9.C语言位域(位段)

有些信息在存储时，并不需要占用一个完整的字节，而只需占几个或一个二进制位。例如开关只有通电和断电两种状态，用0和1表示足以，也就是用一个二进位。所以C语言又提供了一种数据结构，称为位域或位段。

> 位域在应用开发中较少使用，你可以暂时跳过，遇到相关问题再回来温习。

所谓“位域”是把一个字节中的二进位划分为几个不同的区域，并说明每个区域的位数。每个区域有一个域名，允许在程序中按域名进行操作。

位域的定义和位域变量的说明

位域的定义与结构类似，形式为：

```
struct 位域结构名{
    位域列表
};
```

其中位域列表的形式为：

```
类型说明符 位域名: 位域长度;
```

例如：

```
01.   struct bs{
02.       int a:8;
03.       int b:2;
04.       int c:6;
05.   };
```

可以先定义位域再定义位域变量，也可以同时定义。例如：

```
struct bs data;
```

或者：

```
01.   struct bs{
02.       int a:8;
03.       int b:2;
04.       int c:6;
05.   } data;
```

说明data为bs变量，共占两个字节。其中位域a占8位，位域b占2位，位域c占6位。

对位域的几点说明

1) 一个位域必须存储在同一个字节中，不能跨两个字节。如果一个字节所剩空间不够存放另一位域时，应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。例如：

```
struct bs{
    unsigned a:4;
    unsigned :0;  //空域
    unsigned b:4;  //从下一单元开始存放
    unsigned c:4
}
```

在这个位域定义中，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。

2) 由于位域不允许跨两个字节，因此位域的长度不能大于一个字节的长度，也就是说不能超过8位二进位。

3) 位域可以无位域名，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 k{
    int a:1;
    int  :2;  //该2位不能使用
    int b:3;
    int c:2;
};
```

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;
```

位域允许用各种格式输出。

【示例】位域的使用。

```
#include <stdio.h>
int main(){
    struct{
        unsigned a:1;
        unsigned b:3;
        unsigned c:4;
    } bit, *pbit;
    bit.a=1;
    bit.b=7;
    bit.c=15;
    printf("%d, %d, %d\n", bit.a, bit.b, bit.c);
    pbit=&bit;
```

```
13.         pbit->a=0;
14.         pbit->b&=3;
15.         pbit->c|=1;
16.         printf("%d, %d, %d\n", pbit->a, pbit->b, pbit->c);
17.         return 0;
18.     }
```

运行结果：
1, 7, 15
0, 3, 15

程序第15行使用了复合的位运算符 `&=` ，该行相当于：

```
pbit->b=pbit->b&3;
```

位域b中原有值为7，与3作按位与运算的结果为3（111&011=011，十进制值为3）。同样，程序第16行中使用了复合位运算符 `|=` ，相当于：

```
pbit->c=pbit->c|1;
```

其结果为15。