

数据结构课程设计

——勇闯迷宫游戏

项目说明文档

作 者 姓 名： 韦世贸

学 号： 2351131

指 导 老 师： 软件学院

专 业： 软件工程

同济大学

Tongji University

1 介绍

1.1 背景介绍

迷宫指的是充满复杂通道，很难找到从其内部到达入口或从入口到达中心的道路，道路复杂难辨，人进去不容易出来的建筑物。通常比喻复杂艰深的问题或难以捉摸的局面。

求解迷宫问题对现实世界也有很大的帮助，所以此次的项目就迷宫问题展开探究。

1.2 功能介绍

作为一个最简易的勇闯迷宫游戏，首先应该有的功能就是展现迷宫。其次是解决迷宫问题进而找到离开路径，以保证游戏可以完成，完成后还会打印出行走的路径。最后，为了增加趣味性还设置了各种大小的迷宫可以随便选择。

2 设计

2.1 数据结构设计

该本项目的实质是对迷宫的路径搜索，所以本项目数据结构采用数组存储迷宫，将问题转化到数组中解决，可以更加简化，同时还使用了栈来保存迷宫的路径。当然，对于寻找迷宫通路时我们进行了比较基础的递归算法。

2.2 类结构设计

在该项目中创建一个迷宫结构体和一个链式栈，用栈来保存迷宫通路，大大简化了代码的复杂度。

2.3 成员与操作设计

迷宫结构体的代码如下：

struct Maze //把rows, cols,maze数组,dir数组封装起来

{

// 迷宫的行数和列数

int rows, cols;

// 迷宫地图

int maze[MAX\_ROWS][MAX\_COLS];

// 方向数组，分别表示上、右、下、左

int directions[4][2] = { {-1, 0}, {0, 1}, {1, 0}, {0, -1} };

Maze():rows(0),cols(0)

{

// 初始化迷宫为全障碍

for (int i = 0; i < NewMaze.rows; ++i) {

for (int j = 0; j < NewMaze.cols; ++j) {

NewMaze.maze[i][j] = 1;

}

}

}

}NewMaze;

链式栈的实现省略。

3 实现

3.1 创建迷宫功能的实现

3.1.1 创建迷宫功能流程简介

创建迷宫功能为了实现用户更好的体验由用户自定义数组大小，迷宫形式不变的是起始点和终点的坐标都是两对角，迷宫初始化时设定为全障碍，并使用深度优先搜索随机生成迷宫，同时保证生成的迷宫都有通路。

3.1.2 创建迷宫功能核心代码（深度优先搜索）

void dfsGenerateMaze(int x, int y)

{

// 将当前位置设为通路

NewMaze.maze[x][y] = 0;

// 随机选择一个方向

int dir[4] = { 0, 1, 2, 3 };

for (int i = 0; i < 4; ++i) {

int r = rand() % 4;

swap(dir[i], dir[r]);

}

// 尝试四个方向

for (int i = 0; i < 4; ++i) {

int newX = x + NewMaze.directions[dir[i]][0] \* 2;

int newY = y + NewMaze.directions[dir[i]][1] \* 2;

if (MazeisValid(newX, newY)) {

// 打通中间的墙

NewMaze.maze[x + NewMaze.directions[dir[i]][0]][y + NewMaze.directions[dir[i]][1]] = 0;

dfsGenerateMaze(newX, newY);

}

}

}

3.2 寻找迷宫路径功能的实现

3.2.1 寻找迷宫路径功能流程简介

迷宫问题的求解过程采用回溯法即在一定的约束条件下试探地搜索前进，若前进中受阻，则及时回头纠正错误另择通路继续搜索。从入口出发，按某一方向向前探索，若能走通，即某处可达，则到达新点，否则探索下一个方向；若所有的方向均没有通路，则沿原路返回前一点，换下一个方向再继续试探，直到所有可能的道路都探索到，或找到一条通路，或无路可走又返回入口点。在求解过程中，为了保证在达到某一个点后不能向前继续行走时，能正确返回前一个以便从下一个方向向前试探，则需要在试探过程中保存所能够达到的每个点的下标以及该点前进的方向，当找到出口时试探过程就结束了。，如果未找到出路也会进行相应提示。

3.2.2 寻找迷宫路径功能核心代码

// 回溯法求解迷宫问题

bool solveMaze(int startX, int startY, int endX, int endY)

{

// 如果当前位置是出口，则找到路径

if (startX == endX && startY == endY) {

// 将当前位置入栈

Path.Push(startX, startY);

NewMaze.maze[endX][endY] = 2;

return true;

}

// 标记当前位置为已访问

NewMaze.maze[startX][startY] = 2;

// 尝试四个方向

for (int i = 0; i < 4; ++i) {

int newX = startX + NewMaze.directions[i][0];

int newY = startY + NewMaze.directions[i][1];

if (PathisValid(newX, newY)) {

if (solveMaze(newX, newY, endX, endY)) {

// 将当前位置入栈

Path.Push(startX, startY);

return true;

}

}

}

// 如果所有方向都无法走通，则回溯

Path.Pop();

NewMaze.maze[startX][startY] = 0; // 恢复当前位置为通路

return false;

}

3.3 打印迷宫及其路径功能的实现

3.3.1 打印迷宫及其路径功能流程简介

打印函数的输出为两部分一部分为迷宫的输出，一部分为路径的输出，迷宫输出直接历遍迷宫输出即可，路径的输出通过递归将栈顶的元素输出即可。

3.3.2 打印迷宫及其路径功能核心代码

// 打印迷宫

void printMaze()

{

cout << " ";

for (int i = 0; i < NewMaze.cols; i++)

cout << i << "列 ";

cout << endl;

for (int i = 0; i < NewMaze.rows; ++i) {

for (int j = 0; j < NewMaze.cols; ++j) {

if (j == 0)

cout << i << "行 ";

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

cout << "#";

else if (NewMaze.maze[i][j] == 2)

cout << "x";

else

cout << NewMaze.maze[i][j];

cout << " ";

}

cout << endl;

cout << endl;

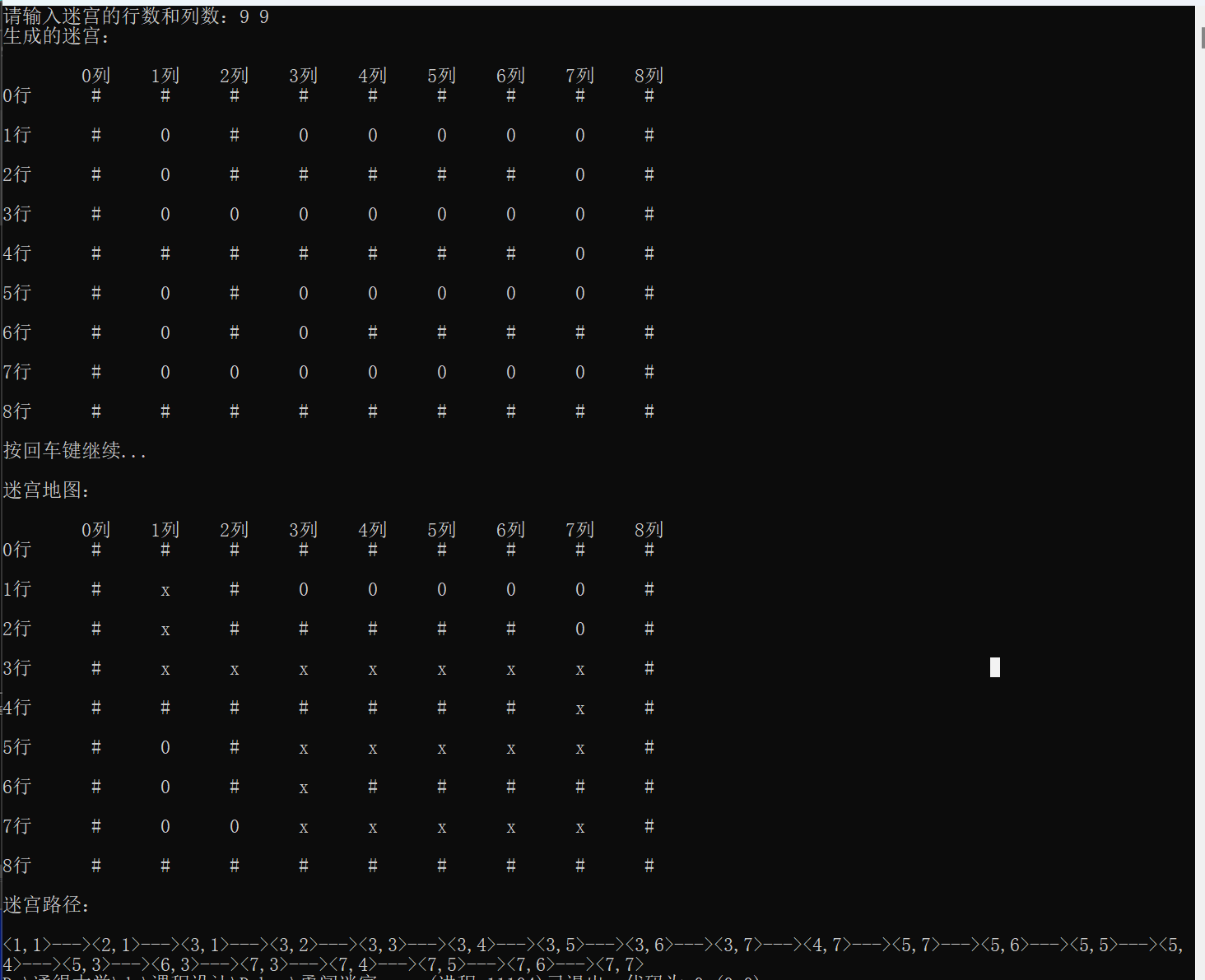
}

}

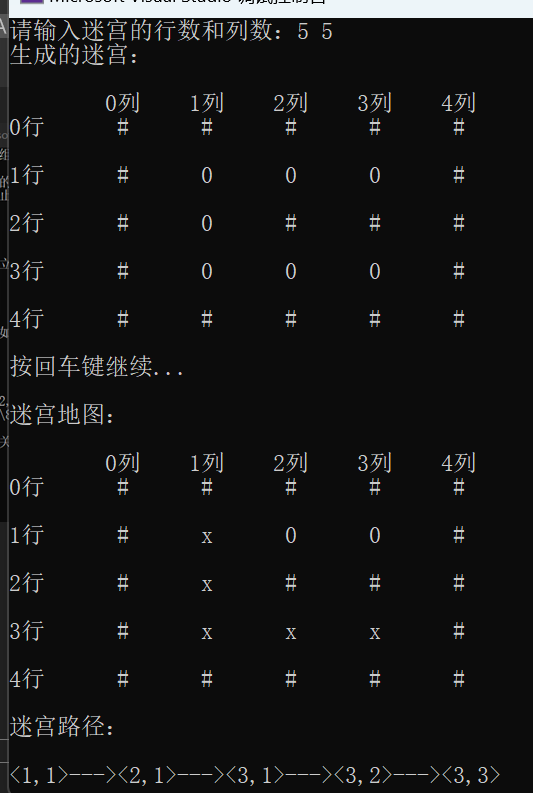
3.4 总体功能的实现

3.4.1 总体功能截屏示例

9\*9的迷宫正常输出：

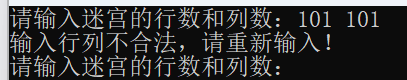


5\*5正常输出：



3.4.2 总体功能测试以及边界测试

1.输入迷宫行列不合法：



4 .设计小结

在这次的勇闯迷宫游戏制作过程中我学到了许多关于搜索的方法，有回溯法以及深度优先搜索，也体会到栈的方便之处，就是不需要为输出顺序而烦恼。这次的代码其实也存在有一些不足，就是迷宫的行列必须为奇数。但是也做出了一些修改：建立起迷宫后不会立即输出迷宫通路，而是显示“按空格键继续”，给了玩家独立思考的时间