目录

1.项目背景

迷宫只有两个门，一个门叫入口，另一个门叫出口。一个骑士骑马从入口进入迷宫，迷宫设置很多障碍，骑士需要在迷宫中寻找通路以到达出口。 该程序通过读入地图文件，输出该地图中的迷宫路径。

2.地图说明

地图文件采用txt文档保存，第一行参数为行数与列数，其余参数为迷宫地形描述。

符号表:

符号	意义
#	障碍物
0	可行走点
1	起点
2	终点
X	已经走过的点

示例:

```
7 7
#######
#1#000#
#0#0###
#000#0#
#0#000#
#0#0#2#
#######
```

3.需求分析

读取用户提交的迷宫地图，按照坐标系[1] 中坐标输出从 起点到终点的路径

4.设计思路

迷宫问题的求解过程可以采用 **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的方法。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。

用递归实现回溯法，函数的功能为判断该点能否走通，如果可以，则继续调用剩下可以行走的方向的判断函数（已经走过的点标注为`x`），直至找到出口标志`2`为止，如果递归调用结束后仍未找到路线，则输出“未找到路线”

5.核心代码说明

地图信息

坐标点结构

```
struct Point {
    int x;
    int y;
};
```

用于表示二维坐标点

地图结构

```
struct Map {
    char **map;
    int x;
    int y;
};
```

map指针用于指向地图信息头结点，采用二重指针的方式实现动态的二维数组，x,y则表示地图的长和宽

打印地图

```
//print the map with row and column number
void PrintMap(char **map,int a,int b) {
  cout << "迷宫地图:" << endl;
  for (int i = 0; i <= a; i++) {
    for (int j = 0; j <= b; j++) {
      if (i == 0 && j == 0) {
        cout << "\t";
      }
      else if (i == 0 && j != 0) {
        cout << j - 1 << "列" << "\t";
      }
      else if (i != 0 && j == 0) {
        cout << i - 1 << "行" << "\t";
      }
      else {
        cout << map[i - 1][j - 1] << "\t";
      }
    }
    cout << endl << endl;
  }
}
```

递归函数

```
//use the recursion type of DFS to find the path
void FindPath(Map t_map, int x, int y,stack<Point> &path) {
  if (t_map.map[x][y] == '2') {
    t_map.map[x][y] = 'X';
    Point thisPoint = { x,y };
    path.push(thisPoint);
    exitFinded = true;
    return;
  }
  if (exitFinded == true)
    return;

  //mark the point
  t_map.map[x][y] = 'X';
  Point thisPoint = { x,y };
  path.push(thisPoint);
  if (x > 0 && x < t_map.x - 1 && y > 0 && y < t_map.y - 1) {
    if (t_map.map[x + 1][y] == '0' || t_map.map[x + 1][y] == '2')
      FindPath(t_map, x + 1, y,path);
    if (t_map.map[x - 1][y] == '0'||t_map.map[x - 1][y] == '2')
      FindPath(t_map, x - 1, y,path);
    if (t_map.map[x][y + 1] == '0'|| t_map.map[x][y + 1] == '2')
      FindPath(t_map, x, y + 1,path);
    if (t_map.map[x][y - 1] == '0'|| t_map.map[x][y - 1] == '2')
      FindPath(t_map, x, y - 1,path);
  }

  //Delete the mark
```

```
  if (exitFinded == false) {
    t_map.map[x][y] = '0';
    path.pop();
  }
  return;
}
```

用全局变量`exitFinded`来作为找到路径与否的标签，采用一个栈来保存路径信息，每一次访问一个节点，先判断其是否为出口，若是，则结束递归并输出结果，如果不是出口，在`exitFinded`为true的情况下（即其他递归函数已经找到路径），同样结束递归，这样就可以避免已经找到路径后仍要继续未完成的递归调用浪费资源。随后判断该节点的剩余四个方向，如果没有走过且可以走，则递归调用判断函数。在该函数末尾，如果`exitFinded`仍为false，则说明此前的递归函数均为找到路径，将该点重置为0，同时将坐标从路径栈中弹出，避免对其余递归调用的影响。

函数接口说明

返回值类型	函数名	参数	属性	功能
void	PrintMap	(char **map,int a,int b)	非成员函数	输出地图
void	FindPath	(Map t_map, int x, int y,stack &path)	非成员函数	回溯法寻找路径

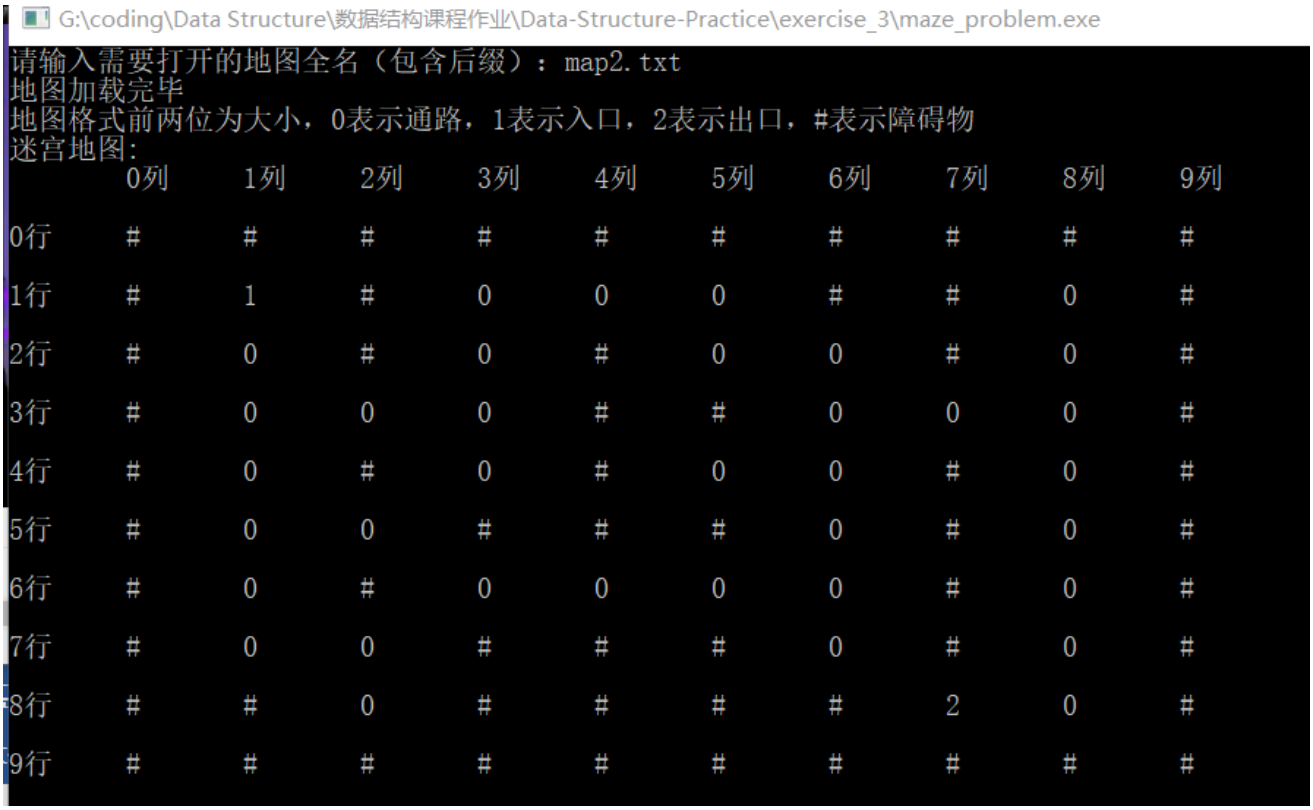
6.使用方法及函数功能演示

输入地图

按照地图说明中的要求新建txt文档并按格式输入地图信息后保存在exe文件 **相同目录下**

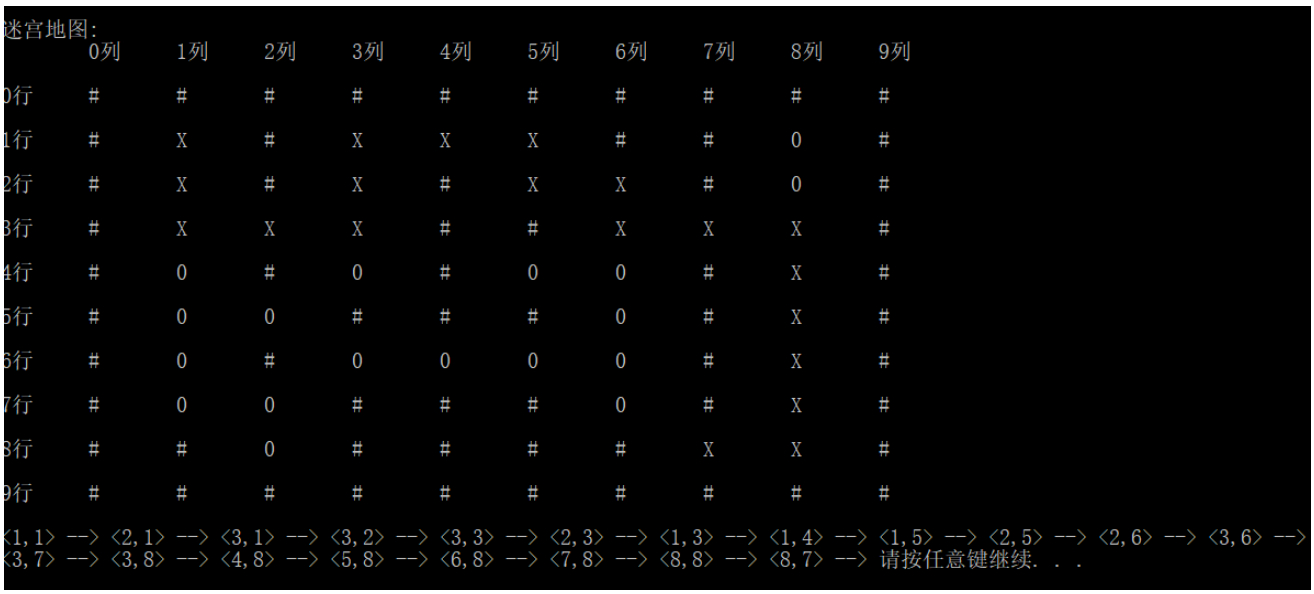
读取地图

运行`maze_problem.exe`，按照系统提示输入地图文件名称（包含后缀），随后若读取成功，系统将会打印地图，若读取失败，则会提示“没有找到该地图！”



输出结果

打印完地图后，如果为活迷宫（即有通路的迷宫），系统将会打印出迷宫路径并按坐标输出，如果是死迷宫（即没有通路），系统会提示“ 没有找到出口”



1. 以输入地图左下角为原点的常规平面直角坐标系↩