为什么在字长为32的系统中， C语言int型变量存储的数的范围是-2147483648~2147483647

在字长为32的系统中二进制原码最大值为

0111 1111 1111 1111 1111 1111 1111 1111 =231-1=**2147483647**

二进制原码最小值为

1111 1111 1111 1111 1111 1111 1111 1111 =-(231-1)=-**2147483647**

正0和负0：

+0=0000 0000 0000 0000 0000 0000 0000 0000

-0=1000 0000 0000 0000 0000 0000 0000 0000

所以，二进制原码表示时，范围是-**2147483647**～-0和0～**2147483647**，因为有两个零的存在，所以不同的数值个数一共只有232-1个，比32位二进制能够提供的232个编码少1个。但是计算机中采用二进制补码存储数据。正数编码不变，从

0000 0000 0000 0000 0000 0000 0000 0000到

0111 1111 1111 1111 1111 1111 1111 1111

补码依旧表示0到**2147483647。**

原码负数是从-**2147483647到-0**

1111 1111 1111 1111 1111 1111 1111 1111到

1000 0000 0000 0000 0000 0000 0000 0000

补码是从1000 00000 0000 00000 0000 0000 0000 0001到

0000 0000 0000 0000 0000 0000 0000 0000

从需要把除符号位以后的部分取反加1，即-**2147483647**的补码为10…30个…1。

也就是正0和负0的补码编码是一样的。但是，我们知道，32位二进制数可以表示232个编码，而在补码中零的编码只有一个，也就是补码中会比原码多一个编码出来，这个编码就是1000 0000 0000 0000 0000 0000 0000 0000，因为任何一个原码都不可能在转成补码时变成1000 0000 0000 0000 0000 0000 0000 0000。**所以，人为规定**1000 0000 0000 0000 0000 0000 0000 0000**这个补码编码为**-**2147483648。**所以，**补码系统中，范围是-**-**2147483648～**-**2147483647**。

因此，实际上，32bits二进制的最小数确实是1111 1111 1111 1111 1111 1111 1111 1111，只是二进制补码的最小值才是1000 0000 0000 0000 0000 0000 0000 0000，而1111 1111 1111 1111 1111 1111 1111 1111的补码是-1。

#include "stdio.h"

/\*int型变量取值范围为-2147483648(2^31)~2147483647(2^31-1)\*/

main()

{

int j,i=2147483647;

j=i+1; /\*超过int型数的存储范围\*/

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

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

}

运行结果为：

2147483647

-2147483648