ALGORITHM 48
LOGARITHM OF A COMPLEX NUMBER

JOHN R. HERNDON
Stanford Research Institute, Menlo Park, California

```
procedure LOGC(a, b, c, d); value a, b; real a, b, c, d;
comment This procedure computes the number, c+di, which
  is equal to loge(a+bi);
begin c := sqrt (a × a + b × b);
      d := arctan (b/a);
      c := log (c);
      if a < 0 then d := d+3.1415927
end LOGC;
```

CERTIFICATION OF ALGORITHM 48
LOGARITHM OF A COMPLEX NUMBER (J. R. Herndon, *Comm. ACM 4* (Apr., 1961), 179)

A. P. RELPH
Atomic Power Div., The English Electric Co., Whetstone, England

Algorithm 48 was translated using the DEUCE ALGOL compiler, after certain modifications had been incorporated, and then gave satisfactory results.

The original version will fail if $a = 0$ when the procedure for arctan is entered. It also assumes that $-\pi/2<d<3\pi/2$, whereas the principal value for logarithm of a complex number assumes $-\pi<d\leqq\pi$.

Incidentally, the ALGOL 60 identifier for natural logarithm is ln, not log.

The modified procedure is as follows:

```
procedure LOGC (a,b,c,d); value a,b; real a,b,c,d;
comment This procedure computes the number c + di which is
  equal to the principal value of loge (a + bi). If a = 0 then c is
  put equal to −₁₀47 which is used to represent "− infinity";
begin integer m,n
  m := sign (a); n := sign (b);
  if a = 0 then begin c := −₁₀47;
                      d := 1.5707963 × n;
                      go to k
  end;
  c := sq rt(a × a + b × b);
  c := ln (c);
  d := 1.5707963 × (1−m) × (1+n−n×n) + arctan (b/a);
k: end LOGC;
```

REMARK ON ALGORITHM 48
LOGARITHM OF A COMPLEX NUMBER [John R. Herndon, *Comm. ACM 4* (Apr. 1961)]

MARGARET L. JOHNSON AND WARD SANGREN
Computer Applications, Inc., San Diego, Calif.

Considerable care must be taken in using the arctan function. In Algorithm 48 two such difficulties are ignored. First it is necessary, because of a resulting division by zero, to deal separately with the case where the real part of the complex number is zero. Second, if the real part of the complex number is negative and the argument of the logarithm is to have a value between $-\pi$ and π then the action depends upon the sign of the imaginary part of the complex number. For clarity the following procedure exhibits in sequence the alternatives:

```
procedure LOGC (a, b, c, d); value a, b; real a, b, c, d;
comment This procedure computes the number c+di which is
  equal to loge (a+bi). It is assumed that the arctan has a value
  between −π/2 and π/2.
begin if a>0 then begin THETA := 0; go to SOL end;
      if a<0∧b≧0 then begin THETA := 3.1415927;
        go to SOL end;
      if a<0∧b<0 then begin THETA := −3.1415927;
        go to SOL end;
      if a=0∧b=0 then begin c := d := 0;
        go to RETURN end;
      if a=0∧b>0 then begin c := ln(b); d := 1.570963;
        go to RETURN end;
      if a=0∧b<0 then begin c := ln(abs(b));
        d := 1.570963; go to RETURN end;
SOL:  d := arctan (b/a) + THETA;
      c := sqrt(a×a+b×b);
      c := ln(c);
RETURN: end LOGC
```

REMARK ON REMARKS ON ALGORITHM 48 [B3]
LOGARITHM OF A COMPLEX NUMBER [John R. Herndon, *Comm. ACM 4* (Apr. 1961), 179; *5* (Jun. 62), 347; *5* (Jul. 62), 391]

DAVID S. COLLENS (Recd. 24 Jan. 1964 and 1 Jun. 1964)
Computer Laboratory, The University, Liverpool 3, England

This procedure was designed to compute $\log_e(a+bi)$, namely $c+di$, and although some very necessary precautions about its use have already been stated, some points seem to have escaped notice. In particular, A. P. Relph [*Comm. ACM*, June 1962, 347] remarked that if $a = 0$, then c becomes '−infinity', but this is only the case if $b = 0$ also. Margaret L. Johnson and Ward Sangren [*Comm. ACM*, July 1962, 391] conceded that $a = b = 0$ was a special case, but wrongly gave zero as the result. The only reasonable way of dealing with this case is to exit to some nonlocal label and to let the user decide whether to terminate his program or to assign particular values to c and d. The obvious values to use here are, for c, a negative number, larger than the largest which would be given by the procedure, and possibly zero for d. (In an implementation where 2^{-129} is the smallest representable nonzero number, the largest negative value of c possible is -89.416.) Finally, in the Johnson-Sangren version of the procedure, the last conditional statement should read

```
if a = 0 ∧ b < 0 then begin c := ln(abs(b));
    d := −1.570963; go to RETURN end;
```

the omission of the minus sign in the original being probably typographical in origin.