C 프로그래밍 프로젝트

Chap 22. 구조체와 사용자 정의 자료형1

2013.10.10.

오 병 우

컴퓨터공학과

kit 국립금오공과대학교
Kumoh National Institute of Technology

구조체

- 구조체의 정의 (Structure)
  - 하나 이상의 기본 자료형을 기반으로 사용자 정의 자료형(User Defined Data Type)을 만들 수 있는 문법 요소
  - 배열 vs. 구조체
    - 배열: 한 가지 자료형의 집합
    - 구조체: 여러 가지 자료형의 집합

사용자 정의 자료형

```
struct point          // point라는 이름의 구조체 선언
{
    int x;            // 구조체 멤버 int x
    int y;            // 구조체 멤버 int y
};
```

구조체의 정의

```
int xpos;    // 마우스의 x 좌표
int ypos;    // 마우스의 y 좌표
```

마우스의 좌표정보를 저장하고 관리하기 위해서는
x좌표와 y좌표를 저장할 수 있는 두 개의 변수가 필요하다.

xpos와 ypos는 서로 독립된 정보를 표현하지 않고 하나의 정보를 표현한다. 따라서 이 둘은 늘 함께한다.

```
struct point   // point라는 이름의 구조체 정의
{
    int xpos;    // point 구조체를 구성하는 멤버 xpos
    int ypos;    // point 구조체를 구성하는 멤버 ypos
};
```

구조체를 이용해서 xpos와 ypos를 하나로 묶었다.
이 둘을 묶어서 point라는 이름의 새로운 자료형을 정의!

int가 자료형의 이름인것 처럼 point도 자료형의 이름이다.
단, 프로그래머가 정의한 자료형이기에 '사용자 정의 자료형(user defined data type)'이라 한다.

```
struct person
{
    char name[20];      // 이름 저장
    char phoneNum[20];    // 전화번호 저장
    int age;    // 나이 저장
};
```

개인의 이름과 전화번호 나이 정보를
person이라는 구조체 정의를 통해서 묶고 있다.

배열도 구조체의 멤버로 선언이 가능!

구조체 변수의 선언과 접근

포인터는 -> 사용

마침표 (.) 사용

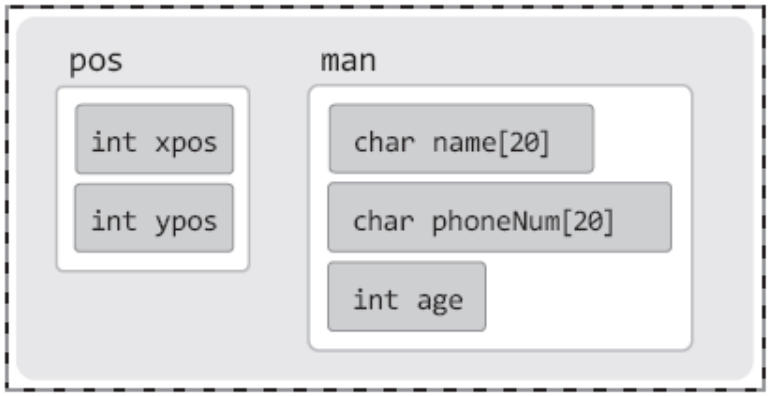
구조체 변수선언의 기본 형태

struct *type_name val_name ;*

struct point pos;

struct person man;

구조체 변수선언의 예



구조체 변수선언의 결과

멤버의 접근방식

구조체 변수의 이름 . 구조체 멤버의 이름

pos.xpos=20;

구조체 변수 pos의 멤버 xpos에 20을 저장

printf("%s \n", man.name);

man의 멤버 name에 저장된 문자열 출력

구조체 변수의 선언과 접근관련 예제1

Main.c

```
#include <stdio.h>
#include <math.h>

struct point {
        int x;
        int y;
};
```

```
int main (void)
{
    struct point p1, p2;
    double distance;

    printf("첫 번째 점의 x, y 좌표 입력: ");
    scanf("%d %d", &p1.x, &p1.y);

    printf("두 번째 점의 x, y 좌표 입력: ");
    scanf("%d %d", &p2.x, &p2.y);

    distance = sqrt ((p1.x-p2.x)*(p1.x-p2.x)
                    + (p1.y-p2.y)*(p1.y-p2.y));

    printf("두 점의 거리는 %5.1lf입니다.\n", distance);

    return 0;
}
```

이 예제에서 호출하는 함수 sqrt는 제곱근을 반환하는 함수로서 헤더파일 math.h에 선언된 수학관련 함수이다.

Point 예제의 모듈화

Point.h

```
// Point.h : Header File of the Point Module
//////////////////////////////////////////
#ifndef _POINT_H_
#define _POINT_H_

struct point {
        int x;
        int y;
};

extern double pointGetDistance( struct point p1,
                                struct point p2);

#endif// _POINT_H_
```

Point.c

```
// Point.c : Implementatin of the Point Module
//////////////////////////////////////////
//#include "stdafx.h“ // Precompiled Header 사용시
#include <math.h>

#include “Point.h”

double pointGetDistance(struct point p1, struct point p2)
{
   return sqrt ((p1.x-p2.x)*(p1.x-p2.x)
                + (p1.y-p2.y)*(p1.y-p2.y));
}
```

Main.c

```
// Main.c : Main Implementation File
//////////////////////////////////////////

//#include "stdafx.h“ // Precompiled Header 사용시


#include <stdio.h>

#include “Point.h”

int main (void)
{
   struct point p1, p2;
   double distance;

   printf("첫 번째 점의 x, y 좌표 입력: “);
   scanf("%d %d", &p1.x, &p1.y);

   printf("두 번째 점의 x, y 좌표 입력: “);
   scanf("%d %d", &p2.x, &p2.y);

   printf("두 점의 거리는 %5.1lf입니다.\n",
                pointGetDistance(p1, p2));

   return 0;
}
```

구조체 변수의 선언과 접근관련 예제2

```
struct person
{
    char name[20];
    char phoneNum[20];
    int age;
};
int main(void)
{
    struct person man1, man2;
    strcpy(man1.name, "안성준");
    strcpy(man1.phoneNum, "010-1122-3344");
    man1.age=23;
    printf("이름 입력: "); scanf("%s", man2.name);
    printf("번호 입력: "); scanf("%s", man2.phoneNum);
    printf("나이 입력: "); scanf("%d", &(man2.age));
    printf("이름: %s \n", man1.name);
    printf("번호: %s \n", man1.phoneNum);
    printf("나이: %d \n", man1.age);
    printf("이름: %s \n", man2.name);
    printf("번호: %s \n", man2.phoneNum);
    printf("나이: %d \n", man2.age);
    return 0;
}
```

구조체의 멤버라 하더라도 일반적인 접근의 방식을 그대로 따른다. 구조체의 멤버로 배열이 선언되면 배열의 접근방식을 취하면 되고, 구조체의 멤버로 포인터 변수가 선언되면 포인터 변수의 접근방식을 취하면 된다.

```
이름 입력: 김수정
번호 입력: 010-0001-0002
나이 입력: 27
이름: 안성준
번호: 010-1122-3344
나이: 23
이름: 김수정
번호: 010-0001-0002
나이: 27
```

실행결과

구조체 정의와 동시에 변수 선언하기

```
struct point
{
     int xpos;
     int ypos;
} pos1, pos2, pos3;
```

point라는 이름의 구조체를 정의함과 동시에
point 구조체의 변수 pos1, pos2, pos3를 선언하는 문장이다.

```
struct point
{
     int xpos;
     int ypos;
};
struct point pos1, pos2, pos3;
```

위와 동일한 결과를 보이는 구조체의 정의와 변수의 선언이다.

구조체를 정의함과 동시에 변수를 선언하는 문장은 잘 사용되지 않는다.

그러나 문법적으로 지원이 되고 또 간혹 사용하는 경우도 있다.

구조체 변수의 초기화

```
struct point
{
    int xpos;
    int ypos;
};
struct person
{
    char name[20];
    char phoneNum[20];
    int age;
};
int main(void)
{
    struct point pos={10, 20};
    struct person man={"이승기", "010-1212-0001", 21};
    printf("%d %d \n", pos.xpos, pos.ypos);
    printf("%s %s %d \n", man.name, man.phoneNum, man.age);
    return 0;
}
```

초기화 방식이 배열과 유사하다.

초기화 할 데이터들을 중괄호 안에 순서대로 나열하면 된다..

실행결과

```
10 20
이승기 010-1212-0001 21
```

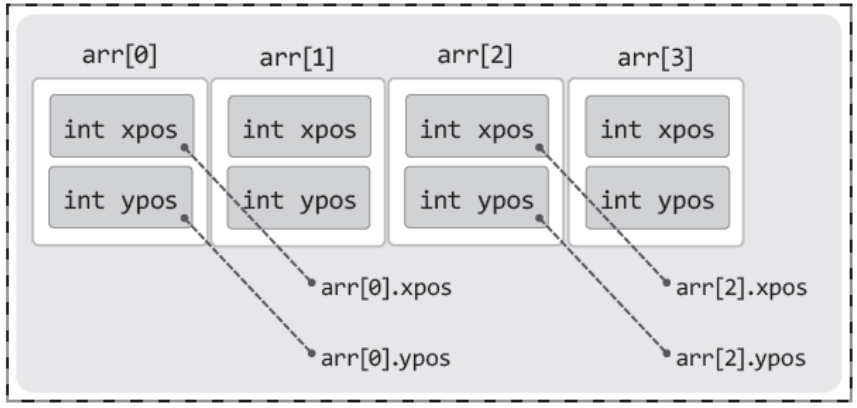
구조체 배열의 선언과 접근

```
struct point
{
    int xpos;
    int ypos;
};

int main(void)
{
    struct point arr[3];
    int i;

    for(i=0; i<3; i++)
    {
        printf("점의 좌표 입력: ");
        scanf("%d %d", &arr[i].xpos, &arr[i].ypos);
    }

    for(i=0; i<3; i++)
        printf("[%d, %d] ", arr[i].xpos, arr[i].ypos);

    return 0;
}
```

```
struct point arr[4];
```

길이가 4인 구조체 배열의 선언방법

선언된 배열의 형태



실행결과

```
점의 좌표 입력: 2 4
점의 좌표 입력: 3 6
점의 좌표 입력: 8 9
[2, 4] [3, 6] [8, 9]
```

구조체 배열의 초기화

```
struct person man={"이승기", "010-1212-0001", 21};
```

구조체 변수의 초기화

구조체 변수 하나를 초기화하기 위해서 하나의 중괄호를 사용하듯이…

```
struct person arr[3]={
    {"이승기", "010-1212-0001", 21},    // 첫 번째 요소의 초기화
    {"정지영", "010-1313-0002", 22},    // 두 번째 요소의 초기화
    {"한지수", "010-1717-0003", 19}     // 세 번째 요소의 초기화
};
```

구조체 배열의 초기화

구조체 배열을 초기화하기 위해서 배열요소 각각의 초기화 값을 중괄호로 묶어서 표현한다.

구조체 배열의 초기화 예제

```
struct person
{
    char name[20];
    char phoneNum[20];
    int age;
};

int main(void)
{
    struct person arr[3]={
        {"이승기", "010-1212-0001", 21},    // 첫 번째 요소의 초기화
        {"정지영", "010-1313-0002", 22},    // 두 번째 요소의 초기화
        {"한지수", "010-1717-0003", 19}     // 세 번째 요소의 초기화
    };

    int i;
    for(i=0; i<3; i++)
        printf("%s %s %d \n", arr[i].name, arr[i].phoneNum, arr[i].age);

    return 0;
}
```

실행결과

```
이승기 010-1212-0001 21
정지영 010-1313-0002 22
한지수 010-1717-0003 19
```

구조체 변수와 포인터

```
struct point pos={11, 12};
struct point * pptr=&pos;
```

구조체 point의 포인터 변수 선언

```
(*pptr).xpos=10;
```

pptr이 가리키는 구조체 변수의 멤버 xpos에 접근

```
(*pptr).ypos=20;
```

pptr이 가리키는 구조체 변수의 멤버 ypos에 접근

구조체 포인터 변수를 대상으로 하는 포인터 연산 및 멤버의 접근방법

```
(*pptr).xpos=10;  ⟷  pptr->xpos=10;

(*pptr).ypos=20;  ⟷  pptr->ypos=20;
```

포인터는 -> 사용

-> 연산자를 기반으로 하는 구조체 변수의 멤버 접근 방법

구조체 변수와 포인터 관련 예제

```
struct point
{
    int xpos;
    int ypos;
};

int main(void)
{
    struct point pos1={1, 2};
    struct point pos2={100, 200};
    struct point * pptr=&pos1;
    (*pptr).xpos += 4;
    (*pptr).ypos += 5;
    printf("[%d, %d] \n", pptr->xpos, pptr->ypos);
    pptr=&pos2;
    pptr->xpos += 1;
    pptr->ypos += 2;
    printf("[%d, %d] \n", (*pptr).xpos, (*pptr).ypos);
    return 0;
}
```

프로그래머들이 주로 사용하는 연산자이니

-> 연산자의 사용에 익숙해지자.

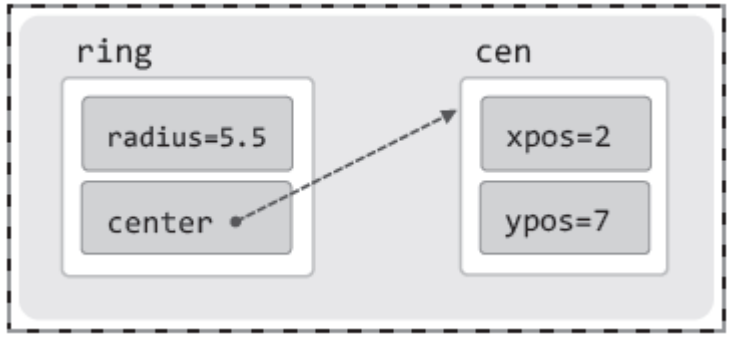
실행결과

```
[5, 7]
[101, 202]
```

포인터 변수를 구조체의 멤버로 선언하기1

```
struct point
{
    int xpos;
    int ypos;
};
struct circle
{
    double radius;
    struct point * center;
};

int main(void)
{
    struct point cen={2, 7};
    double rad=5.5;
    struct circle ring={rad, &cen};
    printf("원의 반지름: %g \n", ring.radius);
    printf("원의 중심 [%d, %d] \n", (ring.center)->xpos, (ring.center)->ypos);
    return 0;
}
```

구조체 변수의 멤버로 구조체 포인터 변수가 선언될 수 있다!



실행결과

```
원의 반지름: 5.5
원의 중심 [2, 7]
```

포인터 변수를 구조체의 멤버로 선언하기2

```
struct point
{
    int xpos;
    int ypos;
    struct point * ptr;
};

int main(void)
{
    struct point pos1={1, 1};
    struct point pos2={2, 2};
    struct point pos3={3, 3};
    pos1.ptr = &pos2;     // pos1과 pos2를 연결
    pos2.ptr = &pos3;     // pos2와 pos3를 연결
    pos3.ptr = &pos1;     // pos3를 pos1과 연결
    printf("점의 연결관계... \n");
    printf("[%d, %d]와(과) [%d, %d] 연결 \n",
        pos1.xpos, pos1.ypos, pos1.ptr->xpos, pos1.ptr->ypos);
    printf("[%d, %d]와(과) [%d, %d] 연결 \n",
        pos2.xpos, pos2.ypos, pos2.ptr->xpos, pos2.ptr->ypos);
    printf("[%d, %d]와(과) [%d, %d] 연결 \n",
        pos3.xpos, pos3.ypos, pos3.ptr->xpos, pos3.ptr->ypos);
    return 0;
}
```

type형 구조체 변수의 멤버로 type형 포인터 변수를 둘 수 있다.

실행결과

```
점의 연결관계...
[1, 1]와(과) [2, 2] 연결
[2, 2]와(과) [3, 3] 연결
[3, 3]와(과) [1, 1] 연결
```

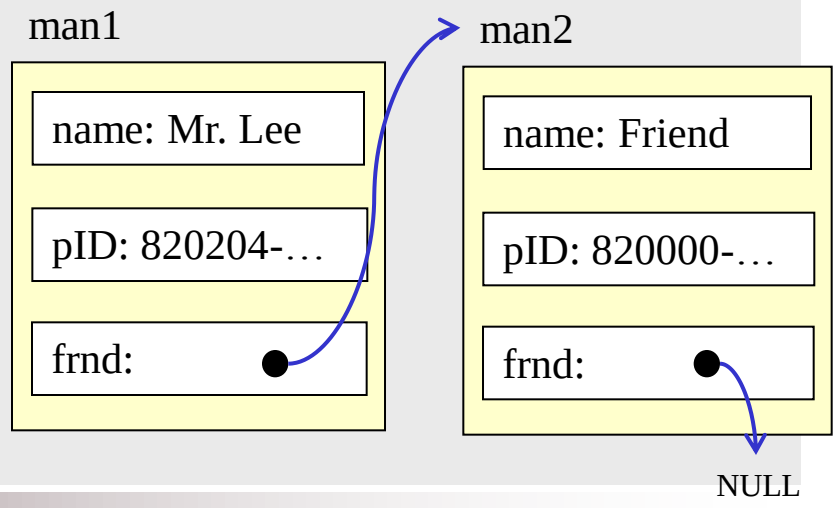
자기 참조형 구조체

Linked List

메모리 동적 할당

```
struct person {
    char name[20];
    char pID[20];
    struct person* frnd;
};

int main()
{
    struct person man1={"Mr. Lee", "820204-0000512"};
    struct person man2={"Mr. Lee's Friend", "820000-0000101"};

    man2.frnd = NULL;
    man1.frnd=&man2;

    printf("[Mr. Lee]\n");
    printf("name : %s\n", man1.name);
    printf("pID  : %s\n", man1.pID);

    printf("[His Friend]\n");
    printf("name : %s\n", man1.frnd->name);
    printf("pID  : %s\n", man1.frnd->pID);
    return 0;
}
```

자기 참조형 구조체



구조체 변수와 첫 번째 멤버의 주소 값

```
struct point
{
    int xpos;
    int ypos;
};
struct person
{
    char name[20];
    char phoneNum[20];
    int age;
};
int main(void)
{
    struct point pos={10, 20};
    struct person man={"이승기", "010-1212-0001", 21};

    printf("%p %p \n", &pos, &pos.xpos);
    printf("%p %p \n", &man, man.name);
    return 0;
}
```

구조체 변수의 주소 값과 구조체 변수의 첫 번째 멤버의 주소 값은 일치한다.

응용 프로그램 분야에서는 이 사실을 이용해서 프로그램을 작성하기도 한다.

실행결과

```
003EF7B8 003EF7B8
003EF784 003EF784
```