symbols for the British pound (£, Latin *libra*), the dollar ($, an abbreviation of *pesos*), the cent (¢), and the sign ℞ (for the Latin *recipe*, or the imperative "take") displayed by pharmacists.[19,20] Cardan used ℞ for "root" (Latin *radix*) in 1539, and we still talk of "extracting" (pulling out) the root. Although Euler believed the square root symbol ($\surd$) to be the deformed letter *r* (abbreviating *radix*), Cajori doubts this, suggesting its origin might be a dot.[21]

We are taught that it is a simple step from exponents to logarithms, and few developments have been more important. Laplace recognized our immense debt to Napier in his well-known remark about logarithms, that, by halving the labor, they had doubled the life of the astronomer and mathematician; but we seldom think of the primitive state of the conceptual tools available in 1614, or recognize Napier's genius. In his day, algebra differed little from arithmetic, and the notation we take for granted was almost nonexistent. Napier's discovery came three years before he invented the decimal point, and less than 60 years after Recorde introduced the equals sign and first used the signs + and − in an English book. Just how Napier succeeded in calculating his table of logarithms is well described by Gittleman.[22]

In a volume commemorating the 300th anniversary of Napier's *Description of the Marvellous Canon of Logarithms*, Glaisher well expressed the power of good notation: "Nothing in the history of mathematics is to me so surprising or impressive as the power it has gained by its notation or language. . . . By his invention [of logarithms] Napier introduced a new function into mathematics. . . . When mathematical notation has reached a point where the product of n xs was replaced by x^n, and the extension of the law $x^m \cdot x^n = x^{m+n}$ has suggested $x^{1/2} \cdot x^{1/2} = x$, so that $x^{1/2}$ could be taken to denote the square root of x, then the fractional exponents would follow as a matter of course, and the tabulation of x in the equation $10^x = y$ for integral values of y might naturally suggest itself as a means of performing multiplication by addition. But in Napier's time, when there was practically no notation, his discovery or invention was accomplished by mind alone without any aid from symbols."[23] (See also Reference 24.)

"We who live in an age when algebraical notation has been extensively developed can realise only by an effort how slow and difficult was any step in mathematics until its own language had begun to arise, and how great was the mental power shown in Napier's conception and its realisation. . . . In our days when the rules of computation are precise, and when the construction of instruments has reached a high state of efficiency, the processes of multiplication and other arithmetical operations can be performed by machines designed for the purpose. These apparatuses which save mental strain and time are effective aids to calculation, and they may be regarded as the modern successors to Napier's rods."[23]

APL and functional programming

APL's concise notation helps us grasp the intellectual content of an algorithm without the distraction of extraneous and irrelevant matters prescribed by a machine. APL is a succinct and admirably consistent language that not only uses verbs (functions) to act on nouns (data arrays), but uses adverbs and conjunctions (operators) to derive new verbs, and permits definition of new verbs, adverbs, and conjunctions. It has the subtlety and suggestiveness which, as Bertrand Russell said, makes a good notation "seem almost like a live teacher,"[25] and, to quote Pledge, "Suggestiveness is the essential service of symbolism."[26]

With APL, the goal of *functional programming* (Backus, 1978) can be achieved. The word *function* (derived from *functio*, meaning a performance or execution) was used at the end of the 17th century by mathematicians writing in Latin. Leibniz, who gave us many terms such as *constant*, *variable*, and *parameter*, used "function" in our sense in 1673. Euler used the symbol *f* for a function in 1734, and in 1754 used the notation *f:(a,n)* for a function of the variables a and n, i.e., to state that the result depends upon the current values of a and n. Iverson does better than this; in 1976 his method of *direct definition*[27] of functions shows formally exactly how the result is derived from the arguments, and Euler's parentheses are not needed.

The relationship between ordinary APL and direct definition is illustrated by the following examples:

In ordinary APL:

```
     ∇Z← A PLUS B
[1]    Z← A + B
[2]  ∇

      3 PLUS 4
7
```