

# C언어 (CSE2035)

## (Chap11. Derived types-enumerated, structure, and union) (1-1)

Sungwon Jung, Ph.D.

Dept. of Computer Science and Engineering  
Sogang University  
Seoul, Korea  
Tel: +82-2-705-8924  
Email : [kimjihwan@sogang.ac.kr](mailto:kimjihwan@sogang.ac.kr)

컴퓨터공학과

C 프로그래밍

## Enumerated types

- **enum**은 상수 대신 상징적인 이름을 사용할 수 있도록 해준다. 다음과 같이 두 가지 선언 방식이 있다.

Format 1: enum type 변수를 하나만 정의해서 사용하는 방법

```
enum {enumVal1 [= initialValue1], enumVal2 [=initialValue2], ...} variable_name;
```

Format 2: 특정한 enum type을 정의하고, 그 enum-type의 변수를 선언하는 방법

```
enum tag {enumVal1 [= initialValue1], enumVal2 [=initialVal2], ...};  
enum tag variable_name;
```

- **enumVal**은 상수 대신 사용하는 상징적인 이름이다.
  - 내부적으로 **initializer**에 의해 결정된 상수로 취급된다.
  - 만약 **initValue**가 존재하지 않는다면 가장 첫 이름은 0으로, 이후는 계속 1씩 증가한다.
  - **enumVal1, enumVal2, ...**, 도 변수 이름과 같으므로 **이름이 중복되면 에러가 발생한다.**

## Enumerated types

- 다음은 enum 타입 선언의 예이다.

```

1 #include <stdio.h>
2
3 int main(){
4     enum months{jan = 1, feb, mar, apr, may, jun,
5                 jul, aug, sep, oct, nov, dec};
6     enum months mon;
7
8     for(mon = jan; mon <= dec; mon++)
9         printf("%d ", mon);
10    printf("\n");
11
12    return 0;
13 }

```

```

vore@nlpsag:~$ ./a.out
1 2 3 4 5 6 7 8 9 10 11 12
vore@nlpsag:~$

```

- months에서 최초의 값인 jan만 1로 초기화 되었다.  
이후의 값은 이전 이름에 1 증가된 값인 feb = 2, mar = 3, ... , 을 가진다.

## Enumerated types

- 위 프로그램은 다음과 같이 작성 가능하다.

```

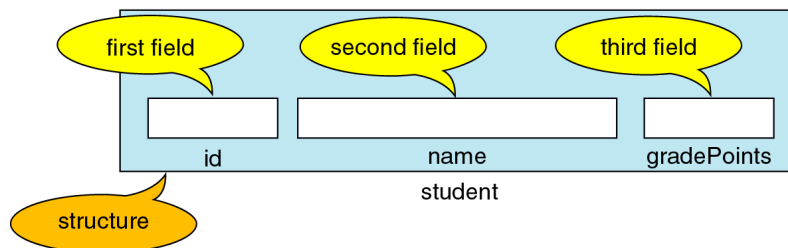
1 #include <stdio.h>
2
3 int main(){
4     int jan = 1, feb = 2, mar = 3, apr = 4, may = 5, jun = 6,
5         jul = 7, aug = 8, sep = 9, oct = 10, nov = 11, dec = 12;
6     int mon;
7
8     for(mon = jan; mon <= dec; mon++)
9         printf("%d ", mon);
10    printf("\n");
11
12    return 0;
13 }

```

- 그러나, 관리해야 할 변수가 많고 그것들이 1씩 증가하는 값을 가진다면 enum을 사용하는 편이 위와 같이 코드를 작성하는 것 보다 유용하다고 할 수 있다.

# Structure

- 구조체(structure)는 여러 개의 변수를 하나의 구조로 가지는 변수이다.
- 배열이 같은 타입의 변수들의 집합이라면, 구조체는 여러 가지 타입의 변수가 포함된 집합이라고 볼 수 있다. 구조체는 C에서 제공하는 모든 타입이 들어갈 수 있으므로, 구조체 내부에는 배열을 가질 수도 있다.
- 예를 들어 학생마다 정보 (id, name, gradePoints)를 유지해야 한다면, 이 정보들을 하나로 묶어서 하나의 변수로 관리하면 편할 것이다. 이렇게 여러 데이터 타입을 하나로 묶은 것을 구조체라 한다.



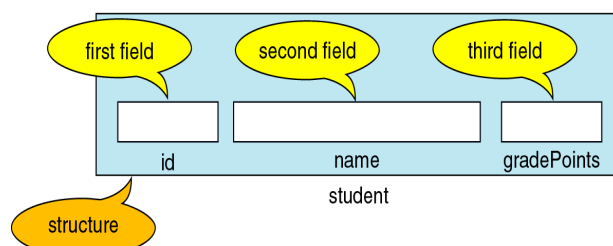
# Structure

- 구조체도 하나의 타입처럼 선언한다.

```
struct {
    type1  fieldName1;
    type2  fieldName2;
    ....
    typeN  fieldNameN;
} variable_identifier;
```

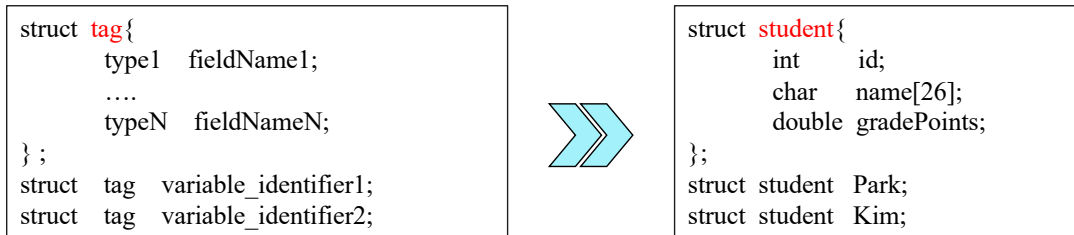
- 앞장의 student구조체를 만들면 다음과 같다.

```
struct {
    int    id;
    char   name[26];
    double gradePoints;
} student;
```



# Structure

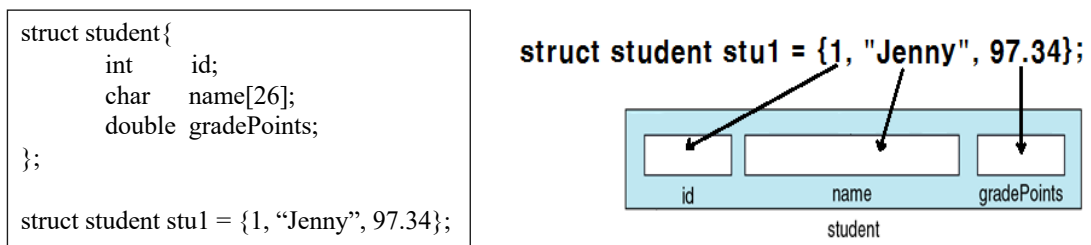
- 하지만, 학생이 여러 명인 경우라면 **student**와 같은 변수가 여러 개 필요하다. 하지만 그 때마다 **struct {}** 문을 이용해서 변수를 만들어 준다면 너무나 복잡할 것이다. 그래서 C에서는 구조체 자체를 하나의 **data type**으로 만들어 사용할 수 있도록 해준다. 다음과 같이 **tag**를 이용하면 구조체를 하나의 **data type**으로 만들 수 있다.



- 즉, 구조체는 사용자가 정의한 타입(**user-defined type**)이다.  
→ 하나의 **int**, 하나의 **char** 배열, 하나의 **float**를 저장할 수 있는 **student** 타입을 만들었다고 생각하면 된다.

# Structure

- 다음은 구조체 변수의 선언 후 초기화 모습을 보여준다.
- 변수를 선언할 때 초기값을 지정해줄 수 있다.



- 구조체의 변수에 접근하기 위해서는 member operator인 **'.'**을 사용하여 접근할 수 있다. 단, 이 경우 string을 복사할 때는 **strcpy**를 사용해야 한다.

