项目说明文档

数据结构课程设计

——勇闯迷宫游戏

作 者 姓 名： 曹峰源

学 号： 1951328

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目录

[1. 分析 3](#_Toc59911427)

[1.1 背景分析 3](#_Toc59911428)

[1.2 功能分析 3](#_Toc59911429)

[2. 功能实现 3](#_Toc59911430)

[2.1 地图输入功能实现 3](#_Toc59911431)

[2.2 判断出口功能实现 4](#_Toc59911432)

[2.3 判断是否可以停留功能实现 4](#_Toc59911433)

[2.4 寻找路径功能实现 4](#_Toc59911434)

[2.4.1 流程图 4](#_Toc59911435)

[2.4.2 代码实现 5](#_Toc59911436)

[3. 测试 6](#_Toc59911437)

[3.1 功能测试 6](#_Toc59911438)

[3.2 错误测试 6](#_Toc59911439)

1. 分析
   1. 背景分析

计算机的发展使得游戏的发展也变得快速起来，不仅在于PC端或者网页游戏的发展，在传统游戏问题的解决上计算机也有卓越的贡献，比如说这段程序解决的迷宫问题，用DFS算法可以很快的进行解决。

* 1. 功能分析

对于迷宫问题的解决，首先要能进行迷宫地图的输入，在这段程序中方便起见，将迷宫地图规定好，不单独设置地图的输入，其次就是进行问题的求解，要能在规定起点终点的情况下，寻找到起点到终点的路径，并将其输出。

1. 功能实现
   1. 地图输入功能实现

代码实现：

1. **void** Mazeget(Maze\* maze)//将迷宫地图输入进去
2. {
3. **if** (maze==NULL)
4. {
5. **return**;
6. }
7. **char** temp[maze\_row][maze\_col] =
8. {
9. {'#','#','#','#','#','#','#'},
10. {'#','\*','#','\*','\*','\*','#'},
11. {'#','\*','#','\*','#','#','#'},
12. {'#','\*','\*','\*','#','\*','#'},
13. {'#','\*','#','\*','\*','\*','#'},
14. {'#','\*','#','\*','#','\*','#'},
15. {'#','#','#','#','#','#','#'}
16. };
17. **int** row = 0;
18. **for** (**int** row=0; row<maze\_row;row++)
19. {
20. **for** (**int** col=0;col<maze\_col;col++)
21. {
22. maze->map[row][col] = temp[row][col];
23. }
24. }
25. }

在该函数中地图是规定好的，如果需要自行输入地图的话，可以在该函数中用流输入。

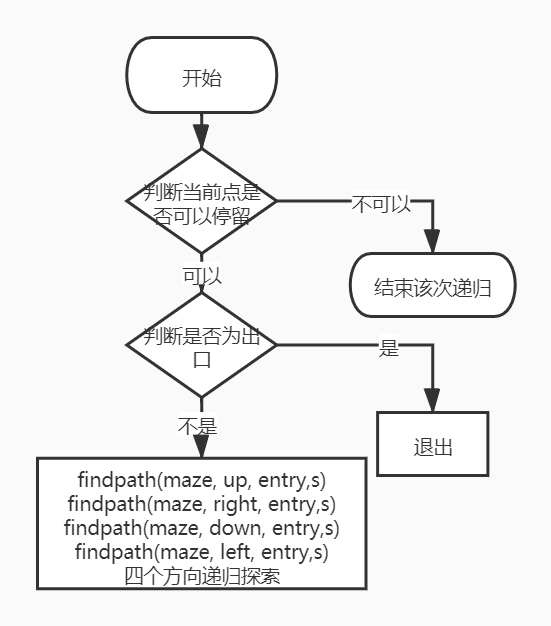
* 1. 判断出口功能实现

代码实现：

1. **bool** isexit(point cur)//判断是否为出口
2. {
3. **if** (cur.row == exit\_row && cur.col == exit\_col)
4. {
5. **return** **true**;
6. }
7. **else** **return** **false**;
8. }
   1. 判断是否可以停留功能实现

代码实现：

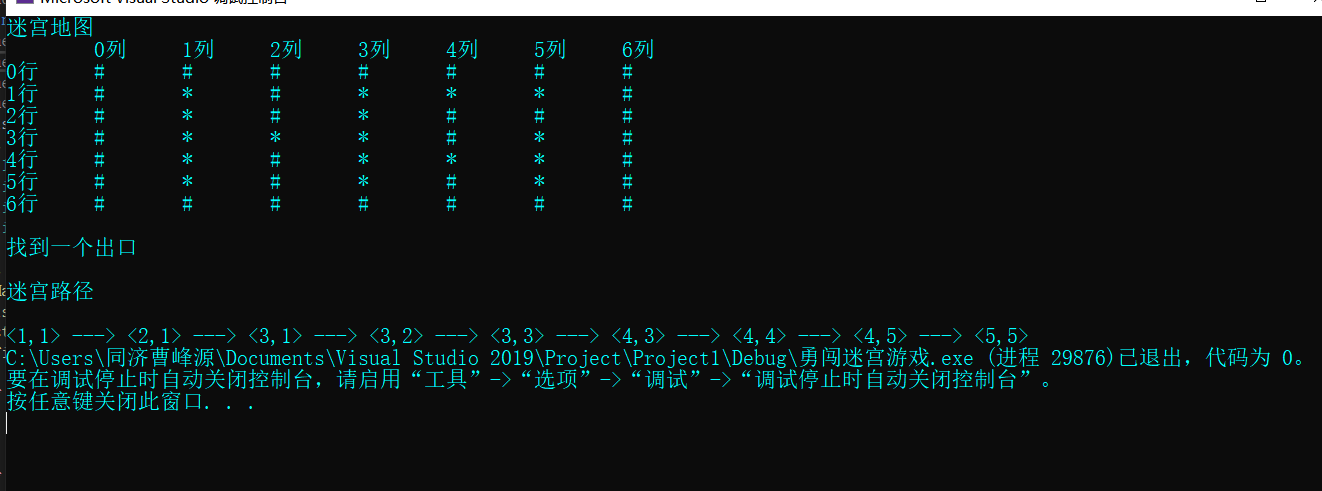
1. **bool** stay(Maze\* maze, point cur)//判断该点cur是否可以落脚
2. {
3. **if** (cur.row < 0 || cur.row >= maze\_row || cur.col < 0 || cur.col >= maze\_col)
4. {
5. **return** 0;
6. }
7. **if** (maze->map[cur.row][cur.col] == '\*')
8. {
9. //说明是路，且是未走过的路，说明可以落脚
10. **return** 1;
11. }
12. **else**
13. {
14. **return** 0;
15. }
16. }
    1. 寻找路径功能实现
       1. 流程图



* + 1. 代码实现

1. **void** findpath(Maze\* maze, point cur, point entry, linklist& s)//寻找路径
2. {
3. **if** (maze == NULL)
4. {
5. **return**;
6. }
7. **if** (stay(maze, cur) == 0)
8. {
9. **return**;
10. }
11. maze->map[cur.row][cur.col] = '0';
12. s.insert(cur.row, cur.col);
13. **if** (isexit(cur) == 1)//是否为出口，是的话寻找迷宫出口完成
14. {
15. cout << "找到一个出口" << endl << endl;
16. cout << "迷宫路径" << endl << endl;
17. s.print();
18. isfind = 1;
19. **return**;
20. }
21. //向四个方向进行递归
22. point up = cur;
23. up.row -= 1;
24. findpath(maze, up, entry,s);
25. s.del(cur.row,cur.col);
26. point right = cur;
27. right.col += 1;
28. findpath(maze, right, entry,s);
29. s.del(cur.row, cur.col);
30. point down = cur;
31. down.row += 1;
32. findpath(maze, down, entry,s);
33. s.del(cur.row, cur.col);
34. point left = cur;
35. left.col -= 1;
36. findpath(maze, left, entry,s);
37. s.del(cur.row, cur.col);
38. }
39. 测试
    1. 功能测试

结果显示：



* 1. 错误测试

当不存在从起点到终点的路径时，我们不希望程序不作出任何反应，而是进行报错。

结果显示：

