An Introductory Problem in Symbol Manipulation for the Student

Robert F. Rosin

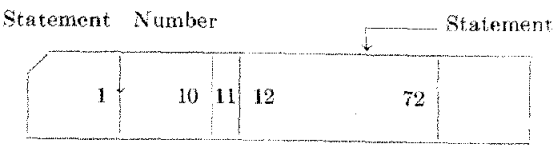
One objective of the course on Introduction to Digital Computers given by B. A. Galler at the University of Michigan is to equip the students with some knowledge of the techniques of automatic programming and to indicate some of the problems involved in such work. The final problem assigned in the course consists of a program to be written in some compiler language which involves symbol manipulation, automatic programming, etc. With the hope others in the computing field will find it of interest, an example of a final problem in this course is given here.

Along the way in the course, which assumes no previous knowledge of computers or programming, students are given general counsel on how to approach the various stumbling blocks in programming. However, no part of the program is ready made for them. By the time they have reached this problem they have programmed solutions to four small problems using both assembly and compiler languages and they have heard several lectures on the fundamental structure of a translator, along with other topics covered in the course.

The problem is formalized (briefly) as follows:

Write (using the MAD language) an interpretive routine which will accept and execute statements constructed in the following way:

A. Variable name: all single letters except A, E, R, P, T
B. Arithmetic operators: +, −, *, /, .P. (exponentation), =, A (absolute value)
C. R and P as input and output operators respectively, each followed in a statement by up to six variable names, with each R statement followed by a card containing the values to be read into the variable locations (in a specified format).
D. Punctuation:) and (
E. T used to indicate a conditional transfer, which will appear as "TnV$_1$ = V$_2$" where n is an integer, V$_1$ and V$_2$ are variable names. The transfer is executed if and only if the values of V$_1$ and V$_2$ are equal. The transfer merely skips all statements following until the first card with n in the appropriate columns is found.
F. An arithmetic substitution statement in the form V = (Expression), where V is the name of a variable and the right side consists of some algebraic expression. The intent is that the value of the expression is the new value of the variable V.
G. In all statements, blank spaces to be ignored and the conventional algebraic hierarchy to be observed.
H. Card format as follows:



The description given to the students suggests the use of internally defined functions to determine whether or not any character is an operator and to find the precedence of any operator. It is also suggested that right and left termination symbols (referred to below as RTERM and LTERM, respectively) should be created internally. The students are instructed to use a statement decomposition similar to that used in MAD, and the flow chart shown in figure 1 is furnished to them. (Note the similarity of this figure to that published by Arden and Graham in the Letters to the Editor section of the *Communications*, November, 1959.) Of course, there are many ways to effectively program the solution to this problem. The illustrations which follow are in the MAD language, which was written for the IBM 704 at The University of Michigan by Arden, Galler, and Graham.

(a) A region is usually established which contains the current values of all variables. To index the appropriate location of any variable the value of the BCD variable name itself may be used, provided this is shifted from its normally left justified position to the low order (or right justified) position (as an integer). One way to do this is to look up the appropriate name in a table and substitute for it the corresponding value in another table. Of course, the entire list of octal combinations is not needed and the variable names may be mapped onto the set of integers from 1 to 26 (actually 0 to 26 since it is handy to have the zero constant on this list). For example:

```
Q       THROUGH Q, FOR A = 1, 1, NAME(A) .E. S(A1)
          .OR. A .G. 26
        WHENEVER A .LE. 26, S(A1) = NEWNAM(A)
        VECTOR VALUES NAME(1) = $A$,$B$,$C$,$D$,$E$,
      1   $F$,$G$,$H$,$I$,$J$,$K$,$L$,$M$,$N$,$O$,$P$,$Q$,
      2   $R$,$S$,$T$,$U$,$V$,$W$,$X$,$Y$,$Z$
        VECTOR VALUES NEWNAM(1) = 1,2,3,4,5,6,7,8,9,10,
      1   11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26
```

This sequence will execute statement Q with A initially 1, then incremented by 1 *until* NAME(A) is equal to S(A1) or A is greater than 26. The next statement will substitute NEWMAN(A) into S(A1) if and only if A is less than or equal to 26, i.e., the name was found in the list.

(b) An internal function may be used to determine whether or not a character is an operator. It is declared to be Boolean in the main program, having the value 1B (True) if its argument is an operator, 0B (False) otherwise.

```
      INTERNAL FUNCTION (X)
      ENTRY TO OPCHK.
      THROUGH L, FOR VALUES OF DATA = $+S,$*S,
    1   $-$,$/$,$+U$,$EXP$,$00000A$,$LTERM$,$RTERM$,
        $-U$
L     WHENEVER X .E. DATA, FUNCTION RETURN 1B
      FUNCTION RETURN 0B
      END OF FUNCTION
```

(c) Compressing the statement (removal of blanks), insertion of termination symbols, conversion of variable names (as described earlier) and determination and translation of unary operators and .P. might be programmed as follows:

```
      NORMAL MODE IS INTEGER
      BOOLEAN OPCHK.
      S(0) = $LTERM$
      HPRIME = 1
      THROUGH T, FOR H=12,1,H .G. 72
        WHENEVER S(H) .E. $ $, TRANSFER TO T
        HPRIME = HPRIME + 1
        WHENEVER S(H) .E. $.$ .AND. S(HPRIME -1)
          .E. 16 .AND.
    1   S(HPRIME -2) .E. $.$
          HPRIME = HPRIM - 2
          S(HPRIME) = $EXP$
        OR WHENEVER (S(H) .E. $+$ .OR. S(H) .E.
          $-$) .AND.
    1   (OPCHK.(S(HPRIME-1)) .OR. S(HPRIME-1) .E.
          $($ )
          WHENEVER S(H) .E. $+$
            HPRIME = HPRIME -1
            S(HPRIME) = $+U$
          OTHERWISE
            S(HPRIME) = $-U$
          END OF CONDITIONAL
        OTHERWISE
Q         THROUGH Q, FOR A = 1,1,NAME(A) .E. S(H)
            .OR. A .G. 26
          WHENEVER A .LE. 26, S(HPRIME) = NEW-
            NAM(A)
      VECTOR VALUES NAME(1) = $A$,$B$,$C$,$D$,$E$,
    1   $F$,$G$,$H$,$I$,$J$,$K$,$L$,$M$,$N$,$O$,$P$,$Q$,
    2   $R$,$S$,$T$,$U$,$V$,$W$,$X$,$Y$,$Z$
      VECTOR VALUES NEWNAM(1) = 1,2,3,4,5,6,7,8,9,10,
    1   11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26
      END OF CONDITIONAL
T     CONTINUE
      HPRIME = HPRIME + 1
      S(HPRIME) = $RTERM$
```

(d) The list of temporaries used (corresponding to the T's in the flowchart) can be implemented by extending the table of variables and starting I at 60 instead of 0.

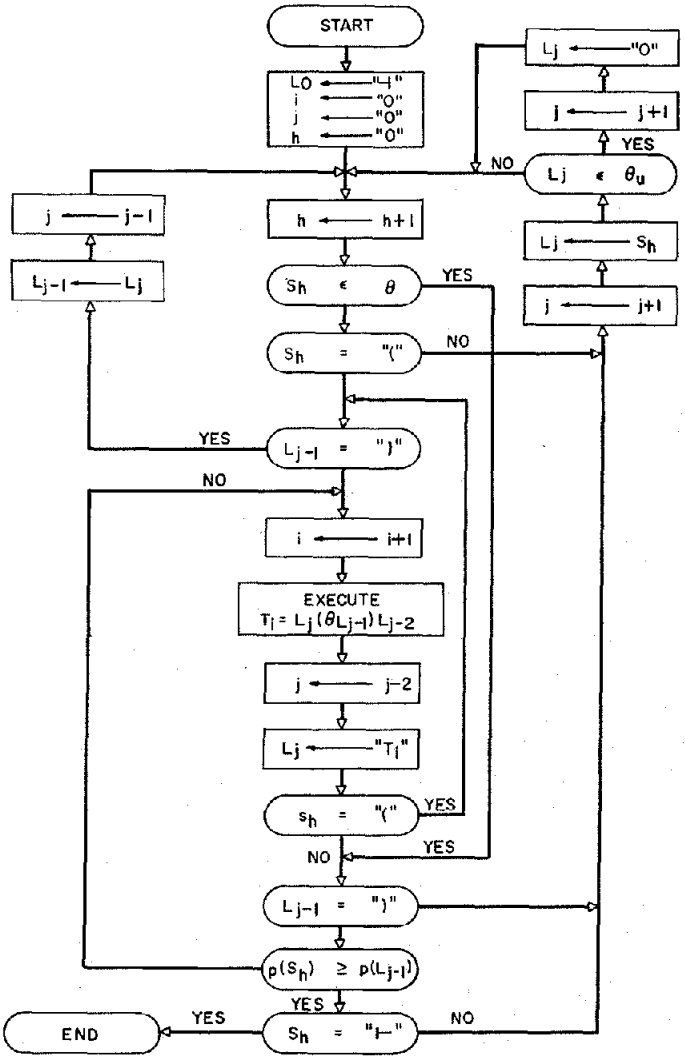
(e) Usually an input-output buffer is used to read and print. The interpretation of a P (print) statement, spaces having been compressed out as indicated above, is programmed simply as follows:

```
      THROUGH P, FOR A = 2,1,A .E. APRIME
P     BUFR(A-1) = VAR(X(A))
      PRINT FORMAT OUT, BUFR(1)...BUFR(A-1)
```

(f) Normally only columns 12 to 72 of a card are used. In looking for a given statement number for (stored in location TRANS) for a conditional transfer statement columns 1–10 are checked:

```
LOOK      READ FORMAT AHEAD,S(1)...S(72)
CHECK     THROUGH CHECK, FOR A = 1,1,A .G. 10
      1     .OR. S(A) .LE. $9$. AND.S(A) .GE. $0$
          WHENEVER A .LE. 10 .AND. S(A) .E. TRANS,
              TRANSFER TO INPUT
          TRANSFER TO LOOK
          VECTOR VALUES AHEAD = $72C1*$
```

We have not yet been able to evaluate the full benefits of this type of problem assignment beyond the enthusiastic response from our students and the fact that several of our junior staff have been able to deal more effectively with research problems in these areas. (Approximately 30 percent of the 75 students in the course completed this problem successfully in the allotted time.) However, the experience and understanding gained cannot but aid the students in their appreciation of the complexities of automatic programming as well as perhaps interest them in further work in this area.



θ IS THE SET OF OPERATORS; $\theta_u = \{A, +u, -u\} \subset \theta$