**Joint DSA & Programming Assignment**

**Ian Mac Mathuna**

**C00206527**

**Date: 17/11/2017**

**Advanced Programming – Noel O Hara**

**Discrete Structures and Algorithms – Aine Byrne**

****

Table of Contents

[Description of the Problem 3](#_Toc498683760)

[Description of underlying data structures 3](#_Toc498683761)

[Pseudocode of the Algorithms used 4](#_Toc498683762)

[Copy of C++ Code 7](#_Toc498683763)

[Description of all the functions/routines 11](#_Toc498683764)

[Testing 12](#_Toc498683765)

# **Description of the Problem**

Part 1:

Create a 2D world is represented by a 2D grid of 8x8 squares (2d Array) each square contains a creature in it or not.

All the creatures are safe in the world given these conditions;

* The world is divided into rows, columns and diagonals.
* There are exactly 8 creatures in the world. All creatures must be able to survive.
* They can only survive if no creature can “see” another. For them to survive there must be exactly one creature in each row and exactly one creature in each column, and at most one creature in the 30 diagonals.

Part 2:

Using SFML you must draw the world on screen. Empty squares are drawn in white. Safe creatures are coloured green and unsafe creatures are drawn in red. If all are safe the program must display a message to indicate it. Then read in the next 8 creatures.

# **Description of underlying data structures**

I used two 2D arrays in this assignment as we were required to create an 8 x 8 grid to represent the world space in which the creatures were to be placed. A 2D array is an box that has N x N spaces in it for values.

# **Pseudocode of the Algorithms used**

**Reading in .txt file:**

ifstream Creatures;

Creatures.open("Creatures.txt", ios::in);

if (Creatures.is\_open())

{

getline(Creatures, line);

std::istringstream stream(line);

}

**Placing Creatures into Array:**

for (int i = 0; i < 16; i += 2)

{

int XAxis = line[i] - 48 - 1;

int YAxis = line[i + 1] - 48 - 1;

CreaturePos[XAxis][YAxis] = 2;

}

**Checking Rows:**

for (int column = 0; column < 8; column++)

{

int count = 0;

for (int row = 0; row < 8; row++) {

if (CreaturePos[row][column] != 0)

{

count++;

if (count > 1)

{

for (int row = 0; row < 8; row++)

{

if (CreaturePos[row][column] == 2)

{

CreaturePos[row][column] = 1;

}

}

}

}

}

}

**Checking columns:**

for (int row = 0; row < 8; row++)

{

int count = 0;

for (int column = 0; column < 8; column++)

{

if (CreaturePos[row][column] != 0)

{

count++;

if (count > 1)

{

for (int column = 0; column < 8; column++)

{

if (CreaturePos[row][column] == 2)

{

CreaturePos[row][column] = 1;

}

}

}

}

}

}

**Checking Down left Diagonal:**

for (int column = 0; column > 8; column++) // For every across

{

int count = 0;

for (int row = 0; row > 8; row++)

{

for (int x = row, y = column; x >= 0, y >= 0; x++, y++)

{

if (CreaturePos[x][y] != 0)

{

count++;

if (count > 1)

{

if (CreaturePos[x][y] != 0)

{

CreaturePos[x][y]= 1;

}

}

}

}

}

}

**Checking Down right Diagonal:**

for (int column = 8; column > 0; column--) // For every across

{

int count = 0;

for (int row = 0; row > 8; row++)

{

for (int x = row, y = column; x >= 0, y >= 0; x++, y++)

{

if (CreaturePos[x][y] != 0)

{

count++;

if (count > 1)

{

if (CreaturePos[x][y] != 0)

{

CreaturePos[x][y]= 1;

}

}

}

}

}

**Printing 2D array of numbers :**

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

cout << CreaturePos[j][i];

}

cout << "\n";

}

**Setting Creature status in SFML array:**

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

if (CreaturePos[i][j] == 0)

{

shapeArray[i][j] = SquareNoCreature;

}

else if (CreaturePos[i][j] == 1)

{

shapeArray[i][j] = SquareUnsafeCreature;

}

else if (CreaturePos[i][j] == 2)

{

shapeArray[i][j] = SquareSafeCreature;

}

shapeArray[i][j].setPosition(sf::Vector2f((55 \* i), (450 / 8)\*j));

}

**Drawing SFML Array:**

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

window.draw(shapeArray[i][j]);

}

}

# **Copy of C++ Code**

#include <C:/Library/SFML-2.4.2/include/SFML/Graphics.hpp>

#include <iostream>

#include <cstdlib>

#include <fstream>

#include <sstream>

#include <string>

using namespace std;

int main()

{

ifstream Creatures;

Creatures.open("Creatures.txt", ios::in);

int CreaturePos[8][8] = { 0 };

string line;

int row = 8;

int column = 8;

if (Creatures.is\_open())

{

getline(Creatures, line);

std::istringstream stream(line);

for (int i = 0; i < 16; i += 2)

{

int XAxis = line[i] - 48 - 1;

int YAxis = line[i + 1] - 48 - 1;

CreaturePos[XAxis][YAxis] = 2;

}

//CHECK ROWS

for (int column = 0; column < 8; column++) // For every column

{

int count = 0;

for (int row = 0; row < 8; row++) // For every square in every row

{

if (CreaturePos[row][column] != 0)

{

count++;

if (count > 1)

{

for (int row = 0; row < 8; row++)

{

if (CreaturePos[row][column] == 2)

{

CreaturePos[row][column] = 1;

}

}

}

}

}

}

//CHECK COLUMNS

for (int row = 0; row < 8; row++) // For every across

{

int count = 0;

for (int column = 0; column < 8; column++) // For down in every across

{

if (CreaturePos[row][column] != 0)

{

count++;

if (count > 1)

{

for (int column = 0; column < 8; column++)

{

if (CreaturePos[row][column] == 2)

{

CreaturePos[row][column] = 1;

}

}

}

}

}

}

//CHECK DIAGONAL

/\* for (int column = 8; column > 0; column--) // For every across

{

int count = 0;

for (int x = row, y = column; x >= 0, y >= 0; x++, y++)

{

if (CreaturePos[x][y] != 0)

{

count++;

if (count > 1)

{

if (CreaturePos[x][y] != 0)

{

CreaturePos[x][y] = 1;

}

}

}

}

}

DIAGONAL

count = 0;

for (int x = row, y = column - 1; x >= 0, y >= 0; ++x, --y)

{

if (CreaturePos[x][y] != 0)

{

count++;

if (count > 1)

{

for (x = x + 1, y = column - 1; x >= 0, y >= 0; ++x, --y)

{

if (CreaturePos[x][y] == 2)

{

CreaturePos[x][y] = 1;

}

}

}

}

}

\*/

//PRINT 2D ARRAY

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

cout << CreaturePos[j][i];

}

cout << "\n";

}

//SFML GRID

sf::RenderWindow window(sf::VideoMode(435, 440), "Survival Game!", sf::Style::Close);

window.clear();

sf::RectangleShape shapeArray[8][8];

//Square with an unsafe creature

sf::RectangleShape SquareUnsafeCreature(sf::Vector2f(50, 50));

SquareUnsafeCreature.setFillColor(sf::Color(250, 0, 0));

SquareUnsafeCreature.setOutlineThickness(5);

SquareUnsafeCreature.setOutlineColor(sf::Color(10, 0, 0));

//Square with a safe creature

sf::RectangleShape SquareSafeCreature(sf::Vector2f(50, 50));

SquareSafeCreature.setFillColor(sf::Color(0, 250, 0));

SquareSafeCreature.setOutlineThickness(5);

SquareSafeCreature.setOutlineColor(sf::Color(10, 0, 0));

//Square Without a Creature

sf::RectangleShape SquareNoCreature(sf::Vector2f(50, 50));

SquareNoCreature.setFillColor(sf::Color(250, 250, 250));

SquareNoCreature.setOutlineThickness(5);

SquareNoCreature.setOutlineColor(sf::Color(10, 0, 0));

while (window.isOpen())

{

sf::Event event;

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

if (CreaturePos[i][j] == 0)

{

shapeArray[i][j] = SquareNoCreature;

}

else if (CreaturePos[i][j] == 1)

{

shapeArray[i][j] = SquareUnsafeCreature;

}

else if (CreaturePos[i][j] == 2)

{

shapeArray[i][j] = SquareSafeCreature;

}

shapeArray[i][j].setPosition(sf::Vector2f((55 \* i), (450 / 8)\*j));

}

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

window.draw(shapeArray[i][j]);

}

}

window.display();

while (window.pollEvent(event))

{

if (event.type == sf::Event::Closed)

{

window.close();

}

}

}

system("pause");

return 0;

}

}

}

# **Description of all the functions/routines**

**Reading in .txt file:**

Using ifstream and getline I retrieved the .txt file called “Creatures.txt” which contained the positions of the various creatures.

**Placing Creatures into Array:**

To take the creatures from the txt file and place them into the first array I used a for loop to grab 2 values from the txt file and then set the position with those values to be 2(which in my code means safe creature)

**Checking rows in the Array:**

I use a triply nested for loop to check each row, it goes through each slot in each row and if it comes across more than 1 creature in a row it sets all creatures in said row to be unsafe.

**Checking columns in the Array:**

I use a triply nested for loop to check each column, it goes through each slot in each column and if it comes across more than 1 creature in a column it sets all creatures in said column to be unsafe.

**Checking diagonals in the Array:**

I use a triply nested for loop to check each diagonal, it goes through each slot in every diagonal and if it comes across more than 1 creature in the diagonal it sets all creatures in said diagonal to be unsafe.

**Printing 2D array of numbers:**

Using a nested for loop to print the entire 2D array which contained the numbers representing the creatures.

**Setting Creature status in SFML array:**

Cross referencing with the array of numbers to decide the colours of the squares, this is where the squares were set to Safe, Unsafe, and No Creature depending on whatever number it was set to in the numbers array.

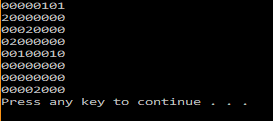
**Drawing SFML Array:**

Using a nested for loop this went through each square in the shape array printing them to the window in whatever position they are meant to be in and whatever colour they are set to be.

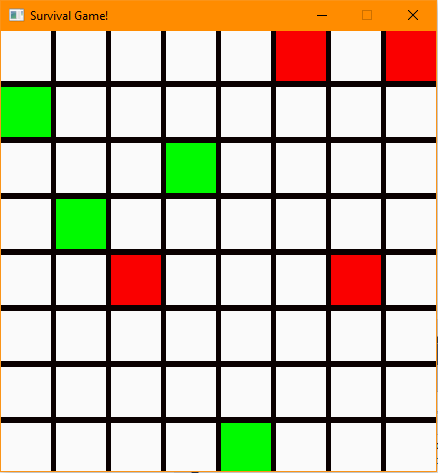
# **Testing**

**Test 1:**

Inputs: 1224354358617581



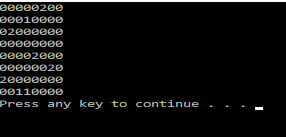
The Numbers Array



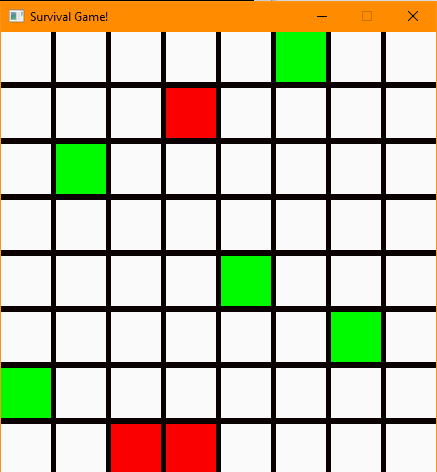
The SFML display

**Test 2:**

Inputs: 1723384255617648



Numbers Array



SFML Display