《电子证据分析》课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 实验名称 |  | 姓名 |  |
| 班级 |  | 学号 |  |
| 实验日期 |  | 指导老师 |  |
| 实验环境  (实验软硬件要求) | 笔记本电脑、Windows 10、python、office、winhex | | |
| 实验目的 | （给出本次实验所涉及并要求掌握的知识点）  熟练掌握浮点型数据转换方式  知道浮点数的ieee标准  了解大端小端表示 | | |

|  |  |
| --- | --- |
| 实验内容 | （1）把自己手机号变成8位浮点数  这个问题比较简单，我理解的题目就是“手机号如果使用IEEE754的单精度浮点数再使用十六进制需要如何表示”。（后面有解释为什么不用8位浮点数表示……）    我的手机号就是13636375421。我想先将其转换为二进制，因为数字实在太大，所以直接使用python的内置函数bin()和hex()来获取需要的二进制和十六进制。  得到结果为：(1100101100110010101001001101111101)2过程如下图    再将其转换为下面的格式：  1.100101100110010101001001101111101\*233  然后就是计算阶码和尾数。  阶码和尾数计算如下：  阶码因为float型的偏移量是127，指数又是33，所以阶码的十进制值就是160，转换到二进制就是(1010 0000)2    尾数因为太长了，超出了范围，这里只能容下23位，剩余的几位都需要进行舍入，得到的值就是(10010110011001010100100)2  因为手机号码是正数，所以第一位符号位为0  综上，将所有的部分拼接起来就得到了需要的数字：  (0101 0000 0100 1011 0011 0010 1010 0100)2  再继续转化为十六进制就是结果了。  故，按照常规的计算办法得到的结果为：0x504b32a4    或者再使用专门的网站进行验算，得到答案一致：    故大端序时值为0x 50 4B 32 A4  小端序时值为0x A4 32 4B 50    另外，假如真的采取8位浮点数的话，会因为阶码（33+3=36 > 6）超过了最大表示范围，表示出来的就是正无穷，也就是(01110000)2，故我才在最开始采用单精度浮点数。  （2）推一下135开头的手机号浮点数表示应该在什么范围里  首先，手机号为11位的十进制数字，所以转换成为二进制必然在一定的范围内，因为如果是使用的单精度浮点数的话，就会像上面的计算自己手机号时出现一样的问题，有部分的数字会不得不舍入。  （下面内容中的以上和以下都包括数字本身）  135开头的手机号都是在13500000000~13599999999这个范围里面的  这里为了避免出现精度丢失，故使用双精度浮点数。    计算13500000000的浮点数表示：  二进制的值为(1100100100101010011010011100000000)2    符号位为0，阶码位为(10000100000)2 过程为：33+1023=1056(偏移量为1023，33为指数)  然后得到尾数：(100100100101010011010011100000000)2，过程如下图：      拼接起来得到浮点数：  (0 10000100000 1001001001010100110100111000000000000000000000000000)2  同理，得到13599999999的浮点数表示：  (0 10000100000 1001010101001111110000111111111110000000000000000000)2  再分别转换成为十六进制得到：  13500000000：0x4209254d38000000  13599999999：0x420954fc3ff80000  以上的十六进制是大端序！！！  所以范围就是0x4209254d38000000到0x420954fc3ff80000  但是，假如考虑的时候是说135开头的手机号就是十六进制的，也就是说是支持像135FFFFFFFF这种特别的类型，那么范围就会发生变化。  但是这样反而更简单了。因为十六进制的135手机号转换成二进制范围就是：  0b10011010100000000000000000000000000000000 ~ 0b10011010111111111111111111111111111111111  分别使用双精度浮点数表示：  0x13500000000：0 10000101000 0011010100000000000000000000000000000000000000000000  0x135FFFFFFFF：0 10000101000 0011010111111111111111111111111111111111000000000000  转换为十六进制就是：  0x4283500000000000 ~ 0x42835ffffffff000  以上为大端序！！！  小端序为0x0000000000508342~0x00f0ffffff5f8342 |
| 实验总结 | （对本次实验涉及到的知识点的重难点归纳、实验心得、思考与建议）  本次实验我认为难点在于浮点数转换规则是否熟练掌握，对于数字本身的进制转换并不是太大的问题(假如数字太大的话，可能是个问题) |
| 评分 |  |