ALGORITHM 240
COORDINATES ON AN ELLIPSOID [Z]
Egon Dorrer (Recd. 8 Jan. 1964 and, rev., 19 May 1964)
Inst. f. Photogrammetrie, Techn. Hochschule, Munich, Germany

```
procedure GEODH 1 (L, B, AZ, S, EPS, lim, A, F, FAIL);
    value S, EPS, lim, A, F;   real L, B, AZ, S, EPS, A, F;
    integer lim;   label FAIL;
comment  GEODH 1 solves the problem of transferring of geo-
    graphical coordinates on an arbitrary ellipsoid of rotation. A is
    the radius of the equator, F is the flattening of the meridian
    ellipse. Before executing GEODH 1, L and B are longitude and
    latitude of a point P₁ on the ellipsoid. AZ is the azimuth at P₁,
    measured from north, of the geodesic to another point P₂, and
    S is the distance from P₁ to P₂, measured in the same unit as A.
    After execution of GEODH 1, L and B represent the longitude
    and latitude of P₂, and AZ is the final azimuth of the geodesic
    at P₂. Here L, B, AZ, and EPS are measured in radians. Arbi-
    trarily long distances S can be used, even more than the circum-
    ference. However, the geodesic must not cross the poles or come
    near to them. The problem has been solved by reiterated use of
    the Runge-Kutta method to solve the system of the three first-
    order differential equations of the geodesic on a rotation ellip-
    soid. EPS is the convergence parameter, e.g. a small number
    indicating the desired accuracy, normally 10⁻⁸ or 10⁻⁹. lim is the
    upper limit on iterations, it depends on EPS, and should not be
    chosen greater than 11 or 12. If lim is reached, computations
    stop, and the FAIL exit is used:
begin
    real EP2, Lo, Bo, AZo, LL, BL, AZL, So, SL, H, DL, DB, DAZ,
        KL, KB, KAZ, BQ, AZQ, W, H1, T, SINBQ;
    integer i, n, j, z;
    array D[1:4];   D[1] := D[4] := 1;   D[2] := D[3] := 2;
    EP2 := F × (2 − F);   Lo := L;   Bo := B;   AZo := AZ;
    n := 1;   z := 0;
ITERATION: if z = lim then go to FAIL;
    So := 0;   LL := Lo;   BL := Bo;   AZL := AZo;
    for i := 1 step 1 until n do
    begin
        SL := S × i/n;   H := (SL − So)/A;
        DL := DB := DAZ := KL := KB := KAZ := 0;
        for j := 1 step 1 until 4 do
        begin
            T := D[j];
            BQ := BL + DB/T;   AZQ := AZL + DAZ/T;   SINBQ :=
                sin(BQ);
            W := 1 − EP2 × SINBQ × SINBQ;   H1 := H × sqrt(W);
            DL := H1 × sin(AZQ)/cos(BQ);
            DB := H1 × W × cos(AZQ)/(1 − EP2);
            DAZ := DL × SINBQ;
            KL := KL + DL × T;   KB := KB + DB × T;   KAZ :=
                KAZ + DAZ × T
        end j;
        So := SL;   LL := LL + KL/6;   BL := BL + KB/6;
            AZL := AZL + KAZ/6
    end i;
    DL := LL − L;   DB := BL − B;   DAZ := AZL − AZ;
    L := LL;   B := BL;   AZ := AZL;
```

```
    if abs(DAZ) < EPS/sin(S/A) ∧ (abs(DL) < EPS/cos(B) ∨
        abs(DB) < EPS) then go to END;
    z := 1 + z;   n := 2 × n;
    go to ITERATION;
    END:
end GEODH 1
```

ALGORITHM 241
ARCTANGENT [B1]
K. W. Mills (Recd. 21 Nov. 1963)
Computing Centre, University of Adelaide, So. Australia

```
real procedure arg(x, y) exit: (error);   value x, y;   real x, y;
    label error;
comment  This procedure calculates the argument of a complex
    number x + iy, using a method which is substantially that of
    E. G. Kogbetliantz, IBM J. Research Develop., Jan. 1958, pp.
    43–53. The result lies in the interval [−π, π] and the exit error
    is provided for the case when x = y = 0. The procedure is es-
    sentially an ALGOL program for the calculation of the arctan-
    gent. arctan(y) is obtained most conveniently by calling the pro-
    cedure with x = 1;
begin
    array ct, csc2[2:5], tn[1:4];   integer k;   real w, v, pi, r, z;
    pi := 3.1415926536;   if x = 0 then
    begin
        if y = 0 then go to error;
L1: arg := pi/2 × sign(y);   go to exit
    end;
    w := y/x;   v := abs(w);
    if v > 1.34₁₀8 then go to L1;
    if v < 2.13₁₀−22 then r := w else
    begin
        ct[2] := tn[4] := 2.7474774195;
        ct[3] := tn[3] := 1.1917535926;
        ct[4] := tn[2] := .57735026919;
        ct[5] := tn[1] := .17632698071;
        csc2[2] := 8.548632169;
        csc2[3] := 2.420276626;
        csc2[4] := 1.333333333;
        csc2[5] := 1.031091204;
        if v < tn[1] then
        begin
            k := 1;   z := .16363636364 × v
        end
        else
        begin
            for k := 2 step 1 until 4 do if v < tn[k] then go to L3;
            k := 5;
L3:         z := .16363636364 × (ct[k]−csc2[k]/(v+ct[k]))
        end;
        r := (pi × (k−1)/9 + z/(z × z + .216649136 − .00270998425/
            (z × z + .0511194591))) × sign(w)
    end;
    arg := if x > 0 then r else
            if y = 0 then r + pi else
            r + pi × sign(y);
    exit:
end arg
```