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

using namespace std;

# include "cordic.hpp"



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

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:

    2012.04.24

  Author:

    Young Won Lim

  Parameters:

*/
{
  double angle = 0.0;
  int i, j;
  char s[32];

  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';
    }
    // printf("i=%d j=%d 1/j=%f atan(1/j)=%f \n",
            // i, j, 1./j, atan(1./j)*180/3.1416);
  }
  s[nIter] = '\0';

  // printf("%d %d %s ---> %f \n", nIter, idx, s, angle*180/3.1416);



  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;
  double delta = 2.*pi / nAngle;


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

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


  A = (double *) malloc((1<<20) * sizeof (double));

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

    A[i] = compute_angle(i, nIter);
  }


  ofstream myout;
  myout.open("angle.dat");

  for (i=0; i<nAngle; i++) {
    myout << "0.0 0.0 " << cos(A[i]) << " " << sin(A[i]) << " "
          << A[i] << " 0.0 0.0 0.5" << endl;
  }

  myout.close();



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

    cordic(&x, &y, &z, nIter+4);

    cout << "A[" << i << "]= " << A[i] << " z= " << z << endl;

  }

  // printf("delta=%f \n", delta);

  return 0;

}
```