ship between the two permitting the translation of "statements" into "actions" on call by appropriate *control* processors. Thus declarative "definitions" (i.e. descriptive specifications) can be transformed into imperative "generators" (i.e. command specifications). The call for such a transformation shifts interpretation from descriptive to command, just as a jump instruction shifts the interpretation of a word from object to instruction (command syntax); it therefore belongs to the *control syntax*.

In general, the *control syntax* determines (selects) sequencing, scope, or context; it selects the processor. Because of control syntax we do not need an infinite hierarchy of syntax languages over syntax languages over . . . , etc. For example, a theorem about a class of programs for processing a fixed class of data is expressed in the descriptive syntax of the command syntax of an object language. These can all be in one language if the control syntax permits us to shift the context (i.e. interpretation) of an object expression to give it a syntactical or even a control interpretation. Natural languages can do this by quotation marks and words like "mean" and "define". Machine languages can do it via the control instructions. Both are examples of languages with "unstratified control".

We can now recognize quite a variety of meanings for the word "specify". The *specification* of a language or processor in descriptive syntax may be an explicit definition or an implicit definition by recursion; in command syntax it may be a program or a manual of instructions (the producers). The specification may also be structural or behavioral. A flow chart is a structural command specification of a processor; its logical design is a structural descriptive specification.

An important control processor for structural specifications is one which permits us to make big processors out of small ones by "linkage". Such a processor is called an *assembler*. The linkage of two processors may be at object level or at control level or at both. Some of the output data of one may be inputs to the other; this is *data linkage*, via *inputs* and *outputs*. Similarly, some of the "exits" from one may be made "entrances" to the other; this is *control linkage*. Programmed switches are often achieved by transforming data links into control links by setting a variable exit.

Corresponding to the descriptive "variable" is the command meaning "storage". An input which has not been linked is called a *free variable*. As soon as a substitution links it, the storage is called a *bound variable*.

We can now summarize the terms introduced in tabular form (Table I).

REFERENCES

1. Gorn, S. Common Programming Language Task. Final Reports AD59UR1 and AD60UR1, U. S. Army Signal Corps Contract No. DA-36-039-SC-75047, Part I, 1959, 1960.
2. Gorn, S. The treatment of ambiguity and paradox in mechanical languages. In "Symposium on Recursive Function Theory," April 7, 1961.

J. H. WEGSTEIN, Editor

ALGORITHM 68
AUGMENTATION

H. G. Rice
Computer Sciences Corp., Palos Verdes, Calif.


```
real procedure Aug(x,y); value x,y; integer x,y;
comment This algorithm makes use of the implicitly defined recursive properties of ALGOL procedures to compute the augment of x by y, using the basic technique of incrementation by unit step size;
begin Aug := if x = 0 then (if y > x then (Aug(y − 1, x) + 1) else y)
else Aug(x − 1, y + 1) end Aug
```

CERTIFICATION OF ALGORITHM 52
A SET OF TEST MATRICES (J. R. Herndon, *Comm. ACM*, Apr. 1961)

H. E. Gilbert
University of California at San Diego, La Jolla, Calif.

The statement

```
c := t × (t+1) × (t+t−5)/6;
```

was changed to

```
c := n × (n+1) × (n+n−5)/6;
```

to make the inverse have the form described in the algorithm. The algorithm was translated to Fortran and tested with a matrix eigenvalue program on the CDC 1604 computer at UCSD.

The eigenvalues for the 20 × 20 test matrix are:

```
 1.   1.000000
 2.   1.000000
 ⋮    ⋮
19.    .01636693
20.  −.02493833
```

Contributions to this department must be in the form stated in the Algorithms Department policy statement (*Communications*, February, 1960) except that ALGOL 60 notation should be used (see *Communications*, May, 1960). Contributions should be sent in duplicate to J. H. Wegstein, Computation Laboratory, National Bureau of Standards, Washington 25, D. C. Algorithms should be in the Publication form of ALGOL 60 and written in a style patterned after the most recent algorithms appearing in this department.

Although each algorithm has been tested by its contributor, no warranty, express or implied, is made by the contributor, the editor, or the Association for Computing Machinery as to the accuracy and functioning of the algorithm and related algorithm material and no responsibility is assumed by the contributor, the editor, or the Association for Computing Machinery in connection therewith.