프로그래밍 1
Lecture Note #16
(chapter 22)

백윤철
ybaek@smu.ac.kr

내용

- 구조체 정의
- 구조체 변수 접근
- typedef를 이용한 구조체 정의
- 초기화
- 구조체의 배열, 포인터

구조체의 정의

```
int xpos; /* 마우스의 x좌표 */
int ypos; /* 마우스의 y좌표 */
```

- 마우스의 좌표 정보를 저장하고 관리하기 위해서는 x좌표와 y좌표를 저장할 수 있는 두 개의 변수가 필요
- xpos와 ypos는 서로 독립된 정보를 표현하지 않고 한 개의 정보를 표현함. 따라서 이 둘은 늘 함께 함
- 이러한 경우 구조체를 이용해서 한 개의 자료형으로 묶을 수 있음

구조체의 정의

```
struct point {
  int xpos; /* point 구조체를 구성하는 멤버 xpos */
  int ypos; /* point 구조체를 구성하는 멤버 ypos */
};
```

- 구조체를 이용해서 xpos와 ypos를 한 개의 자료형인 point라는 이름으로 묶음
- int가 자료형인 것처럼 point도 자료형의 이름
- 단 프로그래머가 정의한 자료형이기에 '사용자 정의 자료형(user defined data type)'이라고 함

구조체의 정의

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

- 개인의 이름과 전화번호, 나이 정보를 person이라는 구조체 정의를 통해 묶고 있음
- 배열도 구조체의 멤버로 선언 가능

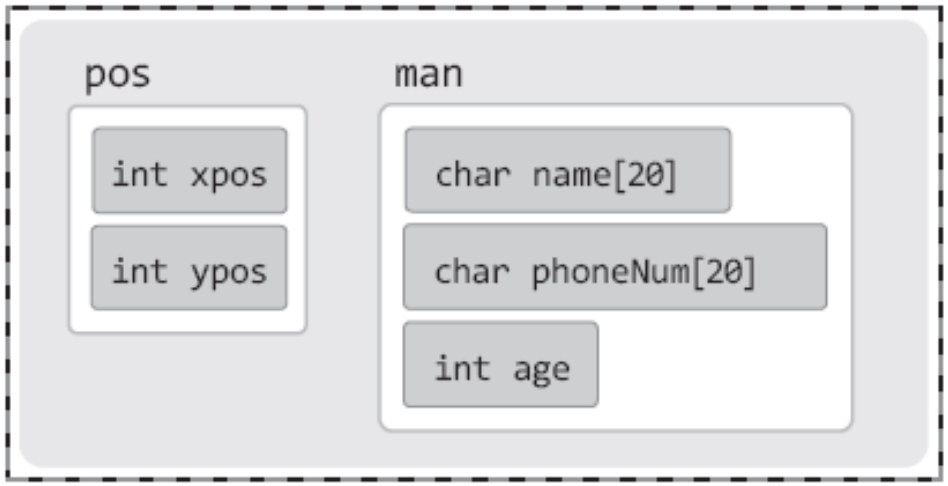
구조체 변수의 정의와 접근

- 구조체 변수 정의의 기본 형태

```
struct  type_name  val_name ;
```

```
struct point pos;
struct person man;
```

구조체 변수선언의 예



구조체 변수의 정의와 접근

- 멤버의 접근 방식

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 예제 1

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

int main(void) {
  struct point pos1, pos2;
  double distance;
  printf("point1 pos: ");
  scanf("%d %d", &pos1.xpos, &pos1.ypos);
  printf("point2 pos: ");
  scanf("%d %d", &pos2.xpos, &pos2.ypos);
```

구조체 변수의 선언과 접근관련 예제 1

```
  /* 두 점간의 거리 계산 공식 */
  distance = sqrt((double)(
  (pos1.xpos-pos2.xpos) * (pos1.xpos-pos2.xpos)
+ (pos1.ypos-pos2.ypos) * (pos1.ypos-pos2.ypos)));
  printf("두 점의 거리는 %g 입니다\n", distance);
  return 0;
}
```

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

실행결과

```
point1 pos: 1 3
point2 pos: 4 5
두 점의 거리는 3.60555 입니다.
```

구조체 변수의 선언과 접근관련 예제 2

```
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;
```

구조체 변수의 선언과 접근관련 예제 2

```
    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;
}
```

실행결과

```
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
```

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.

typedef를 이용한 구조체 정의

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

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

```
point pos1, pos2;
point pos3;
person per1, per2;
```

typedef 는 C/C++에서 자료형 이름을 새롭게 만들 수 있게 해줌

예) typedef int INT;
INT a; /* int a; 와 같음 */

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 };
```

초기화 방식이 배열과 유사하다.
초기화 할 데이터들을 중괄호 안에 순서대로 나열하면 된다. .

구조체 변수의 초기화

```
  printf("%d %d\n", pos.xpos, pos.ypos);
  printf("%s %s %d\n", man.name, man.phoneNum,
          man.age);
  return 0;
}
```

실행결과

```
10 20
이승기 010-1212-0001 21
```

구조체 배열의 정의와 접근

```
#define ARR_SIZE      3
struct point {
  int xpos;
  int ypos;
};

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

구조체 배열의 정의와 접근

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

  for (i = 0; i < ARR_SIZE; i++) {
    printf("[%d, %d] ", arr[i].xpos, arr[i].ypos);
  }
  return 0;
}
```

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

- 구조체 변수의 초기화

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

구조체 변수 하나를 초기화하기 위해서
하나의 중괄호를 사용하듯이...

- 구조체 배열의 초기화

```
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
};
```

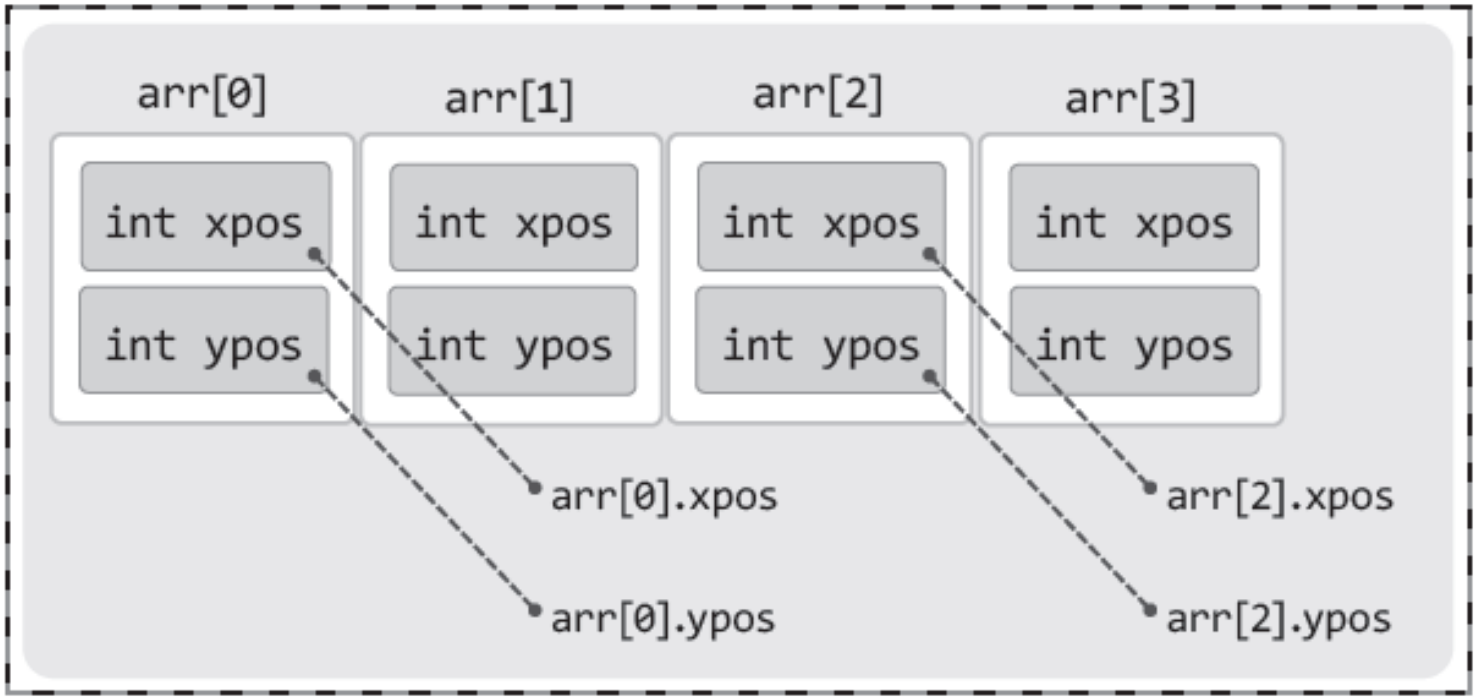
구조체 배열을 초기화하기 위해서
배열요소 각각의 초기화 값을 중괄호로
묶어서 표현한다.

구조체 배열의 정의와 접근

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

길이가 4인 구조체 배열의 선언방법

선언된 배열의 형태



구조체 배열의 초기화 예제

```
#define ARR_SIZE 3
struct person {
  char name[20];
  char phoneNum[20];
  int age;
};

int main(void) {
  struct person arr[ARR_SIZE] = {
    { "이승기", "010-1212-0001", 21 },
    { "정지영", "010-1313-0002", 22 },
    { "한지수", "010-1717-0003", 19 } };
```

구조체 배열의 초기화 예제

```
  int i;
  for (i = 0; i < ARR_SIZE; 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;
```

pptr 이 가리키는 구조체 변수의 멤버 xpos에 접근

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

pptr 이 가리키는 구조체 변수의 멤버 ypos에 접근

구조체 포인터 변수를 대상으로 하는 포인터 연산 및 멤버의 접근방법

-> 연산자를 기반으로 하는 구조체 변수의 멤버 접근 방법

```
(*pptr).xpos = 10;  ⟷  pptr->xpos = 10;
(*pptr).ypos = 20;  ⟷  pptr->ypos = 20;
```

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);
```

구조체 변수와 포인터 관련 예제

```
    pptr = &pos2;
    pptr->xpos += 1;
    pptr->ypos += 2;
    printf("[%d, %d]\n", (*pptr).xpos, (*pptr).ypos);
    return 0;
}
```

많은 코드에서 주로 사용하는 연산자이니 -> 연산자의 사용에 익숙해지자.

실행결과

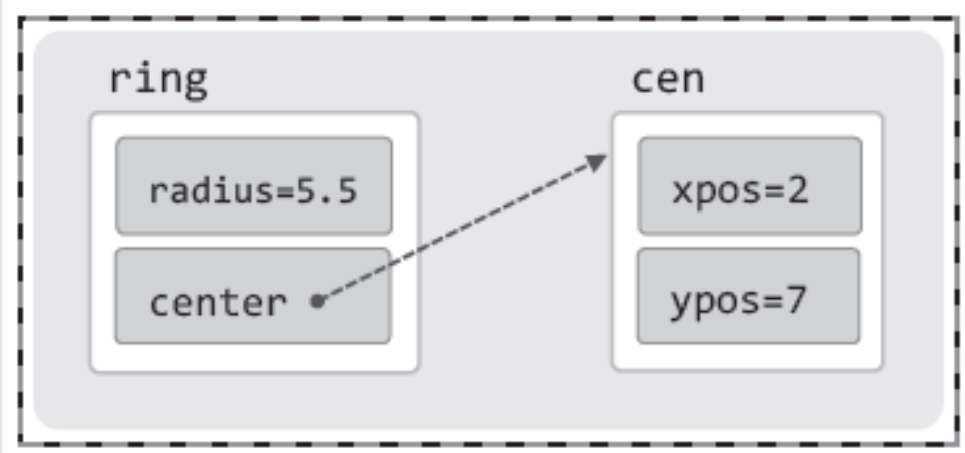
```
[5, 7]
[101, 202]
```

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 };
```

구조체 변수의 멤버로 구조체 포인터 변수 선언 가능



포인터 변수를 구조체의 멤버로 선언하기1

```
  printf("원의 반지름: %d\n", ring.radius);
  printf("원의 중심 [%d, %d]\n",
        (ring.center)->xpos, (ring.center)->ypos));
  return 0;
}
```

실행결과

```
원의 반지름: 5.5
원의 중심 [2, 7]
```

포인터 변수를 구조체의 멤버로 선언하기 2

```
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 */
```

구조체 변수의 멤버로 해당 구조체의 포인터 변수를 둘 수 있다.

포인터 변수를 구조체의 멤버로 선언하기 2

```
  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;
}
```

실행결과

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

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};
```

구조체 변수와 첫 번째 멤버의 주소 값

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

실행결과

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

정리

- 구조체 정의
- 구조체 변수 접근
- typedef를 이용한 구조체 정의
- 초기화
- 구조체의 배열, 포인터