CSCI 381 (C++)

Section 31

Project 6 Chain Coding

Essam Yousry

Due Date of soft copy: 03/19/2018

Due Date of hard copy: 03/20/2018

Algorithm Steps for getChainCode

Step 1: scan imgAry from L to R & T to B

Step 2: if imgAry(iRow, jCol) > 0

output iRow, jCol, imgAry(iRow, jCol) to outFile1

output iRow, jCol, imgAry(iRow, jCol) to outFile2

startP 🡨 (iRow, jCol)

currentP 🡨 (iRow, jCol)

lastQ 🡨 4

Step 3: nextQ 🡨 mod(lastQ+1, 8)

Step 4: PchainDir 🡨 findNextP(currentP, nextQ, nextP)

Step 5: output PchainDir to outFile1

output PchainDir to outFile2

Step 6: lastQ 🡨 nextDirTable[PchainDir]

currentP <-- nextP

step 7: repeat step 3 to step 6 until currentP == startP

Main Program

Step 1: inFile 🡨 open input file, file1 and file2

Read image header and output to outFile1 and outFile2

Step 2: imgAry 🡨 dynamically allocated

Step3: loadImage()

Step 4: zeroFramed ()

Step 5: getChainCode()

Step 6: close all files

Source Code

#include <iostream>

#include <fstream>

#include <string>

#include <algorithm>

using namespace std;

class point{

private:

    int row, col;

public:

    point(int x = 0, int y = 0){

        row = x;

        col = y;

    }

    int getRow(){ return row; }

    int getCol(){ return col; }

    bool comparePoints(point y){

        if ((row == y.row) && (col == y.col)) return true;

        else return false;

    }

    void operator = (const point &y){

        row = y.row;

        col = y.col;

    }

};

void zeroFrame(int \*\*array, int row, int col){

    for (int i = 0; i < row + 2; i++){

        for (int j = 0; j < col + 2; j++){

            array[0][j] = 0;

            array[row + 1][j] = 0;

            array[i][0] = 0;

            array[i][col + 1] = 0;

        }

    }

}

void loadImage(int \*\*array, ifstream &myfile, int row, int col){

    for (int i = 1; i < row + 1; i++){

        for (int j = 1; j < col + 1; j++){

            myfile >> array[i][j];

        }

    }

}

void loadnextDirTable(int dirTable[]){

    dirTable[0] = 6;

    dirTable[1] = 0;

    dirTable[2] = 0;

    dirTable[3] = 2;

    dirTable[4] = 2;

    dirTable[5] = 4;

    dirTable[6] = 4;

    dirTable[7] = 6;

}

void loadNeighborsCoord(int \*\*imgAry, point currentP, point Neighbor []){

    int i = currentP.getRow();

    int j = currentP.getCol();

    Neighbor[0] = point(i, j + 1);

    Neighbor[1] = point(i - 1, j + 1);

    Neighbor[2] = point(i - 1, j);

    Neighbor[3] = point(i - 1, j - 1);

    Neighbor[4] = point(i, j - 1);

    Neighbor[5] = point(i + 1, j - 1);

    Neighbor[6] = point(i + 1, j);

    Neighbor[7] = point(i + 1, j + 1);

}

int getChainDir(int \*\*imgAry, point currentP, int nextQ, point Neighbor []){

    int x = 0;

    int i = nextQ - 1;

    int chainDir;

    while (x < 8){

        if (imgAry[Neighbor[i % 8].getRow()][Neighbor[i % 8].getCol()] > 0){

            chainDir = i % 8;

            break;

        }

        i++;

        x++;

    }

    return chainDir;

}

int findNextP(point currentP, int nextQ, point &nextP, int \*\*imgAry){

    point neighborCoord [8];

    loadNeighborsCoord (imgAry, currentP, neighborCoord);

    int chainDir = getChainDir(imgAry, currentP, nextQ, neighborCoord);

    nextP = neighborCoord[chainDir];

    return chainDir;

}

point findFirstOne(int \*\*imgAry, int row, int col){

    point p;

    int i = 1;

    int j = 1;

    while (j < col + 1){

        if ((imgAry[i][j] > 0) && (imgAry[i][j - 1] == 0)){

            p = point(i, j);

            break;

        }

        else j++;

        if (j == col + 1) i++;

    }

    return p;

}

void getChainCode(int \*\*imgAry, int row, int col, ofstream &myfile2, ofstream &myfile3){

    point startP;

    point currentP;

    point nextP;

    int lastQ;

    int nextQ;

    int nextDirTable [8];

    int nextDir;

    int PchainDir;

    loadnextDirTable(nextDirTable);

    point p = findFirstOne(imgAry, row, col);

    int iRow = p.getRow();

    int jCol = p.getCol();

    myfile2 << iRow;

    myfile2 << ' ';

    myfile2 << jCol;

    myfile2 << ' ';

    myfile2 << imgAry[iRow][jCol];

    myfile2 << ' ';

    myfile3 << iRow;

    myfile3 << ' ';

    myfile3 << jCol;

    myfile3 << ' ';

    myfile3 << imgAry[iRow][jCol];

    myfile3 << endl;

    startP = point(iRow, jCol);

    currentP = point(iRow, jCol);

    lastQ = 4;

    int count = 0;

    do{

        nextQ = (lastQ + 1) % 8;

        PchainDir = findNextP(currentP, nextQ, nextP, imgAry);

        myfile2 << PchainDir;

        myfile2 << ' ';

        myfile3 << PchainDir;

        myfile3 << ' ';

        count++;

        if (count == 15){

            myfile3 << endl;

            count = 0;

        }

        lastQ = nextDirTable[PchainDir];

        currentP = nextP;

    }

    while (currentP.comparePoints(startP) == false);

}

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

{

    int vars[4], row, col, min, max;

    ifstream myfile;

    myfile.open(argv[1]);

    for(int i = 0; i < 4; i++)

        myfile >> vars[i];

    cout << endl;

    row = vars[0];

    col = vars[1];

    min = vars[2];

    max = vars[3];

    string fileName = argv[1];

    string fileNameWithoutExtension = fileName.substr(0, fileName.rfind("."));

    ofstream myfile2;

    myfile2.open(string(fileNameWithoutExtension + "\_Image" + ".txt"));

    ofstream myfile3;

    myfile3.open(string(fileNameWithoutExtension + "\_Debug" + ".txt"));

    int \*\*imgAry = new int \*[row + 2];

    for (int i = 0; i < row + 2; i++)

        imgAry[i] = new int [col + 2];

    myfile2 << row;

    myfile2 << ' ';

    myfile2 << col;

    myfile2 << ' ';

    myfile2 << min;

    myfile2 << ' ';

    myfile2 << max;

    myfile2 << endl;

    myfile3 << row;

    myfile3 << ' ';

    myfile3 << col;

    myfile3 << ' ';

    myfile3 << min;

    myfile3 << ' ';

    myfile3 << max;

    myfile3 << endl;

    loadImage(imgAry, myfile, row, col);

    zeroFrame(imgAry, row, col);

    getChainCode(imgAry, row, col, myfile2, myfile3);

    for (int i = 0; i < row + 2; i++){

        delete imgAry[i];

    }

    delete[] imgAry;

    myfile.close();

    myfile2.close();

    myfile3.close();

    return 0;

}

Output















