



C언어 – HW4

임베디드스쿨1기

Lv1과정

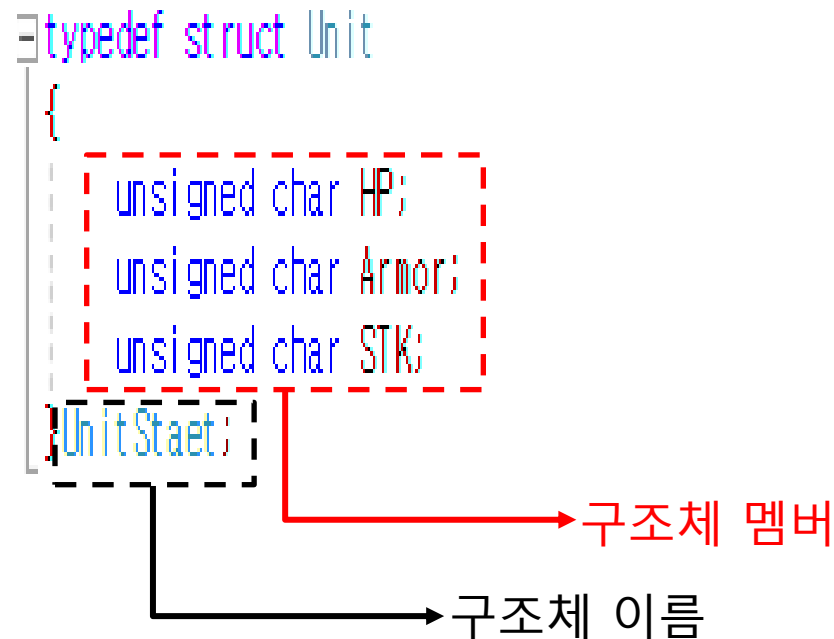
2020. 08. 11

손표훈

1. 구조체

(1) 구조체 : 구조체는 다양한 데이터 타입을 하나로 묶어 "카테고리화"를 할 수 있다.

Ex) 유닛의 스탯



```
int main(void)
{
    UnitStaet Marine;
    UnitStaet Medic;

    Marine.Armor = 1;
    Marine.HP = 50;
    Marine.STK = 10;

    Medic.Armor = 1;
    Medic.HP = 50;
    Medic.STK = 0;

    printf("Marine's Armor = %d\n", Marine.Armor);
    printf("Marine's HP = %d\n", Marine.HP);
    printf("Marine's STK = %d\n", Marine.STK);

    printf("\nMedic's Armor = %d\n", Medic.Armor);
    printf("Medic's HP = %d\n", Medic.HP);
    printf("Medic's STK = %d\n", Medic.STK);

    return 0;
}
```

-> 위와 같이 "유닛의 스탯"이라는 구조체를 선언하고 구조체 변수를 유닛의 이름으로 선언하여 구조체 멤버에 접근하여 사용한다.

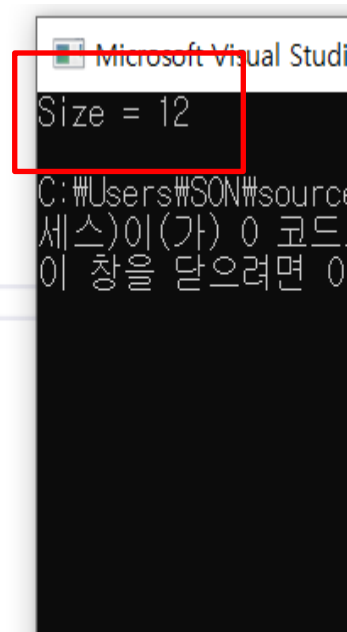
1. 구조체

(2) 구조체의 메모리 할당 : 아래 유닛스택의 구조체는 얼마나 메모리 공간을 차지할까?

```
#include <stdio.h>

struct Unit
{
    unsigned char HP;
    unsigned char MP;
    unsigned char STK;
    unsigned char ARMOR;
    unsigned char LEVEL;
    unsigned int PWR;
}Staets;

int main(void)
{
    printf("Size = %d\n", sizeof(Staets));
    return 0;
}
```



0x1000	HP
0x1001	MP
0x1002	STK
0x1003	ARMOR
0x1004	LEVEL
0x1005	PWR
0x1009	

- > 구조체는 연속된 메모리 공간을 할당 받는다.
- > 먼저 선언된 멤버 변수부터 낮은 메모리 주소에 할당 된다.
- > char형 변수 3개가 멤버니까 9byte가 될 것 같지만 아니다... 12byte가 할당된다..

1. 구조체

```
#include <stdio.h>

typedef struct
{
    unsigned char HP;
    unsigned char Armor;
    unsigned char STK;
}UnitStaet;

int main(void)
{
    UnitStaet Marine;

    printf("HP address : %x\n", &Marine.HP);
    printf("Armor address : %x\n", &Marine.Armor);
    printf("STK address : %x\n", &Marine.STK);
    return 0;
}
```

Microsoft Visual Studio 디버그

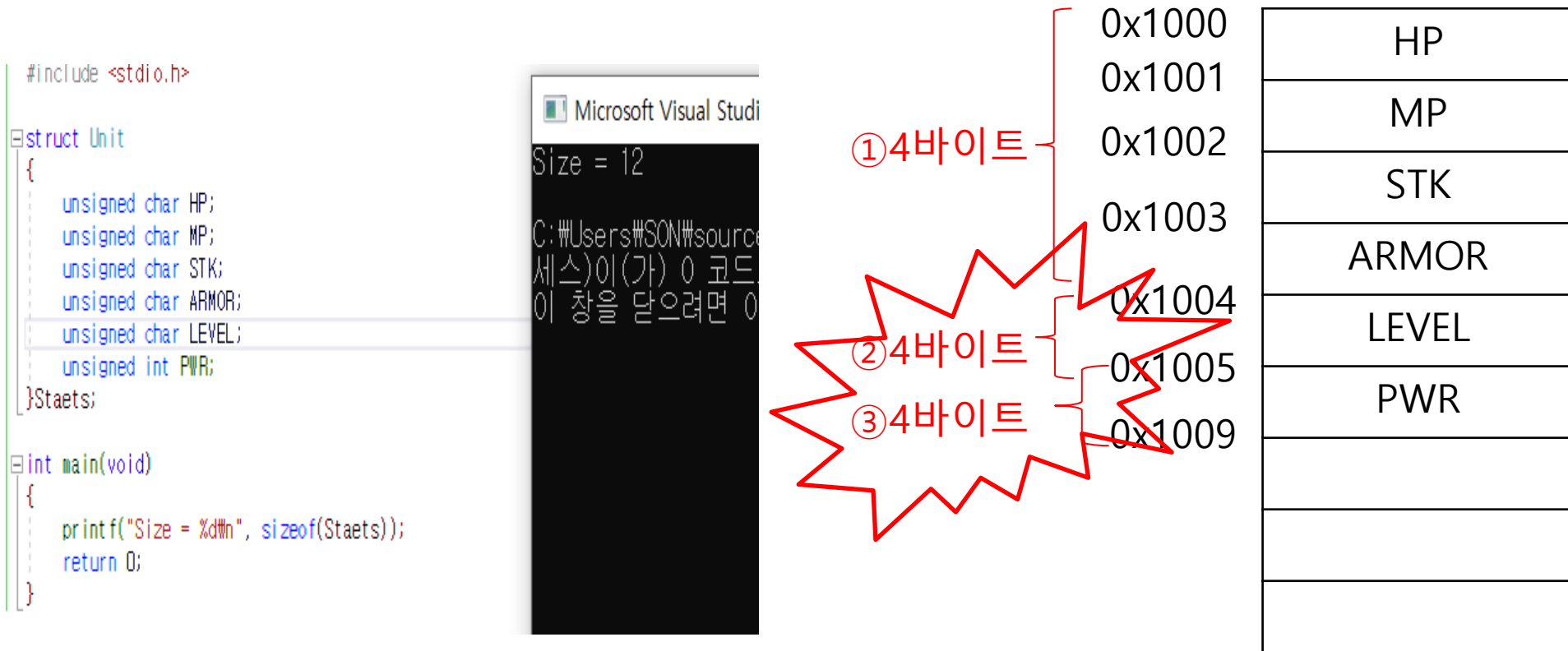
HP address : faf788
Armor address : faf789
STK address : faf78a

C:\Users\SON\source\repo
이 창을 닫으려면 아무 키

-> 선언된 멤버 순서에 따라 차례대로 주소가 할당된다.

1. 구조체

(3) 구조체의 바이트 패딩 : 구조체 멤버 변수를 CPU가 한번에 읽을 수 있도록, 컴파일러가 레지스터 크기에 맞게 바이트를 밀어주는 최적화 작업이다.



- > Staets구조체는 9byte의 멤버를 가지고 있다.
- > 멤버에 접근하기 위해 HP, MP, STK, ARMOR 4byte를 읽는데 문제 없지만, LEVEL부터 4byte를 읽고, 다시 PWR부터 4byte를 읽는 3번 구조체 메모리에 접근해야 멤버 전체에 접근 할 수 있다. 총 12byte를 읽음

1. 구조체

- > 이런 비효율적인 메모리 접근을 막기 위해 컴파일러가 패딩 바이트를 넣는다.
- > LEVEL과 PWR사이에 패딩 바이트를 넣어 구조체 전체 크기를 12byte로 만든다.

```
#include <stdio.h>

struct Unit
{
    unsigned char HP;
    unsigned char MP;
    unsigned char STK;
    unsigned char ARMOR;
    unsigned char LEVEL;
    unsigned int PWR;
}Staets;

int main(void)
{
    printf("Size = %d\n", sizeof(Staets));
    return 0;
}
```

Microsoft Visual Studio

Size = 12

C:\Users\SON\source
세스)이(가) 0 코드
이 창을 닫으려면 0

① 4바이트

② 4바이트

③ 4바이트

0x1000	HP
0x1001	
0x1002	MP
0x1003	
	STK
	ARMOR
0x1004	LEVEL
0x1005	empty
0x1006	empty
0x1007	empty
0x1008	
0x100C	PWR

- > 32bit시스템의 레지스터 크기(4byte)에 맞게 패딩을 해주면 구조체 멤버에 접근할 때 한번에 접근 할 수 있다.

1. 구조체

-> 패딩의 크기는 컴파일러마다 다르지만 보통 시스템의 크기에 따른다.

32bit = 4byte, 64bit = 8byte 크기로 패딩이 발생!!

(4) 패딩은 메모리 누수를 발생한다.

-> 구조체의 크기가 크다면, 패딩으로 인해 다른 메모리 영역을 침범한다.

-> 이를 방지하기 위해 `#pragma pack(바이트 사이즈)`라는 데이터 패킹 전처리기를 선언한다.

패킹 사이즈1로 설정

```
#include <stdio.h>
```

```
#pragma pack(1)
```

```
struct Unit
```

```
{
```

```
    char C; 1byte
```

```
    unsigned char A[8]; 8byte
```

```
    float B; 4byte
```

```
}Staets;
```

```
int main(void)
```

```
{
```

```
    printf("Size = %d\n", sizeof(Staets));
```

```
    return 0;
```

```
}
```

Microsoft Visu

Size = 13

C:\Users\SON\

이 창을 닫으려

1. 구조체

- > 임베디드라는 범용 PC환경과 다른 하드웨어가 한정적인 분야에선 패딩이 치명적!
- > #pragma pack(size)전처리기를 사용도 하지만 공용체 + 비트필드로 더 많이 사용하며, 메모리 할당과 레지스터 설정을 직관적으로 이해시켜 준다.

2. 공용체

- (1) 구조체와 형태는 유사하지만, 메모리 할당에 있어서 다름.
공용체의 메모리 크기는 멤버 중 데이터의 크기가 가장 큰 멤버의 크기에 따른다.

```
#include <stdio.h>

union Data
{
    int i;
    float f;
    char str[20];
};

int main(void)
{
    union Data data;

    printf("Memory Size occupied by data : %d\n", sizeof(data));
    printf("Member str size : %d\n", sizeof(data.str));

    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

```
Memory Size occupied by data : 20
Member str size : 20

C:\Users\SON\source\repos\test2\Debug
이 창을 닫으려면 아무 키나 누르세요.
```

3. 비트 필드

- (1) 비트필드는 구조체와 동일하며, 메모리 할당을 비트단위로 설정할 수 있다.
- (2) 임베디드 같은 한정된 메모리 자원을 사용하는 시스템에서 메모리 절약에 효과적이다.
- (3) 공용체와 같이 사용하면 레지스터 설정에 있어 가독성이 좋다.

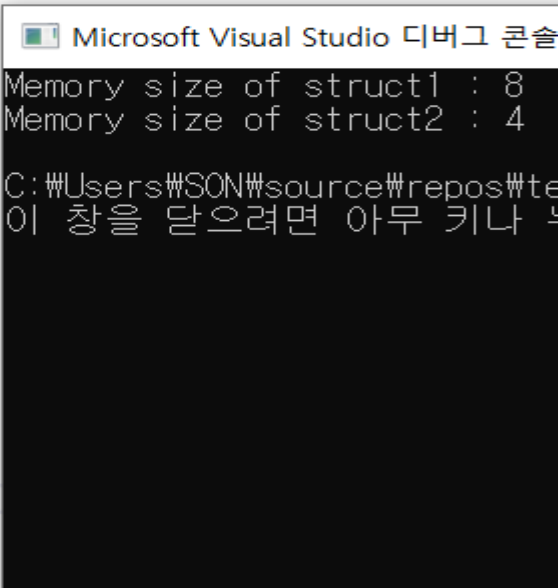
```
#include <stdio.h>

struct {
    unsigned int width;
    unsigned int height;
}struct1;

struct {
    unsigned int width : 1;
    unsigned int height : 1;
}struct2;

int main(void)
{
    printf("Memory size of struct1 : %d\n", sizeof(struct1));
    printf("Memory size of struct2 : %d\n", sizeof(struct2));

    return 0;
}
```



- > 비트필드를 사용하지 않았을 경우 멤버 데이터 총 크기 만큼 할당이 된다.
- > 비트필드를 사용하게 되면 2bit만 할당했으므로 멤버 데이터 타입 4byte만 할당해도 충분하기 때문에 4byte만 할당된다.

4. 구조체와 비트필드, 공용체

```
typedef unsigned short UINT16;

typedef union
{
    UINT16 All;
    struct
    {
        UINT16 LoByte : 8;
        UINT16 HiByte : 8;
    } Byte;

    struct
    {
        UINT16 VAUXLow : 1;
        UINT16 VAUXoverCUR : 1;
        UINT16 CAN5VThShutdown : 1;
        UINT16 CAN5VUV : 1;
        UINT16 CAN5VOC : 1;
        UINT16 VSENSELow : 1;
        UINT16 VSUPUV : 1;
        UINT16 IDDOcNorm : 1;

        UINT16 rvd1 : 3;
        UINT16 VDDThShutdown : 1;
        UINT16 rvd2 : 1;
        UINT16 RSTLow : 1;
        UINT16 VSUPBatFail : 1;
        UINT16 IDDOcLp : 1;
    } Bit;
} USBCREGFlag;

int main(void)
{
    USBCREGFlag reg1;

    reg1.All = 0x1155; //0001 0001 0101 0101

    printf("HiByte = %x\n", reg1.Byte.HiByte);
    printf("LoByte = %x\n", reg1.Byte.LoByte);

    printf("VAUXLow = %x\n", reg1.Bit.VAUXLow);
    printf("VAUXoverCUR = %x\n", reg1.Bit.VAUXoverCUR);
    printf("CAN5VThShutdown = %x\n", reg1.Bit.CAN5VThShutdown);
    printf("CAN5VUV = %x\n", reg1.Bit.CAN5VUV);
    printf("CAN5VOC = %x\n", reg1.Bit.CAN5VOC);
    printf("VSENSELow = %x\n", reg1.Bit.VSENSELow);
    printf("VSUPUV = %x\n", reg1.Bit.VSUPUV);
    printf("IDDOcNorm = %x\n", reg1.Bit.IDDOcNorm);

    printf("rvd1 = %x\n", reg1.Bit.rvd1);
    printf("VDDThShutdown = %x\n", reg1.Bit.VDDThShutdown);
    printf("rvd2 = %x\n", reg1.Bit.rvd2);
    printf("RSTLow = %x\n", reg1.Bit.RSTLow);
    printf("VSUPBatFail = %x\n", reg1.Bit.VSUPBatFail);
    printf("IDDOcLp = %x\n", reg1.Bit.IDDOcLp);

    return 0;
}
```

```
HiByte = 11
LoByte = 55
VAUXLow = 1
VAUXoverCUR = 0
CAN5VThShutdown = 1
CAN5VUV = 0
CAN5VOC = 1
VSENSELow = 0
VSUPUV = 1
IDDOcNorm = 0

rvd1 = 1
VDDThShutdown = 0
rvd2 = 1
RSTLow = 0
VSUPBatFail = 0
IDDOcLp = 0
```

4. 구조체와 비트필드, 공용체

```
typedef unsigned short UINT16;
```

```
typedef union
```

```
{
```

```
    UINT16 All;
```

```
    struct
```

```
    {
```

```
        UINT16 LoByte : 8;
```

```
        UINT16 HiByte : 8;
```

```
    }Byte;
```

```
    struct
```

```
    {
```

```
        UINT16 VAUXLow : 1;
```

```
        UINT16 VAUXoverCUR : 1;
```

```
        UINT16 CAN5VThShutdown : 1;
```

```
        UINT16 CAN5VUV : 1;
```

```
        UINT16 CAN5VOC : 1;
```

```
        UINT16 VSENSELow : 1;
```

```
        UINT16 VSUPUV : 1;
```

```
        UINT16 IDDOcNorm : 1;
```

```
        UINT16 rvd1 : 3;
```

```
        UINT16 VDDThShutdown : 1;
```

```
        UINT16 rvd2 : 1;
```

```
        UINT16 RSTLow : 1;
```

```
        UINT16 VSUPBatFail : 1;
```

```
        UINT16 IDDOcLp : 1;
```

```
    }Bit;
```

```
}USBCREGFlag;
```

