嵌入式C语言之-

浮点型数据类型

讲师: 叶大鹏



数据类型定义

● 浮点数是指带小数点的数值。

类型	关键字	位数	取值范围
单精度浮点型	float	32	-3.4*10 ³⁸ ~-1.18*10 ⁻³⁸ 1.18*10 ⁻³⁸ ~ 3.4*10 ³⁸
双精度浮点型	double	64	-1.80*10 ³⁰⁸ ~ -2.23*10 ⁻³⁰⁸ 2.23*10 ⁻³⁰⁸ ~ 1.80*10 ³⁰⁸

● 变量定义:

float temp = 20.5f; //float类型, 在常数数值后面添加f double temp = 20.5;



```
float temp = 20.5f;
//double temp = 20.5;
temp = temp * 1.2;
```

Program Size: Code=1164 RO-data=992 RW-data=4 ZI-data=1028

```
38:
                float temp = 20.5f;
    39:
                //double temp/= 20.5;
0x000004D4 4C08
                              r4, [pc, #32] ; @0x000004F8
                     LDR
                temp = temp * 1.2;
    40:
    41:
                     MOV
0x000004D6 4620
                              r0.r4
                                 aeabi f2d (0x00000760)
0x000004D8 F000F942/
                     BL.W
0x000004DC 4605
                     MOV
                               r5,r0
0x000004DE F04F3233 MOV
                               r2,#0x333333333
0x000004E2 4B06
                     LDR
                              r3.[pc.#24] ; @0x000004FC
0x000004E4 F000F83E
                     BL.W
                                 aeabi dmul
                                            (0x00000564)
0x000004E8 4607
                     MOV
                               r7,r0
0x000004EA F000F809
                     BL.W
                                aeabi d2f (0x00000500)
0x000004EE 4604
                     MOV
                               r4,r0
```



```
float temp = 20.5f;
//double temp = 20.5;
temp = temp * 1.2f;
```

Program Size: Code=708 RO-data=992 RW-data=4 ZI-data=1028

```
38:
                float temp = 20.5f;
                //double temp = 20.5;
    39:
0x000004D2 4C04
                     LDR
                               r4, [pc, #16] ; @0x000004E4
    40:
                temp = temp * 1.2f;
    41:
0x000004D4 4904
                     LDR
                                            ; @0x000004E8
                               rl, [pc, #16]
0x000004D6 4620
                     MOV
                               r0,r4
                     BL.W
                                 aeabi fmul (0x000004EC)
0x000004D8 F000F808
0x000004DC 4604
                     MOV
                               r4,r0
```

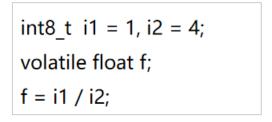


```
//float temp = 20.5f;
            volatile double temp = 20.5;
            temp = temp * 1.2;
   Program Size: Code=820 RO-data=992 RW-data=4 ZI-data=1028
          volatile double temp = 20.5;
   39:
0x000004D2 2100
                    MOVS
                             rl,#0x00
                             r0, [pc, #24] ; @0x000004F0
0x000004D4 4806
               LDR
0x000004D6 E9CD1000 STRD
                             r1, r0, [sp, #0]
   40:
                            1.2;
               temp = temp
   41:
0x000004DA F04F3233
                    MOV
                             r2,#0x333333333
                   LDR
0x000004DE 4B05
                             r3, [pc, #20] ; @0x000004F4
0x000004E0 E9DD0100
                    LDRD
                             r0, r1, [sp, #0]
0x000004E4 F000F808 BL.W
                               aeabi dmul (0x000004F8)
```

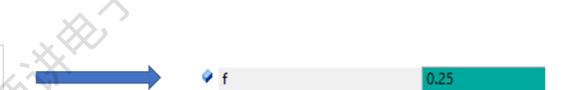
● 浮点数不要直接用等于号做比较



● 整形数转浮点数时要显示的转换类型



正确示例:





Android 计算器惊现超级大 bug! 在 Android 的计算器程序里输入 14.52 - 14.49,计算器竟然说它等于 0.0299999999! 其实,这都是二进制惹的祸,原来在计算机内部,数字并不是用十进制来储存的,所有数字都是以二进制的方式储存的。但一个进制下的有限小数,很可能是另一个进制下的无限小数。比方说,把十进制小数 1.2 转换成二进制小数,将会得到一个无限循环小数 1.001100110011...; 把 1.1 转换成二进制小数则是 1.00011001100110011..., 也是一个无限循环小数。计算机显然不能储存无穷多位数,因而不得不近似地截取有限多位。如果保留 52 位数的话,那么在计算机看来,1.2 - 1.1 其实是这样:

问题出现了:在显示计算结果的时候,计算机需要把它转换回十进制。但上面的结果转换成十进制并不是精确的 0.1,而是一个 52 位小数 0.0999999999999986677323704 49812151491641998291015625。由于 2 的 -52 次方约为 10 的 -16 次方,也就是说 52 位二进制小数的精度大约相当于 16 位十进制小数,因此计算机上通常只保留这个小数的 16 位有效数字。因此,上面这个小数也就成了 0.099999999999987。

注意事项

● 总结:

- 1.尽量选择float类型,减小对单片机资源和性能的开销;
- 2.浮点数常量默认是double型,加后缀f可以声明为float型;
- 3.浮点数不要直接用等于号做比较;
- 4.整形数转浮点数的运行时要显示的转换类型。



THANK YOU!