谨慎使用单精度_双精度数值类型 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/430/2021_2022__E8_B0_A8_E6_85_8E_E4_BD_BF_E7_c97_430373.htm 单精度和双精度数值类型最早出现在C语言中（比较通用的语言里面），在C语言中单精度类型称为浮点类型（Float），顾名思义是通过浮动小数点来实现数据的存储。这两个数据类型最早是为了科学计算而产生的，他能够给科学计算提供足够高的精度来存储对于精度要求比较高的数值。但是与此同时，他也完全符合科学计算中对于数值的观念：当我们比较两个棍子的长度的时候，一种方法是并排放着比较一下，一种方法是分别量出长度。但是事实上世界上并不存在两根完全一样长的棍子，我们测量的长度精度受到人类目测能力和测量工具精度的限制。从这个意义上来说，判断两根棍子是否一样长丝毫没有意义，因为结果一定是False，但是我们可以比较他们两个哪个更长或者更短。这个例子很好地概括了单精度/双精度数值类型的设计初衷和存在意义。 基于上述认识，单精度/双精度数值类型从一开始设计的时候，就不是一个准确的数值类型，他只保证在他这个数值类型的精度之内是准确的，精度之外则不保证，比方说，一个数值5.1，很可能存储在单精度/双精度数值中的实际值是5.100000000001或者5.099999999999。导致这个现象的原因我们可以通过两种方式来解释：简单的解释方法：你可以尝试在任何一个控件的属性面板中，设定他的宽度为：3.2CM，当你输入完毕后，你会发现值自动变成了3.199cm，无论你怎么改，你都无法输入3.200CM，因为实际上在电脑中存储的并不是CM为单位的数值，而是“缇”

为单位的数值，而“ 缇 ”和CM之间的比值，是个很难被除尽的数，因此你输入完毕后，电脑自动转换成了最接近的“ 缇 ”值，然后再转换成厘米显示到属性面板上，这一乘一除，两次四舍五入，误差就出来了。单精度/双精度也是类似的原理，其实在二进制存储的时候，单精度/双精度都采用了类似相近分数的方法，而这样的存储是不可能做到准确的。深入的解释方法： 让我们来看看我们存储到数字介质中的单精度/双精度值到底是怎么样的，我们使用如下代码对单精度类型进行一个解剖：Public Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (Destination As Any, Source As Any, ByVal Length As Long) Public Sub floatTest()Dim dblVar As SingledblVar = 5.731 / 8dblOutput dblVardblVar = dblVar * 2dblOutput dblVardblVar = dblVar * 2dblOutput dblVardblVar = dblVar * 2dblOutput dblVardblVar = dblVar * 2dblOutput dblVardblVar = dblVar * 2dblOutput dblVarEnd SubPublic Sub dblOutput(ByVal dblVar As Single) Dim bytVar(3) As Byte Dim i As Integer, j As Integer Dim strVar As String CopyMemory ByVal VarPtr(bytVar(0)), ByVal VarPtr(dblVar), 4 strVar = dblVar amp. (bytVar(i) And 2 ^ j) / 2 ^ j Next j strVar = strVar &amp. " " Next i Debug.Print strVarEnd Sub 运行后我们得到输出结果（输出格式为高位左，低位右）：716375: 00111111 00110111 01100100 01011010 1.43275: 00111111 10110111 01100100 01011010 2.8655: 01000000 00110111 01100100 01011010 5.731: 01000000 10110111 01100100 01011010 11.462: 01000001 00110111 01100100 01011010 22.924: 01000001 10110111 01100100 01011010 这里，我们把单精度类型转化成了二进制数据输出，这里我们看到，虽然这六

个数字完全不同，但是他们的二进制存储惊人地相似，我们看到红色标记部分，每次都是加1，事实上，单精度数据类型使用从高位开始第1位作为正负标记位（绿色），第2位到第9位，是一个跨字节的有符号字节类型数据，这个数值决定了小数点移动的方向和位数（红色），第10位到32位保存一个整数（蓝色）在存储过程中，电脑首先把输入的值不断移位（乘除2）直到这个数的整数部分占用了全部24位的整数位，然后把移动的位数写入浮点部分（红色），而移位后的结果写入整数部分（蓝色和绿色），小数部分则舍弃。求值的时候则是反向过程，先根据正负位和整数位求值，然后根据红色部分的整数来进行移位（乘除2的次方），最终才是我们得到的单精度数值。双精度数值也是同样原理，只是位数更多而已。 通过解剖单精度数值的二进制存储格式，我们可以清楚看到，实际上单精度/双精度的存储，都要通过乘法和除法，其中必有舍入，如果恰好你的数值在除法中被舍入了，那么你赋的初值就很可能与你最终存储的值不完全相同，其中的微小差异，并不与单精度/双精度的设计目标相违背。 当我们在数据库中或者VBA代码中使用一个单精度/双精度数值的时候，也许你从界面上看不到区别，但是在实际的存储中，这个差别却真真切切地就在那里，当你对其进行相等比较的时候，系统只是简单地作二进制的比较，界面上无法体现的微小差异，在二进制比较面前却无处遁形，于是，你的等于比较返回了一个意料之外的False。结束语 通过本文，我们介绍了单精度/双精度数据类型的实质以及其特点（优点和缺点），通过比较和解剖我们了解到单精度/双精度实际上存储的是一个近似值，浮点的特性决定了他可以存储非常小的数，

也可以存储极大的数，他的数据精度并不是一个绝对值，而是存储值的百分比，如果你存储10的100次方，误差就可能是10的80次方，如果你存储10的-100次方，误差就可能是10的-120次方。因此单精度/双精度数据类型不能进行相等的比较（或数据库关联）。 如果你需要进行等值比较或关联，那么有以下几种方案：1、使用专为准确度而设计的货币类型。2、使用整数类型存储，代码中移位。3、某些特定情况下可以用文字存储。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com