The language can be further simplified without introducing any complications.

First eliminate Boolean variables. We can declare two variables true and false in the outermost block and set true := 1; false := 0. We can then later write a := true; **if** b = true **then** etc.

Next eliminate integers. Integers are used as such only in indices and as operands for the operator ÷. These cases are syntactically identifiable and the compiler can insert the proper rounding-off operations.

Mr. Wirth touches upon the advisability of not declaring arrays since the final program will run slowly. To this one may say that:

1. It is high time we scientists began to tell the manufacturers what we want. In many cases ability to process lists efficiently is more valuable than multiple access storage modules and rapid carry propagation.

2. There are, broadly speaking, two types of problems. One type is exemplified by reactor calculations and simulation. They are few in number, but run for very long times. The other type is short and most of the time used for solving them is taken by programming and debugging. This type of problem is solved fastest by having the best possible programming language. Running efficiency here is of minor concern.

As there are few problems of the first type, one can afford to use a primitive language (e.g., FORTRAN), but the second category requires the best programming language (from the users point of view) that we can produce.

I am certain other scientists in this field will add their remarks to mine so that ALGOL 6x may ultimately appear.

REFERENCE:

1. WIRTH, N. A generalization of ALGOL. *Comm. ACM 6*, 9 (Sept. 1963), 547–554.

JAN V. GARWICK
Norwegian Defence Research Establishment
Kjeller, Norway

RECEIVED FEBRUARY, 1964

Characteristics of the FORTRAN CEP Language

O. G. MANCINO
Centro Studi Calcolatrici Elettroniche, C.N.R., Pisa, Italy

The FORTRAN CEP languages differs from FORTRAN II mainly because: (1) it extends the variety of the modes for real quantities; (2) it allows suitable mixtures, in an input/output list or in an expression, of quantities that occur under different modes; (3) it makes it possible to address a greater number of input/output equipment; and (4) it removes the restrictions on the complexity of the list of quantities to be transmitted between the magnetic core memory and the drum or the magnetic tape units.

The FORTRAN CEP language (defined in [1]) differs from FORTRAN II [2] for the following features.

A quantity may be either Boolean or in any real mode. Generally the mode of one or more quantities is specified by a *mode indicator*. This may appear either on the left or on the right of the opening character of a statement, and is composed respectively, either of a single letter or of a single letter and a colon. There are available without special provisions mode indicators that specify quantities as Boolean, or floating-point single precision, or floating point double precision, or fixed-point (noninteger) single precision, or fixed-point (noninteger) double precision. For real quantities other mode indicators may be used provided that they are introduced by a special statement called MODE, but they will be effective only if a suitable group of pseudoinstructions [3] is built to perform the operations required in the indicated modes. The integer quantities are, as in [2], specified by their forms or names; therefore mode indicators have no effect on them.

An expression may contain suitable mixtures of quantities that occur under different modes. Stating it in detail for arithmetic expressions, call *total expression* an expression not contained in another and *argument expression* an expression argument of a function. The value of an arithmetic expression, total or argument, may be obtained in any real mode, called *mode of the total or of the argument arithmetic expression.* Generally this mode must be specified with the suitable mode indicator immediately preceding the total or the argument arithmetic expression. If $\mathcal{A}$ is a total or an argument arithmetic expression immediately preceded by a mode indicator, those noninteger constants, variables and functions, that are contained in $\mathcal{A}$ but not in an argument expression inside $\mathcal{A}$, are thought of in the mode specified by that indicator. Mode indicators are not necessary before or inside a total arithmetic expression, when it is valid in [2] and it is intended as in [2]. Moreover, it is not necessary to indicate the mode of a total arithmetic expression $\mathcal{A}_1$ when the constants, variables and functions, contained in $\mathcal{A}_1$ but not in an argument expression inside $\mathcal{A}_1$, are intended as in [2]. Finally, it is not necessary to indicate the mode of an argument arithmetic expression $\mathcal{A}_2$ when the noninteger constants, variables and functions, contained in $\mathcal{A}_2$ but not in an argument expression inside $\mathcal{A}_2$, are thought of in the mode of the function of which $\mathcal{A}_2$ is an argument. Integer quantities may appear everywhere in any arithmetic expression.

Each actual argument of a library or of an arithmetic statement function may be either a Boolean expression or an arithmetic expression in any real mode.

Each actual argument of a FORTRAN function or of a CALL statement may be: a Boolean expression, an arithmetic expression in any real mode, an array name, the name of a library function without the terminal F, the name of a FORTRAN function, the name of a SUBROUTINE subprogram or a Hollerith field.

An arithmetic statement function may be either arithmetic in any real mode or Boolean.

The list of quantities to be transmitted may be, in any input/output statement with which it is associated, as described in [2, pp. 37–38]. In the list several mode indicators may appear, each of which may be introduced before any list element and also before any variable which is not part of a subscript or of indexing information. A mode indicator specifies the mode of the noninteger variables that follow it, that is, until another mode indicator occurs. For variables of the list which are not preceded by a mode indicator, the mode is as in [2]. Extraparentheses are not required.

Several readers and punches may be addressed, and reading of paper tapes may be done with or without the control of a FORMAT statement.

Output on a typewriter is allowed and is caused by a suitable statement called TYPE.

A FORMAT statement may include up to three levels of parentheses.

In a DIMENSION statement several mode indicators may appear and may be introduced before any array name. A mode indicator specifies the mode of noninteger arrays, referred to with names following it, until another mode indicator occurs. For matrices whose names are not preceded by a mode indicator, the mode is as in [2].

Acknowledgments. Thanks are expressed to Dr. A. Caracciolo di Forino and Dr. I. Galligani who contributed valuable ideas for the FORTRAN CEP language.

RECEIVED JANUARY, 1964; REVISED MARCH, 1964

REFERENCES

1. MANCINO, O. G. FORTRAN CEP language, a FORTRAN II version for the CEP. Centro Studi Calcolatrici Elettroniche, Pisa, Italy, 1963.
2. IBM Reference Manual 704 FORTRAN Programming System, Form C28-6106.
3. Manuale delle istruzioni CEP. Centro Studi Calcolatrici Elettroniche, Pisa, Italy, 1960.

Letters to the Editor

GEORGE E. FORSYTHE, Editor

Remark on Gladwin's Integer Conversion

Dear Editor:

Mr. Harmon T. Gladwin in his article "An Algorithm for Converting Integers from Base Alpha to Base Beta" [*Comm. ACM 7* (Apr. 1964), 241] indicated that the algorithm for converting from octal to decimal was not widely known. In fairness, I would point out that the algorithm was published some three years ago in *IRE Transactions on Electronic Computers* [1].

REFERENCE:

1. CROY, J. E. Rapid technique of manual or machine binary-to-decimal integer conversion using decimal radix arithmetic. *IRE EC-10*, 4 (Dec. 1961), 777.

ROBERT M. MCCLURE
Texas Instruments, Inc.
Dallas, Texas

Comments on the ALCOR Group Representation of ALGOL Symbols

Dear Editor:

These comments refer to the 5-track paper tape representation described in [1]. We find it hard to believe that a group of computer users could produce such a code. Even harder to believe is the fact that anyone would *want* to start from the I.T.C. code as a suitable code for any form of computer input. The arbitrary arrangement of symbols and lack of any form of parity checking militates against common sense.

In Great Britain most computers using paper tape employ codes specifically designed for computers and radically different from the I.T.C. code. We think it is significant that no group using such a code is a member of the ALCOR group. That we have not agreed among ourselves on a standard code is unfortunate; however, *any* of these codes is preferable, for computer work, to the I.T.C. code.

A representation of ALGOL, based on an existing computer code but otherwise in the general spirit of the ALCOR group, has recently been proposed by Gerard and Sambles [2]. We give in Table 1 a variant of the Ferranti 5-hole code (with which we are most familiar) designed specifically for ALGOL.[1] The printed version looks like ALGOL and achieves with 5-hole tape results that are similar to the 7- and 8-hole codes now being introduced. In the Ferranti code all the numerals and some other frequently used characters have an odd number of holes, hole 5 being used as a parity digit. The partiy checking may then be done by program or by hardware (for instance the Ferranti Pegasus allows "direct" and "checked" input).

The nonfeed characters, _ and |, can be used to synthesize other characters, such as **begin**, $\geq$, $\leq$ and $\neq$. The symbols $\times$ and $\uparrow$ could be conveniently represented by * and **, as in FORTRAN. String quotes could be represented by $\not<$ and $\not>$, re-

[1] A similar code is in use at the Royal Radar Establishment, Malvern.

Letters are continued on page 447