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。例如在存放一个开关量时，只有0和1 两种状态，用一位二进位即可。为了节省存储空间，并使处理简便，Ｃ语言又提供了一种数据结构，称为“位域”或“位段”。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。每个域有一个域名，允许在程序中按域名进行操作。 这样就可以把几个不同的对象用一个字节的二进制位域来表示。

一、位域的定义和位域变量的说明

位域定义与结构定义相仿，其形式为：

struct 位域结构名

{

 类型说明符  位域名：位域长度

......

 };

例如：（wyp原创）

#include <iostream>

typedef struct

{

/\*int a:8;

int b:2;

int c:6; //sizeof(bs) = 4;（1）同类型可以紧挨着排放，先申请的放在低位。

int d:5;

int f:5;\*/

/\*int e:33;\*/ //错误，（2）单次申请的位数不能大于类型本身限额。

/\*unsigned a:4;

unsigned :0; //sizeof(bs) = 8；（3）没有位域名、位域长度0时，类型剩余位默认全占。

unsigned b:28;

unsigned c:4;\*/

/\*int a:1;

int :32; //sizeof(bs) = 12;（4）没有位域名、位域长度指定位数时，可能从下一类位开始。

int b:3;

int c:2;\*/

/\*unsigned char b1:5;

unsigned char b2:5;

unsigned char b3:5; //sizeof(bs) = 5; （5）剩余位域不能放下后续申请长度则重新从下一类位开始。

unsigned char b4:5;

unsigned char b5:5;\*/

/\*unsigned int b1:5;

unsigned int b2:5;

unsigned int b3:5;

unsigned int b4:5; //sizeof(bs) = 8; （6）总占字节数为（最大）类型的整数倍。

unsigned int b5:5;

unsigned int b6:5;

unsigned int b7:5;\*/

/\*char a:1;

int b:5;

char :8;

long long c:3; //sizeof(bs) = 12;（7）偏移量为最长类型的位宽

char d:2;

double e;\*/

}bs;

int main()

{

std::cout << sizeof(bs) << std::endl; //空结构体占一个字节

return 0;

}