Letters to the Editor

A Machine Algorithm for Processing Calendar Dates

Key Words and Phrases: calendars, calendar date, Julian date, Gregorian date, Gregorian calendar, Julian calendar, time interval, continuous day count, Fortran statement function, arithmetic statement function, function
CR Categories: 3.1, 3.10, 3.11, 3.15, 3.2, 4.9

EDITOR:

The need to determine the elapsed number of days between any two given calendar dates seems to be a common problem in writing computer programs. Generally speaking, rather elaborate logic is needed to take into account the varying number of days in each month, plus the occurrence of leap years, and perhaps also the omission of a February 29 in years divisible evenly by 100 but not by 400. The following algorithm takes advantage of the truncation feature of integer arithmetic in the FORTRAN programming language to solve this problem in a very compact way. It converts any given calendar date (I = year; J = month, a number from 1 to 12; K = day of month) to a Julian Date (JD)—a continuous count of days from an epoch in the very distant past. For example, noon at Greenwich, England, on January 1, 1970, is the beginning of Julian Date 2,440,588. So if I = 1970, J = 1, and K = 1, then the algorithm gives JD = 2440588. Clearly, the interval between any two calendar dates (on the *Gregorian Calendar*) can be found by obtaining the Julian Date for each, and finding the difference.

The algorithm is given below (presented as a FORTRAN arithmetic statement function). It is valid for any Gregorian Date producing a Julian Date greater than zero.

```
JD (I, J, K) = K - 32075 + 1461*(I + 4800 + (J - 14)/12)/4
             + 367*(J - 2 - (J - 14)/12*12)/12 - 3
             *((I + 4900 + (J - 14)/12)/100)/4
```

The authors have yet to discover the algorithm of comparable compactness for converting a Julian Date back to a calendar date. But in preference to leaving the problem undiscussed, the following is offered (presented as a FORTRAN subroutine):

```
SUBROUTINE DATE (JD, I, J, K)
L = JD + 68569
N = 4*L/146097
L = L - (146097*N + 3)/4
I = 4000*(L + 1)/1461001
L = L - 1461*I/4 + 31
J = 80*L/2447
K = L - 2447*J/80
L = J/11
J = J + 2 - 12*L
I = 100*(N - 49) + I + L
RETURN
END
```

HENRY F. FLIEGEL
Georgetown University Observatory
Washington, D.C.
AND
THOMAS C. VAN FLANDERN
U.S. Naval Observatory
Washington, DC 20390

On "Prime Phrase" in Feldman and Gries Paper

Key Words and Phrases: compilers, operator precedence, translator writing systems
CR Categories: 4.12

EDITOR:

The article by Feldman and Gries on Translator Writing Systems [*Comm. ACM 11*, 2 (Feb. 1968), 77–113] is an excellent one, but one error in it ought to be corrected. In their description of operator precedence parsing on page 82, they give a definition of a prime phrase as "a phrase which contains no phrase other than itself but at least one terminal character." In Floyd's original article on the subject [*J.ACM 10* (Jul. 1963), 316–333], a prime phrase was defined to be a phrase which contains no *prime* phrase other than itself but at least one terminal character. It may not be obvious that the two definitions are not equivalent. The difference shows up when the grammar in question has a production whose right side consists of one nonterminal symbol.

Consider the grammar:

$$S \rightarrow aU_1b$$
$$U_1 \rightarrow U_2$$
$$U_2 \rightarrow b$$

Then in the sentence "aU_2b", "U_2" is a phrase but not a prime phrase by either definition. Hence "aU_2b" is a prime phrase by Floyd but not by Feldman and Gries.

If Feldman and Gries's definition is used, a parse may reach a state where there are no prime phrases and hence the parse cannot be continued.

PAUL ABRAHAMS
New York University
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, NY 10012

Do You Use Microfiche?

Key Words and Phrases: microfiche, user study, document surrogates
CR Categories: 3.7, 3.72, 3.79

EDITOR:

There has been a growing tendency for Federal agencies to encourage, usually through differential pricing, the distribution of microfiche instead of full size copies of reports. The economic advantages of microfiche are obvious to the issuing agencies (and to the General Accounting Office); agency distribution lists show that some libraries actually prefer to receive microfiche. We have little information, however, on the acceptance and use of microfiche by individual scientists and engineers.

I have been asked by COSATI (the Committee on Scientific and Technical Information of the Federal Council for Science and Technology) to look into this matter. Those of your readers who have actually been offered the opportunity of using microfiche and have strong opinions on such subjects as legibility, convenience, availability, and quality of readers and reader-printers and kindred topics are encouraged to write to me. I am especially interested in hearing from those who have found it possible, or even preferable, to use microfiche in maintaining their personal

report collections. I cannot guarantee to answer individual letters, but all respondents will receive copies of a summary report—in full size, hard copy!

HAROLD WOOSTER
AFOSR/SRI, 1400 Wilson Blvd.
Arlington, VA 22209

Comment on Curriculum 68

Key Words and Phrases: computer science curriculum, computer case studies, system case studies
CR Categories: 1.52

EDITOR:

May I point out a deficiency in Curriculum 68 [*Comm. ACM 11*, 3(Mar. 1968) 151–197]: its lack of orientation to the practitioners of computer systems analysis.

A good education should not be solely directed toward academicians whose only economic justification is to teach in order to turn out recursively new generations of academicians. Such has been the problem in the teaching of economics since it was defined as a subject without institutional content. University economic departments (notably not in the schools of business) have turned out trained economic theoreticians who have found little relationship between their academic knowledge and the existing practices which guide business firms and government. A balanced education in economics must properly emphasize the descriptions of existing economic institutions as well as the inadequate theories of economics.

Thus, in the education of the undergraduate computer scientist (?), emphasis must be given to a description of what a practitioner of computer science does as well as to the teaching of the inadequate theories of the science (?).

If this is not done, practical men will place the required "institutional" courses in other departments of the university. This would be comparable to the current practice of taking business "institutional" courses in the school of business and not in the economics department.

Concretely, I find it difficult to accept an undergraduate curriculum in this field which would not include six academic hours in the study of existing computer systems, i.e. case studies. The college graduate trained in computer science will work most likely in the environment of such systems. Why not, therefore, give the apprentice scientist a frame of reference for the application of theories that are being taught.

In a way, the report attempts to circumvent this criticism by stating:

> It is also likely that the majority of application programmers in such areas as business data processing, scientific research and engineering analysis will continue to be specialists educated in the related subject matter areas, although such students can undoubtedly profit by taking a number of computer courses.

The implication is that computer science can be isolated from a system of application. However, upon close examination the recommended course work is highly slanted towards the needs of physical scientists and engineers. Very much neglected are the knowledge requirements of business systems designers and information technologists.

RAYMOND P. WISHNER
American University
Center for Technology
Washington, D. C.

On Binary Notation

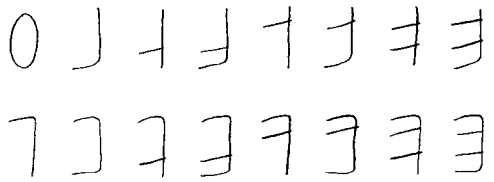
Key Words and Phrases: representation, notation, binary numbers, binary exponential numbers, memory, two
CR Categories: 1.9, 2.44, 4.0, 5.0, 5.29

EDITOR:

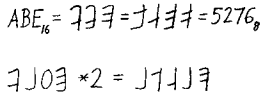
Although more accurate, unambiguous notation is badly needed for numbers of binary origin, I don't think the suggestion [1, 2, 3] of a special symbol for 2^{10} is the best solution. Would it not be more precise and convenient if numbers to be expressed in binary were written in terms of a coefficient and an exponent of two (e.g. 3×2^{12}), rather than using an exponent of the decimal number 1024?

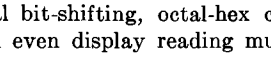
FORTRAN uses the letter "E" to separate the coefficient from its decimal exponent (e.g. $5E7=5\times10^7$); why not use a symbol—perhaps the letter "B"—in the same manner, for the base two? (Some assembly languages use B this way, but some FORTRAN use B to indicate octal. This conflict could be easily resolved by choice of another symbol, or by the use of different symbols for octal or hex digits.) I think 3B20, the decimal binary power notation for 3×2^{20}, makes more sense than the deci-power notation 3bK2 $[3\times(2^{10})^2]$. Furthermore, a memory of B15 is addressable with 15 bits—a fact not apparent from the expression 2^5bK1 or 32bK. An additional benefit is that this scheme is almost analogous to the internal representation of floating-point numbers in most computers. The handy rule of thumb: $2^{10}\approx10^3$, remains equally handy without inventing another unit. Memory sizes of "131K" could be described as 2^{17} or B17 (or verbally, as "bee" seventeen).

With the ridiculous choice of letters A, B, C, D, E, F as hexadecimal number symbols adding to already troublesome problems of distinguishing octal (or hex) numbers from decimal numbers (or variable names), the time is overripe for reconsideration of our number symbols. This should have been done before poor choices gelled into a de facto standard! Why represent some of the nondecimal numbers with the symbols which imply to us a base-ten place-value scheme? Why not use entirely new symbols (and names) for the seven or fifteen nonzero digits needed in octal or hex. Even use of the letters A through P would be an improvement, but entirely new symbols could reflect the binary nature of the system,



e.g.





making mental bit-shifting, octal-hex conversion, binary-point fractions, and even display reading much easier. I believe we would profit from the trade-off between the addition of 15 new symbols and the elimination of monstrosities such as

$$ABE_{16}=5276_8=2750 \text{ and } X'123B9''.$$

REFERENCES:

1. TRYTTEN, JOHN. Letter. *Datamation 10*, 2 (Feb. 1964), 6.
2. MORRISON, DONALD R. Letter to the Editor. *Comm. ACM 11*, 3 (Mar. 1968), 150.
3. GIVENS, WALLACE. Letter to the Editor. *Comm. ACM 11*, 6 (June 1968), 391.

BRUCE A. MARTIN
Applied Mathematics Department
Brookhaven National Laboratory
Associated Universities, Inc.
Upton, Long Island, NY 11973