链滴

关于 C 语言浮点数 float 浮点数舍入错误问题

作者：Doss

原文链接：https://ld246.com/article/1660938368545

来源网站：链滴

关于C语言浮点数 float浮点数舍入错误问题

背景

《C Primer Plus》 3.4.6 float、double和long double小节中浮点数舍入错误问题。

代码：

```
/* float_rounding_error.c -- 实验浮点数舍入错误问题 */
#include <stdio.h>

int main(void)
{
  float a, b;

  b = 2.0e20 + 1.0;
  a = b - 2.0e20;

  printf("%f \n", a);

  return 0;
}
``
```

输出如下：

```
0.000000  <-- Linux系统下的老式gcc
-13584010575872.000000  <-- Turbo C 1.5
4008175468544.000000  <-- XCode 4.5、Visual Studio 2012、当前版本的gcc
```

问题

- 为什么输出结果不等于1?
- 需要具备哪些知识点才能理解这个问题出现的原因?

1. IEEE 754标准。
2. C标准 即float.h头文件。
3. 保存浮点数的原理。

书中给出的解释：

> 得出这些奇怪答案的原因是，计算机缺少足够的小数位来完成正确的运算。2.0e20是 2后面有20个0 如果把该数加1，那么发生变化的是第21位。要正确运算，程序至少要储存21位数字。而float类型的字通常只能储存按指数比例缩小或放大的6或7位有效数字。在这种情况下，计算结果一定是错误的。一方面，如果把2.0e20改成2.0e4，计算结果就没问题。因为2.0e4加1只需改变第5位上的数字，floa类型的精度足够进行这样的计算。

结论

1. 因为保存单精度浮点数的内存大小为32位，其中表示有效数字的尾数部分为23个比特，最多表示1

进制下的7位数。所以不管是C标准还是IEEE 754内的定义，都是只保证从第一个非0数字开始往后数位数字的精度。例如：123.456789（只保证1，2，3，4，5，6，7的精度）

2. C标准定义10进制下的精度为6是因为计算机需要把数值转换成为2进制的数值保存，而2进制下的数队列与10进制下的队列不是一一对应的。2进制小数能保证完整表示从0到9的只有10进制下的小数分第6位。

解析

● 2.0e20是 2后面有20个0。如果把该数加1，那么发生变化的是第21位。要正确运算，程序至少要存21位数字。

2.0e20 + 1 = 2.00000000000000000001e20（要这么保存才能保持精度）

● 而float类型的数字通常只能储存按指数比例缩小或放大的6或7位有效数字。

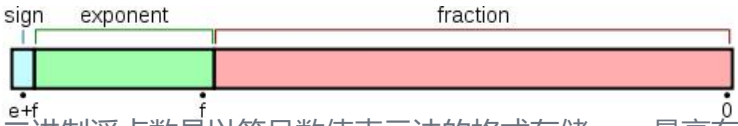
有效数字（维基百科）：

有效数字指科学计算中用以表示一定长度浮点数精度的那些数字。一般指一个用小数形式表示的浮点中，从第一个非零的数字算起的所有数字，因此，1.24和0.00124的有效数字都有3位。并且在取有效字时一般会遵循四舍五入的进位规则。例如取1.23456789为三位有效数字后的数值将会是1.23，而四位有效数字后的数值将会是1.235。

为什么是6或7位?

先看看C语言采用的IEEE 754标准

IEEE二进制浮点数算术标准（IEEE 754）：



二进制浮点数是以符号数值表示法的格式存储——最高有效位被指定为符号位（sign bit）；“指数分”，即次高有效的e个比特，存储指数部分；最后剩下的f个低有效位的比特，存储“有效数”（significand）的小数部分（在非规约形式下整数部分默认为0，其他情况下一律默认为1）。

float类型也就是单精度二进制小数，使用32个比特存储。

S	Exp	Fraction
1	8	23位长
31	30至23偏正值（实际的指数大小+127）	2至0位编号（从右边开始为0）

S为符号位，Exp为指数字，**Fraction为有效数字**。 指数部分即使用所谓的偏正值形式表示，偏正值实际的指数大小与一个固定值（32位的情况是127）的和。采用这种方式表示的目的是简化比较。因，指数的值可能为正也可能为负，如果采用补码表示的话，全体符号位S和Exp自身的符号位将导致不简单的进行大小比较。正因为如此，指数部分通常采用一个无符号的正数值存储。单精度的指数部分−126～+127加上偏移值127，指数值的大小从1～254（0和255是特殊值）。浮点小数计算时，指值减去偏正值将是实际的指数大小。

尾数部分占23个比特（其实位数隐含了整数部分1，这23位是小数部分。但这里可以先忽略）。需要道23位比特所能表达的最大值在10进制下至少需要几位数。

- 可以根据公式 N=b^n-1（求在b进制下n个位数所能表达的最大数值）得出 2^23-1=8388607
- 再根据 n=log(b)N （求在b进制下表达数值N至少需要的位数）得出 log8388607≈6.9 向上取

得7位。23位比特所能表达的最大值在10进制下至少需要7位数。

结论：

也就是说23位比特所能表达的最大数也就只能到7位。这正好就能解释“float类型的数字通常只能储按指数比例缩小或放大的6或7位有效数字”。

验证：

```
/* significant_figures.c -- 单精度浮点类型的有效数字 */
#include <stdio.h>

int main(void)
{
  float a, b, c, d, e, f;

  a = 1234567;
  b = 1234567.89;
  c = 1234.56789;
  d = 123456789.7654321;
  e = 1.1234567;
  f = 0.123456;

  printf("a: %f \n", a);
  printf("b: %f \n", b);
  printf("c: %f \n", c);
  printf("d: %f \n", d);
  printf("e: %f \n", e);
  printf("f: %f \n", f);

  return 0;
}
```

输出如下:

```
a: 1234567.000000
b: 1234567.875000
c: 1234.567871
d: 123456792.000000
e: 1.123457
f: 0.123456
```

可以看到在有效数字超出7位以后就失去了精度。

为什么默认只展示小数点后6位?

C标准定义 FLT_DIG 10进制的精度位数 为6，也就是小数点后6位

C标准函数库中的头文件float.h

```
/* float_header_file.c -- float.h是C标准函数库中的头文件 */
#include <stdio.h>
#include <float.h>

int main(void)
```

```
{
  printf("The precision of float = %d\n", FLT_DIG );
  return 0;
}
```

输出如下：

```
The precision of float = 6
```

那为什么会定义为小数点后6位而不是7位或8位呢?

浮点数的精度

原因在于二进制小数与十进制小数没有完全一一对应的关系，二进制小数相比十进制小数来说，是离而不是连续的，我们来看看下面这些数字：

二进制小数	十进制小数
2^-23	1.00000011920928955078125
2^-22	1.00000023841857910156 25
2^-21	1.00000047683715820312 5
2^-20	1.00000095367431640625
2^-19	1.00000190734863281 25
2^-18	1.00000381469726562 5

这里只需要关注F，上面列出了1.xxx这类浮点数中的6个最小的二进制小数，及其对应的十进制数。以看到使用二进制所能表示的最小小数是1.00000011920928955078125，其次是1.0000002384185591015625，这两个数之间是有间隔的，如果想用二进制小数来表示8位有效数（只算小数部分，小数前面的1是隐藏的默认值）1.00000002、1.00000003、1.00000004.....这些数是无法办到的，而7位效数1.0000001可以用2-23来表示，1.0000002可以用2-22来表示，1.0000003可以用2-23+2-22来示。从这个角度来看，float型所能精确表示的位数只有7位，7位之后的数虽然也是精确表示的，但却法表示任意一个想表示的数值。

但还是有一些例外的，比如说7位有效数1.0000006这个数就无法用F表示，这也表明二进制小数对于进制小数来说相当于是离散的，刚好凑不出1.0000006这个数，从这点来看float型所能精确表示的位只有6位。因此float型的有效位数是6-7位，但这个说法应该不是非常准确，准确来说应该是6位，C言的头文件中规定也是6位。对于一个很大的数，例如1234567890，它是由于指数E系数而被放大了，但它的有效位仍然是F所能表示的6~7位有效数字。1234567890用float表示后为1234567936，只高7位是有效位，后3位是无效的。int型可以准确的表示1234567890，而float浮点数则只能近似的示1234567890，精度问题决定了float型无法取代int型。

结论：

从上文中可知，二进制存储的23位小数部分，在10进制数的小数部分中能完整且精准表示的只能到第位。我想这就是C标准定义10进制的精度位数为6的原因。