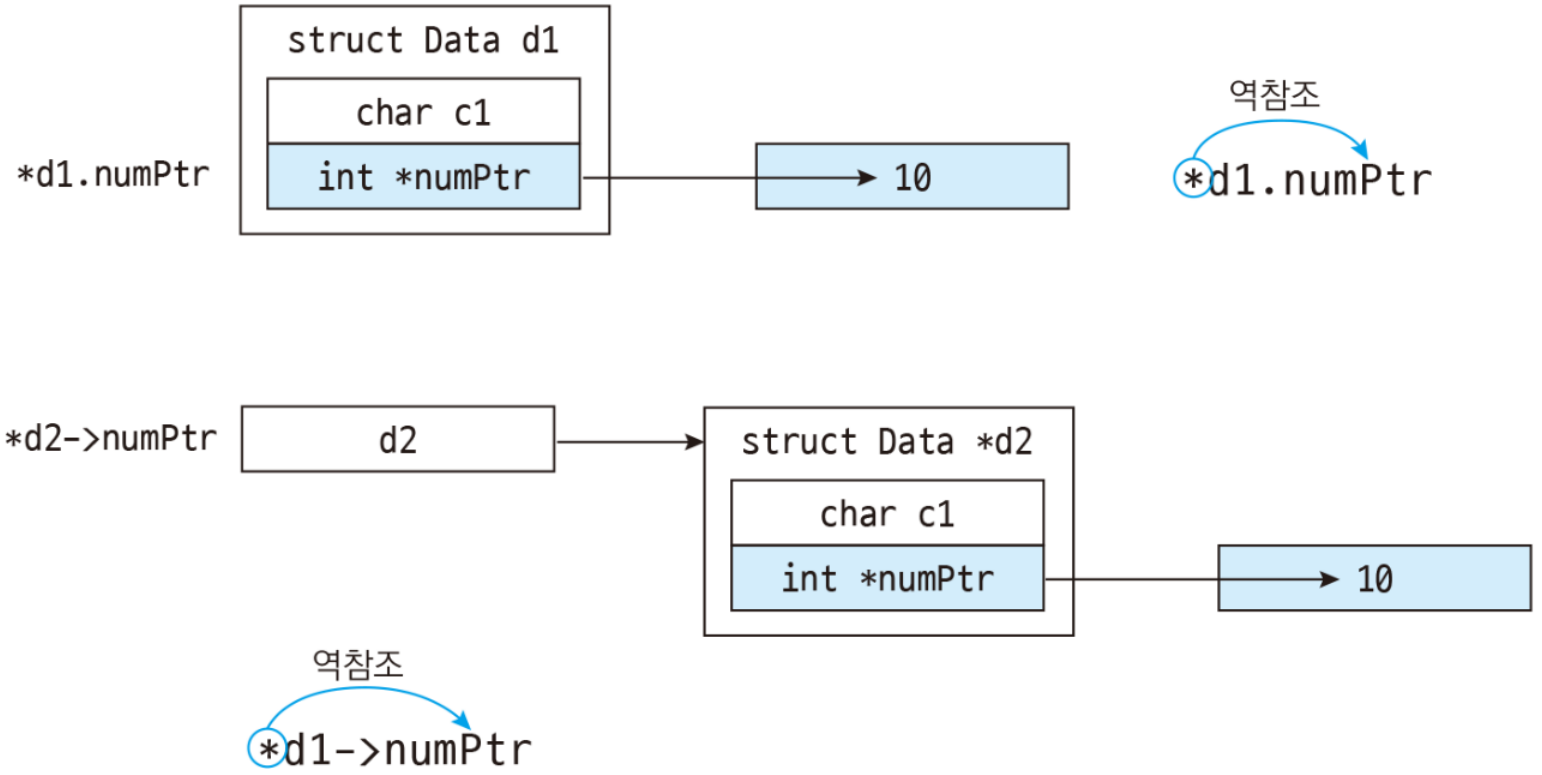
구조체 포인터

- 구조체의 멤버가 포인터일 때 역참조
  - *구조체변수.멤버
  - *구조체포인터->멤버

```
#include <stdio.h>
#include <stdlib.h>
struct Data {
    char c1;
    int *numPtr;    // 포인터
};
int main(){
    int num1 = 10;
    struct Data d1;    // 구조체 변수
    struct Data *d2 = malloc(sizeof(struct Data));    // 구조체 포인터에 메모리 할당
    d1.numPtr = &num1;
    d2->numPtr = &num1;
    printf("%d\n", *d1.numPtr);     // 10: 구조체의 멤버를 역참조
    printf("%d\n", *d2->numPtr);    // 10: 구조체 포인터의 멤버를 역참조
    d2->c1 = 'a';
    printf("%c\n", (*d2).c1);      //  a: 구조체 포인터를 역참조하여 c1에 접근// d2->c1과 같음
    printf("%d\n", *(*d2).numPtr); // 10: 구조체 포인터를 역참조하여 numPtr에 접근한 뒤 다시 역참조//*d2->numPtr과 같음
    free(d2);
    return 0;
}
```

구조체 포인터

- 구조체 멤버가 포인터일 경우 역참조하는 방법
  - 구조체의 멤버가 포인터일 때 역참조를 하려면 맨 앞에 *를 붙임
  - 구조체 변수 앞에 *가 붙어있더라도 멤버의 역참조이지 구조체 변수의 역참조가 아님



구조체 포인터

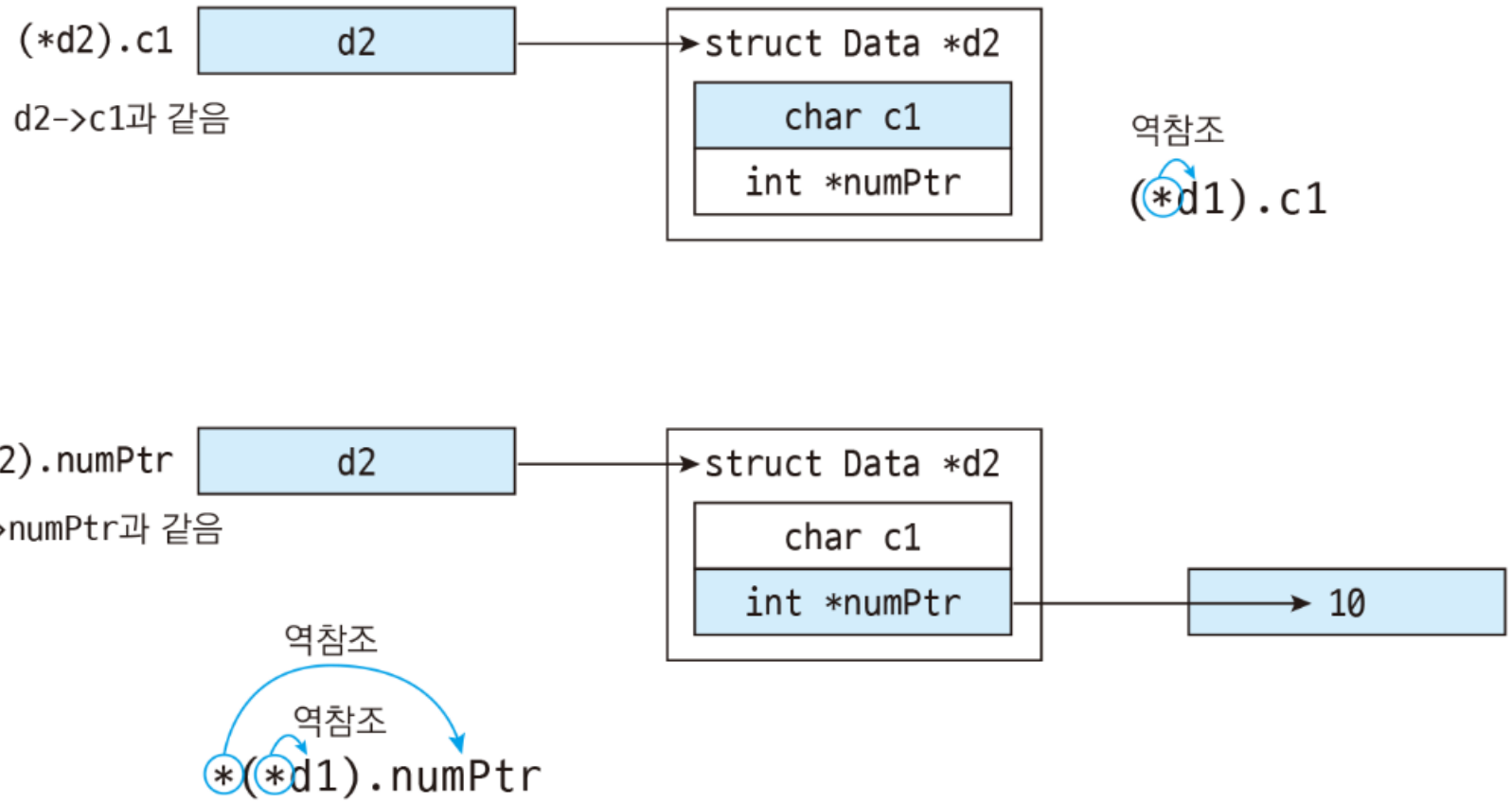
- 구조체 포인터를 역참조한 뒤 괄호로 묶기
  - 구조체 변수를 역참조한 뒤 멤버에 접근
  - (*구조체포인터).멤버
  - *(*구조체포인터).멤버

```
d2->c1 = 'a';
printf("%c\n", (*d2).c1);       //  a: 구조체 포인터를 역참조하여 c1에 접근
                                // d2->c1과 같음
printf("%d\n", *(*d2).numPtr); // 10: 구조체 포인터를 역참조하여 numPtr에 접근한 뒤 다시 역참조
                                // *d2->numPtr과 같음
```

  - (*d2).c1 == d2->c1
  - (*d2).numPtr == *d2->numPtr
  - 구조체 포인터를 역참조한 뒤 괄호로 묶으면 -> 연산자에서 . 연산자를 사용하게 되므로 포인터가 일반 변수로 바뀜

구조체 포인터

▶구조체 포인터를 역참조한 뒤 괄호로 묶기



구조체 포인터

- 구조체 별칭으로 포인터를 선언하고 메모리 할당
  - struct 키워드로 포인터를 선언하고 메모리를 할당 가능
  - typedef로 정의한 구조체 별칭으로도 포인터를 선언하고 메모리를 할당 가능

```
구조체별칭 *포인터이름 = malloc(sizeof(구조체별칭));
```

```
Person *p1 = malloc(sizeof(Person));    // 구조체 별칭으로 포인터 선언, 메모리 할당
```

  - 구조체 별칭으로 선언한 포인터도 구조체 멤버에 접근할 때는 -> (화살표 연산자)를 사용

```
// 화살표 연산자로 구조체 멤버에 접근하여 값 할당
strcpy(p1->name, "홍길동");
p1->age = 30;
strcpy(p1->address, "서울시 용산구 한남동");

// 화살표 연산자로 구조체 멤버에 접근하여 값 출력
printf("이름: %s\n", p1->name);        // 홍길동
printf("나이: %d\n", p1->age);         // 30
printf("주소: %s\n", p1->address);     // 서울시 용산구 한남동
```

구조체 포인터

▶구조체 별칭으로 포인터를 선언하고 메모리 할당하는 예제

```
#define _CRT_SECURE_NO_WARNINGS    // strcpy 보안 경고로 인한 컴파일 에러 방지
#include <stdio.h>
#include <string.h>    // strcpy 함수가 선언된 헤더 파일
#include <stdlib.h>    // malloc, free 함수가 선언된 헤더 파일
typedef struct _Person {    // 구조체 이름은 _Person
    char name[20];              // 구조체 멤버 1
    int age;                    // 구조체 멤버 2
    char address[100];          // 구조체 멤버 3
} Person;                   // typedef를 사용하여 구조체 별칭을 Person으로 정의
int main(){
    Person *p1 = malloc(sizeof(Person));    // 구조체 별칭으로 포인터 선언, 메모리 할당
    // 화살표 연산자로 구조체 멤버에 접근하여 값 할당
    strcpy(p1->name, "홍길동");
    p1->age = 30;
    strcpy(p1->address, "서울시 용산구 한남동");
    // 화살표 연산자로 구조체 멤버에 접근하여 값 출력
    printf("이름: %s\n", p1->name);       // 홍길동
    printf("나이: %d\n", p1->age);        // 30
    printf("주소: %s\n", p1->address);    // 서울시 용산구 한남동
    free(p1);    // 동적 메모리 해제
    return 0;
}
```

```
이름: 홍길동
나이: 30
주소: 서울시 용산구 한남동
```

구조체 포인터

▶ 익명 구조체 형식 - 구조체 별칭으로 포인터를 선언하고 메모리 할당하는 예제

```
#define _CRT_SECURE_NO_WARNINGS    // strcpy 보안 경고로 인한 컴파일 에러 방지
#include <stdio.h>
#include <string.h>    // strcpy 함수가 선언된 헤더 파일
#include <stdlib.h>    // malloc, free 함수가 선언된 헤더 파일
typedef struct {    // 구조체 이름이 없는 익명 구조체
    char name[20];          // 구조체 멤버 1
    int age;                // 구조체 멤버 2
    char address[100];      // 구조체 멤버 3
} Person;           // typedef를 사용하여 구조체 별칭을 Person으로 정의
int main(){
    Person *p1 = malloc(sizeof(Person));    // 구조체 별칭으로 포인터 선언, 메모리 할당
    // 화살표 연산자로 구조체 멤버에 접근하여 값 할당
    strcpy(p1->name, "홍길동");
    p1->age = 30;
    strcpy(p1->address, "서울시 용산구 한남동");
    // 화살표 연산자로 구조체 멤버에 접근하여 값 출력
    printf("이름: %s\n", p1->name);        // 홍길동
    printf("나이: %d\n", p1->age);         // 30
    printf("주소: %s\n", p1->address);     // 서울시 용산구 한남동
    free(p1);    // 동적 메모리 해제
    return 0;
}
```

```
이름: 홍길동
나이: 30
주소: 서울시 용산구 한남동
```

구조체 포인터

- 구조체 별칭으로 포인터를 선언하고 메모리 할당
  - 구조체 별칭을 사용하여 포인터를 선언하고 메모리를 할당 가능
  - Person *p1과 같이 구조체 별칭으로 포인터를 바로 선언한 뒤 malloc 함수로 메모리를 할당 가능
    - 이때 할당할 메모리 크기도 sizeof(Person)처럼 구조체 별칭으로 사용 가능

```
#define _CRT_SECURE_NO_WARNINGS    // strcpy 보안 경고로 인한 컴파일 에러 방지
#include <stdio.h>
#include <string.h>    // strcpy 함수가 선언된 헤더 파일
#include <stdlib.h>    // malloc, free 함수가 선언된 헤더 파일
typedef struct _Person {    // 구조체 이름은 _Person
    char name[20];             // 구조체 멤버 1
    int age;                   // 구조체 멤버 2
    char address[100];         // 구조체 멤버 3
} Person;                   // typedef를 사용하여 구조체 별칭을 Person으로 정의
int main(){
    Person *p1 = malloc(sizeof(Person));    // 구조체 별칭으로 포인터 선언, 메모리 할당
    // 화살표 연산자로 구조체 멤버에 접근하여 값 할당
    strcpy(p1->name, "홍길동");
    p1->age = 30;
    strcpy(p1->address, "서울시 용산구 한남동");
    // 화살표 연산자로 구조체 멤버에 접근하여 값 출력
    printf("이름: %s\n", p1->name);       // 홍길동
    printf("나이: %d\n", p1->age);        // 30
    printf("주소: %s\n", p1->address);    // 서울시 용산구 한남동
    free(p1);    // 동적 메모리 해제
    return 0;
}
```

구조체 포인터

구조체 포인터 인수

- 구조체 형 포인터 인수를 받는 함수를 정의할 수 있으며 당연히 호출도 가능
- 앞서 구조체 변수를 인수로 전달할 경우 값이 복사되었는데 구조체 변수의 주소나 구조체 포인터 변수로 넘겨줄 경우 동일한 데이터를 공유할 수 있음
- 만약 호출된 함수(student_print)에서 구조체 멤버 값을 변경한다면 호출한 함수(main)에서 인자로 넘긴 구조체 변수(student1)값도 변경됨

```
void student_print(student * pdata){
   printf("학년 : %d\n", pdata->grade);
   printf("학급 : %d\n", pdata->group);
   printf("이름 : %s\n", pdata->name);
   …
}
void main(){
   student student1;
   …  //멤버 참조 연산자를 이용하여 student1의 멤버에 데이터저장
   student_print(&student1);
}
```

구조체 배열

구조체 배열

- 구조체도 배열로 작성 가능하며 기존의 배열 선언방식과 동일
- 예를 들어 여러 명의 학생 정보를 사용할 경우 학생 정보를 담고 있는 구조체 변수를 한꺼번에 선언가능하며 접근도 편리해짐
  - 10명의 학생 정보를 다룰 경우 각 학생 정보를 저장하는 구조체를 배열[10]로 선언하여 다룸

```
void student_print(student pdata[], int count){
   int i;
   for(i=0; i<count; i++){
       printf("학년 : %d\n", pdata[i]->grade);
       printf("학급 : %d\n", pdata[i]->group);
       printf("이름 : %s\n", pdata[i]->name);
       …
   }
}
void main(){
   student studentData[10];
   …  //구조체 배열에 학생 정보 입력
   student_print(studentData, 10);//배열의 이름은 시작 주소, 10은 크기
}
```

연습문제 1

- 3명의 이름, 나이, 성별을 키보드로 입력 받고 표시하는 프로그램을 작성하시오.
  - 데이터는 구조체로 저장하고 데이터 입력받는 함수와 출력하는 함수를 각각 구현

```
첫번째 사람 : Jason 28 m
두번째 사람 : Ailee 39 f
세번째 사람 : yumi 32 f
jason은 남자이고 28살이다.
Ailee은 여자이고 39살이다.
yumi은 여자이고 32살이다.
```

연습문제 2

- 앞의 연습문제 1을 활용하여 입력한 나이로 오름차순 정렬하시오.
  - 0미만 120초과되는 숫자를 입력할 경우에는 오류 메시지를 출력하여 다시 입력 받도록 함

```
첫번째 사람 : Jason 28 m
두번째 사람 : Ailee 139 f
입력된 나이 확인
두번째 사람 : Ailee 39 f
세번째 사람 : yumi 32 f
jason은 남자이고 28살이다.
yumi은 여자이고 32살이다.
Ailee은 여자이고 39살이다.
```

연습문제 3

- 앞의 연습문제 1을 활용하여 입력한 이름을 비교하여 아스키코드 값에 따라 오름차순으로 출력하시오.
  - 입력되는 모든 이름은 다르다고 가정
  - 아스키 코드 값을 기준으로 오름차순으로 정렬

```
첫번째 사람 : Jason 28 m
두번째 사람 : Ailee 39 f
세번째 사람 : yumi 32 f
Ailee은 여자이고 39살이다.
jason은 남자이고 28살이다.
yumi은 여자이고 32살이다.
```

연습문제 4

- 앞의 연습문제2,3을 활용하여 3명의 정보를 입력한 후 사용자에게 나이와 이름 중 원하는 기준을 선택하게 한 후 오름차순 정렬하시오.

```
첫번째 사람 : Jason 28 m
두번째 사람 : Ailee 39 f
세번째 사람 : yumi 32 f
1, 나이순 2. 이름순?2
Ailee은 여자이고 39살이다.
jason은 남자이고 28살이다.
yumi은 여자이고 32살이다.
```

Q & A