

/\*

Programmer: Josh Lee

Program Name: Maze.cpp

Version: 1.9.1

Date: 4/26/20

Description:

Creates an algorithm for a mouse to solve the maze given.

\*/

#include <iostream>

#include <Windows.h>

#include <fstream>

#include <string>

using namespace std;

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

//global constants for maze size

const int mazeSIZE = 3000;

const int rows = 30;

const int columns = 100;

//moves the cursor with the given gotoxy function

void gotoxy(int x, int y)

{

COORD pos = { x, y };

HANDLE output = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleCursorPosition(output, pos);

}

//creates a basic stack

class Stack {

public:

int spaces[mazeSIZE];

int top;

Stack() {

top = -1;

}

//checks if stack is full

bool isFull() {

return (top == mazeSIZE - 1);

}

//checks if stack is empty

bool isEmpty() {

return (top == -1);

}

//pushes values into stack

void push(int x, int y) {

if (!isFull()) {

top++;

spaces[top] = y \* 100 + x;

}

}

//pops values out of stack

int pop() {

if (!isEmpty()) {

int num;

num = spaces[top];

top--;

return num;

}

return -1;

}

}; // end class Stack

//class Maze

class Maze {

public:

//0 = path 1 = wall 2 = landmine 3 = exit 4 = visited

int maze[rows][columns];

//rows and columns

int xpos, ypos, direction;

Stack myStack;

Maze() {

xpos = 1;

ypos = 1;

direction = 1;

for (int i = 0; i < 100; i++) {

for (int j = 0; j < 30; j++) {

maze[j][i] = 0;

}

}

maze[ypos][xpos] = 4;

myStack.push(ypos, xpos);

}

//gets maze from file and outputs

void getMaze(string f) {

{

ifstream inFile;

char c;

inFile.open(f);

for (int i = 0; i < 30; i++) {

for (int j = 0; j < 100; j++) {

inFile.get(c);

maze[i][j] = (int)c - 48;

gotoxy(j, i);

SetConsoleTextAttribute(hConsole, 7);

//walls

if (maze[i][j] == 1)

cout << char(219);

//path

else if (maze[i][j] == 0)

cout << " ";

//mines

else if (maze[i][j] == 2) {

SetConsoleTextAttribute(hConsole, 12);

cout << char(233);

}

//exit

else if (maze[i][j] == 3) {

SetConsoleTextAttribute(hConsole, 10);

cout << char(176);

}

}

inFile.get(c);

cout << endl;

}

inFile.close();

gotoxy(xpos, ypos);

SetConsoleTextAttribute(hConsole, 14);

cout << char(255);

}

}

//moves the actual mouse

void moveMouse() {

//changes direction in a clockwise motion

//1 = R

//2 = D

//3 = L

//4 = U

gotoxy(xpos, ypos);

cout << "-";

myStack.push(xpos, ypos);

//moves right

if (direction == 1)

xpos++;

//moves down

else if (direction == 2)

ypos++;

//moves left

else if (direction == 3)

xpos--;

//moves up

else if (direction == 4)

ypos--;

gotoxy(xpos, ypos);

SetConsoleTextAttribute(hConsole, 14);

cout << char(254);

if (maze[ypos][xpos] == 0)

maze[ypos][xpos] = 4;

Sleep(60);

}

void backTrack(int space) {

gotoxy(xpos, ypos);

cout << " ";

ypos = space / 100;

xpos = space % 100;

gotoxy(xpos, ypos);

cout << char(254);

Sleep(60);

}

//function to change direction

void changeDirection() {

// 1 = right 2 = down 3 = left 4 = up

if (direction == 1)

direction++;

else if (direction == 2)

direction++;

else if (direction == 3)

direction++;

else if (direction == 4)

direction = 1;

}

//checks to see if there are any open spaces above, below, to the left, or to the right

bool checkSpaces() {

return (maze[ypos][xpos + 1] == 0 || maze[ypos][xpos + 1] == 3 ||

maze[ypos - 1][xpos] == 0 || maze[ypos - 1][xpos] == 3 ||

maze[ypos][xpos - 1] == 0 || maze[ypos][xpos - 1] == 3 ||

maze[ypos + 1][xpos] == 0 || maze[ypos + 1][xpos] == 3);

}

//checks the space in front of the mouse to see if it's free

bool checkNext() {

if (direction == 1)

return (maze[ypos][xpos + 1] == 0 || maze[ypos][xpos + 1] == 3 || maze[ypos][xpos + 1] == 2);

else if (direction == 2)

return (maze[ypos + 1][xpos] == 0 || maze[ypos + 1][xpos] == 3 || maze[ypos + 1][xpos] == 2);

else if (direction == 3)

return (maze[ypos][xpos - 1] == 0 || maze[ypos][xpos - 1] == 3 || maze[ypos][xpos - 1] == 2);

else if (direction == 4)

return (maze[ypos - 1][xpos] == 0 || maze[ypos - 1][xpos] == 3 || maze[ypos - 1][xpos] == 2);

return 0;

}

//solves the maze

void solveMaze() {

while (maze[ypos][xpos] != 3 && maze[ypos][xpos] != 2)

{

if (checkNext()) {

moveMouse();

}

else if (checkSpaces())

{

changeDirection();

}

else

backTrack(myStack.pop());

}

//ends game if hits landmines

if (maze[ypos][xpos] == 2)

{

system("cls");

system("Color CE");

gotoxy(0, 0);

for (int i = 0; i < 30; i++) {

for (int j = 0; j < 100; j++) {

cout << char(219);

Sleep(1);

}

}

SetConsoleTextAttribute(hConsole, 12);

gotoxy(57, 10);

cout << "BOOM!";

gotoxy(44, 12);

cout << "Better luck next time little mouse";

}

//ends game with victory if maze solved with errors

if (maze[ypos][xpos] == 3)

{

system("cls");

system("Color 0A");

gotoxy(50, 10);

cout << "You did it!";

}

}

}; //end class Maze

//creates instance of maze and solves

int main() {

char mines;

string path;

cout << "Toggle mines? (Y/N)" << endl;

cin >> mines;

if (mines == 'y' || mines == 'Y')

path = "maze.txt";

else

path = "maze-nomines.txt";

system("cls");

SetConsoleTextAttribute(hConsole, 7);

Maze m;

m.getMaze(path);

gotoxy(0, 0);

m.solveMaze();

gotoxy(0, 24);

SetConsoleTextAttribute(hConsole, 7);

return 0;

}

// create stack with locations and current locations

// create game

// check directions

// output maze

// move forward

// solve maze