8.4 Pointer to a structure

* Pointer to a structure
  - Similar to normal pointer definition
    ex) struct book *b ;
* Accessing members
  - Using operator -> : member accessing operator to structure pointer
    the same as (*).
    ex) struct book b ;
        struct book *pb ;
        pb = &b ;
        pb->price = 1.98 ;
        printf("%s \ n", pb->title) ;
        printf("%s \ n", (*pb).title) ;

* Using structures with function
  1) using structure as arguments

```
struct complex
{
        double re ;
        double im ;
} ;
struct complex c1 ;
        :
struct complex add(complex a, complex b)
{
        struct complex c ;
        c.re = a.re + b.re ;
        c.im = a.im + b.im ;
}
```

  2) using structure pointer as arguments

```
void complex add(complex *a, complex *b, complex *c)
{
        c->re = a->re + b->re ;
```

```
        c->im = a->im + b->im ;
    }
```

- method 2 is more efficient than 1
- memory copy is not required
- 구조체에 스트링, 포인터, 배열 등이 섞인 경우 내용 복사가 정확이 안될 수 도 있다.

* Using memory allocation and structure

- 구조체는 덩치가 크므로 구조체 포인터와 memory allocation을 이용하여 효율적으로 활용할 수 있다.

ex)

```
struct book *b[100] ;

for(i=0;i<100;i++)
        b[i] = malloc(sizeof (struct book) ) ;
        :
        구조체 이용
        b[i]->prine = 1.98 ;
        :
for(i=0;i<100;i++)
        free(b[i]) ;
```

8.5 Union

- A kind of derived data type
- Mechanism to store different size of data in shared storage

ex)

```
union REG
{
        int ax ;
        char ah, al ;
} ;
        -> union template
union REG r1 ;
r1.ax = 1 ;
printf(" %d %d ", r1.ah, r1.al) ;
```