Standard Mathematical Input Notation

This paper is concerned with the need for a language interface that allows users to 'write' mathematical symbols on the computer screen in as simple a way as possible, and outlines one way in which this could be done.

Introduction

There is an increasing use of the computer for educational purposes and an essential part of that is getting input from the user - whether for a word-processor or in response to a question in a test. And wouldn't it be a great help if it was available for use in writing Email? It is at that point, in mathematics, that difficulties arise, in 'writing' the necessary mathematical symbols. It is specifically those difficulties that are addressed here.

At present, tests in mathematics are very restricted in what they can ask for and it is not surprising that the multiple-choice format is very popular. It is relatively easy to set, and very easy to mark.

Writers and programmers show much ingenuity in overcoming the difficulty of input and format but, unfortunately, this usually results in a display which carries a 'suggestion' as to what the answer must be. What we need is the ability to write mathematics on the screen (via the keyboard) in the way that we would normally write it on paper, or as we would expect to see it printed.

There are already several programs which allow mathematics to be written to the screen, but these are dedicated to the purpose, and only really suitable for specialist users. MathType is one example, very good it is too, and was used in the writing of this paper. (It is the basis of the Equation Editor in Word.) But it is certainly not simple enough for the more casual user.

Contents

There is no difficulty when the answer is simply a numeric one -

Question	Answer
Solve $3x + 1 = 7$	x = 2 (or just simply 2)

but difficulties arise with the answers needed to these -

Write 35 as the product of two primes	5×7
Write a division sum with an answer of 4	$12 \div 3$
Multiply out $(x + 2)(x - 3)$	$x^2 - x - 6$
Simplify $x^3y \times xy^4$	x^4y^5
Reduce $\frac{25}{40}$	$\frac{5}{8}$
Give the two roots of $x^2 = 5$	$x = \pm\sqrt{5}$
Express the length of AC as a surd.	$\sqrt{41}$
Give the formula for finding the area of a circle	$A = \pi r^2$

What is it that the user cannot do?

Dealing only with elementary mathematics, for the moment, none of these symbols is available from the conventional keyboard -

$\times$ $\div$ $\sqrt{}$ $\leq$ $\geq$ $\pm$ π

and the correct representation of **fractions** and **indices** is not possible.

Some 'fudging' can be done, such as -

*	for	$\times$
/	for	$\div$
<=	for	$\leq$
>=	for	$\geq$
5^2	for	5^2
+/-	for	$\pm$
5/8	for	$\frac{5}{8}$

But none of this is ideal in that it is another form of representation on screen which is different from the way

- we want pupils to write down their mathematics
- it is printed in text-books or any other printed source on paper
- the question is presented on screen most likely.

Another (minor) drawback is that the shift-key is needed for some of the symbols which are available, such as: + ( ) * ^ < > % for example, but not for others.

All of this adds up to an unfortunate state of affairs having nothing to do with the mathematics itself at all. There does seem to be a clear need for the development of as simple and consistent a method as possible for putting the **correct symbol** on the screen. The digits (and decimal point) present no problem.

Outline of a Possible Solution

First thoughts were given to dealing only with symbols which could not be directly formed at all on screen, but further consideration suggested that in the interest of uniformity all the symbols needed should be generated by a **single key-stroke**. One way this can be done is by matching each symbol to a single ketter in as memorable a way as possible. Thus -

A for **A**dd **S** for **S**ubtract **M** for **M**ultiply by **D** for **D**ivide by

The letters themselves would not appear on-screen - only the symbols.

So keying in 3 A 5 would show 3 + 5

4 S 7 4 - 7

2 M 6 2 × 6

10 D 2 10 ÷ 2

Notice the chosen letters encourage the expression to be said as it is entered so that the appropriate letters are easily remembered.

The system would not be case sensitive so: 3 a 5 4 s 7 2 m 6 10 d 2 could all be input and produce the same effects, though CAPITALS will be used throughout this document for clarity.

Spacing would not be needed in the input, but would be automatically set by the system controlling the display.

The system would accept available equivalent alternatives.
So, + - * / would be treated as A S M D.

Variables Only three would be used at this level - x y and z. Keying in any of those letters would result in them appearing on-screen in lower case italic - x y and z.

Brackets would be generated by

B for **(** Brackets open **C** for Close brackets **)**

So keying in B 4 A X C B Y S 3 C would show $(4 + x)(y - 3)$

Remember that the first expression is never seen, and is certainly not meant to be read, as is happening here. Each key-press produces a response just as when writing so the build-up is gradual and it is easy to see just what has to come next. The whole system would function just like a word-processor and the cursor moved to any point where a forgotten item could be inserted, or a deletion made.
In this case the available ( ) would be accepted and treated as B C.

Equality and **inequality** signs would be generated by

E for **=** Equals **G** for **>** Greater than **L** for **<** Less than

So keying in 5 A X G 1 9 would show $5 + x > 19$

In this case the available = > < would be accepted and treated as E G L.

Indices would be made by keying in

I for Index number follows

So keying in 3 I 4 would show 3^4

X I 3 Y I 2 x^3y^2

6 B X I 2 A 3 C B 2 S X I 2 $6(x^2 + 3)(2 - x^2)$

In this case the available ^ would be accepted and treated as I.

The end of the index number would be signalled by almost anything that was not a number (or a decimal point)

Pressing the Enter key signals the expression is finished (the answer is complete)

Fractions would be formed by the (suitable) use of three **F**'s

1st **F** would signal the **start** of the top line of the fraction

2nd **F** would signal the **end** of the top line, and the **start** of the bottom line

3rd **F** would signal the **end** of the entire fraction

So keying in F 5 F 8 F would show $\frac{5}{8}$

F 3 B X A 4 C F X I 2 S 4 F gives $\frac{3(x+4)}{x^2-4}$

The Enter ↵ key would be acceptable in place of the 2nd and/or the 3rd F

Roots, in a way similar to fractions, requires the use of two **R**'s

1st **R** would signal the **start** of the Root $\sqrt{}$

2nd **R** would signal the **end** of the Root

So keying in R 3 R would show $\sqrt{3}$

R 3 X I 2 A Y R gives $\sqrt{3x^2+y}$

The Enter ↵ key would be acceptable in place of the last R

It might be that only **square roots** are accepted at this level but,

keying in I 3 R X R would show $\sqrt[3]{x}$

Negative numbers would be made by pressing

N for Negative *before* a number or variable

So keying in N 3 would show $^{-}3$

N X would show ^{-}x

Negation is also signalled by pressing

N but this time *before* the equality sign

So keying in N E would show $\neq$ (Not Equal to)

.

Summary Table of a Possible Notation

Letter	Symbol	Meaning	Notes
A	+	**A**dd	
B	(	**B**racket open	
C	)	**C**lose bracket	
D	÷	**D**ivided by	
E	=	**E**quals	
F		**F**raction	Used 3 times:- start; line-change; finish
G	>	**G**reater than	
H		*unallocated*	Could be used for 'Help'
I		**I**ndex number starts	
J		*unallocated*	
K		*unallocated*	
L	<	**L**ess than	
M	×	**M**ultiplied by	
N		**N**egative or **N**ot	Depends upon what follows
O		*unallocated*	Could be used to generate degree º symbol and (press twice for) %
P	π	**P**i	
Q		*unallocated*	
R	√	**R**oot	Used twice:- start; finish
S	-	**S**ubtract	
T		*unallocated*	
U		*unallocated*	
V		*unallocated*	
W		*unallocated*	
X	x	a variable	
Y	y	a variable	
Z	z	a variable	

Combinations (order does not matter)

AS	±	**A**dd/**S**ubtract
NE	≠	**N**ot **E**qual
GE	≥	**G**reater than or **E**qual to
LE	≤	**L**ess than or **E**qual to

The letters are NOT case sensitive.

Alternative acceptable inputs not listed here.

Here is shown how a complex expression such as

$$\sqrt[3]{\frac{x^2(y+8)(z-4)^2}{\pi(x+y)-\pi^4}}$$

is built up by a series of simple (and easily memorized) key-strokes.

The three columns show

- the "thinking"
- the action
- what is seen on the screen.

To do this -	*key in -*	*and see -*
put Index **3**	**I 3**	^{3}
start a Root	**R**	$\sqrt[3]{}$
start Fraction top with x	**F X**	$\sqrt[3]{x}$
put Index **2**	**I 2**	$\sqrt[3]{x^2}$
open Bracket	**B**	$\sqrt[3]{x^2(}$
y Add **8**	**Y A 8**	$\sqrt[3]{x^2(y+8}$
Close bracket	**C**	$\sqrt[3]{x^2(y+8)}$
open Bracket	**B**	$\sqrt[3]{x^2(y+8)(}$
z Subtract **4**	**Z S 4**	$\sqrt[3]{x^2(y+8)(z-4}$
Close bracket	**C**	$\sqrt[3]{x^2(y+8)(z-4)}$
put Index **2**	**I 2**	$\sqrt[3]{x^2(y+8)(z-4)^2}$
start Fraction bottom with π	**F P**	$\sqrt[3]{\frac{x^2(y+8)(z-4)^2}{\pi}}$
open Bracket	**B**	$\sqrt[3]{\frac{x^2(y+8)(z-4)^2}{\pi(}}$
x Add y and Close bracket	**X A Y C**	$\sqrt[3]{\frac{x^2(y+8)(z-4)^2}{\pi(x+y)}}$
Subtract	**S**	$\sqrt[3]{\frac{x^2(y+8)(z-4)^2}{\pi(x+y)-}}$
π Index **4**	**P I 4**	$\sqrt[3]{\frac{x^2(y+8)(z-4)^2}{\pi(x+y)-\pi^4}}$

finish Fraction, finish Root
or press Enter to complete everything

The broad outline of a system has been covered in the previous pages, but that is no more than an idea and there are clearly many other things to be considered, such as

- How "intelligent" should the system be. A close analogy here is with modern word-processors. These have a spell-checker incorporated as standard and a grammer-checker is usually an available extra. So, should a program concerned with the input of mathematical notation offer similar facilities? And then, would you want them in a test?
- Writing words (such as the units in an answer) is not possible since the keyboard is dedicated to producing symbols at this point. This could be accomodated by a mode-change (say press F5 for ON/OFF)
- In a similar way to the above, any letter could be made a variable (*and printed in italic*). Or possibly this could be better managed by use of the more memorable V-key.
- Help needs to be available throughout at the press of a single key. This could include some tutorials. The bottom two lines of the screen should carry a continuous display summary of (most of) the letters and their meaning.

The most commonly-known character code which is used as the background standard for the generation and transmission of text is that known as **ASCII** which is the

American **S**tandard **C**ode for **I**nformation **I**nterchange

What is it?

In any character coding system, each of the characters, no matter what it actually looks like, is identified by a distinct number, and for efficiency, this number should be no longer than one byte.

In the earlier days of computing a byte was made up of 8 bits, which meant that the largest number that could be carried by one byte was 255 (= 2^8 - 1). So a total of 256 numbers (0 to 255) could be registered. The ASCII defined what 256 characters these numbers should be related to. For instance, 49 *stands for* 0, 65 *stands for* A, 97 *stands for* a, 40 *stands for* (, 93 *stands for*] and so on.Obviously the complete alphabet (both capitals and lower case), the digits, punctuation, and many symbols (such as + £ - * # etc.) had to be in. But, codes were also needed to show where lines ended and new ones started, where tabs were placed, what spacing was required, and so on. This absorbed about one-half of the available 256 numbers. But that was only for the English language, and non-technical at that - there was no allocation for things like a multiplication sign. For the remaining numbers other languages were covered provided that they were basically variations on the English alphabet, so letters like á å ë ø æ ñ plus several others were coded - in both capital and lower case.

However, Arabic, Chinese, Japanese, Gujarati, Braille and many other languages did not get a look in, and nor did hundreds of symbols or technical signs - there was just no room. The consequence of this was that each country had to have its own version of the ASCII coding system. Clearly this worked fine within their own borders, but it was not much help when it came to international working.

As computers became bigger and more powerful the byte was expanded to 16 bits and that allowed 65,536 (=2^{16}) numbers to be defined by just one byte. Which also allowed a completely new, and bigger, coding system to be devised. This system is know as **Unicode**. And, while much has been done, the work is still in progress. To date (2000, and they are now on version 3.1), only about 50,000 of the numbers have been allocated. Over 40 languages and derivatives of them are now covered, as well as a very wide range of symbols and ideographs.

Unfortunately, defining a coding system is one thing - implementation is another, and that has some way to go yet. So, while it is possible to see, for instance, Chinese characters on the screen, it will still be necessary to make arrangements for that - it will not yet happen automatically.

However the importance of Unicode for the proposalsoutlined in this paper is that all (or nearly all) of the ingredients required are available ‘out there’ waiting to be used.

For those interested in Unicode the Web-site to visit is

http://www.unicode.org/