数据的机器表示与处理-实验

实验1、C语言中允许无符号整数和带符号整数之间的转换，转换前、后的机器数不变，只是转换前、后对其解释发生了变化。转换后数的真值是将原二进制机器数按转换后的数值类型重新解释得到。例如，对于以1开头的一个机器数，如果转换前是带符号整数类型，则其值为负整数，若将其转换为无符号数类型，则它被解释为一个无符号数，因而其值变成了一个大于等于2n-1的正整数。也就是说，转换前的一个负整数，很可能转换后变成了一个值很大的正整数。由于上述原因，程序在某些情况下会发生意想不到的结果。

用C语言实现下列主要代码：

int x = -1;

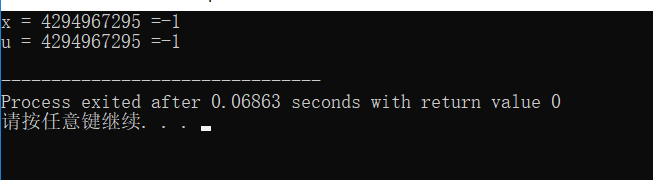
unsigned u = 2147483648; （32位的计算机时）

printf (“x = %u =%d\n”, x, x);

printf (“u = %u =%d\n”, u, u);

上述代码中，x为带符号整数，u为无符号整数，初始值为231。

分析程序输出结果，说明原因。



两次输出结果一致，说明，在计算机中储存的是高低电平，即二进制数，但是到底读出了什么结果，与机器对于这个数的定义有关

实验2、用C语言实现下列主要代码：

int func1 (unsigned word)

{

return (int) ((word << 24) >>24);

}

int func2 (unsigned word)

{

return ((int) word << 24) >>24;

}

假设机器采用二进制补码表示带符号整数，无符号数采用逻辑移位，带符号整数采用算术移位。请填写下表，并说明函数func1 和func2的功能。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| w | | func1 (w) | | func2 (w) | |
| 机器数 | 值 | 机器数 | 值 | 机器数 | 值 |
| 0011 1111 | 127 | 0011 1111 | 127 | 0011 1111 | 127 |
| 0100 0000 | 128 | 0100 0000 | 128 | 0100 0000 | -128 |
| 0111 1111 | 255 | 0111 1111 | 255 | 0111 1111 | -1 |
| 1000 0000 | 256 | 1000 0000 | 0 | 1000 0000 | 0 |

Fun1，先将无符号数放大2^24，将高位的24位移除，然后再复原，也就是只保留前8位，然后返回int型，最后结果是一个小于255的数。

Fun2，先将无符号数放大2^24，将高位的24位移除，然后定义其为int型，也就是第一位的二进制位在第二次移动的时候不参与移动，结果如果大于128，会变为一个小于零的数。

实验3：

输入并运行下面的两组程序（程序A和程序B） ，对照程序和运行结果并分析以下问题：

(1)将一个负整数赋给一个无符号的变量，会得到什么结果？（同一）

(2)将一个浮点数赋给一个整数，会得到什么结果？ （小数被截掉）

(3)将一个浮点数赋给一个无符号的变量，会得到什么结果？

程序A：

#include <stdio.h>

main()

{

   int a,b;

   unsigned c,d;

 float  e,f;

a=57;

b=-57;

  e=5.7;

  f=-5.71;

c=a;

    d=b;

   printf("%d,%d\n",a,b);

printf("%u,%u\n",a,b);

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

  c=a=e;  （间接转换可理解。直接转换？）

  d=b=f;

   printf("%d,%d\n",a,b);

  printf("%u,%u\n",c,d);

}

程序B：

#include <stdio.h>

int main( ) {

int i\_number;

unsigned int u\_number;

float f\_number;

printf("Input an integer:");

scanf("%d", &i\_number);

printf("i\_number=%d, u\_number=%u, f\_number=%f\n", i\_number, (unsigned int)i\_number, (float)i\_number);

printf("Input an unsigned integer:");

scanf("%u", &u\_number);

printf("i\_number=%d, u\_number=%u, f\_number=%f\n", (int)u\_number, u\_number, (float)u\_number);

printf("Input a float number:");

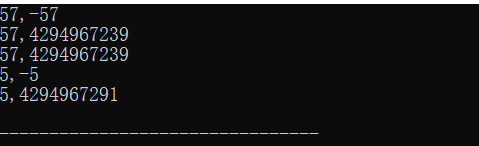
scanf("%f", &f\_number);

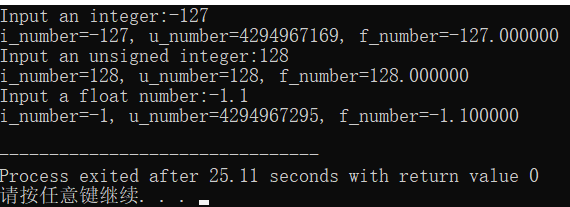
printf("i\_number=%d, u\_number=%u, f\_number=%f\n", (int)f\_number, (unsigned int)f\_number, f\_number);

return 0;

}

分析实验结果，在编程时，你有哪些问题需要注意？





有运行结果可以看出，编程的时候务必要正确的定义变量类型，int型在做除法的时候可能会出现在整数位截断，printf输出的时候选错格式也会出现错误，cout虽然效率低一些但是更不容易出现错误。