08_레코드 타입

- 레코드
  - 집합체의 원소를 이름으로 식별하는 이질형 데이터의 모임
  - 1960년대 초기에 COBOL에 도입
  - 레코드 정의와 선언 예 : Pascal
    - name, number, address 데이터를 하나의 레코드로 묶음

```
type
  student = record
    name: packed array[1..20] of char;
    number: integer;
    address: packed array[1..30] of char
  end;
```

    - 정의된 레코드인 student 타입의 변수 A 선언

```
var A: student;
```

08_레코드 타입

  - 레코드 정의와 선언 예 : C

```
struct student {
  char name[20];
  int number;
  char address[30];
};

struct student A;
```

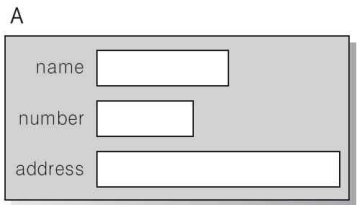
  - 레코드 정의와 선언 예 : Ada

```
type student is record
  name: string (1..20);
  number: integer;
  address: string (1..30);
end record;

A: student;
```

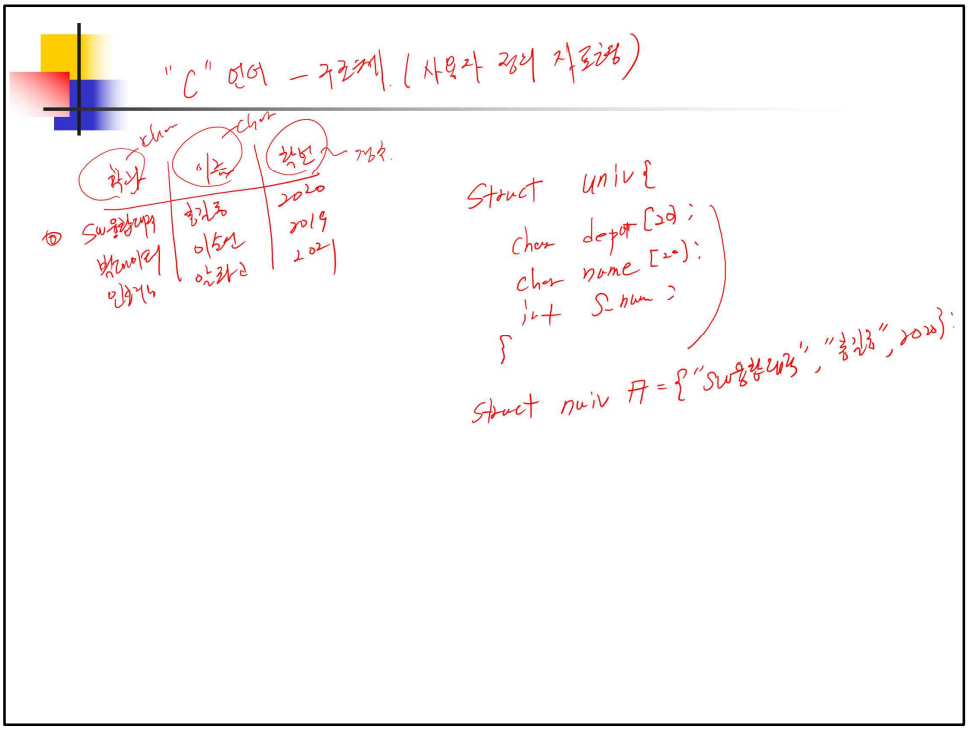
08_레코드 타입

- 레코드 타입 변수 A의 구조도



  - 필드(field): 레코드를 이루고 있는 데이터
  - 필드 접근 시, C/Pascal/Ada는 변수 이름.필드이름 사용

- 예 : 변수 A의 number 필드에 10 저장하기
  - A.number = 10; … C
  - A.number := 10; … Pascal, Ada
  - NUMBER OF A … COBOL



08_레코드 타입

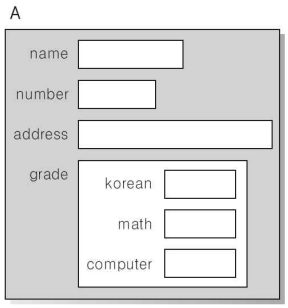
- 다른 레코드를 필드로 지정할 수 있음
  - Ada) student 레코드의 grade 필드가 score 레코드 타입

```
type score is record
   korea: integer;
   math: integer;
   computer: integer;
end record;
type student is record
   name: string (1..20);
   number: integer;
   address: string (1..30);
   grade: score;
end record;

A: student;
```

08_레코드 타입

- 중첩된 레코드 타입 변수 A의 구조도



- 예) math 필드 접근 방법 : A.grade.math

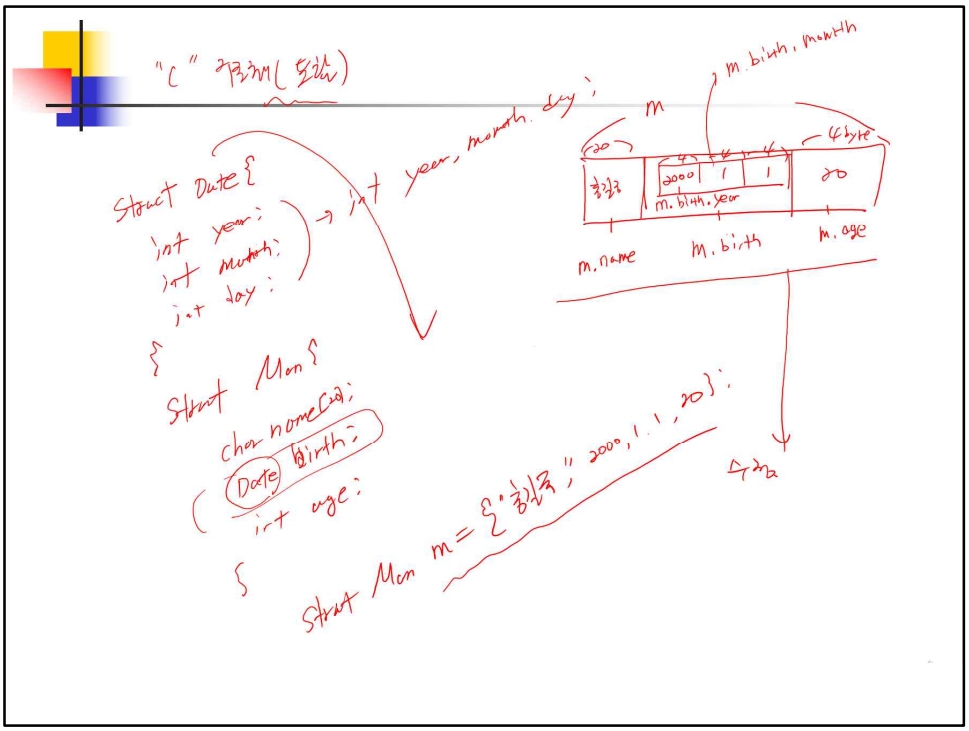
08_레코드 타입

- Ada) 한 번에 레코드 타입 변수의 모든 필드에 값 배정 가능

```
type score is record
   korean: integer;
   math: integer;
   computer: integer;
end record;
student: score;
student := (70, 80, 90);
```

- Ada) 동등 연산자 적용 가능

```
type score is record
   math: integer;
   computer: integer;
end record;
student1, student2: score;
student1 := (70, 80);
student2 := (70, 80);
if student1 = student2 then
   put("equal");
else
   put("not equal");
end if;
```


"C" 방식(동일)
struct Date{
int year;
int month;
int day;
}
int year, month, day;
m
m.birth.month
4byte
20
2000
1
1
m.birth.year
m.name
m.birth
m.age
struct Man{
char name[20];
Date birth;
int age;
}
struct Man m={"홍길동", 2000, 1, 1, 20};

09_공용체 타입

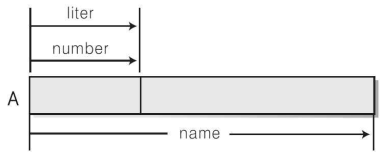
- 공용체
  - 레코드와 형식이 유사
  - 구조체와 달리 모든 필드가 같은 메모리를 공유하면서 필요에 따라 한 필드만을 사용할 수 있음
  - 개념 설명 예
    - 음료수는 float 타입의 liter(용량), 과일은 int 타입의 number(개수), 꽃은 char 배열의 name(이름)으로 가정
      - liter, number, name은 동시에 사용되는 경우는 없고 상황에 따라 하나만 사용되는데 이러한 경우에 공용체를 사용하는 것이 바람직
  - union을 사용한 C/C++ 공용체 표현

```
union product {
   float liter;
   int number;
   char name[10];
};

union product A;
```

09_공용체 타입

- 공용체 타입 변수 A의 구조도



  - 예) liter 필드 접근 방법 : A.litter
  - 프로그래밍 예

```
switch(type) {
   case DRINK:
      printf("%f\n", A.liter);
      break;
   case FRUIT:
      printf("%d\n", A.number);
      break;
   case FLOWER:
      printf("%s\n", A.name);
      break;
   default:
      printf("bad type\n");
}
```

09_공용체 타입

- Ada) 가변 레코드 – 판별자 이용
  - 판별자 kind에 따라 사용되는 필드가 달라짐

```
type Class is (DRINK, FRUIT,
FLOWER);
type product(kind: Class) is record
  name: string(1..5);
  no: integer;
  case kind is
    when DRINK =>
      liter: float;
    when FRUIT =>
      size: integer;
      number: integer;
    when FLOWER =>
      bunch: integer;
  end case;
end record;
```

09_공용체 타입

- 정의된 가변 레코드 product 타입의 변수 선언

```
goods1: product(DRINK);
goods2: product(FRUIT);
```

  - goods1 : name, no, liter 필드로 이루어짐
  - goods2 : name, no, size, numbe 필드로 이루어짐

- 각 필드에 값을 배정하는 문장

```
goods1.name := "cider";
goods1.no := 1;
goods1.liter := 1.5;
goods2:= (kind=>FRUIT, name=>"apple", no=>2, size=>3, number=>5);
```