****

**计算机组成原理实验报告**

**题目：**整数和浮点数的表示

**姓 名： 臧祝利**

**专 业： 计算机科学与技术**

**年 级： 2020级**

**学 号： 202011998088**

**任课教师： 王志春**

**完成日期： 2021年04月01日**

**人工智能学院**

1. 实验要求
2. **验证**表 1 中的关系表达式的结果，并**编程得出表 2 中的结果**；

（2）通过编程验证 float 和 double 类型的**精度**（即十进制有效位的位数），在实验报告中给出**结果及解释**；编程计算“-8.0/0”、“sqrt(-4.0)”的运算结果， **并对结果给予解释**。

（3）float a = (100+1.0/3) - 100; b=1.0/3; 请回答逻辑表达式 a==b 的**取值**是什么？通过程序计算和检验，对**结果**进行说明

（4）类型转换和移位操作：

编程实现以下各种操作：

· 给定一个 short 型数据-12345，分别转为 int、unsigned short、

unsigned int、float 类型的数据；

·给定一个 int 型数据 2147483647，分别转换为 short、unsigned short、

unsigned int、float 类型的数据；

·给定一个 float 型数据 123456.789e5，转换成 double 型数据；

·给定一个 double 型数据 123456.789e5，转换成 float 型数据；

·按 short 和 unsigned short 类型分别对-12345 进行左移 2 位和右移 2

位操作。

要求分别用**十进制**和**十六进制**形式打印输出以上各种操作的结果，并根据实

验结果，回答下列问题：

·无符号数和带符号整数的扩展操作方式是否相同？各是如何进行的？

·有符号整数是否总能转换为等值的 float 类型数据？为什么？

· float 型数据是否总能转换成等值的 double 型数据？为什么？

·长数被截断成短数后可能发生什么现象？为什么？

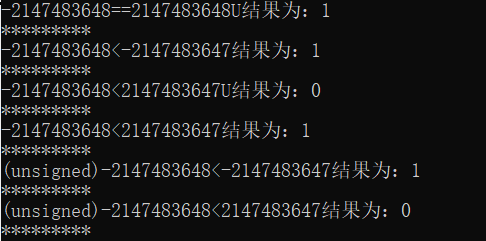
·C 语言中移位操作规则与操作对象的数据类型有关吗？

·左移 2 位和右移 2 位操作分别相当于扩大和缩小几倍？

1. 实验结果与分析

## 验证表1，填写表2

表1验证结果如下，具体代码见test1.cpp



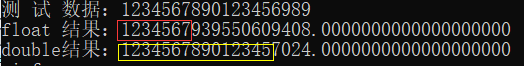
表二结果填写如下，代码见test2.cpp

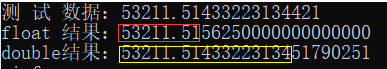
|  |  |  |  |
| --- | --- | --- | --- |
| 关系表达式 | 运算类型 | 结果 | 说明 |
| 0==0U  -1<0  -1<0U  2147483647>-2147483647-1  2147483647U>-2147483647-1  2147483647>(int)2147483648U  -1>-2  (unsigned)-1>-2 | 无符号整数  有符号整数  无符号整数  有符号整数  无符号整数  有符号整数  有符号整数  无符号整数 | 1  1  0  1  0  1  1  1 | 00……00B = 00……00B  11……11B(-1)<00……00B(0)  11……11B()>00……00B(0)  01……11B()>10……00B(-)  01……11B()<10……00B()  01……11B()>10……00B(-)  11……11B(-1)>11……10B(-2)  11……11B()>11……10B(-2) |

表2

## 验证精度

令double和float变量为不同值，分别让其输出19位，结果如下，代码见test3.cpp





由测试结果可知：

float的精度约为6~7位（不同类型的计算机略有不同）

而double的精度约为15~16位

解释：float的尾数为23位，加上小数点前的1位1， = 16777216，因此单精度浮点数的有效位数为7位左右。

而双精度的尾数用52位存储，= 9007199254740992，因此双精度的有效位数是15~16位。

经过编程验证，

-8.0/0的结果为-inf

原因：-8.0为浮点型，而0是整型，在运算中发生了类型转换，0由整数0变成了浮点类型的0，与整除中的0在存储中不同，计算机认为此时的0.0为一个无穷小的数，一个数除一个无穷小的数，得到的是无穷大，再加上是负数，因此计算得到负无穷。

而sqrt(-4.0)的结果为nan，即not a number

原因：对负号进行了开方，得到的是一个复数，而非实数，不在double的范围内，因此无法表示出来，输出nan。

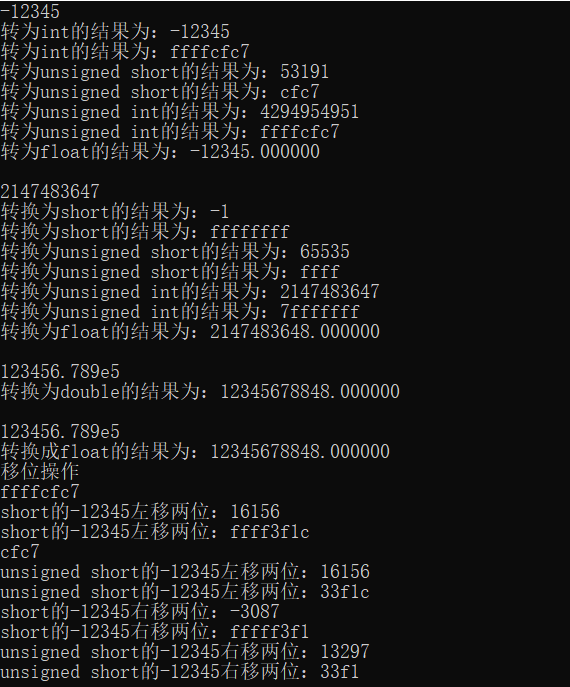
## a==b?

经过程序验证，在a和b为float类型下，a==b（但是在double下a!=b）;即逻辑表达式a==b的取值为1（true）

在计算过程中，由于存在精度问题，导致a在计算每一步时都不是严格的与常规数学计算相等，存在精度的误差，但是由于float的精度过低，前几位的精度依然正确，结果仍然为相等，但是如果变成double类型，由于double的精度高，会使得结果的精度发生变化，从而导致结果不同；因此在判断浮点数是否相等时，用==是不合适的，可以使用精度判断，如果两数之差精度在某一范围内，认为其相等。

## 类型转换和移位操作结果如下

## 代码见test5.c



·无符号数和带符号整数的扩展操作方式是否相同？各是如何进行的？

不相同。

无符号整数扩展时，由于没有符号位，总是无符号扩展，前面的位置补0

有符号整数扩展时，由于具有符号位，总是有符号扩展，前面的位置补符号位

·有符号整数是否总能转换为等值的 float 类型数据？为什么？

不能。

因为有符号整数的表示为31位，而float的尾数部分有23位，float的精度小于有符号整数的精度，转换过程中会出现精度损失。

例子：令int a=123456789,转换成float b=a后输出为，与a不同

· float 型数据是否总能转换成等值的 double 型数据？为什么？

能。

float的尾数为23位，而double的尾数为52位，double的精度要高于float，意味着精度约为7位的float类型一定可以转换成等值的精度约为17位的double类型。

·长数被截断成短数后可能发生什么现象？为什么？

长数被截断成短数后，有可能会发生溢出。

长数被截断后，可能会表示的数字超过对应短数类型的上限，因而会发生溢出。

·C 语言中移位操作规则与操作对象的数据类型有关吗？

有关。

①整数类型的数据类型(char,short,unsigned short,int,unsigned int,long,unsigned long ,long long, unsigned long long等)可以进行移位操作；而浮点类型(float, double等)不能进行移位操作。

②对于无符号数来说，左移右移为逻辑移位，即无论是左移还是右移，都在位置上补0

对于有符号数来说，为算数移位，左移时补0，右移时补符号位

·左移 2 位和右移 2 位操作分别相当于扩大和缩小几倍？

左移2位，相当于之前所有的位数向左移动，末尾两位补0,相当于乘2^2，即扩大了4倍

右移两位，相当于之前所有的位数向右移动，末尾两位舍掉，相当于除2^2，即缩小四分之一

1. 实验小结

通过此次实验，掌握了整型数据和浮点型数据的存储形式和移位过程；深刻感受到了float和double作为浮点型数据在存储和数据范围上的不同，在一些刻板印象上面进行了改观，比如float表示的范围虽然必int大，却无法等值转换，经过调查后发现是由于浮点数存储的数字并不均匀导致的；之前对于移位存在了解误差，只知道移位是发生大小变化，具体移动后的位置补的数并没有做了解，通过这次实验，了解到了对于不同类型的整数而言，发生移位的过程是不一样的，也了解到了算数移位与逻辑移位的相关内容。

1. 参考资料（如有参考资料可以列出）