

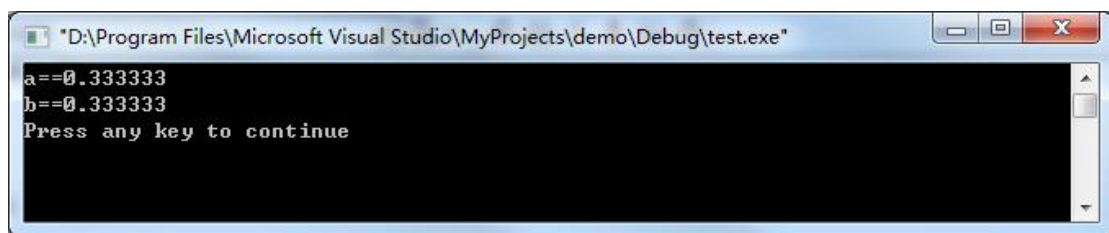
课外阅读材料之 3: float 与 double 有效数字问题

我们在学习 C 语言课本的时候会发现书上说: float 的有效数字为 6 位 , double 的有效数字为 15 位 . 那么我们应该怎样去看它真正有效数字是不是所说的那样呢?

用代码简单实现了一下, 代码如下:

```
#include <stdio.h>
void main()
{
    float a;
    double b; //变量声明
    a = 1.0/3;
    b = 1.0/3;
    printf("a==%f\n", a); //输出 a
    printf("b==%f\n", b);
}
```

发现结果为:




可以发现值是一样的, 那么我们应该怎样去看它的有效数字是不是书上所说的那样呢?

我们用格式控制它输出的有效数字位数, 你就能看到它真正有效数字是多少了, 超过有效数字的值不确定, 代码如下:

```
#include <stdio.h>
void main()
{
    float a;
    double b; //变量声明
    a = 2.0/3;
    b = 2.0/3;
    printf("a==%.20f\n", a); //输出 a, 并要求强制保留 20 位有效数字
    printf("b==%.20f\n", b);
}
```

那么结果为：



选定 "D:\Program Files\Microsoft Visual Studio\MyProjects\demo\Debug\test.exe"

```
a==0.666666666653488159000
b==0.66666666666666663000
Press any key to continue
```

我们可以在结果上看出： float 的有效值为 7 位， double 的有效值为 15 位。

说明：连续多个“6”之后出现的数字是随机数，不是变量保存的数字。因此，数连续“6”的个数为变量中保存的小数长度。

结论:

有些编译器 float 的有效数字位是 8 位，有些有效数字位是 7 位

有些编译器 double 的有效数字位是 15 位，有些是 16 位

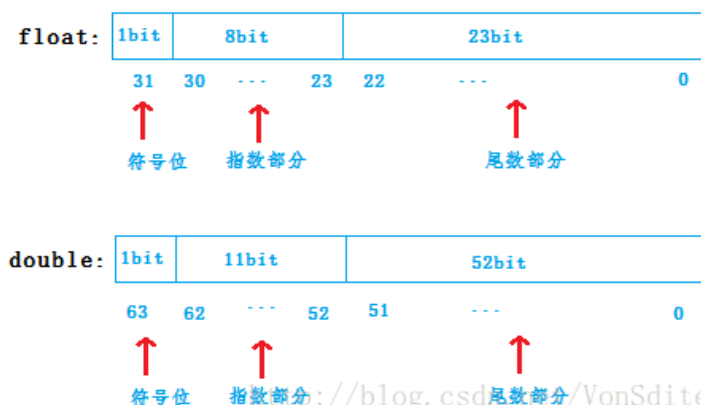
注意 `printf("%f", x);` // 默认输出 6 位小数(不要和有效数字混淆)

分析

C/C++编译器标准都遵照 IEEE 制定的浮点数表示法来进行 float, double 运算

无论是 float 还是 double，在内存中的存储主要分成三部分，分别是：

- (1) 符号位 (Sign) : 0 代表正数, 1 代表负数
- (2) 指数位 (Exponent) : 用于存储科学计数法中的指数部分, 并且采用移位存储方式
- (3) 尾数位 (Mantissa) : 用于存储尾数部分



由图可知:

float 是 32 位，其中有 23 位用于存放尾数，带有一个固定隐含位... 所以 float 的有 24 个二进制有效位位数。23 位二进数转换为 10 进制是 7.2 位。 2^{24} 共有 8 个十进制位。所以有些编译器 float 的有效数字位是 8 位，有些有效数字位是 7 位。(注意不是小数的位数，是有效数字位)。

同样, double 也一样, 是 64 位, 其中有 52 位用于存放尾数, 一个固定隐含位. 共有 53 个二进制有效位数. 2^{53} 次方有 15 个十进制位, 所以有些编译器 double 的有效数字位是 15 位, 有些是 16 位。

自己分析：为什么有同学采用循环语句通过实数比较(例如， $(C*10 - (\text{int}) C) == 0$.)计算出来总是 8 位的原因。