OOP With Design Pattern

Student Name: Bichi Zhang

ID#: 1073838

Program1

Due Date: 02/25/2019

Date of Submission: 02/25/2019

**Table of Contents**

Documented Source Code…………………………………………………………………………………………………….3

Screenshots ………………………………………………………………………………………………………………………….9

Things I learned……………………………………………………………………………………………………………………10

Documented Source Code:

File Name: main.cpp

Author: Bichi Zhang

Date: 02/18/2019

Complier Used: C++ by Visual Studio 2017

Description: In this file, read from the “input.txt” file by the overloaded >> and do a loop for next genarations.

#include <iostream>

#include <fstream>

#include <iomanip>

#include "game\_of\_life.h"

using namespace std;

int main()

{

ifstream ifs("input.txt");

if (!ifs.is\_open())

{

return -1;

}

GameOfLife gameOfLife;

ifs >> gameOfLife;

ifs.close();

ofstream ofs("output.txt");

for (int i = 0;i < 5; ++i)

{

gameOfLife.generate();

ofs << gameOfLife << endl;

}

ofs.close();

ofs << setw(100);

return 0;

}

File Name: game of life. h

Author: Bichi Zhang

Date: 02/19/2019

Complier Used: C++ by Visual Studio 2017

Description: In this header file, I overloaded the “<<” and “>>”, and named the functions to create matrices.

#ifndef \_\_GAME\_OF\_LIFE\_H\_\_

#define \_\_GAME\_OF\_LIFE\_H\_\_

#include <iostream>

using namespace std;

class GameOfLife {

public:

GameOfLife(): generations(0) {}

void setOrganism(int i, int j)

{

matrix[i][j] = 'x';

}

void generate();

friend istream& operator>>(istream& in, GameOfLife& life);

friend ostream& operator<<(ostream& out, const GameOfLife& life);

private:

bool isValid(int i, int j)

{

return 0 <= i && i < nrows && 0 <= j && j < ncols;

}

int countNeibors(int i, int j);

int nrows;

int ncols;

int generations;

char\*\* matrix;

};

File Name: game of life. cpp

Author: Bichi Zhang

Date: 02/23/2019

Complier Used: C++ by Visual Studio 2017

Description: In this file, I use the “countNeibors” function to get the number of each organism, use the “generate” function to create a new matrix to store the next generation information then copy and delete it. By overloading “>>”, I get the size and information of the first generation and then output files by overloading “<<”.

#include "game\_of\_life.h"

void GameOfLife::generate()

{

if (generations != 0)

{

//create new matrix

char\*\* newMatrix = new char\*[nrows + 2];

for (int i = 0;i < nrows + 2; ++i)

{

newMatrix[i] = new char[ncols + 2];

for (int j = 0;j < ncols + 2; ++j)

{

newMatrix[i][j] = '#';

}

}

for (int i = 0;i < nrows + 2; ++i)

{

for (int j = 0;j < ncols + 2; ++j)

{

int neibors = countNeibors(i, j);

if (neibors == 3)

{

newMatrix[i][j] = 'x';

}

else if (neibors > 3 || neibors < 2)

{

newMatrix[i][j] = '#';

}

else

{

if (1 <= i && i <= nrows && 1 <= j && j <= ncols)

{

newMatrix[i][j] = matrix[i - 1][j - 1];

}

else

{

newMatrix[i][j] = '#';

}

}

}

}

for (int i = 0;i < nrows; ++i)

{

delete[] matrix[i];

}

delete[] matrix;

matrix = newMatrix;

nrows += 2;

ncols += 2;

}

generations += 1;

}

int GameOfLife::countNeibors(int i, int j)

{

int count = 0;

i -= 1;

j -= 1;

//left-top

if (isValid(i - 1, j - 1) && matrix[i - 1][j - 1] == 'x') ++count;

//top

if (isValid(i - 1, j) && matrix[i - 1][j] == 'x') ++count;

//right-top

if (isValid(i - 1, j + 1) && matrix[i - 1][j + 1] == 'x') ++count;

//left

if (isValid(i, j - 1) && matrix[i][j - 1] == 'x') ++count;

//right

if (isValid(i, j + 1) && matrix[i][j + 1] == 'x') ++count;

//left-bottom

if (isValid(i + 1, j - 1) && matrix[i + 1][j - 1] == 'x') ++count;

//bottom

if (isValid(i + 1, j) && matrix[i + 1][j] == 'x') ++count;

//right-bottom

if (isValid(i + 1, j + 1) && matrix[i + 1][j + 1] == 'x') ++count;

return count;

}

istream& operator>>(istream& in, GameOfLife& life)

{

in >> life.nrows >> life.ncols;

life.matrix = new char\*[life.nrows];

for (int i = 0;i < life.nrows; ++i)

{

life.matrix[i] = new char[life.ncols];

for (int j = 0;j < life.ncols; ++j)

{

life.matrix[i][j] = '#';

}

}

int i, j;

while (true)

{

in >> i;

if (i == -1)

{

break;

}

in >> j;

life.setOrganism(i, j);

}

return in;

}

ostream& operator<<(ostream& out, const GameOfLife& life)

{

out << "Generation " << life.generations << ":" << endl;

int minRow = life.nrows, maxRow = -1, minCol = life.ncols, maxCol = -1;

for (int i = 0;i < life.nrows; ++i)

{

for (int j = 0;j < life.ncols; ++j)

{

if (life.matrix[i][j] == 'x')

{

if (minRow > i)

minRow = i;

if (maxRow < i)

maxRow = i;

if (minCol > j)

minCol = j;

if (maxCol < j)

maxCol = j;

}

}

}

out << "# ";

for (int j = minCol;j <= maxCol; ++j)

{

out << "# ";

}

out << "#" << endl;

for (int i = minRow;i <= maxRow; ++i)

{

out << "# ";

for (int j = minCol;j <= maxCol; ++j)

{

out << life.matrix[i][j] << " ";

}

out << "#" << endl;

}

out << "# ";

for (int j = minCol;j <= maxCol; ++j)

{

out << "# ";

}

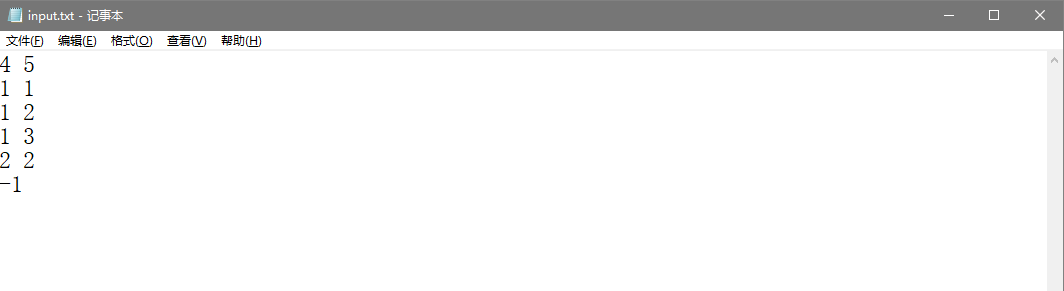
out << "#" << endl;

return out;

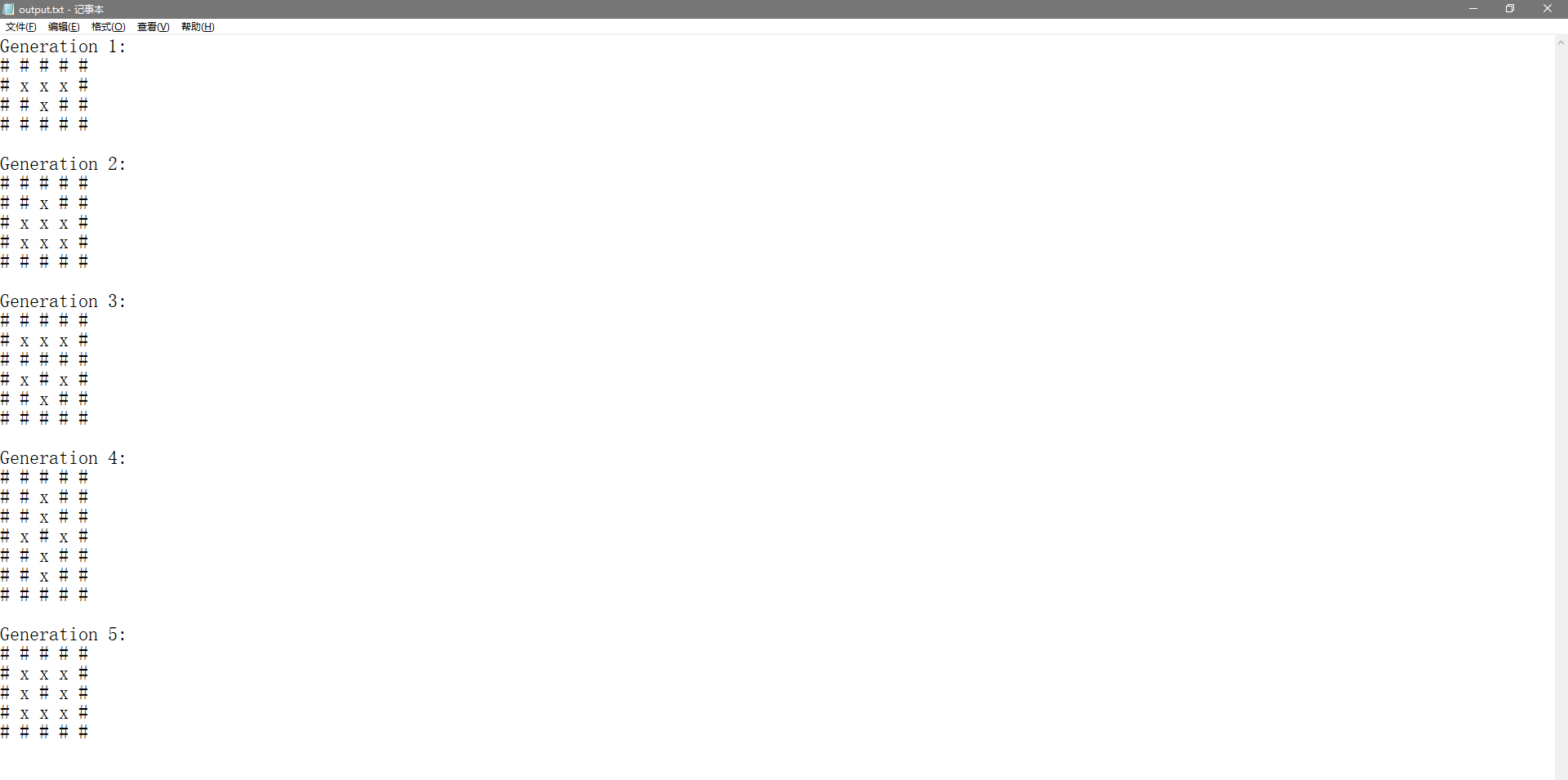
}

Screenshot:

1. Input file:



1. Output File:



Things I learned:

In the process of completing this project, I really learned a lot. I reviewed the knowledge I learned in the class such as how to create, copy, resize a matrix by using pointer and how to delete it. The loop code written to determine different situations improves my logical thinking ability. The requirement of using <iomanip> also helps me understand more.