大家好，我是瑞生，我的个人微信号是：253 057 617，喜欢结交电子行业的朋友们。

前几天，有个小伙伴在做实验过程中，发现了一个奇怪的现象，这个现象就是...

他在用 printf 输出浮点数的时候，想把数据保留到小数点后的两位，他是这么写的...

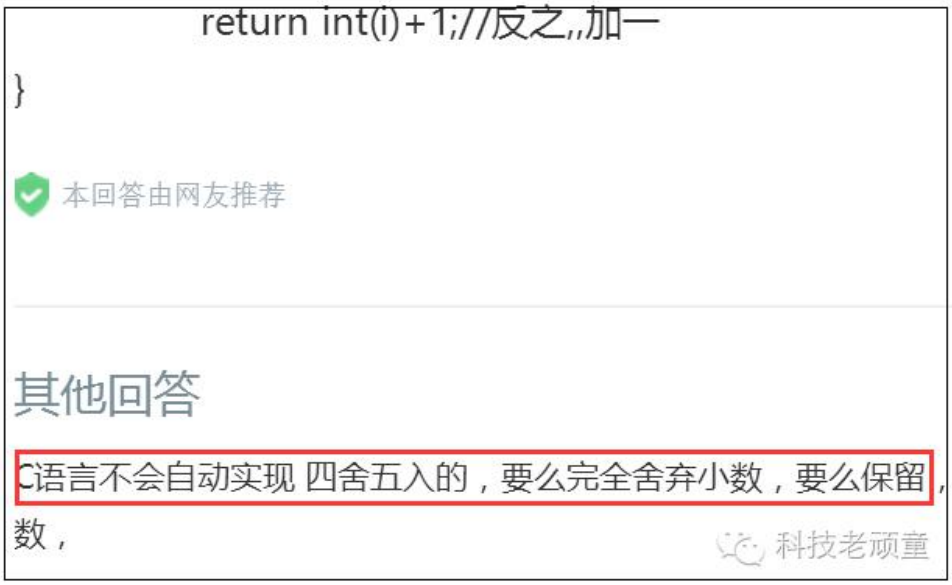
```
float c=1.155;printf("%.2f",c);
```

他的书写是对的，没有错误。但是他发现，当 c 等于 1.555 时，保留两位小数输出是 1.55，而当 c 等于 3.555 时，保留两位小数输出是 3.56。这个结果，就让人捉摸不透了，因为...

如果是程序运算会自动四舍五入的话，结果应该是 1.56 和 3.56；如果程序运算不会自动四舍五入的话，结果应该是 1.55 和 3.55。可是结果却是 1.55 和 3.56，这是什么鬼？

如果你去百度输入关键词“浮点数 四舍五入”，你会发现，有些人会说浮点数会自动四舍五入，如下图...

而有些人会说，不会自动四舍五入，如下图...



到底会不会自动四舍五入呢？

我刚才拿老顽童 STM32 开发板做了一个实验，我定义了 6 个浮点数，他们分别是...

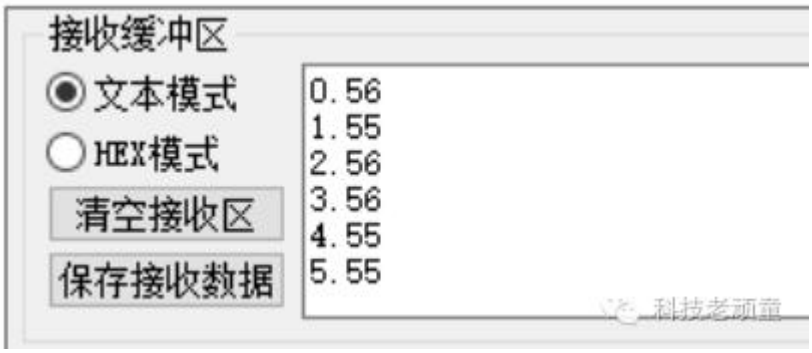
```
float temp1,temp2,temp3,temp4,temp5,temp6;

temp1=0.555;
temp2=1.555;
temp3=2.555;
temp4=3.555;
temp5=4.555;
temp6=5.555;
```


然后我用 printf 给他们保留 2 位小数后输出，程序如下...

```
  while (1)
  {
    Delay(8000000);
    printf("%.2f\r\n",temp1);
    printf("%.2f\r\n",temp2);
    printf("%.2f\r\n",temp3);
    printf("%.2f\r\n",temp4);
    printf("%.2f\r\n",temp5);
    printf("%.2f\r\n",temp6);
  }
}
```


在串口调试助手上看到的结果是...



结果是，6 个数，有 3 个数自动四舍五入了，有 3 个数没有四舍五入。

不管理论是什么，我们只看结果。结果是：浮点数保留小数点后的数据，有时会自动四舍五入，有时不会自动四舍五入。但是...

如果把一个浮点数赋给一个整数变量后，一定不会四舍五入。

所以，我们在保留浮点数的小数点精度时，**必须要人工处理四舍五入**。

很多人一直在用的一个的方法，就是**加 0.5 法**。

这个方法的理论依据是：

```
float f;//定义了一个浮点数

int t;//定义了一个整数
```

我们执行（t=f；）这条语句，不管 f 的小数点后面是小于 5 的数，还是大于等于 5 的数，都不会四舍五入，例如当 f=3.2 和 f=3.8，结果都是 t=3。

那么怎么样让 f=3.8 时，t=4 呢？我们可以给 f+0.5 来解决，例如当 f=3.8 时，f+0.5=4.3，执行完 t=f 后，t 就等于 4 了。而当 f 的小数点后的的数都小于 5 时，加一个 0.5 不会大于 4，所以执行完 t=f 后，结果还都是 3。这正好符合我们四舍五入的要求。

这里**需要注意**的是：其实这个…

加 0.5 的方法

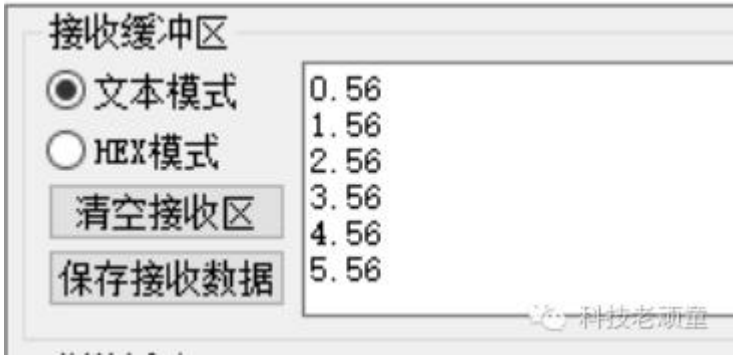
只适合用于保留整数位的应用

很多人都不知道这一点，下面我就给大家实践一下。

按照加 0.5 法的原理，如果要保留 2 位有效数据的话，需要给数据加 0.005，我们做个实验，把 temp1~6 都加 0.005。

```
temp1=temp1+0.005;
temp2=temp2+0.005;
temp3=temp3+0.005;
temp4=temp4+0.005;
temp5=temp5+0.005;
temp6=temp6+0.005;
```


然后我们看输出结果，如下图…



看到了正确的结果，你不要高兴，因为…

你现在可以把 temp1~6 的小数点后面都改为 554，如下图…

```
temp1=0.554;
temp2=1.554;
temp3=2.554;
temp4=3.554;
temp5=4.554;
temp6=5.554;
```


这时候，正确的结果，小数点后两位应该都是 55，但是你看看结果，还是照样是 56。这时候，就输出了完全错误的结果。

有的朋友会说，既然加 0.005 不行，那我们想办法还是加 0.5 吧。好的，下面我有一个方法…

先把 temp1*100，然后再+0.5，然后把这个浮点数赋值给一个整数，然后再把这个整数除以 100。

例如 temp5 的程序写为…

```
temp5=temp5*100+0.5;

t5=temp5;

temp5=(float)t5/100;
```

其中，t5 是我定义的一个 uint32_t 类型的整型变量。我们来分析一下，因为 temp5=4.555，4.555 乘以 100 以后是 455.50，然后再加 0.5 以后是 456.00，把 456.00 取整后是 456，然后 456 除以 100 就是 4.56。

一切都算计的很好，但是实际的结果却还是 4.55。但是如果你这样写的话，结果就是正确的…

```
temp5=4.555*100+0.5;

t5=temp5;

temp5=(float)t5/100;
```

看这个程序和上边的程序对比一下，只是这里直接用了 4.555，而上边的程序用了 temp5，看似一样，结果却不一样。上边程序的结果是 4.55，下边程序的结果是 4.56。

上边两个加 0.5 的实践，你一定要试一下。

那保留 2 位小数，怎么做才能确保完全正确？

送给大家一句话：捷径有可能是歧途，最笨的办法，其实是最保险的办法。（顽童哥语录一定要收藏）

我们可以把小数点后的第三位数取出来，然后判断它和 5 的大小，然后四舍五入。就是这个方法，绝对正确。写成程序的话，是这个样子的...

```
t1=(uint32_t)(temp1*1000)%100%10;
if(t1>4)
{
  temp1=(float)((uint32_t)(temp1*100)+1)/100;
}
else
{
  temp1=(float)((uint32_t)(temp1*100))/100;
}
```


其中，t1 是定义的 uint32_t 类型的整型变量。我们把数据带进去看一下，temp1 是 0.555，0.555 乘以 1000 是 555，555 除以 100 取余数是 55，55 再除以 10 取余数是 5，那 t1 就等于 5。下面用 if 语句判断是否要进位，如果需要进位的话，4.555 乘以 100 就是 455.5，455.5 加 1 就是 456.5，然后我们把 456.5 强制类型转换成整型数据，就是 456，456 除以 100，就是 4.56；如果不需要进位的情况，大家自行分析。

这时候，你去换 temp1~6 的值去吧，不管换什么，结果都会四舍五入。

做一个稳定的电子产品，基础知识很重要！

我已经把这个**源程序**打包放到**坚果云**盘：

https://www.jianguoyun.com/p/DZ1IfiMQlfCTBhi8uCE

你可以自己下载看看。

如果有什么问题需要探讨，或者有其他的什么问题，
请加我的个人微信号：253 057 617