```c
# include <stdlib.h>
# include <stdio.h>
# include <math.h>
# include <time.h>

# include "cordic.h"

# include <gmp.h>


/******************************************************************************/

void cordic ( double *x, double *y, double *z, int n )

/******************************************************************************/
/*
  CORDIC returns the sine and cosine using the CORDIC method.
  GMP (GNU Multiple Precision Library) is used

  Licensing:

    This code is distributed under the GNU LGPL license.

  Modified:

    2013.01.12

  Author:

    Based on MATLAB code in a Wikipedia article.

    Modifications by John Burkardt

    Further modified by Young W. Lim

  Parameters:

    Input:
      *x: x coord of an init vector
      *y: y coord of an init vector
      *z: angle (-90 <= angle <= +90)
       n: number of iteration
         A value of 10 is low.  Good accuracy is achieved
         with 20 or more iterations.

    Output:
      *xo: x coord of a final vector
      *yo: y coord of a final vector
      *zo: angle residue

  Local Parameters:

    Local, real ANGLES(60) = arctan ( (1/2)^(0:59) );

    Local, real KPROD(33), KPROD(j) = product ( 0 <= i <= j ) K(i),
    K(i) = 1 / sqrt ( 1 + (1/2)^(2i) ).
*/
{
# define ANGLES_LENGTH 60
# define KPROD_LENGTH 33

  const int precision = 512;

  mpf_t angle;
  mpf_t angles[ANGLES_LENGTH];
  mpf_t kprod[KPROD_LENGTH];
```

```
  int j;

  mpf_t factor;

  mpf_t pi;
  mpf_t poweroftwo;
  mpf_t sigma;
  mpf_t sign_factor;
  mpf_t theta;


  mpf_t xn, yn;
  mpf_t xt, yt;
  mpf_t tt;


  mpf_init2(angle, precision);
  for (j=0; j<ANGLES_LENGTH; ++j)
    mpf_init2( angles[j], precision);
  for (j=0; j<KPROD_LENGTH; ++j)
    mpf_init2( kprod[j], precision);


  mpf_init2(factor, precision);
  mpf_init2(pi, precision);
  mpf_init2(poweroftwo, precision);
  mpf_init2(sigma, precision);
  mpf_init2(sign_factor, precision);
  mpf_init2(theta, precision);
  mpf_init2(xn, precision);
  mpf_init2(yn, precision);
  mpf_init2(xt, precision);
  mpf_init2(yt, precision);
  mpf_init2(tt, precision);

  mpf_set_d(pi, 3.141592653589793);

  mpf_set_str(angles[ 0], "7.8539816339744830962E-01", 10);
  mpf_set_str(angles[ 1], "4.6364760900080611621E-01", 10);
  mpf_set_str(angles[ 2], "2.4497866312686415417E-01", 10);
  mpf_set_str(angles[ 3], "1.2435499454676143503E-01", 10);
  mpf_set_str(angles[ 4], "6.2418809995957348474E-02", 10);
  mpf_set_str(angles[ 5], "3.1239833430268276254E-02", 10);
  mpf_set_str(angles[ 6], "1.5623728620476830803E-02", 10);
  mpf_set_str(angles[ 7], "7.8123410601011112965E-03", 10);
  mpf_set_str(angles[ 8], "3.9062301319669718276E-03", 10);
  mpf_set_str(angles[ 9], "1.9531225164788186851E-03", 10);
  mpf_set_str(angles[10], "9.7656218955931943040E-04", 10);
  mpf_set_str(angles[11], "4.8828121119489827547E-04", 10);
  mpf_set_str(angles[12], "2.4414062014936176402E-04", 10);
  mpf_set_str(angles[13], "1.2207031189367020424E-04", 10);
  mpf_set_str(angles[14], "6.1035156174208775022E-05", 10);
  mpf_set_str(angles[15], "3.0517578115526096862E-05", 10);
  mpf_set_str(angles[16], "1.5258789061315762107E-05", 10);
  mpf_set_str(angles[17], "7.6293945311019702634E-06", 10);
  mpf_set_str(angles[18], "3.8146972656064962829E-06", 10);
  mpf_set_str(angles[19], "1.9073486328101870354E-06", 10);
  mpf_set_str(angles[20], "9.5367431640596087942E-07", 10);
  mpf_set_str(angles[21], "4.7683715820308885993E-07", 10);
  mpf_set_str(angles[22], "2.3841857910155798249E-07", 10);
  mpf_set_str(angles[23], "1.1920928955078068531E-07", 10);
  mpf_set_str(angles[24], "5.9604644775390554414E-08", 10);
  mpf_set_str(angles[25], "2.9802322387695303677E-08", 10);
  mpf_set_str(angles[26], "1.4901161193847655147E-08", 10);
  mpf_set_str(angles[27], "7.4505805969238279871E-09", 10);
  mpf_set_str(angles[28], "3.7252902984619140453E-09", 10);
  mpf_set_str(angles[29], "1.8626451492309570291E-09", 10);
```

```
mpf_set_str(angles[30], "9.3132257461547851536E-10", 10);
mpf_set_str(angles[31], "4.6566128730773925778E-10", 10);
mpf_set_str(angles[32], "2.3283064365386962890E-10", 10);
mpf_set_str(angles[33], "1.1641532182693481445E-10", 10);
mpf_set_str(angles[34], "5.8207660913467407226E-11", 10);
mpf_set_str(angles[35], "2.9103830456733703613E-11", 10);
mpf_set_str(angles[36], "1.4551915228366851807E-11", 10);
mpf_set_str(angles[37], "7.2759576141834259033E-12", 10);
mpf_set_str(angles[38], "3.6379788070917129517E-12", 10);
mpf_set_str(angles[39], "1.8189894035458564758E-12", 10);
mpf_set_str(angles[40], "9.0949470177292823792E-13", 10);
mpf_set_str(angles[41], "4.5474735088646411896E-13", 10);
mpf_set_str(angles[42], "2.2737367544323205948E-13", 10);
mpf_set_str(angles[43], "1.1368683772161602974E-13", 10);
mpf_set_str(angles[44], "5.6843418860808014870E-14", 10);
mpf_set_str(angles[45], "2.8421709430404007435E-14", 10);
mpf_set_str(angles[46], "1.4210854715202003717E-14", 10);
mpf_set_str(angles[47], "7.1054273576010018587E-15", 10);
mpf_set_str(angles[48], "3.5527136788005009294E-15", 10);
mpf_set_str(angles[49], "1.7763568394002504647E-15", 10);
mpf_set_str(angles[50], "8.8817841970012523234E-16", 10);
mpf_set_str(angles[51], "4.4408920985006261617E-16", 10);
mpf_set_str(angles[52], "2.2204460492503130808E-16", 10);
mpf_set_str(angles[53], "1.1102230246251565404E-16", 10);
mpf_set_str(angles[54], "5.5511151231257827021E-17", 10);
mpf_set_str(angles[55], "2.7755575615628913511E-17", 10);
mpf_set_str(angles[56], "1.3877787807814456755E-17", 10);
mpf_set_str(angles[57], "6.9388939039072283776E-18", 10);
mpf_set_str(angles[58], "3.4694469519536141888E-18", 10);
mpf_set_str(angles[59], "1.7347234759768070944E-18", 10);


mpf_set_str(kprod[ 0], "0.70710678118654752440", 10);
mpf_set_str(kprod[ 1], "0.63245553203367586640", 10);
mpf_set_str(kprod[ 2], "0.61357199107789634961", 10);
mpf_set_str(kprod[ 3], "0.60883391251775242102", 10);
mpf_set_str(kprod[ 4], "0.60764825625616820093", 10);
mpf_set_str(kprod[ 5], "0.60735177014129595905", 10);
mpf_set_str(kprod[ 6], "0.60727764409352599905", 10);
mpf_set_str(kprod[ 7], "0.60725911229889273006", 10);
mpf_set_str(kprod[ 8], "0.60725447933256232972", 10);
mpf_set_str(kprod[ 9], "0.60725332108987516334", 10);
mpf_set_str(kprod[10], "0.60725303152913433540", 10);
mpf_set_str(kprod[11], "0.60725295913894481363", 10);
mpf_set_str(kprod[12], "0.60725294104139716351", 10);
mpf_set_str(kprod[13], "0.60725293651701023413", 10);
mpf_set_str(kprod[14], "0.60725293538591350073", 10);
mpf_set_str(kprod[15], "0.60725293510313931731", 10);
mpf_set_str(kprod[16], "0.60725293503244577146", 10);
mpf_set_str(kprod[17], "0.60725293501477238499", 10);
mpf_set_str(kprod[18], "0.60725293501035403837", 10);
mpf_set_str(kprod[19], "0.60725293500924945172", 10);
mpf_set_str(kprod[20], "0.60725293500897330506", 10);
mpf_set_str(kprod[21], "0.60725293500890426839", 10);
mpf_set_str(kprod[22], "0.60725293500888700922", 10);
mpf_set_str(kprod[23], "0.60725293500888269443", 10);
mpf_set_str(kprod[24], "0.60725293500888161574", 10);
mpf_set_str(kprod[25], "0.60725293500888134606", 10);
mpf_set_str(kprod[26], "0.60725293500888127864", 10);
mpf_set_str(kprod[27], "0.60725293500888126179", 10);
mpf_set_str(kprod[28], "0.60725293500888125757", 10);
mpf_set_str(kprod[29], "0.60725293500888125652", 10);
mpf_set_str(kprod[30], "0.60725293500888125626", 10);
mpf_set_str(kprod[31], "0.60725293500888125619", 10);
mpf_set_str(kprod[32], "0.60725293500888125617", 10);
```

```
  // Initialize loop variables:
  mpf_set_d(theta, *z);

  mpf_set_d(xn, *x);
  mpf_set_d(yn, *y);

  mpf_set_d(poweroftwo, 1.0);
  mpf_set(angle, angles[0]);


  // Iterations
  for ( j = 1; j <= n; j++ )
  {
    if ( mpf_cmp_d(theta, 0.0) )
    {
      set_mpf_d(sigma, -1.0);
    }
    else
    {
      set_mpf_d(sigma, 1.0);
    }

    mpf_mul(factor, sigma, poweroftwo);

    mpf_mul(xt, factor, yn);
    mpf_sub(xt, xn, xt);

    mpf_mul(yt, factor, xn);
    mpf_add(yt, yt, yn);

    *x = mpf_get_d(xt);
    *y = mpf_get_d(yt);

    mpf_set(xn, xt);
    mpf_set(yn, yt);


    //  Update the remaining angle.
    mpf_mul(tt , sigma, angle);
    mpf_sub(theta, theta, tt);

    mpf_div_ui(poweroftwo, poweroftwo, 2);

    //  Update the angle from table, or eventually by just dividing by two.
    if ( ANGLES_LENGTH < j + 1 )
    {
      mpf_div_ui(angle, angle, 2);
    }
    else
    {
      mpf_set(angle, angles[j]);
    }

    *z = mpf_get_d(theta);
  }


  return;
# undef ANGLES_LENGTH
# undef KPROD_LENGTH
}
```