Algorithms

H. J. WEGSTEIN, Editor

ALGORITHM 121
NORMDEV

DAVID SHAFER
University of Chicago, Chicago, Ill.

```
procedure NormDev(Random,A,x);
  procedure Random;  real A,x;
  comment 'NormDev' uses (1) a procedure 'Random(y)' as-
    sumed to produce a random number, 0 < y < 1, and (2) the
    constant A = sqrt(2/pi) × integral [0:1] exp(−x↑2/2)dx, to
    produce a positive normal deviate 'x';
  begin real y;
    Random(x);  if x > A then go to large;
    x := x/A;
    1:  Random(y);  if y < exp(−x↑2/2) then go to EndND;
        Random(x);  go to 1;
    large:  x := (x − A)/(1 − A);
    2:  x := sqrt(1 − 2 × log(x));
        Random(y);  if y < 1/x then go to EndND;
        Random(x);  go to 2;
  EndND:  end
```

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 Reference form of ALGOL 60 and written in a style patterned after the most recent algorithms appearing in this department. For the convenience of the printer, please underline words that are delimiters to appear in boldface type.

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.

The reproduction of algorithms appearing in this department is explicitly permitted without any charge. When reproduction is for publication purposes, reference must be made to the algorithm author and to the *Communications* issue bearing the algorithm.

ALGORITHM 122
TRIDIAGONAL MATRIX

GERARD F. DIETZEL
Burroughs Corp., Pasadena, Calif.

```
procedure TRIDIAG (n,A,U);
integer n;  array A,U;
comment  This procedure reduces a real symmetric matrix A of
  order n to tridiagonal form (UT)AU (UT = transpose of U) by
  a sequence of at most (n−1)(n−2)/2 binary orthogonal trans-
  formations. Also, the matrix U is calculated. [Cf. W. Givens,
  "Numerical computation of the characteristic values of a real
  symmetric matrix," Report ORNL1574 (1954), Oak Ridge Nat.
  Lab., Tenn., and D. E. Johansen, "A modified Givens method
  for the eigenvalue evaluation of large matrices," J. ACM 8, 3
  (1961)];
begin  real fact,c1,c2,loc1,loc2,temp;  integer i,j,j1,j2,j3,j4,n1;
  comment  Set array U = identity matrix of order n;
  for i := 1 step 1 until n do
  begin
    for j := i+1 step 1 until n do U[i,j] := U[j,i]  = 0;
    U[i,i] := 1.0
  end;
  comment  The reduction of the matrix A begins here. Only the
    upper triangular elements of A are used in the computation;
  n1 := n − 2;
  for i := 1 step 1 until n1 do
  begin
    j1 := i + 1;  j2 := i + 2;
    for j := j2 step 1 until n do
    begin
      if A[i,j] = 0 then go to lab;
      fact := 1 / sqrt(A[i,j1]↑2 + A[i,j]↑2);
      c1 := fact × A[i,j1];  c2 := fact × A[i,j];
      loc1 := A[j1,j1];  loc2 := A[j1,j];
      A[j1,j1] := c1↑2 × loc1 + 2.0 × c1 × c2 × loc2 + c2↑2 ×
        A[j,j];
      A[j1,j] := −c1 × c2 × loc1 + (c1↑2 − c2↑2) × loc2 + c1 ×
        c2 × A[j,j];
      A[j,j] := c2↑2 × loc1 − 2.0 × c1 × c2 × loc2 + c1↑2 ×
        A[j,j];
      j3 := j + 1;
      for k := j3 step 1 until n do
      begin
        temp := A(j1,k];
        A[j1,k] := c1 × temp + c2 × A[j,k];
        A[j,k] := −c2 × temp + c1 × A[j,k]
      end;
      j4 := j − 1;
      for k := j2 step 1 until j4 do
      begin
        temp := A[j1,k];
        A[j1,k] := c1 × temp + c2 × A[k,j];
        A[k,j] := −c2 × temp + c1 × A[k,j]
      end;
      A[i,j1] := c1 × A[i,j1] + c2 × A[i,j];
      A[i,j] := 0;
      for k := 1 step 1 until n do
```

```
      begin
        temp := U[k,j1];
        U[k,j1] := c1 × temp + c2 × U[k,j];
        U[k,j] := −c2 × temp + c1 × U[k,j]
      end;
lab:  end
  end;
  for i := 1 step 1 until n do
    for j := i+1 step 1 until n do
      A[j,i] := A[i,j]
end TRIDIAG
```

ALGORITHM 123
REAL ERROR FUNCTION, ERF(x)

Martin Crawford and Robert Techo
Georgia Institute of Technology, Atlanta, Ga.

```
real procedure Erf(x); real x;
comment Φ(x) = Erf(x) = (2/√π)∫_0^x e^(−u²) du can be computed
  by using the recursive relation for derivatives with Φ¹(x) =
  (2/√π)e^(−x²), where Φ⁽ⁿ⁾(x) = −2xΦ⁽ⁿ⁻¹⁾(x) − 2(n−2)Φ⁽ⁿ⁻²⁾(x),
  for n = 2, 3, ··· . The Taylor's series expansions of Φ(a_k) are
  taken about k+1 points on the interval 0 < a_k ≦ x and summed
  to get Φ(x);
begin   real A, U, V, W, Y, Z, T; integer N;
        Z := 0;  1:  if x ≠ 0 then
begin   if 0.5 < abs (x) then A := − sign (x) × 0.5
        else A := − x;
        U := V := 1.12837917 × exp(−x↑2);  Y := T := −V ×
          A;  N := 1;
2:      if abs(T) ≧ 10− 10 then
begin   N := N + 1;  W := −2 × x × V − 2 × U × (N−2);
          T := T × W × A/(V × N);
        U := V;  V := W;  Y := Y + T;  go to 2 end;
        Z := Z + Y;  x := x + A;  go to 1 end;
        Erf := Z end Erf
```

ALGORITHM 124
HANKEL FUNCTION

Luis J. Schaefer
Purdue University, West Lafayette, Ind.

```
procedure   HANKEL(N,X,H);   value N,X;   integer N;
  real X;  array H;
comment This procedure evaluates the complex valued hankel
  function of the first kind for real argument X and integral order
  N and assigns it to H. The individual Bessel- and Neuman-func-
  tion series are not evaluated separately. Both the real and
  imaginary parts are generated from the same terms;
begin   real K, P, R, A, S, T, D, L;  integer Q;
  A := R := 1;  H[1] := H[2] := S := 0;
  for Q := 1 step 1 until N do begin R := R × Q;  S := S +
    1/Q end;  D := R/N;
  R := 1/R;  K := X × X/4;  P := (X/2)↑N;  T := ln(K) +
    1.1544313298631;
  for Q := 0, Q+1 while Q≦N∨L≠H[2] do
  begin  L := H[2];  H[1] := H[1] + A×K×R;
    H[2] := H[2] + A×(R×K×(T−S) − (if Q<N then D/P
      else 0));
    A := A×K/Q;  R := −R/(Q+N);  S := S+1/Q + 1/(Q+N);
    if Q<N then D := D/(N−Q)
  end;  H[2] := H[2] × .31830989
end
```

Notes on Programming Languages

On the Nonexistence of a Phrase Structure Grammar for ALGOL 60

Robert W. Floyd
Computer Associates, Inc., Woburn, Massachusetts

ALGOL 60 is defined partly by formal mechanisms of phrase structure grammar, partly by informally stated restrictions. It is shown that no formal mechanisms of the type used are sufficient to define ALGOL 60.

Let a phrase structure grammar be defined as a set of definitions in the Backus notation used to define ALGOL 60 [1]. A phrase structure language, then, is a language defined by such a grammar. In such a language, because of the finite number of syntactic types, all sufficiently long programs (blocks) contain a substring which in turn contains a proper substring of the same syntactic type as itself [2, 3] Thus, the program P takes the form $QRSTU$, where RST and S are of the same syntactic type so that either may be substituted for the other, and S is a proper substring of RST. Now $QR^{(i)}ST^{(i)}U$ is a syntactically correct program for any non-negative integer i, where $R^{(i)}$ denotes i occurrences of the string R.

Example. An ALGOL 60 program might contain the primary $(c \times d)$, which in turn contains the primary c. It would be possible, therefore, to replace $(c \times d)$ by c at that point in the program, or to replace c by $(c \times d)$ obtaining $((c \times d) \times d)$, etc., without destroying the syntactic correctness of the program.

The goal of the present paper is to exhibit a set of ALGOL 60 programs of unbounded length, and show that none of them has the property described by the first paragraph. This implies that ALGOL 60 is not definable by a phrase structure grammar alone.

Consider the ALGOL 60 program

begin real $x^{(n)}$; $x^{(n)} := x^{(n)}$ **end**

where $x^{(n)}$ stands for n occurrences of the letter x. If ALGOL 60 is a phrase structure language, we may choose n sufficiently large to make applicable the result of the first paragraph. That is,

begin real $x^{(n)}$; $x^{(n)} := x^{(n)}$ **end**

takes the form $QRSTU$, and $QR^{(i)}ST^{(i)}U$ is a syntactically correct program P_i for all $i \geqq 0$. A block in ALGOL must contain at least one declarator, a semicolon, and the words **begin** and **end**; since $QSU = P_0$ is a block, R and T can contain only the characters x and $:=$. Since the declarator **real** occurs only once and there are no commas, only one identifier is declared in each of P_i, and only that identifier may be used in P_i. Two cases arise:

(1) Neither R nor T contains $:=$. Then $R = x^{(j)}$ and $T = x^{(k)}$ with j and k not both zero. One cannot, however, delete x's from P_1 in two places R and T and still have all identifiers properly declared in P_0; at least three deletions would have to be made.

(2) R or T contains $:=$. Since P_0 does not contain $:=$, it must take the form

begin real $x^{(j)}$; **end**