Gerald's Column
by Gerald Fitton

Over the last few years I have had sent to me (on disc) many spreadsheet problems. There is one set of problems which arise from what I shall call the 'approximation effect'. I shall (successfully?) demonstrate this 'effect' in this article. In its pursuit I shall take you (my dear mature reader) back to your school days and amaze you with what you already know about Place Notation and Integer Arithmetic. For my younger readers I shall assume that you are (or have been) 'top stream' pupils and will have followed a similar course in arithmetic as your parents. In the usual manner of my dissertations I shall digress with matters not too computerate (my spell checker doesn't like that word) but I hope relevant to those of you who wish to become even more numerate (than you undoubtedly are).

Paper or disc

However, before I start, let me have a little moan. Thank you for all your letters but please, please, if you have a specific problem then send me a disc with a typical (or contrived) file demonstrating the effect or problem. I am getting too many letters with reams of printouts and no disc.

Let me demonstrate the difficulty I have with a typical (but fictitious) example. Imagine that you've sent me (on paper only) a massive spreadsheet and, from your description of what happens I suspect that you have a particular formula wrong – but I'm not sure. Even though you have sent me all the formulae (as a printout) I still have to recreate your spreadsheet on my machine (typing in all the data and formulae) before I can test my theory. You might believe that I am so knowledgeable that I can solve your problem 'on paper'; the truth is that I'm not that good! Furthermore, with some packages I am never confident until I've made a solution work for real.

Anyway, I have now hardened my heart towards those who send me paper and no disc. I have started writing back to them asking for a disc copy of a suitable file. Since then I've found that more than half the problems 'go away'. Is it that I offend people; have they solved the problem themselves; have they been given the answer elsewhere; or is their information so confidential they don't want me to have a disc copy?

One and one makes two

I am not sure exactly how old I was when I discovered the number 'two' but I remember the occasion quite clearly. It was the first number that I understood – indeed, I think that 'two' was the first abstract noun which had any meaning for me. Although many purists would argue that the number 'one' is of more fundamental significance, it is my considered opinion that two must have been the first (natural) number to have been discovered.

I never had need of numbers whilst everything I knew was unique (one of each). I had one mother, one mouth and (after a few minor doubts about the functions of the different bits) one me! It was when I came to match the number of hands I had with the number of mittens I was trying to remove that I discovered the number two.

I soon graduated to 'five' and then to 'ten'. I will spare you the details. It was much later that I discovered the number ‘zero’. It is my considered view that the difficulty people have with understanding the number ‘zero’ is matched only by the complex number (given the symbol ***i*** by mathematicians and ***j*** by engineers – in order not to confuse themselves when they use ***i*** as the symbol for electric current) which, when squared, is equal to ‘minus one’.

The Romans did not have a symbol for zero. The mathematical philosopher, Bertrand Arthur William Russell (1872 to 1970) – by the way, he was an Earl (the third Earl Russell) better known for his support of the Campaign for Nuclear Disarmament and the Easter walk to Aldermaston than his early mathematical prowess – argued strongly in favour of zero being included in the so called Natural numbers – numbers used for counting. Earlier mathematicians argued that zero was an Integer (Integers include negative whole numbers) but, following the Romans, that zero was not a Natural number.

Let me put the question of whether zero is a Natural number to you in a much simplified form: “Can you count the number of people in a room if there are no people there? – If there is nobody in the room then what is it that you’re counting?” If you believe that you can count zero people then you believe (like the Noble Earl) that zero is a Natural (counting) number rather than the larger set called Integers (positive and negative whole numbers and the enigmatic zero).

Whether by rote learning or through the power of deep thought, most of us come to accept that “one and one make two”.

Please allow me a short digression. Each year I have a few (16 year old) students who come to us (at the college where I work) suffering from a condition which I call ‘number blindness’. Let me give you an example of the nature of the difficulty they have with the abstract concepts you and I know as numbers. These students first prove that two pencils and three pencils together make five pencils; they do this by counting the two, the three and then the five together. Then I ask them to make the abstract jump to deduce that two pens and three pens together make five pens (without repeating the counting process). They fail. They have to go through the same addition and counting process with the pens that they did with the pencils. Nor can they deduce that with five pencils and five pens they can distribute them to five students in such a way that each student gets one of each!

I have had many discussions with our Learning Support team about this number blindness (some of these students are highly literate) and my discovery is that these students have problems with many other abstractions (like the relationship between yesterday, today and tomorrow) which most of us handle easily. Even the most kind hearted amongst them find the abstract concept of fairness too vague. I do have a fair amount of success with these number blind students. They do learn how to do arithmetic but they do it by working from the concrete to the abstract – the opposite way from that beloved by many (academic) teachers.

Integer arithmetic

The set of Integers consists of all positive and negative whole numbers (plus zero). There are a lot of them. Without explanation (and as a totally irrelevant digression) let me state that “The Cardinality of the set of Integers is denumerably infinite”. You can’t store the big ones (whether positive or negative) with complete accuracy in a computer which has a finite memory.

In BBC BASIC you can specify that a variable is an Integer by including a % symbol after the variable name (eg year% might be a variable for the year and its value must be a whole number). I'm not sure if my memory is playing tricks with me but I seem to recall that, in BBC BASIC V Integers can't be bigger than about 32 million. The better spreadsheets (such as PipeDream!) automatically detect whether a set of numbers are Integers and, if so, then they use a simplified algorithm for storage and for addition. Some spreadsheets use a simplified algorithm for the multiplication of numbers which it recognises as belonging to the set of Integers.

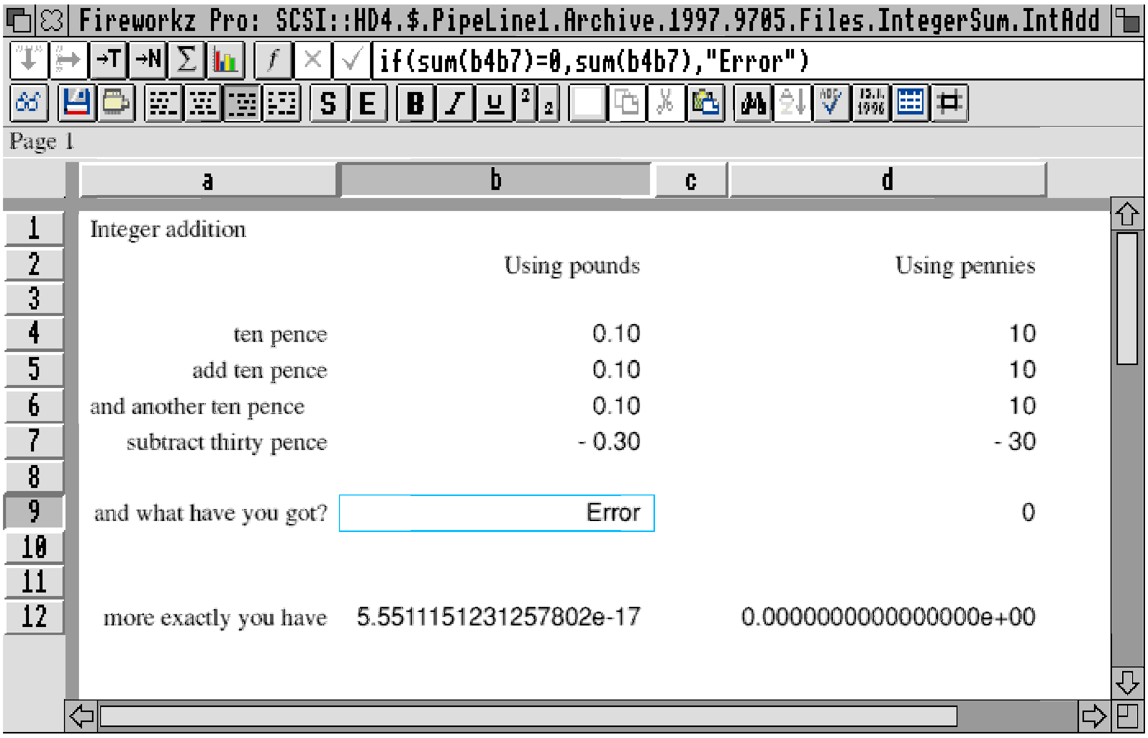
If you set up your spreadsheet in such a way that only Integer Arithmetic is used then it will run much faster and give accurate answers. By accurate I mean that the answer is exact and not an approximation.

For Integers it is is true that "One and One (always) make Two".

(1/10) and (1/10) makes (2/10)

Teaching fractions at school to all but the best pupils began to go out of fashion in the 1960s. There is a strong movement to reintroduce them at primary level now. Many of my 16 to 25 year old students tell me "I was sick (off school) the ***day*** (or week) we did fractions." These students have difficulty entering a height such as sixty two and a quarter inches into a calculator. They enter (621/4) and get 155 and a bit.

Perhaps my example in the heading of this section is not the most appropriate since there are many of you who will cringe at (2/10) and want to reduce the fraction to (1/5). Please bear with my (2/10) whilst I convince you that when you add one tenth to another tenth what you're doing is integer arithmetic and not adding fractions!



Fireworkz Pro: SCSI::HD4.$.PipeLine1.Archive.1997.9705.Files.IntegerSum.IntAdd

```
if(sum(b4b7)=0,sum(b4b7),"Error")
```

Page 1

	a	b	c	d
1	Integer addition			
2		Using pounds		Using pennies
3				
4	ten pence	0.10		10
5	add ten pence	0.10		10
6	and another ten pence	0.10		10
7	subtract thirty pence	- 0.30		- 30
8				
9	and what have you got?	Error		0
10				
11				
12	more exactly you have	5.5511151231257802e-17		0.0000000000000000e+00

Look at the screenshot above and concentrate for the moment on column a. You'll see that my intention is to add together three lots of ten pence and then subtract thirty pence. Move your attention to column d and you'll see the sum executed in integer arithmetic. The answer is zero – and in this case, by zero I mean an exact mathematical zero. This file in Fireworkz format appears on the Archive monthly disc.

Now turn your attention to column b. If you were doing this sum manually (without a calculator or spreadsheet) then I'm sure that you'd use a mathematical concept called 'place notation' in conjunction with integer arithmetic to do the sum. Here's how you would do it. First you would consider the right hand column of figures (the second decimal place column). You would add the numbers of that last column to find zero; you would write down the zero and carry nothing to the one decimal place column. You would then concentrate on the one decimal place column again finding zero and nothing to carry. Finally you would add the pounds column and enter the answer as zero. You have been doing your sum in integer arithmetic and (with a little luck on your side) you would get an ***exact*** answer – the answer zero.

My research with my (not top stream) 16 year old students would indicate that little if anything is taught to them about place notation. By this I mean they are not introduced to Units, Tens and Hundreds columns and they never learn to add up columns of figures starting with the Units column. What happens instead to these (not top stream) students is that they get one token lesson before they are introduced to the calculator. In some of our (worst?) local schools they are given a Resource Pack which gives examples of integer addition and subtraction using place notation. Unfortunately no check is made on whether they have worked through the Pack or just copied the answers from their own secret 'Resource Pack' handed down from an elder brother or sister or friend who has moved on to a higher class. Some Resource Packs include worked answers (which can be copied).

About 80% of my 16 year old students can not add up two two digit numbers without a calculator because they have never learned how to use place notation and integer arithmetic. They even write addition sums horizontally (eg 2 + 3 = 5), which is OK when you're using a calculator, instead of vertically (which is much more convenient when using integer addition and place notation).

$$\begin{array}{r} 2 \\ \underline{+3} \\ 5 \end{array}$$

These students are capable of learning the 'old fashioned' (non calculator) methods and some of them, once motivated, learn quickly. Indeed some find it quite fascinating that addition sums of great complexity (eg using numbers of more than ten digits) can be carried out in columns without a calculator and, unlike a calculator (for these many digit numbers), the answer they get is exact.

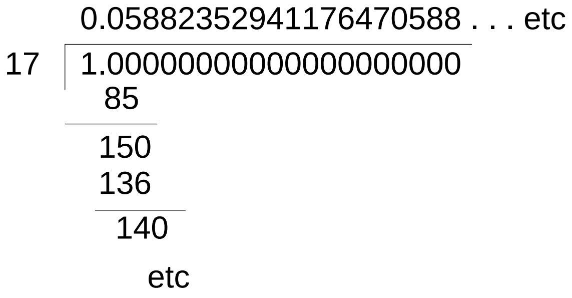
More integer arithmetic

I hope I have explained to you how non calculator addition always uses integer arithmetic even if the numbers in the sum are not Integers. Please allow me to repeat that if you do your sums in integer arithmetic then you'll always get the exact answer and not an approximation. Try adding two 15 digit Integers on a calculator. The answer will be approximate. What does that tell you about the way in which a calculator does sums?

Let me now introduce you to other examples of integer arithmetic which (because most of you will be more mature than my lower stream 16 year olds) you will have learned a long time ago. Long multiplication and long division are executed in integer arithmetic.

The screenshot below shows the conventional layout for executing long division. My example is to convert the fraction (1/17) to a decimal. The answer is a decimal which recurs every 16 digits (that the frequency of this recurrence is sixteen, one less than the prime number 17, is not a coincidence). I have produced this graphic not with a spreadsheet (although I'm sure it's possible in Fireworkz) but using DrawPlus. To save space (and to save me effort) I have shown only the first couple of stages in the long division. The first important one is 5 * 17 = 85. I will assume that you will have tried 6 * 17 and found it to be greater than 100. The 85 is subtracted from 100 to give a remainder of 15. You will remember that a zero is "brought down" from the number which I have shown as 1.000000 etc to make 150. The next step is to discover that 8 * 17 = 136 and that 9 * 17 is more than 150.

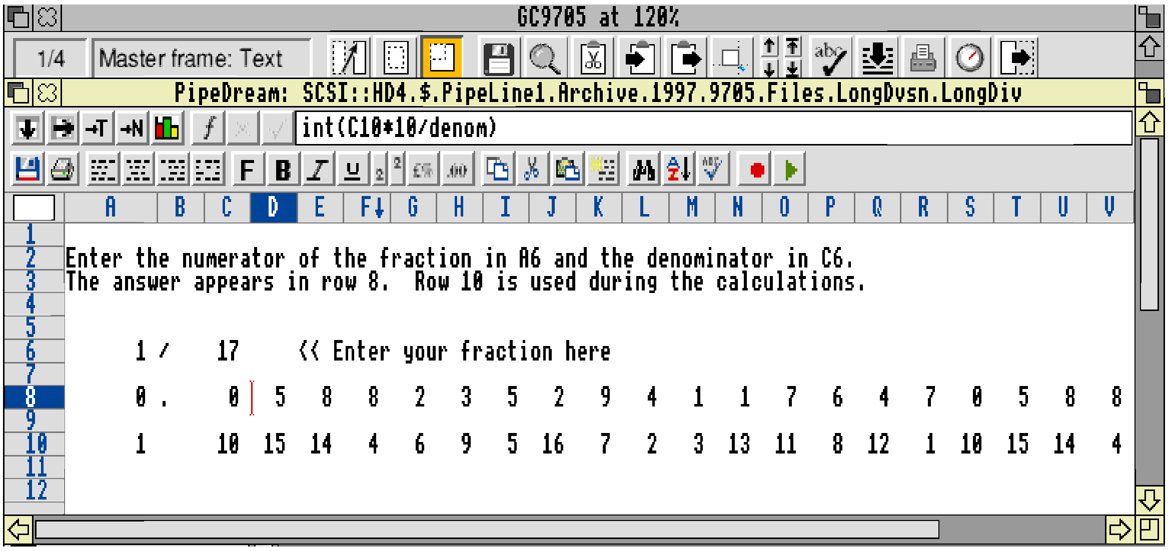
This technique uses integer arithmetic. Every number, the 5 (of the answer) the 85 and the 15 (which becomes 150) are all Integers. You can keep going with this division as far as your patience lasts (the sixteen digit series 0588235294117647 repeats for ever) and your answers will continue to be perfectly exact. If you do the sum (1/17) on a calculator you'll have to accept an approximation of about ten significant figures.



```
        0.05882352941176470588 . . . etc
17    | 1.00000000000000000000
          85
        150
        136
          140
             etc
```

From the screenshot below you will see that I have used PipeDream to calculate the value of (1/17) using the integer arithmetic techniques of the long division sum shown above. You will see that in slot D8 I have arrived at the point where I have decided that 5 * 17 = 85 but that 6 * 17 will be more than 100. The formula in D10 calculates the integer remainder after 5 * 17 has been subtracted from 100. It is mod(C10*10,denom) where *denom* is a PipeDream Name for the content of slot C6 (in this case the number 17).

As an aside I feel an overwhelming need to point out to you that students who don't remember their multiplication tables are unable to do long division. Too many of my students can not multiply 4 * 6 without a calculator. What chance have they with 5 * 17 (which involves carrying a 3 to the tens column)?


GC9705 at 120%
1/4
Master frame: Text
PipeDream: SCSI::HD4.$.PipeLine1.Archive.1997.9705.Files.LongDvsn.LongDiv

```
int(C10*10/denom)

Enter the numerator of the fraction in A6 and the denominator in C6.
The answer appears in row 8.  Row 10 is used during the calculations.

   1 /    17    << Enter your fraction here

   0 .     0  5  8  8  2  3  5  2  9  4  1  1  7  6  4  7  0  5  8  8

   1      10 15 14  4  6  9  5 16  7  2  3 13 11  8 12  1 10 15 14  4
```

This PipeDream file (as all the others of this article) is on the Archive monthly disc. You can replicate the formulae in D8 and D10 as far to the right as you wish. You can change the values in slots A6 and C6; these contain the numerator and denominator of the fraction respectively. In the screenshot I have found the value of (1/17) by entering 1 in A6 and 17 in D6; the sequence of 16 digits (in the answer) repeats for ever. You might find it interesting to see how many digits are generated by a fraction such as (1/101) before the sequence starts to repeat. As a helpful tip I suggest that you transpose the sum so that that which is now row 8 becomes column A and that which is row 10 becomes column B or C; it is easier to replicate down than across.

Let me summarise where we are. The arithmetic that you (my dear mature reader) will have learned at school for addition, subtraction, multiplication and division was integer arithmetic. Integer arithmetic is easy to apply to numbers of any magnitude once you have got to grips with place notation (Units, Tens, Hundreds, etc). You will have learned to do addition in columns rather than with the sum laid out horizontally.

It is unfortunate that many teachers who had charge of my current 16 year old students in their early years never understood the usefulness of place notation (and so they skimped it); a consequence of this lack of understanding (on the part of the teachers) is that I have to spend time on place notation before I can teach those topics for which the college is paid.

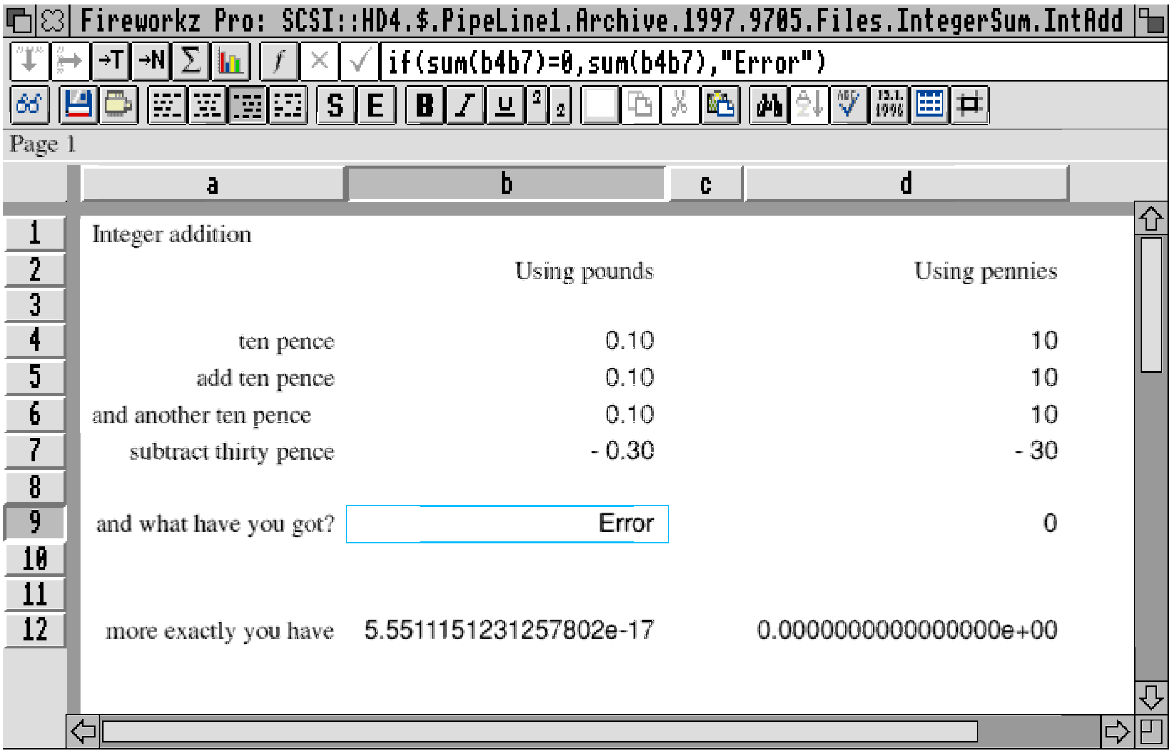
0.1 and 0.1 makes 0.2?

Nowadays decimals are ‘taught’ (with a Resource Pack?) but again, unless the pupil was in the top stream, the formal teaching would have been brief. The pupil will have been encouraged to use a calculator prior to being introduced to decimals and may never have learned how to convert fractions to decimals (with a calculator).

Allow me another digression. About half of my GNVQ Advanced and nearly all my GNVQ Intermediate Students are unable to say how many pennies there are in £2.35 without access to a calculator. Those that have a calculator multiply the 2.35 (pounds) by 100 to reach the correct answer, 235 (pennies)! It is only after many repetitions of similar ‘sums’ that I am able to convince them of the relationship between the number in the question and that in the answer. Perhaps you, my more mature reader, are of a generation that was taught formally (and/or by rote) that “multiplying by 100 shifts the decimal point two places to the right.” If you are not of that generation then this ‘short cut’ of moving the decimal point may be a revelation to you!

Before I get too many letters from proud parents of top stream children let me say that I am not referring to the top 20% of pupils who have been taught multiplication tables, decimals, fractions and even some algebra. My comments refer to the next 20% of pupils below them. I don’t get the remaining 60% of students – I think I’m not clever enough to teach them and I admire those of my colleagues who achieve wonderful results.

Let us return to the Fireworkz file repeated in the screenshot below.


Fireworkz Pro: SCSI::HD4.$.PipeLine1.Archive.1997.9705.Files.IntegerSum.IntAdd
S
E
B
I
U
Page 1

if(sum(b4b7)=0,sum(b4b7),"Error")

	a	b	c	d
1	Integer addition			
2		Using pounds		Using pennies
3				
4	ten pence	0.10		10
5	add ten pence	0.10		10
6	and another ten pence	0.10		10
7	subtract thirty pence	- 0.30		- 30
8				
9	and what have you got?	Error		0
10				
11				
12	more exactly you have	5.5511151231257802e-17		0.0000000000000000e+00

In writing my monthly articles for Archive I have become acutely aware that you are a highly observant and attentive readership; it is hard to slip one by you. I have no doubt that (quite a few paragraphs ago now) you will have noticed that the formula in slot b9 has returned the word “Error”. The formula in b9 is – if(sum(b4b7)=0,sum(b4b7),“Error”). The fact that “Error” is returned shows that when using Fireworkz the result of adding three lots of 0.1 and subtracting 0.3 is not a true (exact) zero.

The conclusion to which we must come is that the sum 0.1 + 0.1 + 0.1 − 0.3 is too hard for Fireworkz – it can’t give us the right answer! You will remember that when Fireworkz used Integer arithmetic (in column d) it did get the right answer. You, my dear mature reader, will be ahead of me. You will have realised by now that it is because Fireworkz has not used Integer arithmetic in column b that it has come to grief.

In slot b12 I have displayed the actual result found by Fireworkz. The (rather difficult to read) number in that slot is: 5.5511151231257802e−17. For those of you unfamiliar with scientific notation the e−17 at the end means “move the decimal point 17 places to the left.

The number in more familiar notation is: 0.000000000000000055511151231257802 – there are 16 zeros in front of the first significant digit (a 5). This is a small number and pretty close to zero; but that’s no excuse – it isn’t zero so something’s gone wrong.

If Fireworkz doesn’t use Integer arithmetic then how does it do the sum? Bear with me a moment whilst I digress in order to assure you that Fireworkz is not alone is finding the sum in column b too hard.

Other spreadsheets

I have four spreadsheets on my machine: PipeDream, Fireworkz, Schema 2 and Eureka. In each of them I entered the following expression into a slot:

1 − (0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1)

It should return zero. Using each of the spreadsheets in turn I have turned up the accuracy of the displayed answer as far as I can go and I found the following:

1.11022302462515654e−16	Fireworkz
1.1102230246251565e−16	PipeDream
1.110223024625157e−16	Eureka
1.11022302462516e−16	Schema

Remember that the e−16 means that you have to ‘shift’ the decimal point 16 places; the number given by Fireworkz (the most ‘accurate’) is:

0.000000000000000111022302462515654

This is a small number – but it is not zero! Why is this? Have just a little more patience.

Other machines

It will not have escaped you that all the answers are similar. This implies that the ‘fault’ does not lie with Fireworkz but is more fundamental. Maybe it’s a fault of our beloved Archimedes.

Again, let me reassure you. The results I have obtained on the Archimedes are not much different from (indeed better than) those using Excel on our college Windows machine.

Floating point arithmetic

As a general rule calculators do not do sums using Integer arithmetic. They use a mathematical technique known as Floating Point arithmetic. When you are in charge of programming a computer to do sums often you have the choice of using Integer arithmetic or Floating Point arithmetic.

In BASIC you can choose your variables to be Integers (declaring year% as your variable name forces it to be an integer) or something incorrectly called Real.

A ‘real’ mathematician would describe the ‘Real’ numbers used by BASIC as being closer to Rational numbers rather than Real numbers. A rough and ready definition of a Rational number is a number which can be expressed as the ratio of two Integers. It can be proved that the square root of two can not be expressed as the ratio of any pair of whole numbers. The square root of two is a Real number but it can not be stored in a computer with complete accuracy. No computer can store with complete accuracy all those numbers which ‘real’ Mathematicians call Real. Allow me to state that “The Cardinality of the set of Real numbers is non-denumerably infinite”. Non-denumerable infinities are infinitely larger than the denumerable infinities; there are a infinitely more Real numbers than there are Integers, indeed, you can’t count the number of Real numbers (using the integers) because there are so many of them.

Also (so that the more knowledgeable amongst you won’t take me to task for spreading misinformation) let me add that not every Rational number can be stored with complete accuracy on a computer. The number (1/17) is Rational (the ratio of two Integers) but it can not be stored with complete accuracy on a computer.

A computer is capable of storing some (indeed billions) of the (infinitely large set of) integers and many but (as we shall see) not all Rational numbers. It is the fact that not all Rational numbers can be stored exactly which is at the root of my correspondents’ spreadsheet problems.

The number (1/10) in floating point

Earlier in this article (in slot b12 of the spreadsheet shown in the first screenshot) we generated the number 5.5511151231257802e−17. I described this format for the number as scientific notation. It is also close to decimal floating point notation. The number 5.5511151231257802 is called the ‘mantissa’ and the −17 is called the ‘exponent’. The difference between scientific notation and floating point notation is that 5.5511151231257802e−17 is regarded as one single number whereas, in floating point notation, the mantissa (without any decimal point) and exponent are stored (in the computer) as two separate numbers.

Similarly binary floating point consists of a ‘mantissa’ and an ‘exponent’. The mantissa is the sequence of binary digits which you get if you divide 1 by 10 in binary (but without the binary equivalent of a decimal point). The exponent is a way of expressing where the ‘binary’ point (like a decimal point) is placed.

Storing (1/10) as a binary floating point number is impossible. If you divide 1 by 10 in binary you do not get a terminating series but a series that goes on for ever. Let me give you an analogy in decimal. You will remember that the number (1/17), a Rational number, when expressed as a decimal consists of a series of sixteen digits which repeat for ever. If you divide the number 1 by 7 using long division then you will find that it goes on for ever with a series that repeats every six digits and starts 0.142857 142857 142857 ....

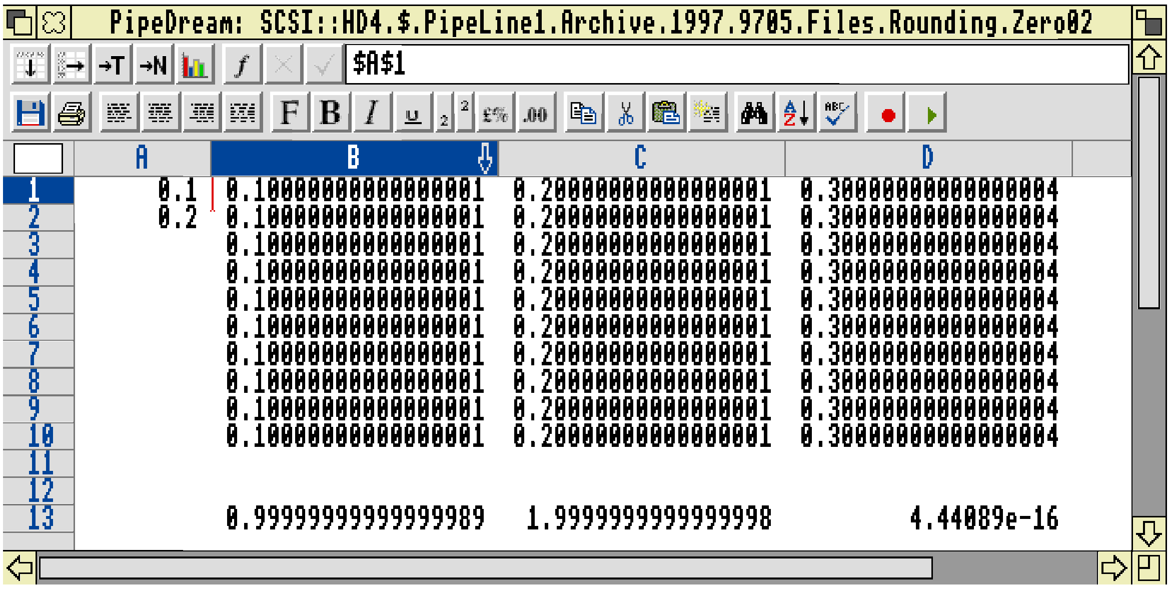
This infinitely long sequence in decimal for (1/17) and (1/7) happens because neither 17 nor 7 can be ‘built up’ from the prime factors of the number base, 10 (whose prime factors are 2 and 5). In decimal the number 16 can be ‘built up’ by multiplying four lots of 2 together. Because of this 1 divided by 16 in decimal does terminate; in fact the answer in decimal is 0.0625 exactly. Fractions with numbers such as 25, 50, 64 and 128 in the denominator will terminate in decimal because they can be ‘built up’ from the product of powers of 2 and 5. Any fraction with numbers in the denominator which can not be built up out of 2s and 5s (eg denominators which have as a factor 3, 7, 11, 13, 17, etc) never terminate.

The number base for binary is 2 and it is prime. The only numbers with mantissae which terminate in binary are powers of 2 such as 2, 4, 8, 16, 32, etc. Numbers which do not terminate in decimal will not terminate in binary. The consequence of this is that an exact representation of such a number can not be stored in floating point binary.

Let us consider numbers which terminate in decimal. Some, such as (1/128) also terminate in binary. Others such as (1/10) which terminate in decimal (1/10 = 0.1) terminate because the denominator is divisible by 5. They will not terminate (viz they form an infinite series) in binary. The true, correct series of digits for the binary mantissa does not terminate so, in the computer, it has to be truncated (chopped off!) and the number used by the computer is approximate. The amount ‘lost’ by truncating 0.1 is about 0.000 000 000 000 000 01.

Is 0.100 000 000 000 000 01 > 0.1 ?

The small error, 0.000 000 000 000 000 01, might seem trivial to you but it isn’t and attempts to minimise it causes some strange effects. Have a look at the screenshot below.



PipeDream: SCSI::HD4.$.PipeLine1.Archive.1997.9705.Files.Rounding.Zero02

A1

	A	B	C	D
1	0.1	0.10000000000000001	0.20000000000000001	0.30000000000000004
2	0.2	0.10000000000000001	0.20000000000000001	0.30000000000000004
3		0.10000000000000001	0.20000000000000001	0.30000000000000004
4		0.10000000000000001	0.20000000000000001	0.30000000000000004
5		0.10000000000000001	0.20000000000000001	0.30000000000000004
6		0.10000000000000001	0.20000000000000001	0.30000000000000004
7		0.10000000000000001	0.20000000000000001	0.30000000000000004
8		0.10000000000000001	0.20000000000000001	0.30000000000000004
9		0.10000000000000001	0.20000000000000001	0.30000000000000004
10		0.10000000000000001	0.20000000000000001	0.30000000000000004
11				
12				
13		0.99999999999999989	1.9999999999999998	4.44089e-16

Let me assure you that I’ve not cheated (well not much). Slot A1 contains 0.1 entered as 0.1 in the formula line. You get exactly the same result if you enter 1/10 into the formula line. You will see that slot B1 contains the slot reference A1 (with $s to help me replicate the slot reference). You might expect that B1 would contain the same number as A1. It might not look like it but it does. If you have the Archive monthly disc then you’ll be able to show the content of A1 to seventeen significant figures is 0.100 000 000 000 000 01.

Now I guess that you’d reckon that this number (as displayed) is slightly larger than 0.1. You’re wrong! The formula in B13 is – sum(B1B10) – and you’ll see that it is smaller than the 1.000 000 000 000 000 1 which you might have expected. Indeed the sum(B1B10) is smaller than 1.0 and this can be confirmed (as I have done off screen in slot B15) by entering the formula – int(B13) – (the integer part of B13) which returns the value zero.

There is another interesting effect in column C. You will see that the sum of column C appears to be only a couple of digits in the last place smaller than 2.0. Wrong again! Once more (off screen in slot C15) the formula – mod(C13,1) – (the remainder after dividing C13 by 1) returns 0.999 999 999 999 999 78 and not 0.999 999 999 999 999 98 as you might have expected.

Spreadsheet 'errors'

I have had many discs (and letters) containing sheets that "don't work" because two values that should be equal aren't.

Let me give you one simple example. A standard check on any (two dimensional) array of numbers is to add down the columns to generate a row of sub-totals, add across the rows to generate a column of sub-totals and then find the grand-total two ways by summing the sub-total column and the sub-total row separately. The two values should be equal. A spreadsheet containing Rational rather than Integer numbers will fail this test.

In conclusion

Integer arithmetic always gives exactly correct answers. Floating Point arithmetic gives answers which are approximate. Calculators (always) and spreadsheets (usually) use Floating Point rather than Integer arithmetic. Most checks on whether a spreadsheet is giving the correct answers (or not) assume that Integer rather than Floating Point arithmetic has been used. Remember too that the int(,) and mod(,) functions all too often give wrong and unexpected answers.

It is my hope that you now understand (at least in part) why the conventional checks devised in the days when Integer arithmetic was commonplace fail and why int(,) and mod(,) give wrong and unexpected answers..

I shall leave until a later article descriptions of those techniques (suitable for Floating Point arithmetic) which will allow you to check that a spreadsheet is working as near correctly as it can. In a later article I shall describe how to ensure that the int(,) and mod(,) functions return 'correct' answers.

Finally

My address is that of Abacus Training (see back inside cover). If you have any comments or contributions to this discussion then please write to me. No! I am not yet on e-mail but Paul is trying hard to persuade me. When I retire (by virtue of my old age) in July 1997 I shall seriously consider whether I have the time to 'surf the net'. Bear with me until then and send me discs and pieces of paper by (what I believe is called) 'snail mail'.