

## FloatStr sometimes produces NUL chars

[illegible]

**#1 - 28 Jan 2021 19:13 - John Abbott**

Of course, it came up while giving a demo :-)

- % Done changed from 0 to 10

The exponent is wrong by 1: it should be -10.

Most odd!

FloatStr simply calls MantissaAndExponent10 to do the main conversion.

Here are the outputs for  $q$ ,  $10 \cdot q$  and  $100 \cdot q$

[illegible]

Aha! It seems that FloorLog10 sometimes give wrong answers... groan.  
With luck that is the root cause.

- Assignee set to John Abbott
- % Done changed from 10 to 80

The problem was in **FloorLogBase**, in ptic the definition of the value **delta**.  
For "efficiency", an approx log was computed using floating-point; delta is an estimate of how inaccurate the fp log might be.  
The old estimate was 5ulp, but apparently this was not always enough.

I have now increased the estimate to 256ulp; probably even 8ulp should be enough.  
The only disadvantage is that a full check will be made a bit more often -- this happens only when the supplied value is "very close to" an integer power of the base.

I have inserted an assert also for the "safe case" (where input is "far from" an integer power).

Analogous change to FloorLogBase for BigInt.

Here is a simpler failing case:

```
N := 10^20-1;
Q := N/10^21;
FloorLog10(Q); --> -1 instead of -2
```

```
N := 10^30-1;
Q := N/10^29;
FloorLog10(Q); --> 1 instead of 0
```

This code prints out many triples -- it should print out none!

```
for B := 18 to 99 do N := 10^B-1; for j := 0 to 200 do F := FloorLog10(N/10^j); if F <> B-1-j then println [B
,j,F]; endif; endfor; endfor;
```

#### #8 - 29 Jan 2021 09:56 - John Abbott

This is not good.

I have increased the factor in the definition of delta to 64: there are still failing cases:

```
/**/ for B := 18 to 199 do N := 10^B-1; for j := 0 to 250 do F := FloorLog10(N/10^j); if F <> B-1-j then print
ln [B,j,F]; endif; endfor; endfor;
[117, 116, 1]
[123, 122, 1]
[128, 127, 1]
[134, 133, 1]
[140, 139, 1]
[146, 145, 1]
[152, 151, 1]
[158, 157, 1]
[164, 163, 1]
[170, 169, 1]
[176, 175, 1]
[182, 181, 1]
[188, 187, 1]
[194, 193, 1]
```

The problem seems to occur only with BigRat; I have tried a similar test loop with BigInt but found no failing examples. I now guess that log for BigRat values with large numer-denom is giving imprecise values: this could result from a simplistic impl  $\log(N/D) = \log(N) - \log(D)$ ; if  $\log(N/D)$  is small but  $\log(N)$  is larger than 1000 (say) then we could indeed lose 10 bits of precision.

I'll check the log code... (sigh)

#### #9 - 29 Jan 2021 10:00 - John Abbott

And indeed the defn of log for BigRat is simply

```
return log(num(Q)) - log(den(Q));
```

Now I need a new defn... [facepalm]

#### #10 - 30 Jan 2021 18:47 - John Abbott

I think everything is nearly sorted out now: the example in comment 8 now finds no problems (even over a much wider range).  
I have a new fn `mpq_get_d_2exp`; actually there was an old one in `RingQQ.C`, but it was sometimes less accurate than the new one.

There is also a new fn `LogAbs`, and the existing log now requires arg to be positive (previously it behaved like the new `LogAbs` fn).

Updated some doc; added new test in `test-BigRat3.C`

Soon I can check in.

#### #11 - 30 Jan 2021 20:21 - John Abbott

- Status changed from *In Progress* to *Feedback*
- % Done changed from 80 to 90

#### #12 - 16 Sep 2021 12:32 - John Abbott

- Status changed from *Feedback* to *Closed*
- % Done changed from 90 to 100
- Estimated time set to 8.11 h