

Name: Jingshi Liu

Section: Image Processing

Project: **Final - Hough Transform**

Due Date: Dec 13th

Source Code:

```
#include <fstream>
#include <iostream>
using namespace std;

namespace Util{
    static int** getArray(int rows, int cols){
        int** array = new int*[rows];
        for(int i = 0; i < rows; i++){
            array[i] = new int[cols];
            for(int j = 0; j < cols; j++){
                array[i][j] = 0;
            }
        }
        return array;
    }

    static int findMax(int** array, int rows, int cols){
        int max = array[0][0];
        for(int i = 0; i < rows; i++){
            for(int j = 0; j < cols; j++){
                if(array[i][j] > max){
                    max = array[i][j];
                }
            }
        }
    }
}
```

```

        return max;
    }

    static int findMin(int** array, int rows, int cols){
        int min = array[0][0];
        for(int i = 0; i < rows; i++){
            for(int j = 0; j < cols; j++){
                if(array[i][j] < min){
                    min = array[i][j];
                }
            }
        }
        return min;
    }
}

class HoughTransform{
public:
    int numRows,
        numCols,
        minVal,
        maxVal,
        houghDist,
        houghAngle;

    int** imgAry;
    int** polarHoughAry;
    int angleInDegree,
        offset;
    double angleInRadian,
        PI;

    HoughTransform(ifstream& inFile){
        inFile >> numRows >> numCols >> minVal >> maxVal;
        int diagonal = sqrt(numRows * numRows + numCols * numCols);
    }
}

```

```

offset = diagonal;
imgAry = Util::getArray(numRows, numCols);
PI = 3.14159265;
houghDist = 2 * diagonal;
houghAngle = 180;
polarHoughAry = Util::getArray(houghDist, houghAngle);
loadImage(inFile);
}

void loadImage(ifstream& inFile){
    int cur = 0;
    for(int i = 0; i < numRows; i++){
        for(int j = 0; j < numCols; j++){
            inFile >> cur;
            imgAry[i][j] = cur;
        }
    }
}

void buildHoughSpace(){
    double angleInR;
    int dist;
    for(int i = 0; i < numRows; i++){
        for(int j = 0; j < numCols; j++){
            if(imgAry[i][j] == 0) continue;

            for(int angleInD = 0; angleInD < 180; angleInD++){
                angleInR = (angleInD / 180.00) * PI;
                dist = (int) polarDist(i, j, angleInR);
                polarHoughAry[dist][angleInD]++;
            }
        }
    }
}

```

```

double polarDist(int i, int j, double angleInR){
    return (i * cos(angleInR) + j * sin(angleInR) + offset);
}

void reformatPrettyPrint(ofstream& outFile){
    int houghMinVal = Util::findMin(polarHoughAry, houghDist, houghAngle),
        houghMaxVal = Util::findMax(polarHoughAry, houghDist, houghAngle);

    outFile << houghDist << " " << houghAngle << " " << houghMinVal << " " << houghMaxVal <<
'\n';

    string str;
    int curWidth,
        pixelWidth = to_string(houghMaxVal).length();

    for(int r = 0; r < houghDist; r++){
        for(int c = 0; c < houghAngle; c++){
            outFile << (polarHoughAry[r][c] == 0 ? "." : to_string(polarHoughAry[r][c]));
            str = to_string(polarHoughAry[r][c]);
            curWidth = str.length();
            while(curWidth < pixelWidth){
                outFile<<' ';
                curWidth++;
            }
            outFile<<' ';
        }
        outFile << '\n';
    }
}
};

```

```

int main(int argc, const char* argv[]){
    ifstream inFile(argv[1]);
    ofstream outFile(argv[2]);
    HoughTransform houghTransform(inFile);
}

```

}

Data 2 Output

90 180 0 2

90 180 0 21

