Numbers in Liberty BASIC

(all you ever wanted to know about numbers but did not know whom to ask ;))
by -
tsh73, January 2012
Note to author: if you need to edit this page, let us know. It has been systematically vandalised, so I'm locking it.

Table of Contents

Liberty BASIC has only two data types: string and numbers. So if you have any numerical result it entitled to be "just a number". However, "under the hood" numbers in any programming language are a bit more complicated. Liberty BASIC is not an exception – and has complications (and unique abilities) of its own. To get most out of Liberty BASIC, you must understand what's going on.

Numbers could be integers (AKA whole numbers) – without decimal point – and real numbers (AKA floating point numbers). Since Liberty BASIC has no Boolean (logical, yes/no true/false) datatype, numbers are used for this purpose as well.

Integer numbers

Whole numbers like 0, 1, 123, -7 are integers.

Combining integers with + - * MOD operators produces a result that is also an integer. Obviously, result of a division could end up as a non-integer (having fractional part). It is less obvious that power operators on integers could produce non-integer results: for example, 2^(-1) is actually 0.5.

Liberty BASIC has no "integer division" operator, but it could be done as INT(a/b), taking only the integer part of result.

Long integers (aka Arbitrary length integers)

Now, there is a somewhat **unique strong point** in Liberty BASIC: in most of languages, the size of an integer is determined by the underlying computer architecture.

QBasic's integer type is stored in 2 bytes, and hence limited to +/- 2^15, from -32,768 to 32,767.

In contemporary C, an integer is stored in 4 bytes, so it is limited to +/-2^31, from -2,147,483,648 to 2,147,483,647.

To get all 158 digits of 100! (factorial of 100, defined as 1*2*3*...*99*100) in these languages one has to invent things, but for Liberty BASIC, numbers of such length pose no problem:

```
f=1

n=100

for i=1 to n

 f=f*i

next

print len(str$(f))

print f
```

output is (line breaks added to the long number for page formatting):

```
158 – number of digits

and factorial itself, hold your breath:

933262154439441526816992388562667004907159682643816214685929638952
```

```
52175999932299156089414639761565182862536979208272237582511852l0

916864000000000000000000000000
```

It looks like the length of those integers is limited only by available memory and computer speed (of course working with these beasts will be slower!). So getting 10^10000 might take several seconds. But it works, and with all digits!

Functions VAL( ) and STR$( ) in Liberty BASIC support long integers.

(Of course) +, -, *, as well as ^ and MOD supports long integers.

- If a is a long integer and a/b is supposed to be whole number,
  - then a/b will be a long integer
  - else a/b will be a real number (and it even could overflow).

There are some beneficial quirks about INT(a/b):

- If a is a long integer, and a/b is supposed to be real number,
  - then result will be a long integer (with all digits preserved). Also we evade real overflow.

It looks like Liberty BASIC somehow bypasses the intermediate step of storing a/b as real.
It looks like when calling standard math functions, the argument is converted to the "real" data type. This way, you can get overflow. So while 10^200 is within double range, you cannot get it by sqr(10^400) – 10^400 will try to convert to real and overflow.

Real numbers

Storing (small) integers is rather straightforward but our data or result turns out to be non-integer. It strongly looks like they end up stored in common in some other languages DOUBLE data type. This data type came with common limitations (existing in many programming languages):

Range limitation

Here is data on limitation for double-precision numbers from QBasic Help:

- Positive 1.79769313486231D+308 4.940656458412465D-324

- Negative -4.940656458412465D-324 -1.79769313486231D+308

If your result exceeds these limits, you'll get overflow error. In Liberty BASIC, it is easy to get when working with long integers.

Precision limitation

Double real value stores only about 16 "real" digits. You can check it with USING function:

```
mask$="#."
for i=1 to 20: mask$=mask$+"#": next '20 digits after "."
print using(mask$, 1/3)
print "*.12345678901234567890"
```

prints

```
0.33333333333333331968
*.12345678901234567890
```

, that is, we get "garbage" after 16-th digit.

Just checking how small number d should be so 1+d will register as d:

```
d=1
for i=1 to 20
 d=d/10
 a=1+d
 print i,1-a
next
```

Precision limitation consequences

This gives some really bad things you should be aware of (and it is not a bug, it is a way real numbers work in pretty much any language):

- **You should not count on real numbers being exact.**
- **You should never test real numbers for being exactly equal.**

```
a = 2.1 - 2

b = 0.1

print a, b

if a=b then print "Equal" else print "Not equal"

print "Difference "; a-b
```

Output:

```
0.1 0.1

Not equal

Difference 0.83266727e-16
```

Instead, you should test for equality with given precision:

```
precision = 1e-10

if abs(a - b) < precision then print
"Equal with precision" else print "Not equal"
```

- **You should not make FOR loops with real step,**
  if the number of steps is important (like in numerical integration).

```
for i = 1 to 2 step 0.1

 print i

next
```

This example misses the last step (end up on 1.9 instead of 2.0).

Instead, you should use loop by integer index.

Saving/loading real number

There is no way in Liberty BASIC to save/load real number in binary representation, making sure we read back exactly what we saved. The only provided way is in text form.
The problem is that PRINT outputs only 7 digits of 16 we could use. If we try STR$, we'll see that it has same limitation.
The USING function could provide the necessary precision, but how do we get right mask? There no support for scientific notation.

This handy function could help. To save 15 digits of x, use usingS$(x,15) :

```
'scientific USING function.

'(keep in mind that there are no more then 16 digits stored in real nu
mber (Double data type)).

function usingS$(n,prec)

 if n = 0 then usingS$="0e+0":exit function

 fmt$ = "#"+left$(".",prec>0)+left$("################",prec)

'fmt of mantissa

 s$=left$("-",n<0)

 n=abs(n)

 log10=log(n)/log(10)

 e=int(log10)-(log10<0)

'QB like INT. Makes mantissa for negative exponents

 'start from digit (not 0 as LB do)

 p=10^e

 if left$(using(fmt$,n/p),1)="%" then p=p*10:e = e+1

 usingS$=s$+using(fmt$,n/p) +"e"+left$("+",e>0) +str$(e)
```

```
'Excel always shows "+" for exponent

end function
```

Using numbers as Booleans

As was stated, Liberty BASIC lacks a Boolean type, but it does have conditional statements. A condition normally evaluates to false or true.

If we print result of any relational operator (> =