Algorithms

G. E. FORSYTHE, J. G. HERRIOT, Editors

ALGORITHM 243
LOGARITHM OF A COMPLEX NUMBER [B3]
REWRITE OF ALGORITHM 48 [*Comm. ACM 4* (Apr. 1961), 179; *5* (Jun. 1962), 347; *5* (Jul. 1962), 391; *7* (Aug. 1964), 485]

David S. Collens [Recd. 24 Jan. 1964 and 1 Jun. 1964]
Computer Laboratory, The University, Liverpool, 3, England

This procedure was tested using the DEUCE ALGOL Compiler and a small sample of the test data and results are given below.

```
procedure LOGC (a, b, c, d, FAIL); value a, b, FAIL; real
  a, b, c, d; label FAIL;
comment This procedure computes the number c + di which is
  equal to the principal value of the natural logarithm of a + bi,
  i.e. such that −π < d ≦ +π. A nonlocal label must be supplied
  as a parameter of the procedure, to be used as an exit when the
  real part of the result becomes −∞. Where required in the body
  of the procedure the numerical values for π, π/2, and the log-
  arithm of the square root of 8 are provided;
  if a = 0 ∧ b = 0 then go to FAIL
  else
  begin
    real e, f;
    e := 0.5 × a; f := 0.5 × b;
    if abs(e) < 0.5 ∧ abs(f) < 0.5 then
    begin
      c := abs(2×a) + abs(2×b);
      d := 8 × a/c × a + 8 × b/c × b;
      c := 0.5 × (ln(c)+ln(d)) −1.03972077084
    end
    else
    begin
      c := abs(0.5×e) + abs(0.5×f);
      d := 0.5 × e/c × e + 0.5 × f/c × f;
      c := 0.5 × (ln(c)+ln(d)) + 1.03972077084
    end;
    d := if a ≠ 0 ∧ abs(e) ≧ abs(f) then arctan(b/a) +
      (if sign(a)≠−1 then 0 else if sign(b)≠−1 then
        3.14159265359 else −3.14159265359) else − arctan(a/b)
        + 1.57079632679 × sign(b)
  end LOGC
```

TEST OF LOGC

a	b	c	d
−2	−2	+1.039721	−2.356194
−2	+1	+0.804719	+2.677945
−1	−1	+0.346573	−2.356194
−1	+0	+0.000000	+3.141593
+0	−2	+0.693147	−1.570796
+0	−1	+0.000000	−1.570796
+0	+1	+0.000000	+1.570796
+0	+2	+0.693147	+1.570796
+1	−1	+0.346573	−0.785398
+1	+0	+0.000000	+0.000000
+2	−2	+1.039721	−0.785398
+2	+1	+0.804719	+0.463647

ALGORITHM 244
FRESNEL INTEGRALS [S20]

Helmut Lotsch* (Recd. 27 May 64 and 11 Jun. 64)
W. W. Hansen Laboratories, Stanford U., Stanford, Calif.
AND
Malcolm Gray†
Computation Center, Stanford U., Stanford, Calif.
(* now at Northrup Space Laboratories, Hawthorne, Calif.)
(† now at The Boeing Company, Seattle, Wash.)

```
procedure FRESNEL (w, eps, C, S); value w, eps; real w,
  eps, C, S;
comment This procedure computes the Fresnel sine and cosine
  integrals C(w) = ∫₀^w cos [(π/2)t²] dt and S(w) = ∫₀^w sin [(π/2)t²]
  dt. It is a modification of Algorithm 213 (Comm. ACM, 6 (Oct.
  1963), 617) such that the accuracy, expressed by eps, is improved.
  eps can arbitrarily be chosen up to eps = 10 − 6 for a computer
  with sufficient word length as, for example, the Burroughs
  B5000 which has 11–12 significant digits. Referring to the formu-
  las of Algorithm 213: if | w | < √(26.20/π) the series expansions
  C(w) and S(w) are terminated when the absolute value of the
  relative change in two successive terms is ≦eps. If | w | ≧
  √(26.20/π) the series Q(x) and P(x) are terminated when the
  absolute value of the terms is ≦eps/2. However, this truncation
  point is not necessarily valid for the range √(26.20/π) ≦ | w |
  < √(28.50/π) when eps = 10 − 6, since the asymptotic series
  must be terminated before arriving at the minimum. In this
  range the ignored terms of the series expansions are < 3 10 − 6,
  and for larger arguments < 10 − 6. This accuracy may be im-
  proved if desired: the switch-over point from the regular to
  the asymptotic series expansions has to be displaced to larger
  arguments;
begin
  real x, x2, term; integer n;
  if abs(w) ≦ 10 − 12 then
    begin C := S := 0; go to aend end
  else x := w × w/0.636619772368;
  x2 := − x × x; if x ≧ 13.10 then go to asympt;
  begin
    real frs, frsi;
    frs := x/3; n := 5; term := x × x2/6;
    frsi := frs + term/7;
loops: if abs((frs−frsi)/frs) ≦ eps then go to send;
    frs := frsi; term := term × x2/(n×n−n);
    frsi := frs + term/(2×n+1);
    n := n + 2; go to loops;
send: S := frsi × w
  end;
  begin
    real frc, frci;
    frc := 1; n := 4; term := x2/2;
    frci := 1 + term/5;
loopc: if abs((frc−frci)/frc) ≦ eps then go to cend;
    frc := frci; term := term × x2/(n×n−n);
    frci := frc + term/(2×n+1);
    n := n + 2; go to loopc;
cend: C := frci × w
  end;
  go to aend;
asympt:
  begin
    real s1, s2, half, temp; integer i;
    x2 := 4 × x2; term := 3/x2; s1 := 1 + term; n := 8;
    for i := 1 step 1 until 6 do
    begin
      n := n + 4;
      term := term × (n−7) × (n−5)/x2;
      s1 := s1 + term;
```

```
        if abs(term) ≦ eps/2 then go to next
      end i;
next:  term := s2 := 0.5/x;  n := 4;
      for i := 1 step 1 until 6 do
      begin
        n := n + 4;
        term := term × (n−5) × (n−3)/x2;
        s2 := s2 + term;
        if abs(term) ≦ eps/2 then go to final
      end i;
final:  half := if w < 0 then −0.5 else 0.5;
      term := cos(x);  temp := sin(x);  x2 := 3.14159265359 × w;
      C := half + (temp×s1−term×s2)/x2;
      S := half − (term×s1+temp×s2)/x2
    end;
aend:
end FRESNEL
```

Revised Algorithms Policy • May, 1964

A contribution to the Algorithms department must be in the form of an algorithm, a certification, or a remark. Contributions should be sent in duplicate to the editor, typewritten double-spaced in capital and lower-case letters. Authors should carefully follow the style of this department, with especial attention to indentation and completeness of references. Material to appear in **boldface** type should be underlined in black. Blue underlining may be used to indicate *italic* type, but this is usually best left to the Editor.

An algorithm must be written in the ALGOL 60 Reference Language [*Comm. ACM 6* (Jan. 1963), 1-17], and normally consists of a commented procedure declaration. Each algorithm must be accompanied by a complete driver program in ALGOL 60 which generates test data, calls the procedure, and outputs test answers. Moreover, selected previously obtained test answers should be given in comments in either the driver program or the algorithm. The driver program may be published with the algorithm if it would be of major assistance to a user.

Input and output should be achieved by procedure statements, using one of the following five procedures (whose body is not specified in ALGOL): [see "Report on Input-Output Procedures for ALGOL 60," *Comm, ACM 7* (Oct. 1964), 628-629].

procedure *inreal* (*channel, destination*)· **value** *channel*; **integer** *channel*; **real** *destination*; **comment** the number read from channel *channel* is assigned to the variable *destination*; . . . ;

procedure *outreal* (*channel, source*); **value** *channel, source*; **integer** *channel*; **real** *source*; **comment** the value of expression *source* is output to channel *channel*; . . ;

procedure *ininteger* (*channel, destination*);
value *channel*; **integer** *channel, destination*; . . . ;

procedure *outinteger* (*channel, source*);
value *channel, source*; **integer** *channel, source*; . . . ;

procedure *outstring* (*channel, string*); **value** *channel*; **integer** *channel*; **string** *string*; . . . ;

If only one channel is used by the program, it should be designated by 1. Examples:

```
        outstring (1, 'x ='); outreal (1, x);
        for i := 1 step 1 until n do outreal (1, A[i]);
        ininteger (1, digit [17]);
```

It is intended that each published algorithm be a well-organized, clearly commented, syntactically correct, and a substantial contribution to the ALGOL literature. All contributions will be refereed both by human beings and by an ALGOL compiler. Authors should give great attention to the correctness of their programs, since referees cannot be expected to debug them. Because ALGOL compilers are often incomplete, authors are encouraged to indicate in comments whether their algorithms are written in a recognized subset of ALGOL 60 [see "Report on SUBSET ALGOL 60 (IFIP)," *Comm. ACM 7* (Oct, 1964), 626-627].

Certifications and remarks should add new information to that already published. Readers are especially encouraged to test and certify previously uncertified algorithms. Rewritten versions of previously published algorithms will be refereed as new contributions, and should not be imbedded in certifications or remarks.

Galley proofs will be sent to the authors; obviously rapid and careful proofreading is of paramount importance.

Although each algorithm has been tested by its author, no liability 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.—G.E.F.

CERTIFICATION OF ALGORITHM 199 [Z] CONVERSIONS BETWEEN CALENDAR DATE AND JULIAN DAY NUMBER [Robert G. Tartzen, *Comm. ACM 8* (Aug. 1963), 444].

DAVID K. OPPENHEIM (Recd. 10 Jul. 64 and 27 Jul. 64)
System Development Corp., Santa Monica, Calif.

Algorithm 199 was translated into JOVIAL J3 and tested on the Philco 2000. Input was generated with a random number generator that produced uniformly distributed dates between the years 1583 and 2583. The results were checked for 50 different dates in that range.

The procedures as written place unnecessary restrictions on some of the parameters. Expressions cannot always be used as inputs to the procedures. Also, the original input to *JDAY*, *JDATE* and *KDAY* will be modified during the operation of the respective procedures. It should also be noted that in many implementations of ALGOL the use of parameters called by name may be more expensive than those called by value. The call by name is a far more powerful tool than is necessary for most of the parameters of these procedures. For these reasons the following changes are suggested:

1. In **procedure** *JDAY*
   change: **integer** *d, m, y, j*;
   to: **value** *d, m, y*; **integer** *d, m, y, j*;
2. In **procedure** *JDATE*
   change: **integer** *j, d, m, y*; to: **value** *j*; **integer** *j, d, m, y*;
3. In **procedure** *KDAY*
   change: **integer** *d, m, ya, k*;
   to: **value** *d, m, ya*; **integer** *d, m, ya, k*;
4. In **procedure** *KDATE*
   change: **integer** *k, d, m, ya*;
   to: **value** *k*; **integer** *k, d, m, ya*;

CERTIFICATION OF ALGORITHM 213 [S20] FRESNEL INTEGRALS [M.D. Gray, *Comm. ACM 6* (Oct. 1963), 617]

Malcolm Gray (Recd. 29 May 1964 and, revised, 11 June 1964)
Computer Science Div., Stanford U., Stanford, Calif.
(now at The Boeing Company, Seattle, Wash.)

Necessary changes to the algorithm are:

(1) in the first line, replace
real *S, C*; with **real** *w, S, C*;

(2) in the formula for $P(x)$, replace $(-i)^{i+1}$ with $(-1)^{i+1}$

(3) the statement beginning
loopc: *if abs(frc−frci)*
should read
loopc: **if** *abs(frc−frci)*

(4) in the body, replace the line
next: **for** *i* := 1 **step** 1 **until** 5 **do begin** *n* := *n* + 4;
with the lines
next: *term* := *S*2 := 0.5/*x*; *n* := 4;
for *i* := 1 **step** 1 **until** 5 **do begin** *n* := *n* + 4;

The procedure (with the above changes) was executed on the Burroughs B5000 at Stanford University and gave results as indicated in the algorithm.

Communications from Helmut Lotsch of the W. W. Hansen Laboratories, Stanford University, and from Harold Butler of the Los Alamos Scientific Laboratory, Los Alamos, New Mexico, state that they found these same errors, and after the corrections were made, similar results were obtained. Mr. Lotsch's work was done on the B5000 and Dr. Butler's work was done on the IBM 7090.