```cpp
# include <cstdlib>
# include <cmath>
# include <iostream>
# include <iomanip>
# include <fstream>

using namespace std;

# include "cordic.hpp"



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

double compute_angle ( int idx, int nIter )

/******************************************************************************/
/*
  Purpose:

    Compute the angle of a leaf node of the binary tree
    whose height is determnined by nIter and
    whose leaf nodes are identified by nIter

  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;
}




int main (int argc, char * argv[]) {

  double pi = 3.141592653589793;
  double K = 1.646760258121;
  int    nIter = 3;
  int    nAngle = 1 << nIter;
  int    i;
  double *A;
  double x, y, z;


  if (argc > 1 ) {
    nIter = atoi(argv[1]);
    nAngle = 1 << nIter;
  }

  cout << "nIter = " << nIter << endl;


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

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


  ofstream myout;

  // ------------------------------------------------
  //   Plot Angles on a unit circle
  // ------------------------------------------------

  // 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");
```

```
  // ------------------------------------------------
  //   Plot residue errors at the leaf node angles
  // ------------------------------------------------

  // 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 0;

}
```