구조체 Part 3

Union (공용체)

- 공용체(union)
  - 하나의 동일한 메모리 영역을 2개 이상의 변수가 공유
  - 어느 시점에서 보면 하나의 변수만이 그 공간을 사용한다
  - 공용체를 선언하고 사용하는 방법은 구조체와 아주 비슷
- 선언 형식은 keyword를 union으로 사용하고 나머지는 구조체와 동일

```
union  tag_name  {
    data_type       member_name ;
    data_type       member_name ;
    …
} ;
```

- 그러므로 공용체 변수가 선언되면 멤버 중 가장 길이가 긴 변수를 저장할 만큼 메모리 공간이 할당된다

- struct와 union의 큰 차이점은 원하는 시점에 메모리 상에 존재하는 데이터의 상황에 있다.
- struct로 정의된 데이터는 어느 시점이든 구조체 멤버들 "모두" 메모리 상에 존재하고 있으며 각각 접근이 가능하다
- union으로 정의된 데이터는 어느 시점에 멤버들 중 "하나만" 메모리 상에 존재한다.
  - 그리고 그것을 어떤 type으로 해석하느냐는 프로그래머에 의존한다

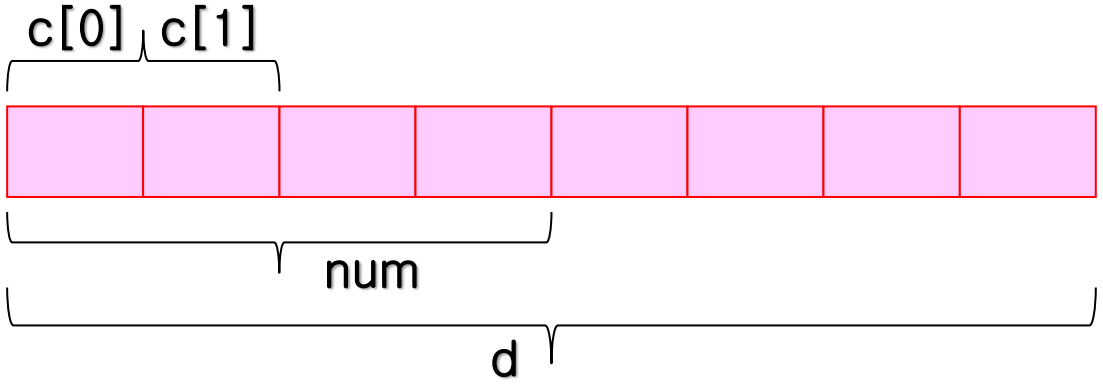
```
union example {
    char c;                    // 같은 기억 공간 공유
    int i;              // 같은 기억 공간 공유
};
```

예

```
union u-type {
    int  num ;
    char c[2] ;
    double d ;
}
union u-type  sample ;
```

```
sample.num = 10 ;
tmp = sample.c[0] ;
sample.c[0] = sample.c[1] ;
sample.c[1] = tmp ;
tmp = sample.num ;
```

tmp **값은** ?



- union**은 어느 데이터를** 2**가지 이상의 방법으로 처리하고자 할 때 매우 유용한 방법**

```
union number {
    int  i ;
    float d ;
}
union number  n ;

main() {
        n.i = 4444 ;
        printf("i : %10d \t f : %16.10e \n", n.i, n.d ) ;
        n.f = 4444.0 ;
        printf("i : %10d \t f : %16.10e \n", n.i, n.d ) ;
}
```

```
i : 4444          f : 0.6227370375e-41
i : 1166729216    f : 4.44400000000e+03
```

예제

```
#include <stdio.h>

union example {
        int i;
        char c;
};

int main(void)
{
        union example v;

        v.c = 'A';
        printf("v.c:%c   v.i:%i\n", v.c, v.i );

        v.i = 10000;
        printf("v.c:%c   v.i:%i\n", v.c, v.i);

}
```

공용체 선언

공용체 변수 선언.

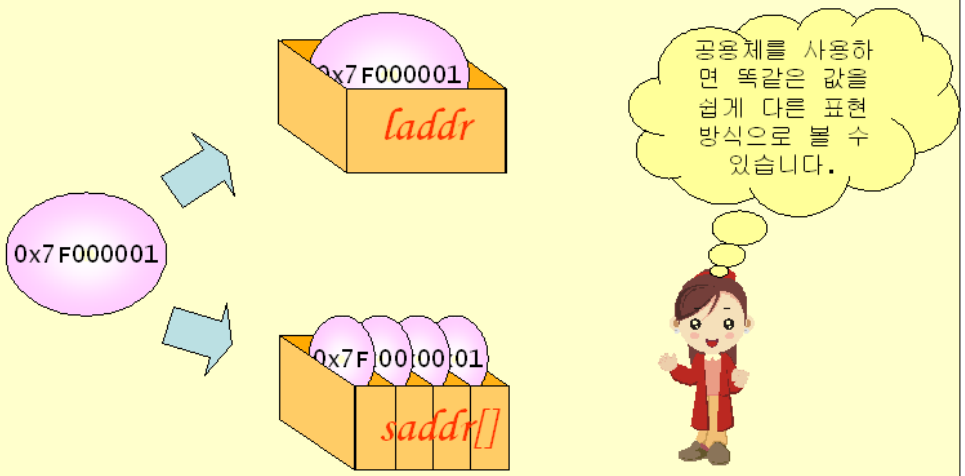
char 형으로 참조.

int 형으로 참조.

```
v.c:A v.i:65
v.c:□v.i:10000
```

IP 주소 예제

```
#include <stdio.h>

union ip_address {
      unsigned long laddr;
      unsigned char saddr[4];
};

int main(void)
{
      union ip_address addr;

      addr.saddr[0] = 1;
      addr.saddr[1] = 0;
      addr.saddr[2] = 0;
      addr.saddr[3] = 127;

      printf("%x\n", addr.laddr);

      return 0;
}
```




```
7f000001
```

타입 필드를 같이 사용하는 예

```
#include <stdio.h>

#define STU_NUMBER 1
#define REG_NUMBER 2

struct  student {
        int type;
        union {
                int stu_number;          // 학번
                char reg_number[15];     // 주민등록번호
        } id;
        char name[20];
};
void print(struct student s)
{
        switch(s.type)
        {
                case STU_NUMBER:
                        printf("학번: %d\n", s.id.stu_number);
                        printf("이름: %s\n", s.name);
                        break
                case REG_NUMBER:
                        printf("주민등록번호: %d\n", s.id.reg_number);
                        printf("이름: %s\n", s.name);
                        break
                default:
                        printf("타입오류\n");
                        break
        }
}
```

타입 필드를 같이 사용하는 예

```
int main(void)
{
	struct student s1, s2;

	s1.type = STU_NUMBER;
	s1.id.stu_number = 20070001;
	strcpy(s1.name, "홍길동");

	s2.type = REG_NUMBER;
	strcpy(s2.id.reg_number, "860101-1058031");
	strcpy(s2.name, "김철수");

	print(s1);
	print(s2);

	return 0;
}
```

```
학번: 20070001
이름: 홍길동
주민등록번호: 1244868
이름: 김철수
```

Enumeration(열거형)

- *열거형(enumeration)* 이란 변수가 가질 수 있는 값들을 미리 열거해 놓은 자료형으로, 열거형 변수는 열거된 이외의 값을 가질 수 없다.
- (예) 요일을 저장하고 있는 변수는 { 일요일, 월요일, 화요일, 수요일, 목요일, 금요일, 토요일 } 중의 하나의 값만 가질 수 있다.
- 열거형은 *enum* 이라는 키워드를 사용하여 만들어진다.

```
enum  tag_name  {
    enumeration_list
};
```

- 열거형 변수를 사용하는 주된 이유는 self-documenting code 작성을 용이하게 하고, 프로그램의 구성을 명료하게 하도록 하는데 있다

```
enum days1 { MON, TUE, WED, THU, FRI, SAT, SUN };
enum days2 { MON=1, TUE, WED, THU, FRI, SAT, SUN };
enum days3 { MON=1, TUE=2, WED=3, THU=4, FRI=5, SAT=6, SUN=7 };
enum days4 { MON, TUE=2, WED=3, THU, FRI, SAT, SUN };

enum days1 d;
d = WED;
```

열거형의 예

```
enum days  { SUN, MON, TUE, WED, THU, FRI, SAT };

enum colors { white, red, blue, green, black };

enum boolean { 0, 1 };

enum months { JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC };

enum major { COMMUNICATION, COMPUTER, ELECTRIC, ELECTRONICS };

enum component { MAIN_BOARD, CPU, GRAPHIC_CARD, DISK, MEMORY };

enum levels { low = 1, medium, high };

enum CarOptions
{
   SunRoof = 0x01,
   Spoiler = 0x02,
   FogLights = 0x04,
   TintedWindows = 0x08,
}
```

열거형과 다른 방법과의 비교

정수 사용	기호 상수	열거형

switch(code) {
  case 1:
  printf("LCD TV\n");
  break;
  case 2:
  printf("PDP TV\n");
  break;
}
	#define LCD 1
	#define PDP 2
	
	switch(code) {
	  case LCD:
	  printf("LCD TV\n");
	  break;
	  case PDP:
	  printf("PDP TV\n");
	  break;
	}
		enum tvtype { LCD, PDP };
		enum tvtype code;
		
		switch(code) {
		  case LCD:
		  printf("LCD TV\n");
		  break;
		  case PDP:
		  printf("PDP TV\n");
		  break;
		}

컴퓨터는 알기 쉬우나 사람은 기억하기 어렵다.	기호 상수를 작성할 때 오류를 저지를 수 있다.	컴파일러가 중복이 일어나지 않도록 체크한다.

예제

```
// 열거형
#include <stdio.h>

enum days { MON, TUE, WED, THU, FRI, SAT, SUN };

char *days_name[] = {
"monday", "tuesday", "wednesday", "thursday", "friday",
    "saturday", "sunday" };

int main(void)
{
    enum days d;

    for(d=MON; d<=SUN; d++)
    {
        printf("%d번째 요일의 이름은 %s입니다\n", d, days_name[d]);
    }
}
```

```
0번째 요일의 이름은 monday입니다
1번째 요일의 이름은 tuesday입니다
2번째 요일의 이름은 wednesday입니다
3번째 요일의 이름은 thursday입니다
4번째 요일의 이름은 friday입니다
5번째 요일의 이름은 saturday입니다
6번째 요일의 이름은 sunday입니다
```

예제

```
#include <stdio.h>
enum tvtype { tube, lcd, plasma, projection };

int main(void)
{
	enum tvtype type;

	printf("TV 종류 코드를 입력하시오: ");
	scanf("%d", &type);
	switch(type)
	{
		case tube:
			printf("브라운관 TV를 선택하셨습니다.\n");
			break;
		case lcd:
			printf("LCD TV를 선택하셨습니다.\n");
			break;
		case plasma:
			printf("PDP TV를 선택하셨습니다.\n");
			break;
		case projection:
			printf("프로젝션 TV를 선택하셨습니다.\n");
			break;
		default:
			printf("다시 선택하여 주십시오.\n");
			break;
	}
	return 0;
}
```

```
TV 종류 코드를 입력하시오: 3
프로젝션 TV를 선택하셨습니다.
```