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

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';
    }

    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, j, k;
  double *A;
  double x, y, z;
  double r;

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

  ofstream myout;

  myout.open("angle.dat");

  for (i=0; i<6; ++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();


  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 << "mean_delta = " << mean << endl;
  cout << "std_delta = " << std << endl;
  myout.open("angle2.dat");

  for (i=0; i<6; ++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();


  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;

}
```