2023.03.29(수)

마이크로프로세서

2402 김나영

구조체와 공용체

mic_0329 > C project.c > __unnamed_union_09d6_1 > __unnamed_struct_09d6_2

```c
//구조체와 공용체(4)
#include<stdio.h>
typedef union{
    unsigned int val;
    struct {//익명 구조체(이름x) 안쓰면 에러남.
        unsigned char r;
        unsigned char g;
        unsigned char b;
        unsigned char a;//투명도
    }rgba;
}color;

int main(){
    color sample;
    sample.val = 0xffffff;
    sample.rgba.b = 0x00;
    printf("%x\n", sample.val);
    return 0;
}
```

unsigend : 비트필드 자료형

열거형

mic_0329 > C project02.c > main(void)

```c
//열거형
#include <stdio.h>
typedef enum Week{
    SUN = 0,//0부터 증감함
    MON,
    TUE,
    WED,
    THU,
    FRI,
    FAT
}Week;
int main(void){
    Week ju;
    ju = TUE;
    printf("%d\n", ju);
    return 0;
}
```

mic_0329 > C project03.c > main(void)

```c
//열거형
#include <stdio.h>
typedef enum subject_code{
    LINUX = 3,
    MICRO= 5,
    TROJECT = 8
}Code;

int main(void){
    Code emb;
    emb = LINUX;
    switch (emb)
    {
        case 3:
            printf("LINUX\n");
            break;

        case 5:
            printf("MICRO\n");
            break;
        case 8:
            printf("PROJECT\n");
            break;
    }
    return 0;
}
```

키값과 밸류로 이뤄짐.
키 값에
따라 값이 나옴.

비트필드

(기계 친화적)

- 구조체 멤버는 각 자료형 크기만큼 공간 차지
- 구조체 멤버를 비트 단위로 저장가능
- 실수는 사용X
- 자료형(unsigend)를 사용함

비트필드

mic_0329 > project04.c > main(void)

```c
//비트필드
#include <stdio.h>
typedef struct bf{
    unsigned int a:1;
    unsigned int b:3;
    unsigned int c:7;
}Bf;
int main(void){
    Bf t;
    t.a = 1;
    t.b = 4;
    t.c = 100;
    printf("%u\n", t.a);//부호없는 10진수
    printf("%u\n", t.b);
    printf("%u\n", t.c);
    printf("%ld\n", sizeof(t));//int 하나를 쪼개어 썼다
}
```

필드의 크기
a : 1비트 b : 3비트 c : 7비트

구조체 변수 t의 메모리 :
총 4바이트(32비트)

비트필드

d1의 byte 멤버에 0xf0 값을 할당
(11110000(2진수), 상위 4비트가 모두 1)

d1.byte의 값은 11110001(2진수)
= 0xf1

mic_0329 > C project05.c > ...

```c
//비트필드
#include <stdio.h>
#include <stdint.h>

typedef union date{
    uint8_t byte;
    struct{
        uint8_t b0:1;
        uint8_t b1:1;
        uint8_t b2:1;
        uint8_t b3:1;
        uint8_t b4:1;
        uint8_t b5:1;
        uint8_t b6:1;
        uint8_t b7:1;
    }bf;

}Date;

int main(){
    Date d1;
    d1.byte = 0xf0;
    d1.bf.b0 = 1;
    printf("data = 0x%x\n", d1.byte);
}
```

비트 연산자

OR 연산 : 11111(2진수) => 31(10진수)

AND 연산: 100(2진수) => 4(10진수)

XOR 연산: 11011(2진수) => 27(10진수)

비트를 반전: 11110000(2진수) => -16(10진수)

mic_0329 > C project06.c > main()

```c
//비트연산
#include <stdio.h>
int main(){
    int num1 = 15;
    int num2 = 20;
    int num3[4]={num1 | num2, num1 & num2, num1 ^ num2, ~num1};
    for (int i=0; i<4; i++){
        printf("연산결과 : %d\n",num3[i]);
    }
    return 0;
}
```

2진수로

num1 : 1111

num2 : 10100

시프트 연산자

mic_0329 > C project07.c > main()

```c
#include <stdio.h>
int main(){
    int num1 = 15;
    int result1 = num1 << 1;
    int result2 = num1 << 2;
    int result3 = num1 << 3;
    printf("1칸 이동 결과 : %d\n",result1);
    printf("2칸 이동 결과 : %d\n",result2);
    printf("3칸 이동 결과 : %d\n",result3);

    result1 = num1 >> 1;
    result2 = num1 >> 2;
    result3 = num1 >> 3;
    printf("1칸 이동 결과 : %d\n",result1);
    printf("2칸 이동 결과 : %d\n",result2);
    printf("3칸 이동 결과 : %d\n",result3);
    return 0;
}
```

<< 연산자 :
num1의 비트 값을 왼쪽으로 이동
빈 비트 자리에는 0이 채워짐
(2배씩 증가)
>> 연산자 :
num1의 비트 값을 오른쪽으로 이동
양수 : 0
음수 : 1로 채워짐

통신 방법(단방향, 반이중, 전이중)

단방향 : 송신측 대답X 예) 라디오
반이중 : 동시에는 X but 양방향 소통 예) 무전기
전이중 : 동시에 가능, 양방향 소통 예) 전화

동지식 통신
-장점 : 많은 양의 데이터를 신속히 전송하려 할 때 적합
-단점 : 비동기 전송보다 더 복잡함, 비용이 높음.

비동기식 통신
-장점 : 발신가자 편할때 데이터를 전송
-단점 : 상대적인 속도가 느림

UART

- 범용 비동기화 송수신기
- 1-bit 씩 전송하고, 두 시스템의 통신 속도 일치 해야함
- 출력되는 TxD핀은 상대 시스템의 입력핀인 RxD와 서로 연결

I2C

- 반 이중통시느 동기 통신
- 직렬버스로 지속의 장치 연결에 사용
- 직렬데이터 전송을 위한 SDA와 클럭을 전달하는 SCL두개의 핀으로 양방향 통신 가능

SPI

- 전이중 통신, 동기 통신
- 마스터와 슬레이브 1대1 통신가능