Truth and Clarity in Arithmetic

Neville Holmes, University of Tasmania

Last year, when I argued that our sequencing of digits in numbers from the highest to the lowest significance was a great blunder ("Seven Great Blunders of the Computing World," *Computer*, July 2002, pp. 112, 110-111), I intended to show that a faulty convention could be so embedded in our everyday life that few would ever think to question it. Several e-mail messages about this blunder chided me for not mentioning that the Arabs got place-value notation from the Indians, while another respondent questioned whether the digit of greatest place value was always the one of greatest significance.

But the most gratifying e-mail supported my view and pointed out that the convention meant that reading out an integer in words could not start until the entire number had been scanned, because only then could the leading digit's place value be known. This reminded me of how I sometimes feel annoyed when using an ATM, because the amount I key in builds up in such a peculiar way: Each digit causes the prior value to be multiplied by 10 by being shunted to the left before the new digit is shown. Something similar happens in most numerical machinery for personal use, particularly electronic calculators.

In a well-designed world, digits would not need to joggle around like this. However, I am realistic enough to know that pointing out this blunder will have no effect on what we so much take for granted and that most people will simply disbelieve it's a blunder at all. Only very young children struggling with the rudiments of numeracy will be conscious of this burden, though unable to explain it.

Nevertheless, in a technological world where numeracy in developing countries shows steady decline, we in the computing profession should do everything we can to remove barriers that keep people from attaining numeracy.

Computing professionals have a duty to combat the spread of innumeracy—starting with better calculator design.

CANTANKEROUS CALCULATORS

The atrociously poor design of the ordinary pocket calculator is the greatest barrier to numeracy that the profession could do much to fix.

In 2001, I received a cute little solar-powered calculator from the IEEE Computer Society as a Christmas gift. This novel present reminded me of the last similar device I can remember owning: a slide rule. But I'd since fiddled skeptically with other people's calculators, so the first thing I did with my gift was to key in 1 ÷ 3 × 3 = upon which the device displayed 09999999 instead of the proper 1. On much closer inspection, it appeared the answer was actually 0.9999999, but your eyesight needs to be better than mine to see the difference easily.

Still, the calculator's accuracy greatly exceeds the slide rule's, so I looked for things I could do with it that I couldn't do with a slide rule. My calculator sported four buttons labeled MC, MR, M+, and M–. I assumed the designers provided these to allow use of a single-value register, but experiment suggested that they serve merely and literally as accumulators. So I haven't used the calculator since, except as a desk decoration.

My experience is not unusual. Harold Thimbleby has made a particular study of electronic calculators. In "Calculators Are Needlessly Bad," Thimbleby summarizes his findings, although in needlessly mild language (www.uclic.ucl.ac.uk/harold/srf/hucalc.pdf). Simply put, the available calculator designs are an utter disgrace to the computing profession.

CALCULATOR REDESIGN

The most important calculator is the most basic one, the one used first in learning arithmetic and most in everyday life. More complex and expensive calculators should be designed to compatibly extend the basic calculator.

My solar-powered calculator is basic in most respects and typical of commodity calculators. Let's consider the flaws in its design and how that design could be improved.

Clarity

A basic calculator must first exhibit clarity. The user should be given a clear depiction of what's going on.

Continued on page 106

The Profession

Continued from page 108

Clarity has many aspects. Visibility provides a static aspect, violated on my calculator by making the decimal point almost invisible. The decimal point affects the value of a number more than any individual digit, and it should be as prominent on the display as any other symbol. The conventional decimal point could be suggested by a low triangle: $12_{\triangle}34$. In any case, it's confusing generally to have the period used as an infix in numeric values but as a suffix in alphabetic text.

Overtness provides a dynamic aspect of clarity. Consider what happens as I key 1 ÷ 3 × 3 = into my calculator. The 1 goes straight to the display, but the division symbol merely causes the display to blink. Nothing has changed on the display otherwise. The 3 goes straight to the display, replacing the 1, but keying the multiplication sign causes the value 0.3333333 to appear on the display. So, tapping the multiplication key causes a division to be carried out. The next 3 goes straight to the display, and the equal sign causes the multiplication to be carried out. This is opaque, not overt.

Two design changes beg to be made. First, before carrying out an operation, both operands should be visible, and after carrying out the operation both the operands and the result should be visible. Second, tapping an operation key should cause that operation to be carried out, and be seen to be carried out, right away. The result of a simple multiplication might be shown overtly as

$$\begin{array}{r} 12 \\ \times\,34 \\ 408 \end{array}$$

Obviously, we need a display of at least four lines. The operands of the most recent operation and its result usually require three lines, and we need a fourth for the next operation's second operand. The result of the previous operation will normally be the next operation's first operand.

Simplicity

Second, a basic calculator must be simple to use. The calculator's operation, particularly in its use of keys, should be as consistent and uniform as possible. Thus we require three key classes: editorial, numeric, and operational, though editorial keys are not strictly necessary.

The *editorial* keys copy and edit values from the stored list of previous operands and results. For many reasons, storing these previous values is preferable to providing only a peculiarly limited and counterintuitive accumulator.

The *numeric* keys must provide the 10 decimal digits and two signs: a prefix for the negative sign, and an infix for the decimal point. The numeric keys should all work the same way, a tap putting the symbol into the second operand at the cursor.

Only exact arithmetic makes complete truth possible without recourse to algebraic notation.

My calculator is defective in respect to these two signs. Although it has a decimal point key, tapping it doesn't change the display. Instead, it just frees up the permanent decimal point to move left. The calculator also lacks a negative-sign key, instead providing a toggle that switches the number in the display between positive and negative. Nor does it show negativeness with a prefixed sign in the display, using instead a suffixed subtraction symbol—all of which is highly inconsistent.

It is most unfortunate that the calculator uses the subtraction operation's symbol also as the negative sign. This fundamental ambiguity severely hampers elementary instruction in arithmetic, for example, and forces the use of the comma as a separator in parameter lists within program code. In a basic calculator, the negative sign could be distinctively shown as a high triangle such that, for example, $^{\triangledown}$100 would represent *negative one hundred*.

The *operational* keys are the most significant because they actually cause arithmetic to be done. The other keys merely set up operands. My calculator has five black keys for *dyadic* operations: +, –, ×, ÷, and =. It also has three blue ones for *monadic* operations, labeled +/-, %, and √, of which the first is the negate operation, the second unfathomable, and the third the square root. Worse than their strange effects, these eight keys fail to provide the most important basic operations.

Presuming the numeric keys are arranged in a 3 × 4 block, it's simplest to provide the four most basic operations straightforwardly as a column of keys alongside the numeric keys: counting, selecting, adding, and multiplying, with key-top symbols of, say, #, ^, +, and ×, respectively. Dyadic counting—for which x # y yields the number of *x*s in *y*—and selection are more central to basic numeracy than addition and multiplication. A count always results in an integer, and this operation is behind the long division taught in elementary education and so often confused with the remainderless division that is multiplication's inverse. The selection operation gives the higher of its two operands—another basic operation that elementary education treats inconsistently.

We need more operations than these four, and they can be provided in two different, simple ways. First, the negative sign and the decimal point—normally a prefix or an infix—can be keyed as number suffixes that will be applied as modifying prefixes to an immediately following basic operation. Second, the operation, with any modifying prefixes, can be keyed without any value being given as the second operand.

Table 1 merely suggests the richness of basic operations that could be provided with a simple keying system. The negative sign $^{\triangledown}$ would best be used systematically to replicate the operand of a monadic operation or to commute the operands of a dyadic operation. Thus, $^{\triangledown}+\ y$ would double y, while $x^{\triangledown}\#\ y$ would be the same as $y\ \#\ x$.

Truth

Only exact arithmetic makes complete truth possible without recourse

Table 1. Possible operations for an improved calculator.

Key	Monadic	Dyadic	Keys	Monadic	Dyadic
×	factor	times	Δ×	reciprocal	divide
+	magnitude	plus	Δ+	negate	minus
^	increment	higher	Δ ^	decrement	lower
#	floor	count	Δ#	round	modulus

to algebraic notation. All the basic calculator's arithmetic operations must yield exact results when applied to exact values, and only exact values can be keyed in. My solar-powered calculator is inexact. It lies under division, as I've shown. While many calculators show a result of 1 under the same sequence, the lie is often revealed when we subtract a real 1 from that result.

To solve this problem, we must provide a representation for nondecimal fractional values. We can do this simply, by using the $^{\triangledown}$key as an infix to separate the numerator of a fraction from its denominator. Although we use this negative-sign key to put the denominator point in, the display could show it as the customary virgule. Among its other virtues, this convention gives a clear notation for infinity as $1^{\triangledown}0$ or 1/0, arising from 0#1, and for indeterminacy as $0^{\triangledown}0$ or 0/0, arising from 0#0.

To maintain the dedication to truth, irrational and transcendental values must be approximated by rational values, but this will emphasize that they are only inexact arithmetically on more advanced calculators. For example, an approximation to π could be keyed in either popularly as $22^{\triangledown}7$ or $3_{\vartriangle}1^{\triangledown}7$, more accurately as $3_{\vartriangle}14159$, or even more accurately still as $355^{\triangledown}113$ or $3_{\vartriangle}16^{\triangledown}113$.

The computing profession—indeed the engineering profession in general—has a particular responsibility to promote numeracy, both within the profession and within the community at large. The commodity calculator is an insult to that responsibility.

The IEEE should establish and promote a rigorous and detailed standard for a universal basic calculator. Professional engineers everywhere should press to have that calculator manufactured and adopted as widely as possible. Computing professionals should work with the teaching profession to have the arithmetic of the universal basic calculator taught throughout elementary schooling, perhaps using calculators with drill and practice built in. ■

***Neville Holmes** is an honorary research associate at the University of Tasmania's School of Computing. Contact him at neville.holmes@utas.edu.au. Details of citations in this essay, links to further material, and further discussion of issues are at www.comp.utas.edu.au/users/nholmes/prfsn.*