勇闯迷宫问题

1651573 刘客

功能分析

- 本执行文件,可针对自定义宽高但随机生成的迷宫寻找走出路径。
- **输入说明** 输入迷宫的宽高,以空格分开
- **输出说明** 会将迷宫图打印,将走的路径置为X,并输出一条路径序列。

  倘若不存在路径,则会输出无路可走。

编译说明

- 在windows平台下的.exe文件
- 在Linux平台下的out文件

一 . 设计

1. 数据结构设计

   > 由题意知,该题目实质即为迷宫问题,需要从入口不断搜索直到到达出口,题目并没有要求是否为最短路径,因此可以使用栈来模拟递归完成搜索过程。

2. 结构体设计

- ## Point 节点
  - 成员变量

成员名称	属性	类型	描述
x	public	int	x坐标
y	public	int	y坐标

  - 成员函数

函数名称	返回值类型	描述
Point	无	构造函数

- **Stack 结构体**
  - 成员变量

成员名称	属性	类型	描述
pVector	public	Point[]	用来存储坐标的栈
currentIndex	public	int	标记栈顶的索引

  - 成员函数

函数名称	返回值类型	描述
Stack	无	构造函数
pop	void	推出栈顶元素
push	void	推入一个元素到栈顶
isEmpty	bool	判断栈是否为空

二 . 函数实现

- 函数

函数名称	返回值类型	描述
drawMap	void	绘制迷宫
judgeDirection	Point	判断当前坐标可朝哪个方向前进
searchForExitPath	bool	搜索路径函数
Main	int	主函数

- drawMap函数

> 此函数用来画出迷宫地图
> Param width 是地图的宽度(即列数)
> Param height 是地图的高度(即行数)
> 仅支持width > 3 && height > 3
> 在地图生成过程中进行逻辑处理
> 当为边界时,全部填充”#”,来代表边界
> 当为入口时(1,1)节点,标志为‘S’,

> 当为出口时(width-1,height-1)节点,标志为'E',
> 在其他条件下,使用以当前时间作为随机数种子的函数,随机置符号,
> "0"(可通过)与"#"(不可通过) 的比率为4:1

```
void drawMap(char**map, bool**judge, int width, int height) {
    cout << "\n\n迷宫地图\n\n";
    for (int i = 0; i < width; i++) {
        cout << "\t" << i << "列";
    }
    cout << endl;
    for (int i = 0; i < height; i++) {
        cout << i << "行\t";
        for (int j = 0; j < width; j++) {
            if (i == 0 || i == height - 1) {
                map[i][j] = initArray[0];
                judge[i][j] = false;
                cout << map[i][j] << "\t";
                continue;
            }
            if (j == 0 || j == width - 1) {
                //输出地图元素
                map[i][j] = initArray[0];
                judge[i][j] = false;
                cout << map[i][j] << "\t";
                continue;
            }
            if (i == 1 && j == 1) {
                //设置起点;
                //输出地图元素
                map[i][j] = 'S';
                judge[i][j] = true;
                cout << map[i][j] << "\t";
                continue;
            }
            if (j == width - 2 && i == height - 2) {
                //设置出口
                //输出地图元素
                map[i][j] = 'E';
                judge[i][j] = true;
                cout << map[i][j] << "\t";
                continue;
            }
            map[i][j] = initArray[rand() % 5];
            map[i][j] == '#' ? judge[i][j] = false : judge[i][j] = true;
            cout << map[i][j] << "\t";
        }
        cout << "\n\n";
    }
}
```

- judgeDirection函数

> 初始化一个direction数组,里面存放四个可能的方向(向右,向下,向左,向上)
> 当获得一个可行的坐标时,将该坐标设为false(即标志为已经走过),同时返回。
> 若无可行坐标,则返回(-1,-1)

```
Point judgeDirection(Point start, bool **judge) {
    int x = start.x;
    int y = start.y;
    Point tempPoint;
    Point direction[] = { Point(1,0),Point(0,1),Point(-1,0),Point(0,-1) };

    for (int i = 0; i < 4; i++) {
        tempPoint.x = x + direction[i].x;
        tempPoint.y = y + direction[i].y;
        if (judge[tempPoint.x][tempPoint.y]) {
            judge[tempPoint.x][tempPoint.y] = false;
            return tempPoint;
        }
    }
    return Point(-1, -1); //代表无路可走
}
```

- searchForExitPath函数

> 首先初始化起点,然后起点入栈,然后进入循环
> 在循环中,对当前坐标执行judgeDirection函数,如果找到后继坐标,则将当前坐标入栈,
> 否则,将当前坐标赋值为栈的顶部元素,同时顶部元素出栈。
> 一直搜索下去,如果找到了终点则返回true。如果循环结束(即栈为空,表明无路可走),则返回false;

```
bool searchForExitPath(char **map, bool ** judge, Stack& countStack) {
    Point start(1, 1);

    countStack.push(start);
    Point temp = start;
    while (!countStack.isEmpty()) {
        temp = judgeDirection(temp, judge);
        if (temp.x != -1 && temp.y != -1) {
            //该条件表示找到了后继前进方向
            if (map[temp.x][temp.y] == 'E') {
                countStack.push(temp);
                return true;
            }
            countStack.push(temp);
        }
        else {
            countStack.pop(temp);
        }
    }
    return false;
}
```

- Main函数中的一些逻辑实现

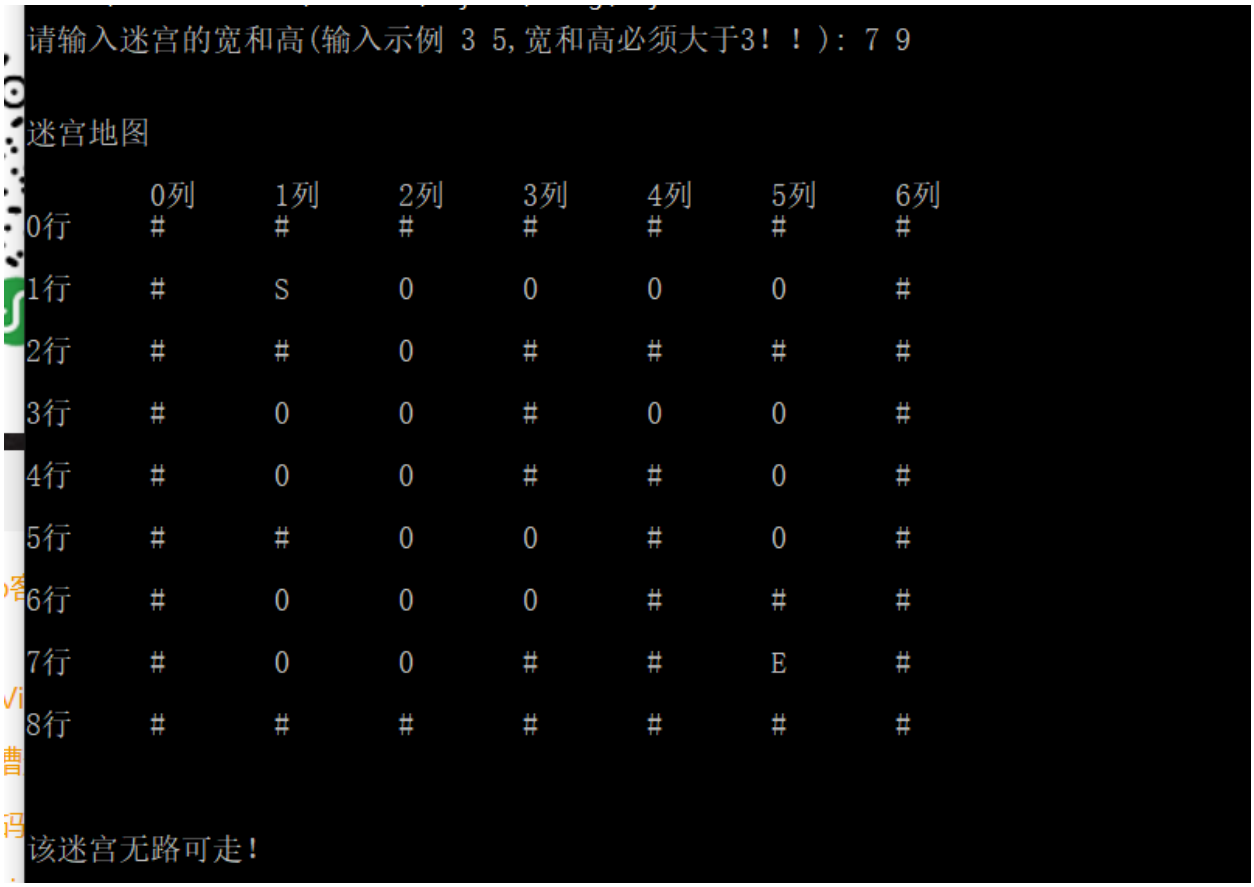
> 当搜索成功时,留在栈内的路径是从终点到起点的,此时新建一个栈变量,将路径逆序过来从而得到从起点到终点的路径

```
if (searchForExitPath(map, judgeV, countStack)) {
        Point temp;
        Stack tempStack;
        while (!countStack.isEmpty()) {
            countStack.pop(temp);
            //刷新地图信息;
            if (map[temp.x][temp.y] != 'S' && map[temp.x][temp.y] != 'E') {
                map[temp.x][temp.y] = 'X';
            }
            //另外设立一个栈,将逆序路线反正
            tempStack.push(temp);
        }
    ```(省略号,省略一些其他实现)
}
```

## 三. 程序运行效果

- 无路可走情况

> **迷宫障碍为以时间作为随机数种子的随机函数生成,所以不同时间条件下,同样的输入结果不一定相同!**


```

- 可以走通情况

```
请输入迷宫的宽和高(输入示例 3 5,宽和高必须大于3！！): 10 8

迷宫地图
        0列     1列     2列     3列     4列     5列     6列     7列     8列     9列
0行     #       #       #       #       #       #       #       #       #       #
1行     #       S       0       0       0       0       #       0       #       #
2行     #       #       #       0       0       #       0       0       0       #
3行     #       0       #       #       0       #       0       0       #       #
4行     #       0       0       0       0       #       #       0       0       #
5行     #       #       #       0       0       0       0       0       0       #
6行     #       0       0       0       0       0       0       0       E       #
7行     #       #       #       #       #       #       #       #       #       #

路径地图
        0列     1列     2列     3列     4列     5列     6列     7列     8列     9列
0行     #       #       #       #       #       #       #       #       #       #
1行     #       S       X       X       0       0       #       0       #       #
2行     #       #       #       X       X       #       0       0       0       #
3行     #       0       #       #       X       #       0       0       #       #
4行     #       0       0       0       X       #       #       0       0       #
5行     #       #       #       0       X       0       0       0       0       #
6行     #       0       0       0       X       X       X       X       E       #
7行     #       #       #       #       #       #       #       #       #       #
迷宫路径
<1,1>----->
<1,2>----->
<1,3>----->
<2,3>----->
<2,4>
----->
<3,4>----->
<4,4>----->
<5,4>----->
<6,4>----->
<6,5>
----->
<6,6>----->
<6,7>----->
<6,8>
```

四. 错误检测

- 输入宽高不合规范

```
D:\数据结构课程设计\课程项目3\Project1\Debug\Project1.exe
请输入迷宫的宽和高(输入示例 3 5,宽和高必须大于3！！): 1 -1
输入数据有误,请检查后重新输入:
```