2.7 C 语言中的位域 2_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 2.7 C 语言中的位域 2 涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

7.C 语言中的位域 2

1. 一个位域存储在同一个字节中，如一个字节所剩空间不够存放另一位域时，则会从下一单元起存放该位域。也可以有意使某位域从下一单元开始。

```
struct info
{
    unsigned char a:4;
    unsigned char :4;
    unsigned char b:4;
    unsigned char c:4
};
```

在上面这个位域定义中，a 占第一字节的 4 位，后 4 位填 0 表示不使用，b 从第二字节开始，占用 4 位，c 占用 4 位。

1. 位域的宽度不能超过它所依附的数据类型的长度，成员变量都是有类型的，这个类型限制了成员变量的最大长度，后面的数字不能超过这个长度。

2. 位域可以是无名位域，这时它只用来作填充或调整位置。无名的位域是不能使用的。例如：

```
struct info
{
    int a:1;
    int  :2;
    int b:3;
    int c:2;
}st;
```

从以上分析可以看出，位域在本质上就是一种结构类型，不过其成员是按二进位分配的。

```
位域的使用和结构成员的使用相同，其一般形式为：
位域变量名.位域名
位域指针变量名->位域名
```

```
#include <stdio.h>

typedef struct bs{
        unsigned int a:1;
        unsigned int b:3;
        unsigned int c:4;
}bit_t;

int main()
{
    bit_t  bit;
    bit_t  *pbit;
    bit.a=1;
    bit.b=7;
    bit.c=15;
    printf("%d,%d,%d\n",bit.a,bit.b,bit.c);
    pbit=&bit;
    pbit->a=0;
    pbit->b&=3;
    pbit->c|=1;
```

```
    printf("%d,%d,%d\n",pbit->a,pbit->b,pbit->c);
}
```

- 如果相邻的两个位域字段的类型相同, 且其位宽之和小于或等于其类型的 sizeof() 大小, 则其后面的位域字段将紧邻前一个字段存储, 直到不能容纳为止;

例如: 一个位域变量有三个位域字段 a、b、c, 且类型完全相同, 位域字段 a 和 b 的位宽之和小于或等于其类型的 sizeof() 大小, 那么位域字段 c 紧接着位域字段 b 后面存储;

用法 1

```
struct Bit
{
unsigned char a:2;
unsigned char b:3;
unsigned char c:3;
}t1;

sizeof(t1) = 1;
```

2 + 3 + 3 = 8,1bytes 刚刚能存储

- 如果相邻的两个位域字段的类型相同, 且其位宽之和大于其类型的 sizeof() 大小, 则较大的位域字段将从下一个存储单元的起始地址处开始存放, 其偏移量恰好为其类型的 sizeof() 大小的整数倍;

比如: 如果位域字段 a 和 b 的位宽之和大于其类型的 sizeof() 大小, 则位域字段 b 就从下一个存储单元的

起始地址初开始存放, 其偏移量恰好是其类型的 sizeof() 大小的整数倍;

用法 2

```
struct Bit
{
    unsigned char a:2;
    unsigned char b:7;
    unsigned char c:4;
}t2;

sizeof(t2) = 3

a a | | | | | |
b b b b b b b |
c c c c | | | |
```

- 如果相邻的两个位域字段的类型不同, 则各个编译器的具体实现有差异, VC6 采取不压缩方式, GCC 和 Dev-C++ 都采用压缩方式;

用法 3:

```
struct bs
{
    unsigned int m: 12;
    unsigned char ch: 4;
    unsigned int p: 4;
}t1;

sizeof (t1) = 4;
```

用法 4:

```
struct bs
{
    unsigned int m: 12;
    unsigned int h;
    unsigned int p: 4;
}t1;
```

注意：

位域成员往往不占用完整的字节，有时候也不处于字节的开头位置，因此使用 & 获取位域成员的地址是没有意义的，C 语言也禁止这样做。地址是字节（Byte）的编号，而不是位（Bit）的编号。

```
struct bit
{
    unsigned int a: 6;
    unsigned int b: 12;
    unsigned int c: 4;
}bt;

求sizeof(bt)的大小。
```

全文完