[구조체]

"사용자 정의 자료형"

데이터를 저장할때, 사용할 수 있는 자료형은 많다.

int / short / long / char / doubble / float 등.

하지만 하나의 데이터를 정의할때, 이 자료형만으로 부족할 때가 많다.

DB를 한번 생각해보면 하나의 데이터는 사실 "여러개"의 데이터 타입의 묶음으로 되어있다.

C 언어 에서도 DB처럼 여러개의 데이터타입으로 묶어 하나의 데이터를 정의 할 수 있도록 "구조체"를 만들 수 있도록 하고 있다.

```
struct phone_num {
    char name[20];
    char num[20];
    int cnt;
};
```

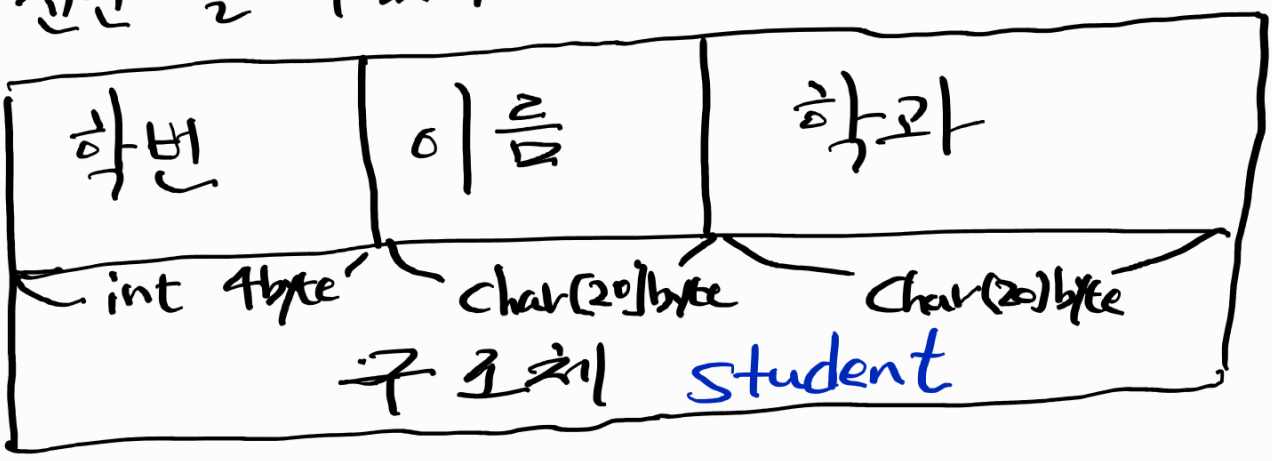
마치 DB 테이블의 열을 만드는 것과 비슷하다.

구조체 선언은 말 그대로 "자료형"을 정의하는 것이므로, 초기값 대입은 불가능하다.

[구조체의 포인터]

"자료형"이기 때문에

구조체도 자료형의 주소값을 저장할 수 있는 포인터 변수를 선언 할 수 있다.




```
struct student s1;
struct student *p = &s1;
```

↳ student 자료형의 주소값을 저장하는 포인터변수

p ⇒ s1의 첫번째 주소값.

간접참조연산

(*p).num 멤버 접근 연산자 .

p → num 멤버 참조 연산자 →

s1 주소값이 가르키는 num

[공용체]

: 불편하지만 메모리 절약이 가능한 구조체

union ← 공용체의 예약어

공용체에서 가장 기억해야 할 것은

구조체처럼 정의한 자료형 마다 메모리의 할당이 되지 않는 다는 것이다.

1. 공용체 변수의 크기는 멤버 중에서 크기를 가장 많이 차지하는 멤버의 자료형으로 설정된다.

2. 공용체 변수의 초기화는 중괄호를 사용하여 첫번째 멤버만 초기화 한다.
   만약 첫번째 멤버가 아닌 멤버를 초기화 할 때는 멤버 접근 연산자 또는 포인터를 사용하여 멤버 참조 연산자를 이용해야 한다.

나의 생각 : 공용체는 하나의 메모리 공간을 모든 멤버가 같이 사용한다는 불편한 점 때문에 실무에서 잘 사용되지 않을 것 같다.

[열거형] enum

"정수들의 나열 ... 열거형"

```
enum Season
{ SPRING,   ← 0
  SUMMER,   ← 1
  FALL,     ← 2
  WINTER    ← 3
};
```

자동 증가.

초기값 설정가능.

[형 재정의]

typedef struct student Student;

typedef 뒤에 재정의 할 자료형의 이름을 적고
뒤이어 새로운 이름을 적는다.

"선언과 동시에 재정의"

```
typedef struct {
    int num;
    double grade;
} Student
```