교재: 쉽게 풀어쓴 C언어 Express, 천인국 지음

제13장 구조체

13.1 단일 구조체

- 구조체(structure) : 자료형이 다른 변수들의 집합체 (배열화)
- 구조체 선언(빵틀) : struct [구조체명] { 멤버변수_1; 멤버변수_2; } [구조체 변수명[={초기값}]];
- 구조체 변수(붕어빵) : struct 구조체명 구조체변수명[={초기값}];
- 구조체 초기화 : 구조체변수명={ 멤버변수값_1, 멤버변수값_2, ... };
- 구조체 멤버의 접근 : 구조체변수명.멤버변수

(교재 568~569쪽) 학생의 인적사항을 구조체로 만든다.

```
#include <stdio.h>
struct student {
   int number;          // 학번
   char name[10];       // 이름
   double grade;        // 학점
};
void main()
{
    struct student s = { 20190001, "홍길동", 4.3 };
//    printf("학번을 입력하시오 : "); scanf_s("%d", &s.number);
//    printf("이름을 입력하시오 : "); scanf_s("%s", &s.name, (unsigned)_countof(s.name));
//    printf("학점을 입력하시오 : "); scanf_s("%lf", &s.grade);
    printf("학번 : %d₩n", s.number);
    printf("이름 : %s₩n", s.name);
    printf("학점 : %.1lf₩n", s.grade);
}
```

13.2 구조체의 활용

- 구조체를 멤버로 가지는 구조체

(교재 572쪽) 사각형을 point 구조체로 만들어 면적과 둘레를 출력한다.

```
#include <stdio.h>
struct point {
    int x;
    int y;
};
struct rect {
    struct point p1;
    struct point p2;
};
```

```
void main()
{
    struct rect r = { {10, 10}, {20, 30} };
    int width =
    int height =
    int area = width * height;
    int peri = (width + height) * 2;
    printf("면적은 %d 이고, 둘레는 %d 입니다.₩n", area, peri);
}
```

- 구조체 변수의 대입 : p2=p1; 또는 p2.x=p1.x; p2.y=p1.y;
- 구조체 변수의 비교 : p1 == p2 (오류 발생) 대신에 p1.x==p2.x && p1.y==p2.y

13.3 구조체 배열

- 선언 : 구조체변수명[배열의 크기];
- 초기화 : 배열의 초기화와 동일하며, 구조체 부분은 중괄호를 사용한다.
- 멤버의 접근 : 구조체변수명[배열의 인덱스].멤버변수

(교재 575쪽) 3명 학생의 인적사항을 구조체에 넣고, 화면에 출력한다.

```
void main()
{
   struct student list[] = {
        { 20190001, "홍길동", 4.3 },
        { 20190002, "김유신", 3.92 },
        { 20190003, "이성계", 2.87 }
   };
   int n = sizeof(list) / [          ];
   for (int i=0; i<n; i++) {
      printf("학번: %d, 성명: %s, 학점: %.1lf₩n",
list[i].number, list[i].name, list[i].grade);
   }
}
```

13.4 구조체와 포인터

- 구조체를 가리키는 포인터 : struct student s={24, "Kim", 4.3}, *p = &s;
- 멤버의 접근 : p->year 또는 (*p).year

(교재 578쪽) 포인터를 사용하여 구조체 멤버를 접근한다.

```
void main()
{
   struct student list[3] = {
        { 20190001, "홍길동", 4.3 },
        { 20190002, "김유신", 3.92 },
        { 20190003, "이성계", 2.87 }
   }, *p = [  ];
   for (int i=0; i<3; i++, p++) {
      printf("학번: %d, 성명: %s, 학점: %.1lf₩n", p->number, p->name, p->grade);
   }
}
```

- 표기법 오류 : *p.year 는 *(p.year) 와 같고, *p->number 는 *(p->number) 와 같다.
- 포인터를 멤버로 가지는 구조체

(교재 579쪽) 학생의 인적사항 구조체에 생일을 추가한다.

```
#include <stdio.h>
strcut date {
   int year;
   int month;
   int day;
};
struct student {
   int number;
   char name[10];
   double grade;
   struct date *dob;
};
```

```
void main()
{
   struct date d = { 1980, 3, 20 };
   struct student s = { 30, "Kim", 167.2, NULL };
   s.dob = [  ];
   printf("학번: %d, 성명: %s, 학점: %.1lf₩n", s.number, s.name,
s.grade);
   printf("생년월일: %d 년 %d 월 %d 일₩n", s.dob->year,
s.dob->month, s.dob->day);
}
```

- 자기 참조 구조체도 가능하다.

교재: 쉽게 풀어쓴 C언어 Express, 천인국 지음

13.5 구조체와 함수

- 구조체를 인수로 가진 함수 : int equal(struct student const *p1, struct student *p2);
- 구조체를 반환하는 함수 : struct student create(int number, char *name, double grade);

(교재 584쪽) 두 벡터의 합을 구한다.

```
struct vector { float x; float y; };
struct vector get_vector_sum(struct vector a, struct vector b);
{
    struct vector result;
    result.x = a.x + b.x;
    result.y = a.y + b.y;
    return result;
}
```

```
void main()
{
    struct vector a = { 2.0, 3.0 };
    struct vector b = { 5.0, 6.0 };
    struct vector sum = get_vector_sum(a, b);
    printf("벡터의 합은 (%.1f, %.1f)입니다.₩n", sum.x, sum.y);
}
```

13.6 공용체

- 정의 : 서로 다른 형의 멤버변수들이 같은 메모리 영역을 사용한다.
- 선언 : union [명칭] { 멤버변수1; 멤버변수2; ... } 변수명;

(교재 586쪽) 공용체의 사용도 구조체와 동일하다.

```
#include <stdio.h>
void main()
{
  union example { int i; char c[4]; } v = { 'A' };
  printf("int=%4x char=%x%x%x%x₩n", v.i, v.c[0], v.c[1], v.c[2], v.c[3]);
  v.i = 10000;
  printf("int=%d char=%x%x%x%x₩n", v.i, v.c[0], v.c[1], v.c[2], v.c[3]);
}
```

- 용도가 다른 2변수가 하나의 메모리를 점유하여 사용하면 메모리 활용 효율을 높일 수 있다.

13.7 열거형 (enumeration[이**누**-머**레**이션])

- 선언 : enum [태그명] { 명칭1, 명칭2, 명칭3, ... };
- 값의 할당 : 0, 1, 2, 3, ...

```
enum boolean { FALSE, TRUE };
enum week { SUN, MON, TUE, WED, THU, FRI, SAT };
enum rainbow { RED, ORANGE, YELLOW, GREEN, BLUE, INDIGO, VIOLET };
```

- 값을 지정하는 경우 : enum { ZERO, TWO=2, FOUR=4 };

13.8 typedef (type define)

- 선언 : typedef old_type new_type;

```
typedef unsigned int size_t;
typedef unsigned int UINT32;
typedef unsigned char BYTE;
typedef double REAL;
typedef enum { FALSE, TRUE } BOOL;
typedef struct { int x; int y; } POINT;
typedef struct { double real; double imag; } COMPLEX;
typedef float VECTOR[2];        // VECTOR는 실수 2개로 이루어진 1차원 배열
typedef float MATRIX[10][10];   // MATRIX는 실수 10*10개로 이루어진 2차원 배열
```

- 사용 : new_type 변수명;

(교재 594쪽) 2차원 공간 상의 점을 평행이동한다.

```
#include <stdio.h>
typedef struct { int x; int y; } POINT;
POINT translate(POINTp, POINT delta)
{
    POINT new_p;
    new_p.x = p.x + delta.x;
    new_p.y = p.y + delta.y;
    return new_p;
}
void main()
{
   POINT p = { 2, 3 }, delta={ 10, 10 };
   POINT result = translate(        );
   printf("새로운 점의 좌표는 (%d, %d)입니다.₩n", result.x, result.y);
}
```

- typedef의 장점

(1) 이식성을 높여준다. (컴퓨터 시스템이 바뀌면 정의만 바꾸면 된다)

(2) #define은 전처리기, typedef은 컴파일러가 처리한다.

(3) 문서화의 역할도 한다. (주석문이 필요없다)