|  |  |  |
| --- | --- | --- |
| **数据结构上机实习报告** | | |
| 图的遍历 | | |
|  | 班级： | 191174班 |
|  | 学号： | 20161001764 |
|  | 姓名： | 牟鑫一 |
|  | 日期： | 2018.11.23 |

一、课程设计题目与要求

1.1 题目：图的遍历

1.2 系统功能要求

【问题描述】

有一个长方形的房间，房间里的地面上布满了正方形的瓷砖，瓷砖要么是红色的，要么是黑色的。一个人站在其中一块黑色的瓷砖上，他可以向四周的瓷砖上移动，但是不能移动到红色的瓷砖上，只能在黑色的瓷砖上移动，那么他可以到达这个房间中的黑色瓷砖的数目为多少？

【测试数据】

输入的测试数据应该是两个整数W和H，分别表示这个房间中x方向和y方向上瓷砖的数目。可以选择相应的符号代替黑色瓷砖和红色瓷砖，例如“#”表示黑色瓷砖，“\*”表示红色瓷砖，“@”表示该位置的黑色瓷砖，且此时人站在上面，注意每个测试数据只有一个“@”符号。输出数据即为从开始位置的黑色瓷砖到最后位置处总共经过的黑色瓷砖数目。并输出移动路径的可视化结果。

二、需求分析

2.1 问题描述

计算房间中可以到达的黑色瓷砖的数目

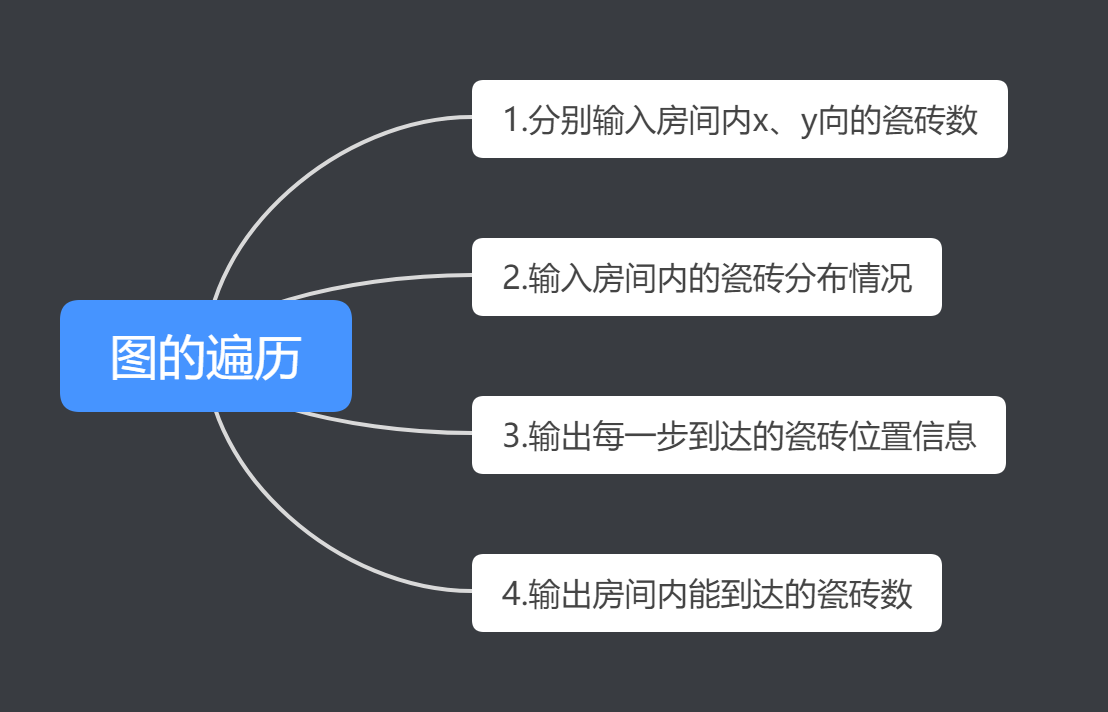
2.2 系统环境

Microsoft Visual Studio 2017

2.3 运行要求

Windows环境下运行

三、概要设计



用一个矩阵保存房间中的瓷砖信息以及开始的时候人所处的位置，同时用一个相同结构的矩阵保存结点是否被访问过的信息，访问过的结点不再重复访问。用递归算法分别向当前结点的上、右、下、左四个方向递归，如果找到黑色瓷砖则瓷砖数count加1，继续以这个节点作为初始位置递归查找能到大的黑色瓷砖；如果找到红色瓷砖则结束当前递归并继续往下一个分支递归回溯。

五、测试

5.1 测试数据

* 分别输入房间中x方向和y方向上的瓷砖数目：4 3
* 输入房间内的瓷砖信息：

@ \* #

# # \*

\* # \*

\* \* \*

* 结果：4
* 分别输入房间中x方向和y方向上的瓷砖数目：4 4
* 输入房间内的瓷砖信息：

# \* # \*

# @ # #

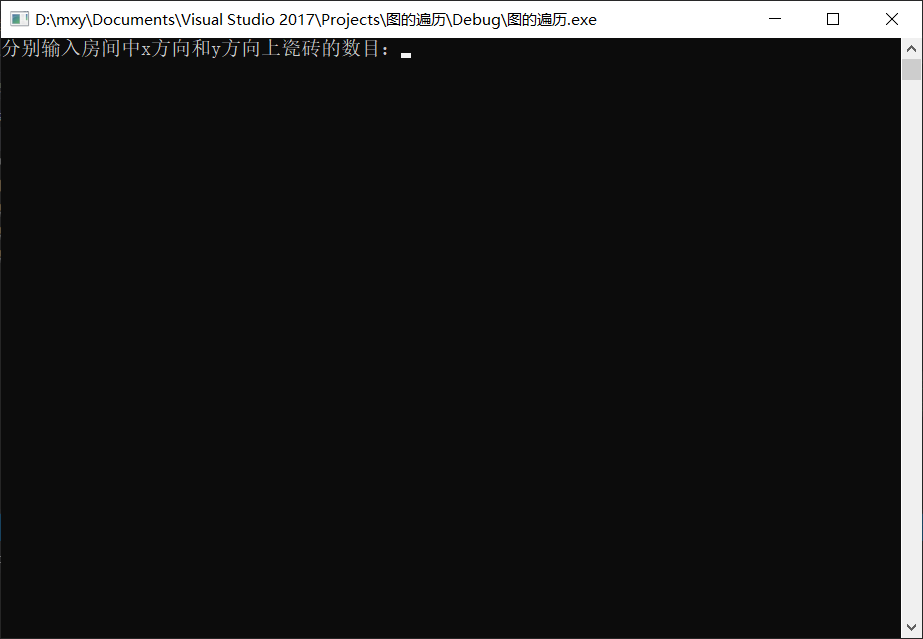
\* \* \* #

# \* # \*

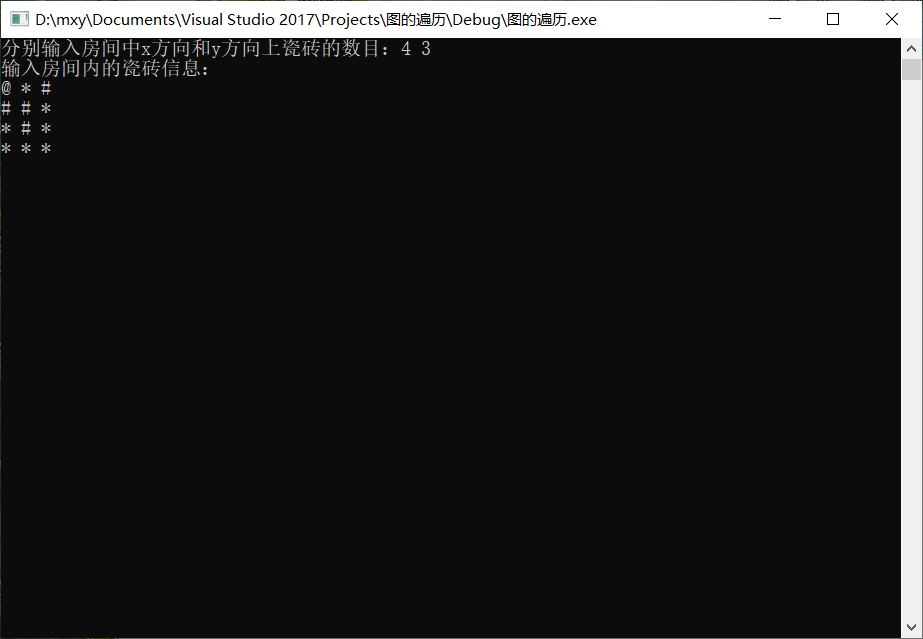
* 结果：7

5.2 测试过程

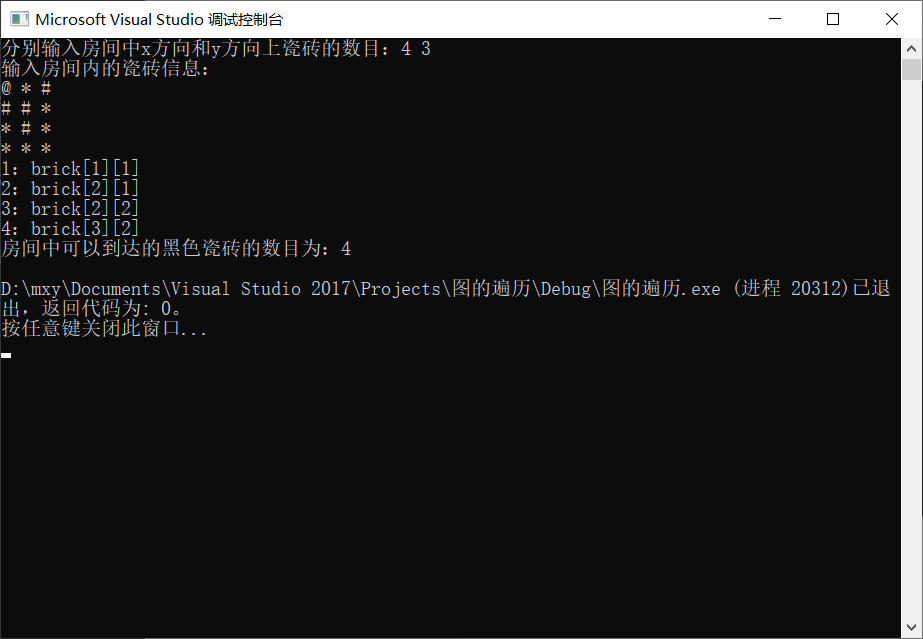
1. 主界面



1. 输入房间内瓷砖相关信息



1. 输出结果



六、源程序代码

1. #include"pch.h"
2. #include <iostream>
3. #include <vector>
4. **using** **namespace** std;
6. **void** DFS(vector<vector<**char**>> brick, vector<vector<**int**>>& visited, **int** X, **int** Y, **int**& count)
7. {
8. **if** (X >= 0 && X < brick.size() && Y >= 0 && Y < brick[0].size())   //检查坐标是否合法
9. {
10. **if** (visited[X][Y] != 1)     //如果没有被访问过
11. {
12. visited[X][Y] = 1;    //将当前瓷砖标记为已访问过
13. **if** (brick[X][Y] == '\*')   //如果当前访问的坐标是红色瓷砖则结束递归
14. **return**;
15. **else** **if** (brick[X][Y] == '#' || brick[X][Y] == '@') //count+1并输出当前瓷砖坐标
16. {
17. ++count;
18. cout << count << " " << "brick[" << X + 1 << "][" << Y + 1 << "]" << endl;
19. }
20. DFS(brick, visited, X - 1, Y, count);   //上
21. DFS(brick, visited, X, Y + 1, count);   //右
22. DFS(brick, visited, X + 1, Y, count);   //下
23. DFS(brick, visited, X, Y - 1, count);   //左
24. }
25. }
26. }
28. **int** main()
29. {
30. **int** W, H;
31. cout << "分别输入房间中x方向和y方向上瓷砖的数目：";
32. cin >> W >> H;
33. **if** (W == 0 && H == 0)
34. **return** 0;   //输入全为0则直接结束程序
35. vector<vector<**char**>> brick(W, vector<**char**>(H, '&'));   //定义瓷砖数组并用'&'初始化
36. vector<vector<**int**>> visited(W, vector<**int**>(H, 0));     //定义标记数组并初始化为全0
37. **int** X, Y;     //起始点的X，Y坐标
38. cout << "输入房间内的瓷砖信息：" << endl;
39. **for** (**int** i = 0; i < W; ++i)   //初始化瓷砖数组
40. {
41. **for** (**int** j = 0; j < H; ++j)
42. {
43. cin >> brick[i][j];
44. **if** (brick[i][j] == '@')  //记录初始位置
45. {
46. X = i;
47. Y = j;
48. }
49. }
50. }
51. **int** count = 0;  //记录符合条件的瓷砖数
52. DFS(brick, visited, X, Y, count);
53. cout << "房间中可以到达的黑色瓷砖的数目为：" << count << endl;
54. **return** 0;
55. }