**CSC 381 (IP\_31) (C++)**

**Project: Project 9.1 (Implementation of the four basic Morphology Operations)**

**Student: Swrajit Paul**

**Due date: soft copy: 5/8/2018**

**Hardcopy : 5/10/2018**

III. Algorithm steps in main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Remarks:

When you call each of the four morphological operations:

1. you need to reset the morphAry to zero,

2. scan imgAry begins at rowFrameSize and colFrameSize

3. when writing the result to the console, you must write which operation was used

4. output the result from morphAry to outfile, begins at rowFrameSize/2 and colFrameSize/2 of morphAry

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 0: open all files

( numRowsImg, numColsImg, minImg, maxImg )  get from input1

( numRowsStrctElem, numColsStrctElem, minStrctElem, maxStrctElem )  get from input2

( rowOrigin, colOrigin)  get from input2

step 1: computeFrameSize

step 2: - dynamically allocate imgAry with extra rows and extra columns

- loadImage // load input file to imgAry

- zeroFrameImg ()

- prettyPrint (imgAry) // pretty print imgAry to the \*console\* of "Input Image"

- dynamically allocate morphAry with extra rows and extra columns

step 3:

- dynamically allocate structElemAry

- loadstruct // load input2 file to structElem array

- prettyPrint (structElemAry)// pretty print to the \*console\* of "Structuring Element"

step 4: - initMorphAry( ) // initialize morphAry to zero

- call dilation // see your lecture note

- prettyPrint (morphAry) // pretty print to the \*console\* the result of dilation

- outputResult //write the delation result to Output1 (argv[3])

step 5: - initMorphAry( ) // initialize morphAry to zero

- call erosion // see your lecture note

- prettyPrint (morphAry) // pretty print to the \*console\* the result of erosion

- outputResult //write the delation result to Output2 (argv[4])

step 6: - initMorphAry( ) // initialize morphAry to zero

- call closing // By call those two morphological ops one after the other

- prettyPrint (morphAry) // pretty print to the \*console\* the result of closing

- outputResult //write the closing result to Output3 (argv[5])

step 7: - initMorphAry( ) // initialize morphAry to zero

- call opening // By call those two morphological ops one after the other

- prettyPrint (morphAry) // pretty print to the \*console\* with "Opening Result"

- outputResult //write the opening result to Output4 (argv[6])

step 8: close all files

**Source Code**

#include <iostream>

#include <fstream>

using namespace std;

int numRowsImg;

int numColsImg;

int minImg;

int maxImg;

int numRowsStructElem;

int numColsStructElem;

int minStrctElem;

int maxStrctElem;

int rowOrigin;

int colOrigin;

int rowFrameSize;

int colFrameSize;

int\*\* imgAry;

int\*\* morphAry;

int\*\* structElemAry;

ifstream inFileone;

ifstream inFiletwo;

ofstream outFileone;

ofstream outFiletwo;

ofstream outFilethree;

ofstream outFilefour;

int\*\* tempAry;

int temp;

void computeFrameSize(){

rowFrameSize = numRowsStructElem;

colFrameSize = numColsStructElem;

}

void loadImage(){

for(int i =rowFrameSize/2; i < (numRowsImg+rowFrameSize/2); i++){

for(int j =colFrameSize/2; j < (numColsImg+colFrameSize/2); j++){

inFileone >> imgAry[i][j];

}

}

}

void loadstruct(){

for(int i =0; i < numRowsStructElem; i++){

for(int j =0; j < numColsStructElem; j++){

inFiletwo >> structElemAry[i][j];

}

}

}

void zeroFrameImg(){

// set it to 0

for(int i =0; i < (numRowsImg + rowFrameSize); i++){

for(int j =0; j < (numColsImg + colFrameSize); j++){

imgAry[i][j] = 0;

}

}

}

void initMorphAry( ){

for(int i =0; i < (numRowsImg + rowFrameSize); i++){

for(int j =0; j < (numColsImg + colFrameSize); j++){

morphAry[i][j] = 0;

}

}

}

void dilation (int i, int j){

for(int r = 0, m = i-1; r < 3 , m < i+2; r++, m++){

for(int c = 0, n = j-1; c < 3, n < j+2; c++, n++){

if(structElemAry[r][c] == 1) {

morphAry[m][n] = 1;

}

}

}

}

void erosion (int i, int j){

for(int r = 0, m = i-1; r < 3 , m < i+2; r++, m++){

for(int c = 0, n = j-1; c < 3, n < j+2; c++, n++){

if(structElemAry[r][c] == 1){

if(imgAry[m][n] != 1){

goto done;

}

else {

continue;

}

}

}

}

morphAry[i][j] = 1;

done: ;

}

void closing (int i, int j){

if(temp ==1){

for(int r = 0, m = i-1; r < 3 , m < i+2; r++, m++){

for(int c = 0, n = j-1; c < 3, n < j+2; c++, n++){

if(structElemAry[r][c] == 1) {

tempAry[m][n] = 1;

}

}

}

}

if(temp ==2){

for(int r = 0, m = i-1; r < 3 , m < i+2; r++, m++){

for(int c = 0, n = j-1; c < 3, n < j+2; c++, n++){

if(structElemAry[r][c] == 1){

if(tempAry[m][n] != 1){

goto done;

}

else {

continue;

}

}

}

}

morphAry[i][j] = 1;

}

done: ;

}

void opening (int i,int j){

if(temp ==1){

for(int r = 0, m = i-1; r < 3 , m < i+2; r++, m++){

for(int c = 0, n = j-1; c < 3, n < j+2; c++, n++){

if(structElemAry[r][c] == 1){

if(imgAry[m][n] != 1){

goto done;

}

else {

continue;

}

}

}

}

tempAry[i][j] = 1;

}

if(temp ==2){

for(int r = 0, m = i-1; r < 3 , m < i+2; r++, m++){

for(int c = 0, n = j-1; c < 3, n < j+2; c++, n++){

if(structElemAry[r][c] == 1) {

morphAry[m][n] = 1;

}

}

}

}

done: ;

}

void prettyPrint (int n){

switch (n){

case 1:

for(int i =0; i < (numRowsImg + rowFrameSize); i++){

for(int j =0; j < (numColsImg + colFrameSize); j++){

cout << imgAry[i][j];

}

cout << endl;

}

cout << endl;

break;

case 2:

for(int i =0; i < (numRowsImg + rowFrameSize); i++){

for(int j =0; j < (numColsImg + colFrameSize); j++){

cout << morphAry[i][j];

}

cout << endl;

}

cout << endl;

break;

case 3:

for(int i =0; i < numRowsStructElem; i++){

for(int j =0; j < numColsStructElem; j++){

cout << structElemAry[i][j];

}

cout << endl;

}

cout << endl;

break;

default:

cout << "incorrect";

break;

}

}

void outputResult (int n){

switch (n){

case 3:

outFileone << numRowsImg << " " << numColsImg << " " << minImg << " " << maxImg << endl;

for(int i =0; i < (numRowsImg + rowFrameSize); i++){

for(int j =0; j < (numColsImg + colFrameSize); j++){

outFileone << morphAry[i][j];

}

outFileone << endl;

}

outFileone << endl;

break;

case 4:

outFiletwo << numRowsImg << " " << numColsImg << " " << minImg << " " << maxImg << endl;

for(int i =0; i < (numRowsImg + rowFrameSize); i++){

for(int j =0; j < (numColsImg + colFrameSize); j++){

outFiletwo << morphAry[i][j];

}

outFiletwo << endl;

}

outFiletwo << endl;

break;

case 5:

outFilethree << numRowsImg << " " << numColsImg << " " << minImg << " " << maxImg << endl;

for(int i =0; i < (numRowsImg + rowFrameSize); i++){

for(int j =0; j < (numColsImg + colFrameSize); j++){

outFilethree << morphAry[i][j];

}

outFilethree << endl;

}

outFilethree << endl;

break;

case 6:

outFilefour << numRowsImg << " " << numColsImg << " " << minImg << " " << maxImg << endl;

for(int i =0; i < (numRowsImg + rowFrameSize); i++){

for(int j =0; j < (numColsImg + colFrameSize); j++){

outFilefour << morphAry[i][j];

}

outFilefour << endl;

}

outFilefour << endl;

break;

default:

cout << "incorrect";

break;

}

}

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

inFileone.open(argv[1]);

inFiletwo.open(argv[2]);

outFileone.open(argv[3]);

outFiletwo.open(argv[4]);

outFilethree.open(argv[5]);

outFilefour.open(argv[6]);

inFileone >> numRowsImg >> numColsImg >> minImg >>maxImg;

inFiletwo >> numRowsStructElem >> numColsStructElem >> minStrctElem >> maxStrctElem >> rowOrigin >> colOrigin;

computeFrameSize();

// initailize the array

imgAry = new int\* [numRowsImg + rowFrameSize];

for(int i =0; i < (numRowsImg + colFrameSize); i++){

imgAry[i] = new int[numColsImg + colFrameSize];

}

zeroFrameImg ();

loadImage();

cout << "Input Image" << endl;

prettyPrint (1);

morphAry = new int\* [numRowsImg + rowFrameSize];

for(int i =0; i < (numRowsImg + rowFrameSize); i++){

morphAry[i] = new int[numColsImg + colFrameSize];

}

structElemAry = new int\* [numRowsStructElem];

for(int i =0; i < (numRowsStructElem); i++){

structElemAry[i] = new int[numColsStructElem];

}

loadstruct();

cout << "Structuring Element" << endl;

prettyPrint (3);

initMorphAry();

for(int i =rowFrameSize/2; i < (numRowsImg+rowFrameSize/2); i++){

for(int j =colFrameSize/2; j < (numColsImg+colFrameSize/2); j++){

if(imgAry[i][j] == 0){

continue;

}

dilation(i,j);

}

}

cout << "The result of dilation" << endl;

prettyPrint(2);

outputResult(3);

initMorphAry();

for(int i =rowFrameSize/2; i < (numRowsImg+rowFrameSize/2); i++){

for(int j =colFrameSize/2; j < (numColsImg+colFrameSize/2); j++){

if(imgAry[i][j] == 0){

continue;

}

erosion(i,j);

}

}

cout << "The result of erosion" << endl;

prettyPrint(2);

outputResult(4);

initMorphAry();

tempAry = new int\* [numRowsImg + rowFrameSize];

for(int i =0; i < (numRowsImg + colFrameSize); i++){

tempAry[i] = new int[numColsImg + colFrameSize];

}

for(int i =0; i < (numRowsImg + rowFrameSize); i++){

for(int j =0; j < (numColsImg + colFrameSize); j++){

tempAry[i][j] = 0;

}

}

for(int h = 1; h < 3; h++){

temp = h;

for(int i =rowFrameSize/2; i < (numRowsImg+rowFrameSize/2); i++){

for(int j =colFrameSize/2; j < (numColsImg+colFrameSize/2); j++){

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

continue;

}

if(tempAry[i][j] == 0 && h ==2){

continue;

}

closing(i,j);

}

}

}

cout << "The result of closing" << endl;

prettyPrint(2);

outputResult(5);

initMorphAry();

for(int i =0; i < (numRowsImg + rowFrameSize); i++){

for(int j =0; j < (numColsImg + colFrameSize); j++){

tempAry[i][j] = 0;

}

}

for(int h = 1; h < 3; h++){

temp = h;

for(int i =rowFrameSize/2; i < (numRowsImg+rowFrameSize/2); i++){

for(int j =colFrameSize/2; j < (numColsImg+colFrameSize/2); j++){

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

continue;

}

if(tempAry[i][j] == 0 && h ==2){

continue;

}

opening(i,j);

}

}

}

cout << "The result of opening" << endl;

prettyPrint(2);

outputResult(6);

inFileone.close();

inFiletwo.close();

outFileone.close();

outFiletwo.close();

outFilethree.close();

outFilefour.close();

}

**Outputs for data A**

**Output one**

Dilation

42 31 0 1

0100000000000000000000000000000100

1110000000000000100000000000001110

1111000000000001110000000000011100

0110011000000011111000000000001000

0000111100000111111100000011000000

0001111100001111111110000111100000

0011111000011111111111000111000000

0011111100111111111111100011100000

0111111001111111111111110011110000

0011110101111111111111110011100000

0001111111111111111111110001000000

0001111111111111111111111111100000

0011111111111111111111111111110000

0001111111111111111111111111100000

0000111001111111111111110000000000

0001110000111111111111111000000000

0000100001111111111111011100000000

0000000011100111111110001110000000

0000000111000011111111000111000000

0000001110000001110011100010000000

0000100100000001110001000000000000

0001110000000001110000000000000000

0011111000000001110000000010000000

0011111000000011111000000111000000

0001110000000011111000001110000000

0000100100000111111100011100000000

0000001110001111111110111111000000

0000011111011111111111111111100000

0001111011111111111111101111100000

0011110001111111111111111111110000

0001100001111111111111111011110000

0000000001111111111111110001100000

0001100001111111111111110000000000

0011110001111111111111110000000000

0011110001111111111111111100000000

0001100001111111111111111110000000

0000000011111111111111111111000000

0000000111011111111111000011100000

0001101110000111111100011011100000

0011111100000011111000111101000000

0111111000000011111000011110000000

1111010000000001110000001100000000

0110000000000000100000000000000000

0000000000000000000000000000000000

0000000000000000000000000000000000

Erosion

42 31 0 1

0000000000000000000000000000000000

0000000000000000000000000000000000

0000000000000000000000000000000000

0000000000000000100000000000000000

0000000000000001110000000000000000

0000000000000011001000000000000000

0000000000000110000100000000000000

0000000000001110001110000000000000

0000000000010110011101000000000000

0000000000000000001000000000000000

0000000000010100000001000000000000

0000000000010110011011000000000000

0000000000100011100111100000000000

0000000000010101000011000000000000

0000000000011000100011000000000000

0000000000000100010100000000000000

0000000000000000011000000000000000

0000000000000000110000000000000000

0000000000000000100000000000000000

0000000000000000000000000000000000

0000000000000000000000000000000000

0000000000000000000000000000000000

0000100000000000000000000000000000

0000100000000000100000000000000000

0000000000000000100000000000000000

0000000000000001110000000000000000

0000000000000011001000000000000000

0000000000000110000100000000000000

0000000000001001000000000000000000

0000000000010000100000000000000000

0000000000010000110000000000000000

0000000000010001100001000000000000

0000000000000011000011000000000000

0000000000010111000001000000000000

0000000000011100100000000000000000

0000000000011000010001000000000000

0000000000000100111000000000000000

0000000000000011111000000000000000

0000000000000001110000000000000000

0000000000000000100000000000000000

0000000000000000100000000000000000

0000000000000000000000000000000000

0000000000000000000000000000000000

0000000000000000000000000000000000

0000000000000000000000000000000000

Closing

42 31 0 1

0000000000000000000000000000000000

0100000000000000000000000000000100

0110000000000000100000000000001000

0000000000000001110000000000000000

0000011000000011111000000000000000

0000111000000111111100000011000000

0001110000001111111110000010000000

0001111000011111111111000001000000

0011110000111111111111100001100000

0001100000111111111111100001000000

0000110101111111111111100000000000

0000111111111111111111110001000000

0001111111111111111111111111100000

0000111001111111111111110000000000

0000010000111111111111100000000000

0000100000011111111111010000000000

0000000000100111111110001000000000

0000000001000011111100000100000000

0000000010000001110010000010000000

0000000100000000100001000000000000

0000000000000000100000000000000000

0000100000000000100000000000000000

0001110000000000100000000000000000

0001110000000001110000000010000000

0000100000000001110000000100000000

0000000000000011111000001000000000

0000000100000111111100011100000000

0000001010001111111110101111000000

0000010001011111111111000111000000

0001100000111111111111101011100000

0000000000111111111111110001100000

0000000000111111111111100000000000

0000000000111111111111100000000000

0001100000111111111111100000000000

0001100000111111111111110000000000

0000000000111111111111111100000000

0000000001011111111111000010000000

0000000010000111111100000001000000

0000000100000011111000000001000000

0001101000000001110000011000000000

0011010000000001110000001100000000

0110000000000000100000000000000000

0000000000000000000000000000000000

0000000000000000000000000000000000

0000000000000000000000000000000000

Opening

42 31 0 1

0000000000000000000000000000000000

0000000000000000000000000000000000

0000000000000000100000000000000000

0000000000000001110000000000000000

0000000000000011111000000000000000

0000000000000111111100000000000000

0000000000001111001110000000000000

0000000000011111011111000000000000

0000000000111111111111100000000000

0000000000010110011101000000000000

0000000000111110011011100000000000

0000000000111111111111100000000000

0000000001110111111111110000000000

0000000000111111100111100000000000

0000000000111101110111100000000000

0000000000011110111111000000000000

0000000000000100111100000000000000

0000000000000001111000000000000000

0000000000000001110000000000000000

0000000000000000100000000000000000

0000000000000000000000000000000000

0000100000000000000000000000000000

0001110000000000100000000000000000

0001110000000001110000000000000000

0000100000000001110000000000000000

0000000000000011111000000000000000

0000000000000111111100000000000000

0000000000001111001110000000000000

0000000000011111100100000000000000

0000000000111001110000000000000000

0000000000111001111001000000000000

0000000000111011110011100000000000

0000000000010111100111100000000000

0000000000111111100011100000000000

0000000000111111110001000000000000

0000000000111100111011100000000000

0000000000011111111101000000000000

0000000000000111111100000000000000

0000000000000011111000000000000000

0000000000000001110000000000000000

0000000000000001110000000000000000

0000000000000000100000000000000000

0000000000000000000000000000000000

0000000000000000000000000000000000

0000000000000000000000000000000000

**Outputs for data B**

**Output one**

Dilation

21 21 0 1

000000111110111110000000

001111111110111111111000

001111111110111111111000

001111111110111111111000

111111111111111111111110

111111111111111111111110

111111111111111111111110

111111111111111111111110

111111111111111111111110

000000111110111110000000

111111111111111111111110

111111111111111111111110

111111111111111111111110

111111111111111111111110

111111111111111111111110

011111111110111111111000

011111111110111111111000

011111111110111111111000

111111111111111111111110

111111111111111111111110

111111111111111111111110

111111111111111111111110

111111111111111111111110

000000000000000000000000

Erosion

21 21 0 1

000000000000000000000000

000000000000000000000000

000000001000001000000000

000000001000001000000000

000000001000001000000000

000000001000001000000000

001111111111111111111000

000000001000001000000000

000000001000001000000000

000000001000001000000000

000000001000001000000000

000000001000001000000000

001111111111111111111000

000000001000001000000000

000000001000001000000000

000000001000001000000000

000000001000001000000000

000000001000001000000000

000000001000001000000000

000000001000001000000000

001111111111111111111000

000000000000000000000000

000000000000000000000000

000000000000000000000000

Closing

21 21 0 1

000000000000000000000000

000000011100011100000000

000111111100011111110000

000111111100011111110000

000111111100011111110000

011111111111111111111100

011111111111111111111100

011111111111111111111100

000000011100011100000000

000000011100011100000000

000000011100011100000000

011111111111111111111100

011111111111111111111100

011111111111111111111100

001111111100011111110000

001111111100011111110000

001111111100011111110000

001111111100011111110000

001111111100011111110000

011111111111111111111100

011111111111111111111100

011111111111111111111100

000000000000000000000000

000000000000000000000000

Opening

21 21 0 1

000000000000000000000000

000000011100011100000000

000000011100011100000000

000000011100011100000000

000000011100011100000000

011111111111111111111100

011111111111111111111100

011111111111111111111100

000000011100011100000000

000000011100011100000000

000000011100011100000000

011111111111111111111100

011111111111111111111100

011111111111111111111100

000000011100011100000000

000000011100011100000000

000000011100011100000000

000000011100011100000000

000000011100011100000000

011111111111111111111100

011111111111111111111100

011111111111111111111100

000000000000000000000000

000000000000000000000000