비트 필드, 공용체

2020년 12월 28일 월요일 오후 11:13

- '프로그래밍 입문' 과목에서 배운 비트 플래그
  - 비트 플래그: bool 여럿을 효율적으로 저장

    - 8개 이하의 bool 값을 하나의 byte에 저장하는 방법
      - C#에는 bool형이 있음
      - C는 없으니까 int형
    - C에서는 원래 4 X 8 = 32 바이트를 쓸 것을
      1바이트에 int(참, 거짓) 8개를 담을 수 있는 것

비트 플래그

- 구조체와 비트 플래그
  - C에서 구조체를 사용하면 매우 간단히 비트 플래그를 구현 가능

    - 크기는 1이다

    - : 1
      - 1비트만 사용하겠다는 것
    - 이를 데이터 패킹(packing) 이라고도 함
  - 비트 플래그 구조체 사용 예

    - 메모리 상태

□ 이진수: 0000 0000

▪ 메모리 상태

□ 이진수: 0001 0000

◆ 2^3 = 8 이므로 16진수로 08이 메모리에 저장된다

○ 플래그 전체를 한 번에 체크하려면?

▪ 지금 방식은 멤버 함수들끼리 비교만 가능

▪ 구조체 전체가 0(모든 플래그가 거짓)인지 비교하고 싶은데 안 됨

○ 포인터를 사용해서 플래그 전체를 한 번에 체크

▪ flags의 주소를 char로 캐스팅해서 val 포인터에 저장하고
val을 역참조하면 8비트씩 읽어오므로 플래그 전체가 0인지 체크 가능

▪ 실수할 여지가 있으므로 다른 방법을 사용하자

□ 공용체(Union)을 사용!

■ 공용체(union)

- 공용체(union)란?
  - 똑같은 메모리 위치를 다른 변수로 접근하는 방법
  - 즉, 공용체 안에 있는 여러 변수들이 같은 메모리를 공유
    - 8바이트의 double 메모리가 있다 하면
      어떤 경우에는 그것을 double로 읽고
      어떤 경우에는 그것을 int로 읽고 싶은 경우
    - 즉, 메모리는 공유하되 그 속에 있는 값을 다르게 해석하고 싶을 때 공용체를 사용

• 공용체로 비트 플래그 구현 코드

  - 공용체 헤더

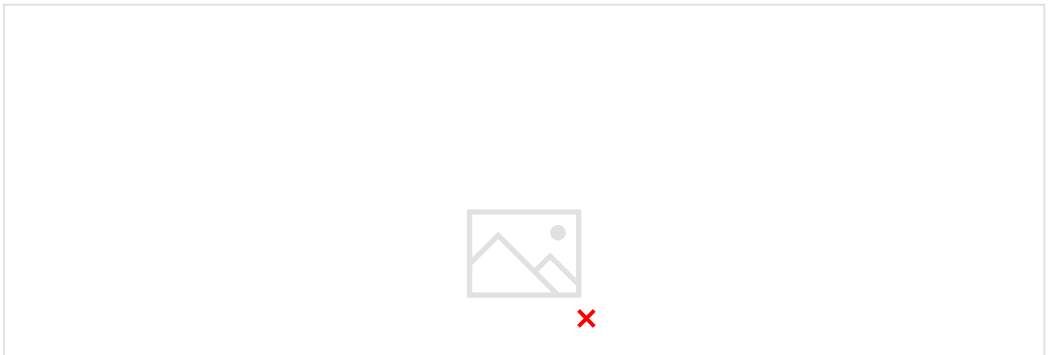
- 공용체는 구조체와 다르게 각 멤버를 다른 메모리로 보지 않음
  - bitflags_t에서 bits와 val은 이름이 다르되 시작하는 메모리는 같다
    - bits와 val을 포인터라 생각하고 그 포인터가 같은 위치를 가리킨다 생각해도 됨

- main.c

- 공용체로 비트 플래그를 구현하면 포인터가 없어도 플래그 전체를 한 번에 체크할 수 있다

메모리 공유만을 위한 공용체의 예

- 공용체 헤더, main.c



- ivalue와 dvalue의 두 비트패턴간 상관관계가 없다
  그냥 그저 메모리 공간을 공유하기 위해 만든 공용체임
- ivalue를 업데이트(대입) 하면 메모리상에서 4바이트만 업데이트가 된다

- dvalue를 업데이트 하면 메모리상에 8바이트가 들어간다

- 함수 헤더

- 똑같은 공용체를 받고 enum 매개변수에 따라 다른 계산을 해주는 함수

- 정리
  - 한 메모리 공간을 용도에 따라 다른 기본데이터형으로 읽을 때 사용
  - 앞의 예보다 덜 유용
    - 비트 플래그 예
  - 사용하기도 어렵고 실수하기도 쉽고
    - 실수 예: op1, op2에 int를 대입했는데
      double로 더하는 경우(OP_DOUBLEADD를 사용한 경우)

- 차라리 함수 자체에서 공용체가 아닌 double로 받는게 안전하겠음