**// HW 2 Maze**

**// Ethan Chiu**

**//system(“CLS”);**

#include <cstdlib>

#include <ctime>

#include <windows.h>

#include <fstream> //for file input

#include <iostream> //for input and output

#include <string> //for using string built in functions

using namespace std;

//Global Vars

const int length = 10;

const int columns = 10;

void createMaze(char maze[][columns], int columns, int rows);

void showMaze(char maze[][columns], int columns, int rows);

void robot(char maze[][columns], int lastColumn, int lastRow, int goalX, int goalY);

int main()

{

//Data

char maze[length][columns];

int lastColumn;

int lastRow;

int goalX;

int goalY;

createMaze(maze, columns, length);

cout << "Starting Grid" << endl;

showMaze(maze, columns, length);

//Find starting point and goal

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

{

for(int b = 0; b < columns; b++)

{

if (maze[i][b] == 'S')

{

lastColumn = i;

lastRow = b;

}

else if (maze[i][b] == 'G')

{

goalX = i;

goalY = b;

}

}

}

cout << endl;

cout << "Robot: " << endl;

robot(maze, lastColumn, lastRow, goalX, goalY);

}

void createMaze(char maze[][columns], int columns, int rows)

{

ifstream fin;

int z = 0;

int row = 0;

int column = 0;

string temp;

//open file

fin.open("maze.txt");

if (!fin.good()) throw "I/O error";

//input maze from file

while(!fin.eof())

{

getline(fin, temp);

for (int a = 0; a < temp.length(); a++)

{

maze[z][a] = temp[a];;

}

for (int k = temp.length(); k < length; k++)

{

maze[z][k] = ' ';

}

z++;

}

fin.close();

}

void showMaze(char maze[][columns], int columns, int rows)

{

//output maze

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

{

for(int b = 0; b < columns; b++)

{

cout << maze[i][b];

}

cout << endl;

}

}

void robot(char maze[][columns], int lastColumn, int lastRow, int goalX, int goalY)

{

int steps;

unsigned seed;

seed = time(0);

srand(seed);

int move = rand() % 5 + 1;

maze[lastColumn][lastRow] = 'X';

while(lastColumn != goalX || lastRow != goalY)

{

maze[lastColumn][lastRow] = 'X';

showMaze(maze, columns, length);

move = rand() % 5 + 1;

//Associate random move with location of X

if (move == 1)

{

if(lastRow - 1 > 0 && maze[lastColumn][lastRow - 1] != '\*' && maze[lastColumn][lastRow - 1] != 'S')

{

//cout << 1;

maze[lastColumn][lastRow] = ' ';

lastRow -= 1;

}

}

else if (move == 2)

{

if(lastColumn - 1 > 0 && maze[lastColumn - 1][lastRow] != '\*' && maze[lastColumn - 1][lastRow] != 'S')

{

//cout << 2;

maze[lastColumn][lastRow] = ' ';

lastColumn -= 1;

}

}

else if (move == 3)

{

if(lastRow + 1 <= length - 1 && maze[lastColumn][lastRow + 1] != '\*' && maze[lastColumn][lastRow + 1] != 'S')

{

//cout << 3;

maze[lastColumn][lastRow] = ' ';

lastRow += 1;

}

}

else if (move == 4)

{

if(lastColumn + 1 <= length - 1 && maze[lastColumn + 1][lastRow] != '\*' && maze[lastColumn + 1][lastRow] != 'S')

{

//cout << 4;

maze[lastColumn][lastRow] = ' ';

lastColumn += 1;

}

}

steps++;

cout << endl;

Sleep(250);

}

maze[lastColumn][lastRow] = 'X';

showMaze(maze, columns, length);

cout << "Total number of steps: " << steps;

}