```
// This file is available in electronic form at http://www.psa.es/sdg/sunpos.htm

#include "sunpos.h"
#include <math.h>

void sunpos(cTime udtTime,cLocation udtLocation, cSunCoordinates *udtSunCoordinates)
{
        // Main variables
        double dElapsedJulianDays;
        double dDecimalHours;
        double dEclipticLongitude;
        double dEclipticObliquity;
        double dRightAscension;
        double dDeclination;

        // Auxiliary variables
        double dY;
        double dX;

        // Calculate difference in days between the current Julian Day
        // and JD 2451545.0, which is noon 1 January 2000 Universal Time
        {
                double dJulianDate;
                long int liAux1;
                long int liAux2;
                // Calculate time of the day in UT decimal hours
                dDecimalHours = udtTime.dHours + (udtTime.dMinutes
                        + udtTime.dSeconds / 60.0 ) / 60.0;
                // Calculate current Julian Day
                liAux1 =(udtTime.iMonth-14)/12;
                liAux2=(1461*(udtTime.iYear + 4800 + liAux1))/4 + (367*(udtTime.iMonth
                        - 2-12*liAux1))/12- (3*((udtTime.iYear + 4900
                + liAux1)/100))/4+udtTime.iDay-32075;
                dJulianDate=(double)(liAux2)-0.5+dDecimalHours/24.0;
                // Calculate difference between current Julian Day and JD 2451545.0
                dElapsedJulianDays = dJulianDate-2451545.0;
        }

        // Calculate ecliptic coordinates (ecliptic longitude and obliquity of the
        // ecliptic in radians but without limiting the angle to be less than 2*Pi
        // (i.e., the result may be greater than 2*Pi)
        {
                double dMeanLongitude;
                double dMeanAnomaly;
                double dOmega;
                dOmega=2.1429-0.0010394594*dElapsedJulianDays;
                dMeanLongitude = 4.8950630+ 0.017202791698*dElapsedJulianDays; // Radians
                dMeanAnomaly = 6.2400600+ 0.0172019699*dElapsedJulianDays;
                dEclipticLongitude = dMeanLongitude + 0.03341607*sin( dMeanAnomaly )
                        + 0.00034894*sin( 2*dMeanAnomaly )-0.0001134
                        -0.0000203*sin(dOmega);
                dEclipticObliquity = 0.4090928 - 6.2140e-9*dElapsedJulianDays
                        +0.0000396*cos(dOmega);
        }

        // Calculate celestial coordinates ( right ascension and declination ) in radians
        // but without limiting the angle to be less than 2*Pi (i.e., the result may be
        // greater than 2*Pi)
        {
                double dSin_EclipticLongitude;
                dSin_EclipticLongitude= sin( dEclipticLongitude );
                dY = cos( dEclipticObliquity ) * dSin_EclipticLongitude;
                dX = cos( dEclipticLongitude );
                dRightAscension = atan2( dY,dX );
                if( dRightAscension < 0.0 ) dRightAscension = dRightAscension + twopi;
                dDeclination = asin( sin( dEclipticObliquity )*dSin_EclipticLongitude );
        }

        // Calculate local coordinates ( azimuth and zenith angle ) in degrees
        {
                double dGreenwichMeanSiderealTime;
                double dLocalMeanSiderealTime;
                double dLatitudeInRadians;
                double dHourAngle;
                double dCos_Latitude;
                double dSin_Latitude;
                double dCos_HourAngle;
```

```
                double dParallax;
                dGreenwichMeanSiderealTime = 6.6974243242 +
                        0.0657098283*dElapsedJulianDays
                        + dDecimalHours;
                dLocalMeanSiderealTime = (dGreenwichMeanSiderealTime*15
                        + udtLocation.dLongitude)*rad;
                dHourAngle = dLocalMeanSiderealTime - dRightAscension;
                dLatitudeInRadians = udtLocation.dLatitude*rad;
                dCos_Latitude = cos( dLatitudeInRadians );
                dSin_Latitude = sin( dLatitudeInRadians );
                dCos_HourAngle= cos( dHourAngle );
                udtSunCoordinates->dZenithAngle = (acos( dCos_Latitude*dCos_HourAngle
                        *cos(dDeclination) + sin( dDeclination )*dSin_Latitude));
                dY = -sin( dHourAngle );
                dX = tan( dDeclination )*dCos_Latitude - dSin_Latitude*dCos_HourAngle;
                udtSunCoordinates->dAzimuth = atan2( dY, dX );
                if ( udtSunCoordinates->dAzimuth < 0.0 )
                        udtSunCoordinates->dAzimuth = udtSunCoordinates->dAzimuth + twopi;
                udtSunCoordinates->dAzimuth = udtSunCoordinates->dAzimuth/rad;
                // Parallax Correction
                dParallax=(dEarthMeanRadius/dAstronomicalUnit)
                        *sin(udtSunCoordinates->dZenithAngle);
                udtSunCoordinates->dZenithAngle=(udtSunCoordinates->dZenithAngle
                        + dParallax)/rad;
        }
}
```