unsigned char a = 128;

char b = a;

short c = b;

printf("%d\n", (unsigned short)a);

printf("%d\n", c);

printf("%d\n", (unsigned short)b);

输出为：128

-128

65408

原因是b的值为-128（128的原码和补码是1000 0000，转换为char型时，1000 0000是-128的补码）.unsigned char与char的字节长度是相同的，在相互赋值过程中，其内存中的补码未发生改变。

将char转换为short时，其字节长度发生改变，但数值未发生变化，在16位中，-128的原码为1000 0000 1000 0000，在计算机中补码为：原码除符号位以外取反加1，就为1111 1111 1000 0000，（计算机存储的为补码）强制转换为无符号数时为65408.

例：

int a = -1;

unsigned int b = 0;

if (a < b)

{

printf("a < b\n");

}

else if (a == b)

{

printf("a = b \n");

}

else

{

printf("a > b \n");

}

输出结果为：a > b

原因：当有符号数与无符号数进行比较运算时，C/C++会将有符号数当成无符号数来处理，-1的二进制补码为32个1，当成无符号数来处理时显然比0大。

例：

unsigned int g = 0;

int h = 0;

if (h < (g - 1))

{

printf("%u\n", (g - 1));

printf("h < (g - 1)\n");

}

输出结果为

4294967295

h < (g - 1)

原因：无符号数0的二进制补码为32个0，而-1的二进制补码为32个1，从而(g - 1)得到的二进制补码为32个1，按照无符号数来处理这个二进制补码，得到的值就是4294947295，显然0小于这个数。

无符号数和有符号数的四则运算：

1. 将有符号数当无符号数处理（结果是无符号数）：

unsigned int i = 1;

int j = -9;

i = j + 8; // j被当成无符号数处理

printf("%u\n", i); // 4294967295，这是-9的补码转换为无符号数的值再加上得到的值

1. 将无符号数当有符号数处理（结果是有符号数）：

short int m = -1;

unsigned short int n = 65535;

m = n - 9;

printf("%d\n", m); // -10，将n当成有符号数处理，其值为-1

int m = -1;

int n = -2;

unsigned int k = m + n; // 4294967293，将m和n看成有符号数，-3转换为无符号数为4294967293。

unsigned int x = 1;

unsigned int y = 2;

int z = x + y; // 3