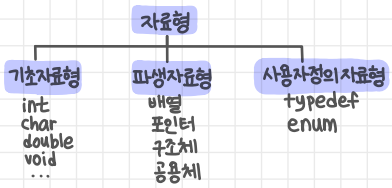
6. 구조체

1 구조체란 무엇인가?

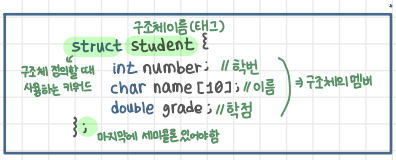


→ 동일한 종류의 데이터 하나로 묶기 ⇒ 배열(array) 사용

→ 서로 다른 종류의 데이터 하나로 묶기 ⇒ 구조체(structure) 사용

2 구조체선언, 초기화, 사용

구조체선언 : 서로 다른 자료형의 변수들을 묶어서 새로운 자료형을 만드는것
: struct 라는 키워드로 정의된다



struct : 구조체를 선언할 때 사용하는 키워드

student : 구조체 **태그(tag)**
↳ 구조체와 구조체를 구별하기 위해 구조체에 붙여지는 이름
태그 다음에 중괄호 쓰고, 중괄호 사이에 원하는 변수 선언

number, name, grade : **구조체 멤버 (structure member)**
↳ 구조체에 포함되는 변수
↳ 유일한 이름 가진 모든 자료형의 변수가 구조체 멤버가 될수 있음

세미콜론 : 구조체의 선언의 끝. 구조체 선언도 하나의 문장에 해당하기 때문

주의할점 : 구조체 선언 ≠ 변수 선언
: 구조체 선언 = 구조체의 형태(틀)만 정의
: 아직 구조체를 이용하여 변수를 만들지 않았음에 유의할 것

구조체를 정의한다는 것 : 자료형나 붙어쓰는 만드는 틀을 정의한다는 것
자료형나 붙어쓰는 실제로 만들기 위해서는 구조체 변수를 선언해야 함.

구조체 변수 생성 :

```
struct student {
    int number;
    char name[10];
    double grade;
};

int main(void) {
    struct student s1;
    ...
};
```

구조체정의
구조체변수선언

: student라는 구조체를 이용해 s1이라는 구조체 변수를 만드는것
: s1이라는 변수 안에는 구조체의 멤버인 number, name, grade가 들어있다
: s1에는 실제 메모리공간이 할당됨 → s1이 차지하는 메모리공간의 크기 = 멤버들 크기의 합
↳ sizeof연산자사용해 구할수 O
: 소스 파일의 첫 부분에서 구조체 정의를 하게 되면 이구조체 정의를 전체소스파일에서 사용가능
: 만약 구조체를 특정 함수 안에서 정의하면 그 함수에서만 구조체 정의 사용해 변수 만들수 있음.
: 하나의 문장에서 여러 구조체 변수를 동시에 선언 가능 ⇒ struct student s2, s3;
: 구조체 정의와 구조체 변수 선언 동시 가능 ⇒

```
struct student {
    int number;
    char name[10];
    double grade;
} s1;
```

구조체정의
구조체 student의 변수

구조체 초기화 : 구조체선언과 변수의 선언이 분리된 경우

```
struct student {
    int number;
    char name[10];
    double grade;
};
Struct student s1 = {24, "kim", 4.3};
```

구조체 멤버 참조

: 구조체의 멤버는 멤버 연산자(.)를 이용해 액세스 할 수 있음

```
s1.grade = 3.8;  // student 구조체 변수 s1의 멤버변수인 grade에 3.8을 대입하는 문장
```

(s1 : 구조체변수, grade : 구조체멤버)

: 멤버가 문자배열이면 strcpy() 사용해 문자열 저장

```
strcpy(s1.name, "kim");
```

이름없는 구조체

: 구조체 정의할 때 태그이름 생략해도 OK.
But 구조체 이름이 없으면 구조체의 변수생성 불가능
➡ 필요한 모든 구조체 변수를 구조체 정의와 함께 선언해야함

ex)

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

struct student {
    int number;
    char name[10];
    double grade;
};

int main(void)
{
    struct student s;

    s.number = 20190001;
    strcpy(s.name, "홍길동");
    s.grade = 4.3;

    printf("학번: %d\n", s.number);
    printf("이름: %s\n", s.name);
    printf("학점: %f\n", s.grade);

    return 0;
}
```

// 구조체 student는 number, name, grade의 3개의 멤버로 정의된다. 아직 구조체 변수 선언 X
// 구조체를 함수의 외부에 선언하면 파일의 모든 함수에서 사용가능
// 구조체를 함수의 내부에 선언하면 해당함수 내부에서만 사용가능

// 구조체 변수 s가 선언됨.
// 선언되는 위치가 함수 내부이므로 지역변수가 되고 따라서 초기값은 쓰레기값이 된다

// 구조체 변수 s의 멤버에 값들 대입
number : 정수형 ⇒ 정수대입
name[] : 문자배열 ⇒ 문자열대입
grade : double형 ⇒ 실수대입

구조체 변수 s에 저장된 값들을 printf()를 이용해 화면에 출력
멤버연산자 (.)는 구조체의 멤버를 참조하는 연산자

↳ 대입대신 직접 입력받고 싶다면?

```
scanf("%d", &s.number);   // 구조체 변수 s의 멤버 number에 사용자로부터 받은 학번 저장
scanf("%s", s.name);      // 구조체 변수 s의 멤버 name에 사용자로부터 받은 이름 저장 → name은 배열의 이름이므로 이미 배열을 가리키는 포인터임. → &필요X
scanf("%lf", &s.grade);   // 구조체 변수 s의 멤버 grade에 사용자로부터 받은 학점값 저장 → &붙이려면 &(s.name[0])
```

ex) 2차원공간상의 점을 구조체로 표현하기

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

Struct point {
    int x;
    int y;
};

int main(void) {
    Struct point p1, p2;
    int xdiff, ydiff;
    double dist;

    printf("점의 좌표입력 : (x, y)");
    scanf("%d %d", &p1.x, &p1.y);

    printf("점의 좌표입력 : (x, y)");
    scanf("%d %d", &p2.x, &p2.y);

    xdiff = p1.x - p2.x;
    ydiff = p1.y - p2.y;

    dist = sqrt((double)(xdiff * xdiff + ydiff * ydiff));
    printf("거리 = %f", dist);
    return 0;
}
```

// 구조체 point 선언
// point는 2개의 멤버 x, y로 정의된다.
// 아직 구조체 변수 선언안됨

// 구조체 student의 변수 p1과 p2가 선언된다. 선언되는 위치가 함수 내부이므로 자동변수가 되고, 초기값은 쓰레기값이 된다.

사용자에게 점의 좌표를 입력받아 구조체 변수에 저장한다

// sqrt() : 제곱근을 계산하는 함수

6. 구조체

구조체를 멤버로 가지는 구조체

```
struct date {
    int year;
    int month;
    int day;
}

struct student {
    int number;
    char name[10];
    struct date dob;   // date of birth
    double grade;
};
```

구조체안에 구조체 포함시킬수있음

Student 구조체 안의 멤버 변수 dob에 값을 대입하려면?

```
s1.dob.year = 1983;
s1.dob.month = 03;
s1.dob.day = 29;
```

사각형을 point 구조체로 나타내기

```
#include <stdio.h>

struct point {   // 구조체 point 선언
    int x;       // point의 멤버는 x와 y
    int y;       // 선언되는 위치가 함수 외부이므로 같은 소스 파일에 있는 모든 함수 사용가능
};

struct rect {    // 구조체 rect 선언
    struct point p1;   // rect의 멤버로 구조체 변수 p1과 p2 포함
    struct point p2;   // 구조체 안에 다른 구조체가 멤버로 포함될수 있음
};

int main(void) {
    struct rect r;   // 구조체 rect를 이용해 구조체 변수 r을 선언한다
    int w,h,area,peri;

    printf("왼쪽상단좌표?: ");
    scanf("%d %d",&r.p1.x,&r.p1.y);   // 사용자로부터 받은 x좌표와 y좌표 저장
    printf("오른쪽 상단좌표?: ");
    scanf("%d %d",&r.p2.x,&r.p2.y);

    w = r.p2.x - r.p1.x;
    h = r.p2.x - r.p1.x;

    area = w*h;
    peri = 2*w + 2*h;
    printf("면적=%d, 둘레=%d", area, peri);

    return 0;
}
```

r의 p1에 직접 값을 대입할 수는 없고 r.p1.x와 같은 중첩된 구조체의 말단 멤버까지 내려가서 각각 대입해야함

구조체 변수의 대입과 비교

구조체를 다른 구조체에 대입 = 하나의 구조체 변수에 들어있는 자료들을 다른 구조체 변수로 복사

ex) 2차원 공간에서 점의 위치를 나타내는 구조체

```
struct point {
    int x;
    int y;
}
Struct point p1 = {10,20};
struct point p2 = {30,40};
```

⇒ `p2 = p1;` // 대입
// p1과 p2의 좌표값이 {10,20}으로 같아진다.

↳
```
p2.x = p1.x;
p2.y = p1.y;   //이것도 대입의 방법
```

But 구조체 변수와 구조체 변수를 서로 비교하는것은 허용 X. // 구조체변수를 비교하려면 멤버마다 별도의 수식을 적어주어야한다.

```
if (p1 == p2) {
    printf("p1과 p2 같음");
}
```
⇒ 컴파일오류

```
if ( (p1.x == p2.x) && (p1.y == p2.y) ) {
    printf("p1과 p2 같음");
}
```
↳올바른방법

4 구조체의 배열

구조체 배열의 선언

```
struct student {
    int number;
    char name[10];
    double grade;
};
Struct student list[100];
```

구조체배열로서 구조체 100개 저장가능

↳ Student 구조체의 배열을 list[]란 이름으로 선언함
↳ 이 배열은 학생 100명의 데이터 저장 가능
↳ 각 학생들의 데이터는 구조체로 표현되어 있음.

ex) 배열에서 인덱스가 2인 요소에 있는 구조체에 값 저장하기

```
list[2].number = 24;
strcpy(list[2].name, "홍길동");   //문자열은 항상 strcpy()를 이용해 대입
list[2].grade = 4.3;
```

구조체 배열의 초기화

⇒ 배열 초기화 만에 구조체 초기화가 들어감 {{}} 콤마필수

```
struct student list[3] = {
    { 1, "Park", 3.42 },
    { 2, "Kim", 4.31 },
    { 3, "Lee", 2.98 }
};
```

구조체배열의 원소개수 자동계산 = 전체배열의 총바이트 수 / 개별원소의 바이트 수

```
n = sizeof(list) / sizeof(list[0]);
```
또는 `n = sizeof(list) / sizeof(struct student);`

ex) 구조체배열사용하기

```
#include <stdio.h>
#define SIZE 3

struct Student {
    int number;
    char name[20];
    double grade;
};

int main(void) {
    Struct student list[SIZE];   // 구조체 Student의 배열 list[]가 선언된다.
    int i;                       // 역시 선언되는 위치가 함수 내부이므로 자동변수가 되고, 따라서 배열 원소의 초기값은 쓰레기값이 된다.

    for (i=0; i<SIZE; i++) {
        printf("학번: ");
        scanf("%d", &list[i].number);
        printf("이름: ");
        scanf("%d", &list[i].name);
        printf("학점(실수): ");
        scanf("%lf", &list[i].grade);
    }

    for(i=0; i<SIZE; i++)
        printf("이름: %S, 학점: %f \n", list[i].name, list[i].grade);

    return 0;
}
```

반복루프 이용해 사용자로부터 값을 입력받아 배열원소들에 대입한다.
배열의 이름은 그자체가 포인터이므로 & 사용안함

5 구조체와 포인터

구조체를 가리키는 포인터

```
Struct student s = { 24, "Kim", 4.3 };
struct student *p;   // 구조체 Student를 가리키는 포인터 선언

p = &s;   //구조체의 주소를 포인터에 대입
printf("학번=%d 이름=%S 학점=%f \n", (*p).number, (*p).name, (*p).grade);
```

포인터를 통해 구조체의 정보에 접근한다. (*p)가 구조체가 된다.

포인터를 이용해 구조체 멤버 가리키기

→ 연산자 : 간접멤버연산자 (indirect membership operator)
: 구조체 포인터 이용해 멤버에 접근하기 위해 사용된다

↳ `p -> number` ⊜ `(*p).number`
↳ 포인터 p가 가리키는 구조체의 멤버 number이라는 의미

정리

1. `(*p).number` — p가 가리키는 구조체 변수 / p가 가리키는 구조체 변수의 멤버 number
2. `p -> number`

⇒둘다 포인터 p가 가리키는 구조체의 멤버 number을 의미함

3. `*p.number` ⇒ 연산자 우선순위에 의하여 *(p.number)과 같음. 구조체 p의 멤버 number가 가리키는 내용을 의미함. 만약 number가 포인터가 아니면 오류임.
4. `*p -> number` ⇒ 연산자 우선순위에 의하여 *(p->number)과 같음. 구조체 p의 멤버 number가 가리키는 내용을 의미함. 만약 number가 포인터가 아니면 오류임.

포인터를 멤버로 가지는 구조체

ex)
```
#include <stdio.h>

struct date {
    int month;
    int day;
    int year;
};

struct student {
    int number;
    char name[20];
    double grade;
    Struct date *dob;   // 포인터가 구조체의 멤버
};

int main(void) {
    Struct date d = { 3, 20, 1990 };
    Struct student s = { 20190001, "Kim", 4.3 };   // 구조체 변수들을 선언하고 초기화한다

    S.dob = &d;   // 구조체 변수 S의 멤버인 포인터 dob에 구조체 d의 주소 대입
    ⋮ 출력부분 (s.dob -> year 사용)
}
```

6. 구조체

문자배열과 문자 포인터의 차이점

```
struct studentA {                    struct studentB {
    int number;                          int number;
    char name[10];  문자배열선언          char *p;  문자포인터선언
    double grade;                        double grade;
};                                   }
```

```
Student studentA s1 = {20180001, "홍길동", 4.3};
Student studentB s2 = {20180002, "김유신", 4.2};
```

⇒ 구조체 s1의 경우, 문자열은 구조체 내부의 배열 name[]에 저장됨.
즉 10 바이트의 공간이 구조체 내부에 할당되는 것임.

```
struct studentA s1;
scanf ("%s", s1.name);
```

⇒ 구조체 s2의 경우, 구조체 내부에는 포인터 p를 위한 4바이트의 공간만 할당된다.
즉 문자열을 위한 공간은 구조체 s2에는 없다.
따라서 어딘가에 문자열이 이미 저장되어 있는 경우에만 포인터 p로 그곳을 가리키게 해야함.
즉 구조체 s2안에 문자열을 저장할 수 없음

```
struct studentB s2;
scanf ("%s", s2.p);   X ⇒ 오류: s2안에 정의된 포인터 p가 올바른 주소로 초기화되지않음
```

⬇

```
Struct studentB s2;
s2.p = "강감찬";   O ⇒ 문자열 "강감찬"은 메모리의 텍스트 세그먼트에 저장되고
                      그주소가 s2.p에 저장된다
```

6 구조체와 함수

: 구조체는 함수의 인수로도, 함수에서의 반환값으로도 반환될수있음 ⇒ 값에의한호출
즉, 구조체 변수의 모든 내용이 복사되어 함수로 전달되고 반환됨.

: 함수에는 구조체의 복사본이 인수로 전달되므로 함수 안에서의 인수값이 변경되더라도 원본구조체에는 영향을 주지 않는다.

; 구조체의 크기가 클 땐 구조체 포인터를 사용하는 것이 좋음.

구조체를 함수의 인수로 넘기는 방법

```
int equal (struct student s1, struct student s2) {
        if (s1.number == s2.number)
            return 1;
        else
            return 0;
}

int main(void) {
        struct student a = { 1, "hong", 3.8 };
        struct student b = { 2, "kim", 4.0 };
        if (equal (a, b) == 1) {
            printf("같은학생\n");
        }
        else {
            printf("다른학생\n");
        }
}
```

구조체의 경우, 복사된다.

↳단점: 구조체의 크기가 크면 적지않은 시간과 메모리 공간을 차지할수 있음

```
int equal ( struct student *p1, struct student *p2 ) {
        if ( p1 -> number == p2 -> number)   // 포인터를 통해 구조체에 접근한다.
            return 1;
        else
            return 0;
}

int main(void) {
        struct student a = {1, "hong", 3.8};
        struct student b = {2, "kim", 4.0};
        if (equal (&a, &b) == 1) {
            printf("같은학생\n");
        }
        else {
            printf("다른학생\n");
        }
}
```

구조체 포인터를 보낸다.

↳단점: 복사본 전달이 아니고, 구조체를 직접 보내기 때문에 원본데이터 훼손가능
↳ 이를 예방하려면 원본을 포인터를 통해 읽기만 하고 쓸 필요는 없는 경우, 매개변수를 정의할때 const 키워드 사용!

const 키워드가 *p1 앞에 있으면 이 포인터가 가리키는 구조체의 값을 변경하려고 하면 컴파일 단계에서 오류발생

```
int equal (struct student const *p1, struct student const *p2) {
        if (p1 -> number == p2 -> number)
            return 1;
        else
            return 0;
}
```

이 포인터를 통하여 구조체를 변경하는 것은 금지됨

구조체를 함수의 반환값으로 넘기는 방법

```
struct student create() {
        struct student s;
        s.number = 3;
        strcpy (s.name, "Park");
        s.grade = 4.0;
        return s;
}

int main(void) {
        struct student a;
        a = create();

        return 0;
}
```

구조체 s가 구조체 a로 복사된다.