```c
#include "complexRobinStuart.h"

void main()
{
	{
	double GHA1 =  DMS2D(6.0, 45.0, 58.06)*15.0;
	double Dec1 = -DMS2D(7.0, 51.0, 30.3);
	double HO1  =  90.0-DMS2D(61.0, 57.0, 30.0);
	double GHA2 =  DMS2D(9.0, 49.0, 11.41)*15.0;
	double Dec2 = -DMS2D(7.0, 48.0, 37.3);
	double HO2  =  90.0-DMS2D(56.0, 34.0, 20.0);

	double B1, L1, B2, L2;

	char Bhms[50], Lhms[50];

	printf( "CoP1 = %lf\t%lf\t%lf\n", GHA1, Dec1, HO1 );
	printf( "CoP2 = %lf\t%lf\t%lf\n", GHA2, Dec2, HO2 );
	printf( "\n" );

	ComplexSolutionForIntersectionOfTwoCoP( GHA1, Dec1, HO1, GHA2, Dec2, HO2,
	                      &B1, &L1, &B2, &L2 );
	H2HMS( B1, Bhms );
	H2HMS( L1, Lhms );
	printf( "I1 = %lf, %lf = %s, %s\n", B1, L1, Bhms, Lhms );
	H2HMS( B2, Bhms );
	H2HMS( L2, Lhms );
	printf( "I2 = %lf, %lf = %s, %s\n", B2, L2, Bhms, Lhms );
	printf( "\n" );
	}
	{
	double GHA = DMS2D( 62, 16, 0 );
	double Dec = DMS2D( 38, 40, 13 );
	double B = DMS2D( 35, 30, 0 );
	double L = -DMS2D( 9, 30, 0 );
	double Hc, Z;
	char hms[50];

	ComplexSolutionForAltitudeAzimuth( GHA, Dec, B, L, &Hc, &Z );

	printf( "%lf\t%lf\t%lf\t%lf\n", GHA, Dec, B, L );
	H2HMS( Hc, hms );
	printf( "Hc = %lf = %s\n", Hc, hms );
	H2HMS( Z, hms );
	printf( "Z = %lf = %s\n", Z, hms );
	printf( "\n" );
	}
	{
	double dap =  DMS2D( 103, 26, 24);
	double HapM =  DMS2D( 35,37, 28 );
	double HapB =  DMS2D( 40, 17, 24 );
	double HoM =  DMS2D( 36, 26, 1 );
	double HoB =  DMS2D( 40, 16, 15 );
	double theta = 0;
	double LD = 0;
	char hms[50];

	ComplexSolutionForClearingLunarDistance( dap, HapM, HapB, HoM, HoB, &theta, &LD ) ;

	H2HMS( LD, hms );
	printf( "LD = %lf = %s\n", LD, hms );
	printf( "\n" );
	}
}
```

```c
/*
  FILE: complexRobinStuart.c

  FUENTE: Applications of Complex Analysis to Celestial Navigation
      Robin G. Stuart
      Valhalla, New York, USA

  STATUS: Finalizado

  TEST: OK

  PENDIENTE: Hc/Z y LD

  This file contains proprietary information of Andrés Ruiz Gonzalez
  Copying or reproduction without prior written approval is prohibited.
  Andrés Ruiz. San Sebastian - Donostia. Gipuzkoa
  Navigational Algorithms
  Copyright (c) 2009
*/

#include "../Matematicas/angulos.h"


double ModulusC( double a, double b )
{
  return( sqrt( a*a+b*b ) );
}


void ProductC( double r1, double i1, double r2, double i2, double *rp, double *ip )
{
  *rp = r1*r2-i1*i2;
  *ip = r1*i2+r2*i1;
}


void InversoC( double a, double b, double *invR, double *invI )
{
  double k = SQ( ModulusC( a, b ) );
  *invR =  a/k;
  *invI = -b/k;
}


void DivisionC( double r1, double i1, double r2, double i2, double *rp, double *ip )
{
  double invR, invI;
  InversoC( r2, i2, &invR, &invI );
  ProductC( r1, i1, invR, invI, rp, ip );
}


void Rec2Polar( double a, double b, double *r, double *fi )
{
  *r = sqrt( a*a+b*b );
  *fi = ATAN2( b,a );
}


void Polar2Rec( double r, double fi, double *a, double *b )
{
  *a = r*COS(fi);
  *b = r*SIN(fi);
}


void Euler( double fi, double *real, double *imag )
{
  *real = COS(fi);
  *imag = SIN(fi);
}



void Zp( double Dec, double GHA, double *real, double *imaginary )
{
  double r = TAN( 45.0+Dec/2.0 );
  Euler( -GHA, real, imaginary );

  *real *= r;
  *imaginary *= r ;
}


double Ro( double zd )
```

```c
{
  return( TAN(zd/2.0) );
}


void Zc( double zpR,double zpI, double ro, double *real, double *imaginary )
{
  double mzp = ModulusC( zpR, zpI );
  double k = (1.0+ro*ro)/(1.0-ro*ro*mzp*mzp);
  *real      = k*zpR;
  *imaginary = k*zpI ;
}


double Radius( double zpR,double zpI, double ro )
{
  double mzp = ModulusC( zpR, zpI );
  return( (1.0+mzp*mzp)/(1.0-ro*ro*mzp*mzp)*ro );
}


void ComplexSolutionForIntersectionOfTwoCoP( double GHA1, double Dec1, double Ho1,
                       double GHA2, double Dec2, double Ho2,
                       double* B1, double* L1,
                       double* B2, double* L2 )
{
  double zp1R, zp1I;
  double zp2R, zp2I;
  double zc1R, zc1I;
  double r1;
  double zc2R, zc2I;
  double r2;

  double d;
  double mu, nu;
  double z1R, z1I;
  double z2R, z2I;

  double ZD1 = 90.0-Ho1;
  double ZD2 = 90.0-Ho2;

  Zp( Dec1, GHA1, &zp1R, &zp1I );
  Zc( zp1R, zp1I, Ro( ZD1 ), &zc1R, &zc1I );
  r1 = Radius( zp1R, zp1I, Ro( ZD1 ) );

  Zp( Dec2, GHA2, &zp2R, &zp2I );
  Zc( zp2R, zp2I, Ro( ZD2 ), &zc2R, &zc2I );
  r2 =  Radius( zp2R, zp2I, Ro( ZD2 ) );

  d = ModulusC( zc1R-zc2R, zc1I-zc2I );
  mu = (SQ(r1)-SQ(r2))/(2.0*SQ(d));
  nu = 1.0/(2.0*SQ(d))*sqrt( 4.0*SQ(r1)*SQ(d)-SQ(SQ(d)+SQ(r1)-SQ(r2)) );

  { // Intersection of the 2 CoP on the complex plane
  double pR, pI;

  ProductC( mu, nu, zc2R-zc1R, zc2I-zc1I, &pR, &pI );
  z1R = 1.0/2.0*(zc1R+zc2R)+pR;
  z1I = 1.0/2.0*(zc1I+zc2I)+pI;

  ProductC( mu, -nu, zc2R-zc1R, zc2I-zc1I, &pR, &pI );
  z2R = 1.0/2.0*(zc1R+zc2R)+pR;
  z2I = 1.0/2.0*(zc1I+zc2I)+pI;
  }

  { // Intersection of the 2 CoP in (B,L)
  double r1, fi1;
  Rec2Polar( z1R, z1I, &r1, &fi1 );

  *B1 = 2.0*(ATAN(r1)-45.0);
  *L1 = fi1;
  }
  {
  double r2, fi2;
  Rec2Polar( z2R, z2I, &r2, &fi2 );

  *B2 = 2.0*(ATAN(r2)-45.0);
  *L2 = fi2;
  }


  printf( "Zp1 = (%lf, %lf i)\n", zp1R, zp1I );
  printf( "ro1 = %lf\n", Ro( ZD1 ) );
  printf( "Zc1 = (%lf, %lf i)\n", zc1R, zc1I );
  printf( "r1 = %lf\n", r1 );
  printf( "\n" );
```

```
  printf( "Zp2 = (%lf, %lf i)\n", zp2R, zp2I );
  printf( "ro2 = %lf\n", Ro( ZD2 ) );
  printf( "Zc2 = (%lf, %lf i)\n", zc2R, zc2I );
  printf( "r2 = %lf\n", r2 );
  printf( "\n" );

  printf( "d = %lf\n", d );
  printf( "mu = %lf\n", mu );
  printf( "nu = %lf\n", nu );

  printf( "Z1 = (%lf, %lf i)\n", z1R, z1I );
  printf( "Z2 = (%lf, %lf i)\n", z2R, z2I );
  printf( "\n" );
}


void ComplexSolutionForAltitudeAzimuth( double GHA, double Dec, double B, double L, double* Hc, double* Z )
{
  double aR, aI;
  double bR, bI, mb;
  double zR, zI;
  double wR, wI;

  double nR, nI, dR, dI;

  Euler( -L/2.0, &aR, &aI );

  mb = TAN( 45.0+B/2.0 );
  Euler( L/2.0+180.0, &bR, &bI );
  bR *= mb;
  bI *= mb;

  Zp( Dec, GHA, &zR, &zI );

  // T(z)
  ProductC( aR, aI, zR, zI, &nR, &nI );
  nR += bR;
  nI += bI;

  ProductC( -bR, bI, zR, zI, &dR, &dI );
  dR += aR;
  dI -= aI;

  DivisionC( nR, nI, dR, dI, &wR, &wI );

  {
  double r, fi;
  Rec2Polar( wR, wI, &r, &fi );

  *Hc = 2.0*(45.0-ATAN(r));
  *Z = fi;

  printf( "w = (%lf, %lf i) = %lf e %lf i\n", wR, wI, r, DegRad(fi) );
  }

  // Hemisferio S (1.2)
  //kk

  printf( "a = (%lf, %lf i)\n", aR, aI );
  printf( "b = (%lf, %lf i)\n", bR, bI );
  printf( "mb = %lf\n", mb );
  printf( "z = (%lf, %lf i)\n", zR, zI );
  printf( "w = (%lf, %lf i)\n", wR, wI );
  printf( "\n" );
}


double ComplexLunarDistance( double z1, double z2, double theta )
{
  double d;

  d = 2.0*ATAN( sqrt( (z1*z1+z2*z2-2.0*z1*z2*COS(theta))/(1.0+z1*z1*z2*z2+2.0*z1*z2*COS(theta)) ) );

  return( d );
}


double ComplexAzimuthDifference( double z1, double z2, double d )
{
  double theta;

  theta = ACOS( (SQ(z1)+SQ(z2)-(1.0+SQ(z1)*SQ(z2))*SQ(TAN(d/2.0)))/(2.0*z1*z2*(1.0+SQ(TAN(d/2.0)))) );

  return( theta );
}
```

```
double Z_H( double H )
{
  printf( "z = %lf\n", TAN( 45.0-H/2.0 ) );
  return( TAN( 45.0-H/2.0 ) );
}


void ComplexSolutionForClearingLunarDistance( double dap, double HapM, double HapB, double HoM, double HoB,
double* theta, double* LD )
{
  double cm = 0; // dip, R, PA, AG, ...
  double cb = 0;
  double z1 = Z_H( HapM+cm );
  double z2 = Z_H( HapB+cb );

  double z1o = Z_H( HoM );
  double z2o = Z_H( HoB );

  *theta = ComplexAzimuthDifference( z1, z2, dap );

  *LD = ComplexLunarDistance( z1o, z2o, *theta );
}
```