```
# include <cstdlib>
# include <cmath>
# include <iostream>
# include <iomanip>
# include <fstream>
# include <vector>
# include <algorithm>

using namespace std;

# include "cordic.hpp"


double pi = 3.141592653589793;
double K = 1.646760258121;



//------------------------------------------------------------------------------

double compute_angle ( int idx, int nIter )

//------------------------------------------------------------------------------
//   Purpose:
//
//     Angle Array in Binary Tree Representation
//
//  Discussion:
//
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    2012.04.26
//
//  Author:
//
//    Young Won Lim
//
//  Parameters:
//    idx - index for leaf nodes of the binary tree
//    nIter - no of iteration (corresponds to the level of the tree)
//
//------------------------------------------------------------------------------
{
  double angle = 0.0;
  char   s[32];
  int    i, j;


  // i - bit position starting from lsb
  // j = 2^i
  // (idx & (1 << i)) - i-th bit of idx
  // if each bit is '1', add atan(1/2^i)
  // if each bit is '0', sub atan(1/2^i)
  // s[32] contains the binary representation of idx

  for (i=0; i<nIter; i++) {

    j = 1 << i;
    if (idx & (1 << i)) {
      angle += atan( 1. / j );
      s[nIter-i-1] = '1';
    } else {
      angle -= atan( 1. / j );
      s[nIter-i-1] = '0';
    }
```

```
    // cout << "i=" << i << " j=" << j << " 1/j=" << 1./j
    //      << " atan(1/j)=" << atan(1./j)*180/3.1416 << endl;

  }
  s[nIter] = '\0';

  // cout << nIter << " " << idx << " " << s
  //      << " ---> " << angle*180/3.1416 << endl;

  return angle;

}



//-----------------------------------------------------------------------------

void plot_unit_circle_angle (double *A, int nIter, int nAngle )

//-----------------------------------------------------------------------------
//  Purpose:
//
//     Plot angle vectors on the unit circle
//
//  Discussion:
//
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    2012.04.26
//
//  Author:
//
//    Young Won Lim
//
//  Parameters:
//
//-----------------------------------------------------------------------------
{
  int i;
  ofstream myout;


  // writing angle data on a unit circle
  myout.open("angle.dat");
  for (i=0; i<nAngle; i++) {
    myout << "0.0 0.0 " << cos(A[i]) << " " << sin(A[i]) << " " << endl;
  }
  myout.close();


  // writing gnuplot commands
  myout.open("command.gp");
  myout << "set size square" << endl;
  myout << "set xrange [-1:+1]" << endl;
  myout << "set yrange [-1:+1]" << endl;
  myout << "set object 1 circle at 0, 0 radius 1" << endl;
  myout << "plot 'angle.dat' using 1:2:3:4  ";
  myout << "with vectors head filled lt 2" << endl;
  myout << "pause mouse keypress" << endl;
  myout.close();


  system("gnuplot command.gp");
```

```
  return;

}


//------------------------------------------------------------------------------

void plot_residual_errors (double *A, int nIter, int nAngle)

//------------------------------------------------------------------------------
//  Purpose:
//
//   plot residual errors
//
//  Discussion:
//
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    2012.04.26
//
//  Author:
//
//    Young Won Lim
//
//  Parameters:
//
//------------------------------------------------------------------------------
{

  int i;
  double x, y, z;
  ofstream myout;

  // writing residue errors
  myout.open("angle.dat");

  for (i=0; i<nAngle; i++) {
    x = 1 / K;
    y = 0.0;
    z = A[i];

    cordic(&x, &y, &z, nIter);

    // cout << "A[" << i << "]= ";
    // cout << fixed << right << setw(10) << setprecision(7) << A[i];
    // cout << " z= " ;
    // cout << fixed << right << setw(10) << setprecision(7) << z << endl;

    myout << fixed << right << setw(10) << i;
    myout << fixed << right << setw(12) << setprecision(7) << A[i];
    myout << fixed << right << setw(12) << setprecision(7) << z << endl;

  }

  myout.close();


  // writing gnuplot commands
  myout.open("command.gp");
  myout << "set autoscale y" << endl;
  myout << "plot 'angle.dat' using 1:3 with linespoints " << endl;
  myout << "pause mouse keypress" << endl;
  myout.close();
```

```
  system("gnuplot command.gp");

  return;

}


//------------------------------------------------------------------------------

void plot_angle_tree (int nIter, int nAngle)

//------------------------------------------------------------------------------
//  Purpose:
//
//   plot angle tree
//
//  Discussion:
//
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    2012.04.26
//
//  Author:
//
//    Young Won Lim
//
//  Parameters:
//
//------------------------------------------------------------------------------
{

  int level = 6;
  int i, j, k;
  ofstream myout;
  double *A;


  A = (double *) malloc((1<<i) * sizeof (double));


  myout.open("angle.dat");

  for (i=0; i<level; ++i) {
    nIter = i;
    nAngle = 1 << nIter;

    for (j=0; j<nAngle; ++j) {
      A[j] = compute_angle(j, nIter);

      // cout << "A[" << j << "] = " << A[j] << endl;
      myout << A[j]*180/pi << " " <<  0.5*i << " 0.0 0.5" << endl;
    }
  }

  myout.close();


  // writing gnuplot commands
  myout.open("command.gp");
  myout << "plot 'angle.dat' using 1:2:3:4  ";
  myout << "with vectors head filled lt 2" << endl;
  myout << "pause mouse keypress" << endl;
  myout.close();
```

```
  system("gnuplot command.gp");

  //---------------------------------------
  // Accumulated Angle Tree
  //---------------------------------------

  myout.open("angle.dat");

  for (i=0; i<level; ++i) {

    for (k=0; k<=i; k++) {
      nIter = k;
      nAngle = 1 << nIter;

      for (j=0; j<nAngle; ++j) {
        A[j] = compute_angle(j, nIter);

        //cout << "A[" << j << "] = " << A[j] << endl;
        myout << A[j]*180/pi << " " <<  0.5*i << " 0.0 0.5" << endl;
      }
    }

  }

  myout.close();


  // writing gnuplot commands
  myout.open("command.gp");
  myout << "plot 'angle.dat' using 1:2:3:4  ";
  myout << "with vectors head filled lt 2" << endl;
  myout << "pause mouse keypress" << endl;
  myout.close();


  system("gnuplot command.gp");



  free (A);
  return;

}


//--------------------------------------------------------------------------

void calc_statistics (double *A, int nIter, int nAngle)

//--------------------------------------------------------------------------
//  Purpose:
//
//   Calculate leaf node angle statistics
//
//  Discussion:
//
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    2012.04.26
//
//  Author:
//
//    Young Won Lim
//
//  Parameters:o
```

```
//
//--------------------------------------------------------------------------
{

  vector <int> B, delta;
  vector <int> ::iterator first, last;
  double mean, std;


  for (int i=0; i < nAngle; ++i)
    B.push_back(A[i]);

  sort(B.begin(), B.end());

  for (int i=0; i < B.size()-1; ++i)
    delta.push_back(B[i+1]- B[i]);


  mean = 0.0;
  for (int i=0; i < delta.size(); ++i)
    mean += delta[i];
  mean /= delta.size();

  std = 0.0;
  for (int i=0; i < delta.size(); ++i)
    std += ((delta[i]-mean) * (delta[i]-mean));
  std /= delta.size();
  std = sqrt(std);

  cout << "delta mean = " << mean << endl;
  cout << "delta std  = " << std << endl;
  return;


  B.clear();
  delta.clear();

}




/*****


  for (i=0; i<20; i+=4) {
    for (j=0; j<4; ++j) {
      r = atan( 1. / (1 << (i+j)) ) / atan( 1. / (1 << i) ) * 100;
      cout << "index = " << i+j << " --> r = " << r << endl;
    }
  }


  return 0;

}

*********************/
```