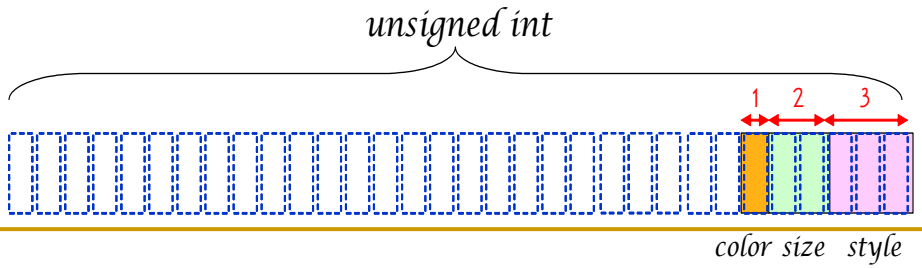
구조체 Part 4

비트 필드(bit field) 구조체

- bit field란 하나 이상의 비트들의 모임을 말하며, bit field를 멤버로 가지는 구조체

```
struct tag_name {
        type name ;  /* 일반 멤버 */
        type name : 비트수 ;   /* bit field */
        type name : 비트수 ;   /* bit field */
        ...
};
```

```
struct product {
        unsigned style : 3;
        unsigned size  : 2;
        unsigned color : 1;
};
```



예

```
1.  // 비트 필드 구조체
2.  #include <stdio.h>
3.
4.  struct product {
5.          unsigned style : 3;
6.          unsigned size  : 2;
7.          unsigned color : 1;
8.  };
9.
10. int main(void)
11. {
12.         struct product p1;
13.
14.         p1.style = 5;
15.         p1.size = 3;
16.         p1.color = 1;
17.
18.         printf("style=%d size=%d color=%d\n", p1.style, p1.size, p1.color);
19.         printf("sizeof(p1)=%d\n", sizeof(p1));
20.         printf("p1=%x\n", p1);
21.
22.         return 0;
23. }
```

```
style=5 size=3 color=1
sizeof(p1)=4
p1=ccccccfd
```

- 비트 필드를 사용하면 하나의 바이트 혹은 워드 내에 비트들을 특정 이름으로 접근할 수 있다
- 어떤 정보를 최소 단위의 메모리 공간으로 팩킹할 필요가 있을 때 매우 유용하다

```
struct telemetry {
        char fuel ;
        char radio ;
        char tv ;
        char water ;
        char food ;
        char waste ;
}

한 item 당 6 bytes 필요
```

일반 구조체 사용

```
struct telemetry {
        unsigned   fuel : 1 ;
        unsigned   radio : 1 ;
        unsigned   tv : 1;
        unsigned   water : 1;
        unsigned   food : 1;
        unsigned   waste : 1;
};

한 item 당 1 byte 필요
```

bit field 구조체 사용

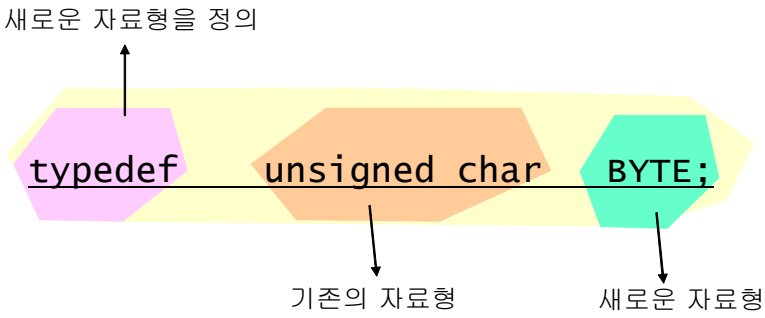
비트 필드 사용시 주의점

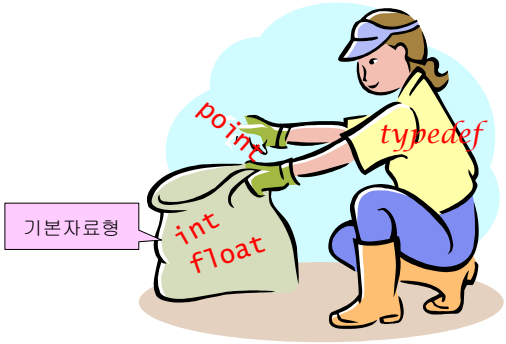
```
struct product {
    long   code;                // ① 일반 멤버도 가능하다.
    unsigned style   : 3;
    unsigned         : 5;       // ② 자리만 차지한다.
    unsigned size    : 2;
    unsigned color   : 1;
    unsigned         : 0;       // ③ 현재 워드의 남아있는 비트를 버린다.
    unsigned state   : 3;       // 여기서부터는 다음 워드에서 할당된다.
};
```

typedef

- typedef은 새로운 자료형(type)을 프로그래머가 정의(define)할 수 있도록 한다
- C의 기본 자료형을 확장시키는 역할도 가능

```
typedef      old_type      new_type;
```





typedef의 예

기본자료형	재정의된 자료형
int	INT32
short	INT16
unsigned int	UINT32
unsigned short	UINT16
unsigned char	UCHAR, BYTE
char	CHAR

```
typedef int  INT32;
typedef unsigned int  UINT32;

INT32 i;                    // int i;와 같다.
UINT32 k;                   // unsigned int k;와 같다.


typedef struct point {
            int x;
            int y;
} POINT;

POINT p,q;
```

```
typedef struct complex {
            double real;
            double imag;
} COMPLEX;

COMPLEX x, y;

typedef enum { FALSE, TRUE } BOOL;

BOOL condition;          // enum { FALSE, TRUE }
condition;

typedef char * STRING_PTR;

STRING_PTR p;            // char *p;
```

typedef과 #define 비교

- 이식성을 높여준다.
  - 코드를 컴퓨터 하드웨어에 독립적으로 만들 수 있다
  - (예) int형은 2바이트이기도 하고 4바이트, int형 대신에 typedef을 이용한 INT32나 INT16을 사용하게 되면 확실하게 2바이트인지 4바이트인지를 지정할 수 있다.
    - Typedef int INT32
    - Typedef short INT16
- #define을 이용해도 typedef과 비슷한 효과를 낼 수 있다. 즉 다음과 같이 INT32를 정의할 수 있다.
  - #define UINT32 unsigned int
  - typedef float VECTOR[2];// #define으로는 불가능하다.
- 문서화의 역할도 한다(self-documenting code)
  - typedef을 사용하게 되면 주석을 붙이는 것과 같은 효과

사용 예

- typedef char * string ;
- typedef int INCHES, FEET, YARDS ;
- Typedef struct point POINT ;
  - struct point x ; → POINT x ;
- typedef float vector[10] ;
  - vector x ; → float x[10] ;
- Typedef float MATRIX[10][10] ;
  - MATRIX m1, m2 ; → float m1[10][10], m2[10][10] ;
- typedef double (*pfd)(double) ;
  - pfd f ; → doule (*f)(double) ;
- typedef int (*pfi)(char *, char *) ;
  - pfi f ; → int (*f)(char *, char *) ;
  - pfi라는 type이 새로이 정의되는데, pfi는 2개의 char * 매개변수를 받아서 int 를 return 하는 함수를 가리키는 pointer 이다

예제

```c
#include <stdio.h>

typedef struct point {
	int x;
	int y;
} POINT;

POINT translate(POINT p, POINT delta);

int main(void)
{
	POINT p = { 2, 3 };
	POINT delta = { 10, 10 };
	POINT result;

	result = translate(p, delta);
	printf("새로운 점의 좌표는(%d, %d)입니다.\n", result.x, result.y);

	return 0;
}

POINT translate(POINT p, POINT delta)
{
	POINT new_p;

	new_p.x = p.x + delta.x;
	new_p.y = p.y + delta.y;

	return new_p;
}
```

```
새로운 점의 좌표는 (12, 13)입니다.
```