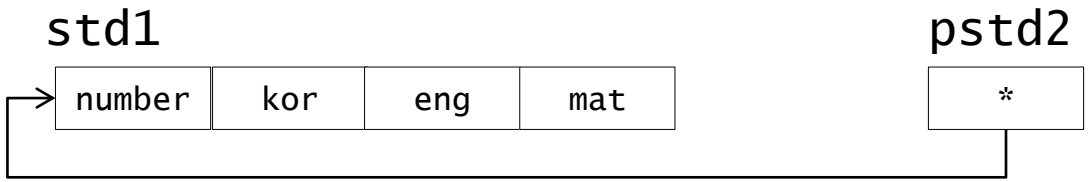
9차시 - 구조체(2)

구조체와 포인터

- 구조체도 자료형이므로 포인터가 존재
  - `typedef struct _STUDENT StudentType;`
  - `StudentType std1, *pstd2 = &std1;`




```
( * pstd2 ) . kor = 10 ;
```

구조체 포인터 연산자 ->

```
(*pstd2).kor = 10;
  pstd2->kor = 10;
```
} 같다.

- 다음의 의미는?
  - *pstd2.kor = 10;
  - *pstd->value = 10;
  - *pstd.number = 20;
  - (*pstd)->score = 30;
  - (pstd->)kor = 90;
  - pstd.*value = 10;

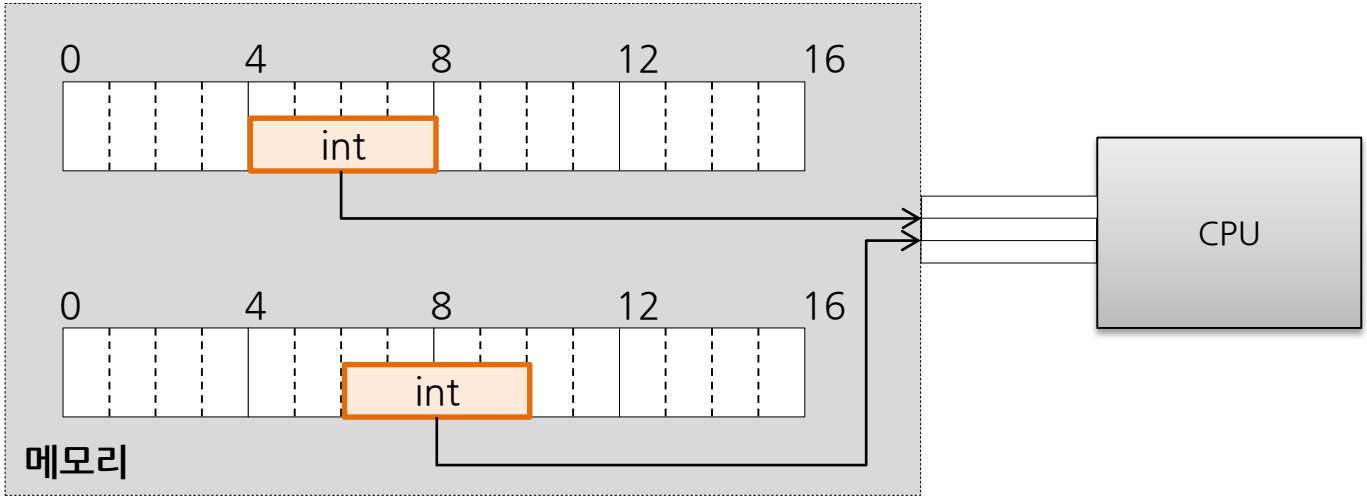
구조체와 메모리

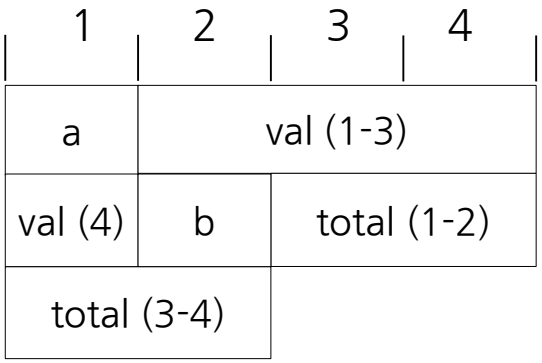
- 결과는?

```
struct X {
        char a;
        int  val;
        char b;
        int  total;
} ;

int main(void)
{
   printf("sizeof %d\n",
            sizeof(struct X));
}
```

```
struct X {
        int  val;
        char a;
        char b;
        int  total;
} ;

int main(void)
{
   printf("sizeof  %d\n“,
            sizeof(struct X));
}
```

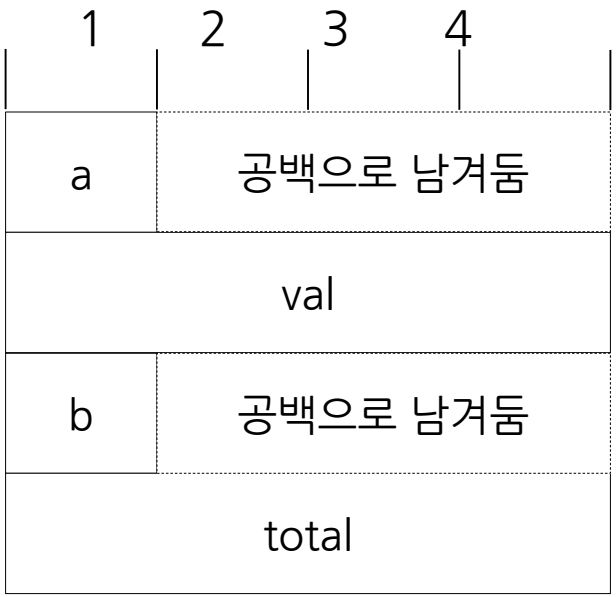
구조체와 메모리

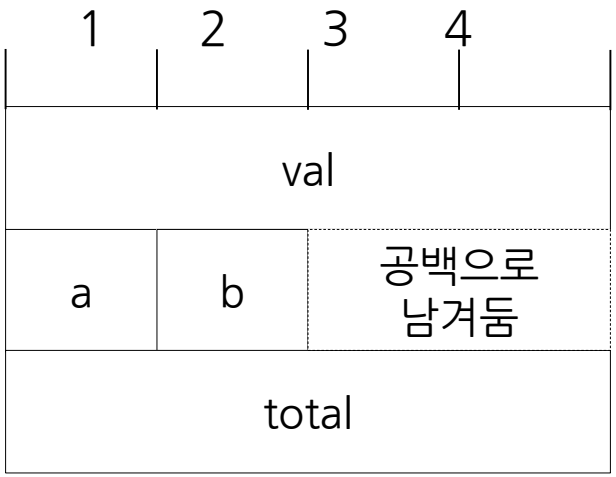

0
4
8
12
16
int
CPU
0
4
8
12
16
int
메모리


1
2
3
4
a
val (1-3)
val (4)
b
total (1-2)
total (3-4)


1
2
3
4
a
공백으로 남겨둠
val
b
공백으로 남겨둠
total

구조체와 메모리





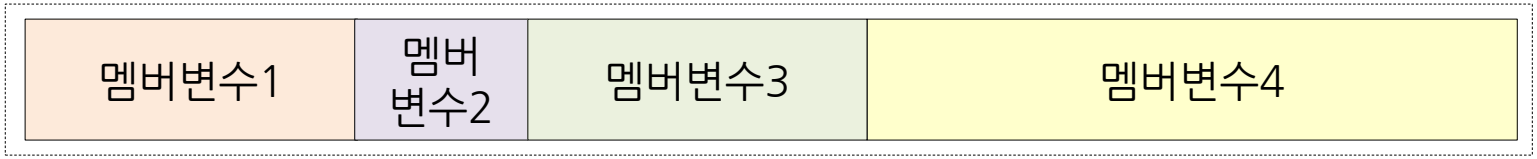
구조체와 메모리

- **컴파일러의 역할**
  - 기계어로 번역
  - 변수의 공간 할당
- **구조체의 효율적 공간 할당도 컴파일러의 일이다.**
- **컴파일러에게 공간 할당을 다르게 하라고 요구할 수 있다.**
  - `#pragma pack(1)`
  - `#pragma pack(4)  // default for 32bit`
  - 언제 사용할까?

공용체

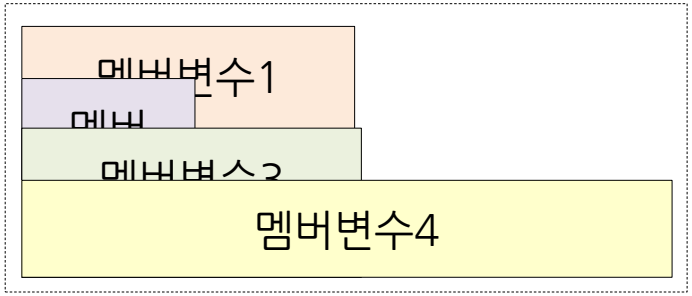
- 구조체의 일종
- 멤버 변수들이 공간을 공유한다

구조체



공간의 크기 = 멤버변수1 + 멤버변수2 + 멤버변수3 + 멤버변수4

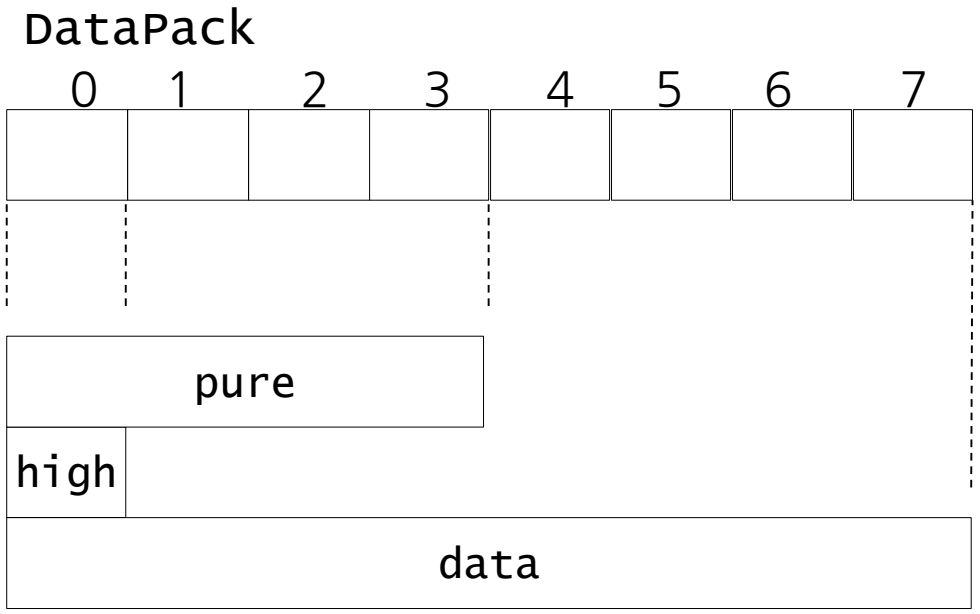
공용체



공간의 크기 = Max(멤버변수1 + 멤버변수2 + 멤버변수3 + 멤버변수4)

공용체의 예

```
union DataPack {
    int    pure;
    char   high;
    double data;
};
```

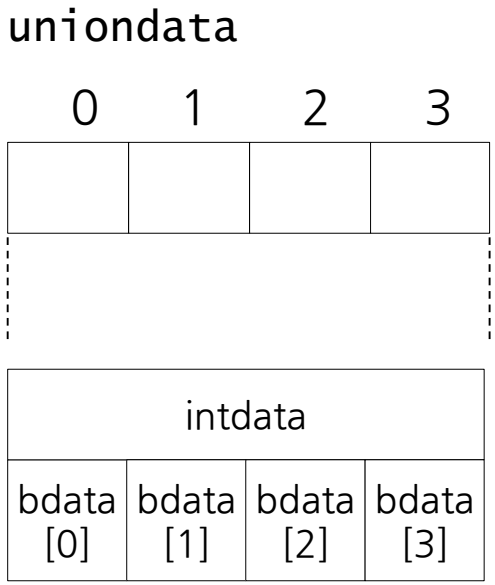


공용체의 예

```
union DataPack {
      int    pure;
      char   high;
      double data;
};

int  main( )
{
      union DataPack dp;

      dp.pure = 10;
      printf("Member : %d\n", dp.pure );
      printf("Member : %c\n", dp.high );
      printf("Member : %f\n", dp.data );
}
```

공용체의 예

```
union uniondata {
      char bytedata[4];
      int  intdata;
} udata ;

udata.intdata = 0x12345678;
printf("%x\n", udata.bdata[0]);
printf("%x\n", udata.bdata[1]);
printf("%x\n", udata.bdata[2]);
printf("%x\n", udata.bdata[3]);
```



비트 구조체

- 구조체 내에 비트 단위 멤버 변수를 넣을 수 있다.

```
struct _B {
    char  topbit : 3;
    char  midbit : 2;
    char  bottombit : 3;
} byte;

sizeof(struct _B) == 1
```

```
union {
    char  s;
    struct {
        char  topbit : 3;
        char  midbit : 2;
        char  botbit : 3;
    } byte;
} discomp;

scanf("%c", &discomp.s);
printf("%d\n",
        discomp.byte.topbit);
```

비트 단위는 구조체, 공용체에서만 사용 가능

중첩 구조체

- 구조체 내에 구조체 변수를 멤버로 가진 경우

```
struct _TIME {
    int   yymmdd;
    int   hhmmss;
}

struct BOOK {
     char   title[20];
     struct  _TIME  in_stock;
};

struct BOOK oldbook;

oldbook.in_stock.yymmdd = 20160901;
```

중첩 구조체

- `stdptr->prev->kor = 100;` 을 해석하면?

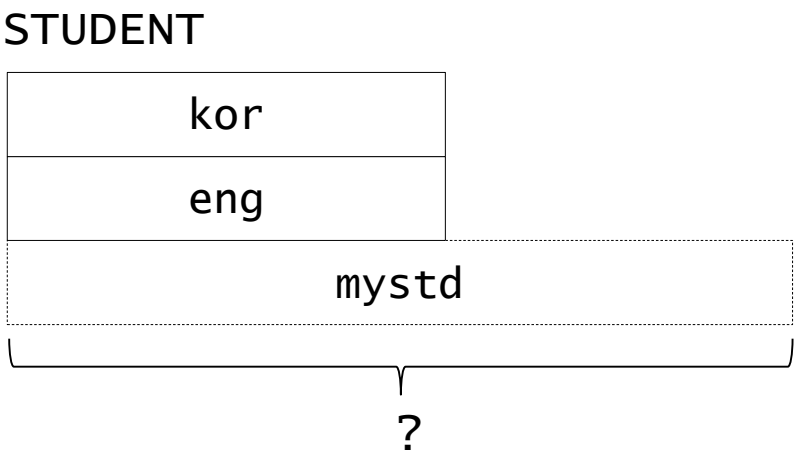
중첩 구조체

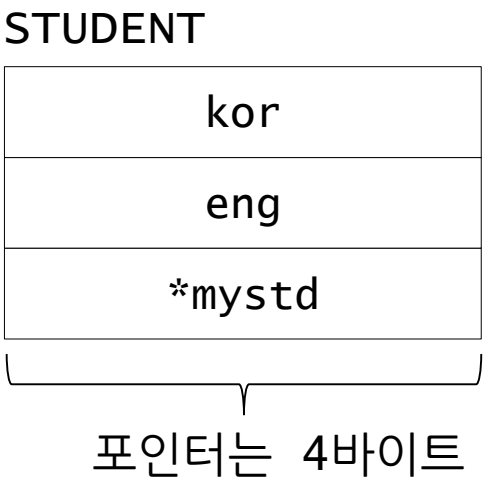
```
struct money {
        char   type;    // 공용체중 어느 멤버를 쓰고 있는지 나타냄
        union {
                double  korea_won;
                double  usa_dollar;
                double  japan_yen;
        };
};

int  main()
{
        struct money m;

        m.korea_won = 1000;         // 값을 넣을 때에는
        m.type = 1;                 // 화폐의 종류를 지정함

        switch (m.type) {
        case 1: printf("%.2f\n", m.korea_won); break;   // korea won
        case 2: printf("%.2f\n", m.usa_dollar); break;  // usa_dollar
        case 3: printf("%.2f\n", m.japan_yen); break;   // japan_yen
        }  // end of switch
}
```

자기 참조 구조체

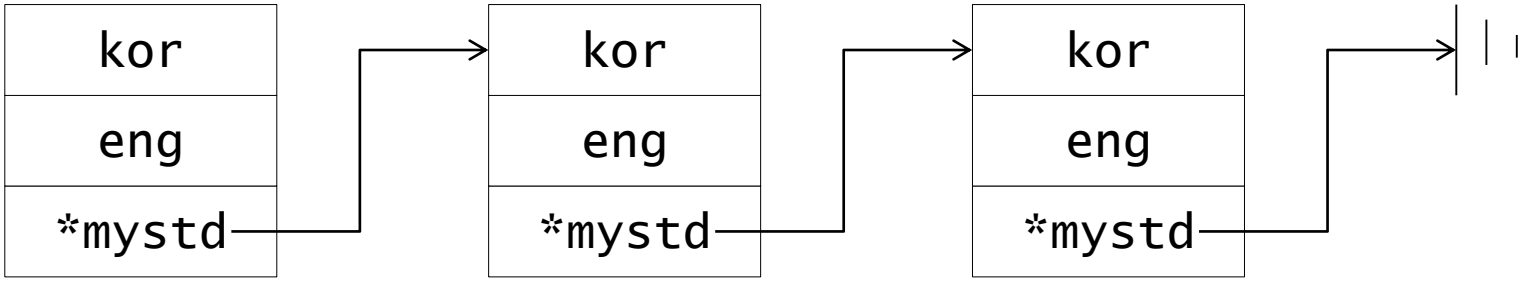
```
struct STUDENT {
    int kor;
    int eng;
    struct STUDENT mystd;
};
// error
```

```
struct STUDENT {
    int kor;
    int eng;
    struct STUDENT *mystd;
};
// no error
```





자기 참조 구조체의 예



링크드리스트(Linked List) 라고 한다.
매우 중요하지만, 동적 할당 이후에.

열거형

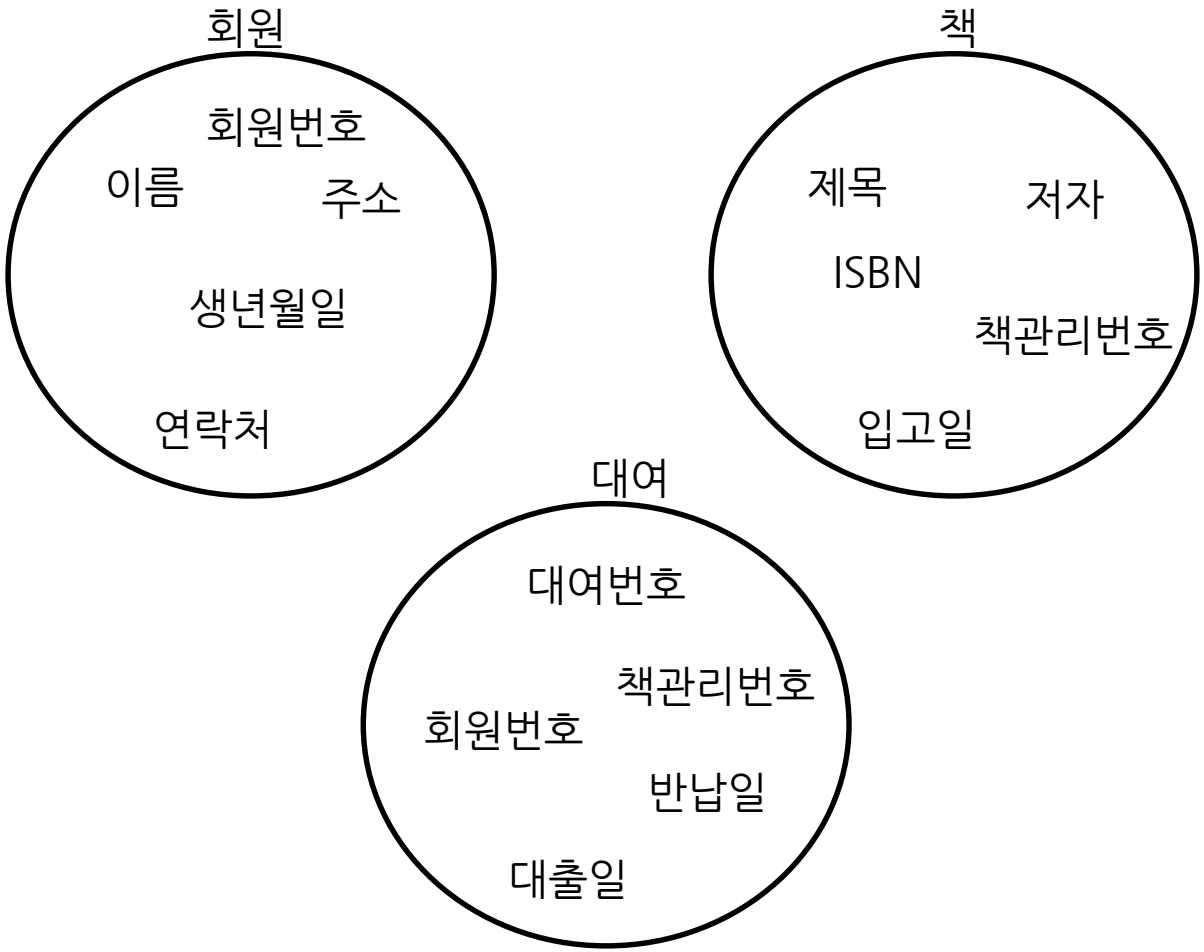
- 상수로 구성된 새로운 자료형
- enum 자료형의 이름 {상수1, 상수2, 상수3, …, 상수n } ;
  - enum GABABO {GAWI, BAWI, BO };
- 변수의 선언
  - enum GABABO user, computer;
  - user = GAWI;
- enum은 int형과 같다.
  - GAWI = 0, BAWI = 1, BO = 2

enum Days {Sat, Sun, Mon, Tue, Wed, Thu, Fri};

enum months_t { JANUARY=1, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER};

구조체의 활용

- 회원 관리 프로그램



구조체의 활용

- 구조체 데이터의 Quick Sort

```
size = sizeof ( struct STUDENT) ;
qsort  ( student, 100, size, compare );

int compare (const void * a, const void * b)
{
  return ( (struct  STUDENT *)a->number -
                                  (struct STUDENT *)b->number );
}
```