数据结构项目三文档

同济大学 软件学院 15级2班 1552651 王依睿

使用说明

操作手册

- 运行程序后，请求用户输入地图规格。

- 请求用户输入地图。

```
请输入地图：  （0代表路，#代表墙，两字符以单个空格为间隔，行末以回车键结束）
# # # # # # #
# 0 # 0 0 0 #
# 0 # 0 # # #
# 0 0 0 # 0 #
# 0 # 0 0 0 #
# 0 # 0 # 0 #
# # # # # # #
```

- 请求用户输入入口坐标。

```
请输入入口坐标：  （从0开始计数）
行：  1
列：  1
```

- 请求用户输入出口坐标。

```
请输入出口坐标：  （从0开始计数）
行：  5
列：  5
```

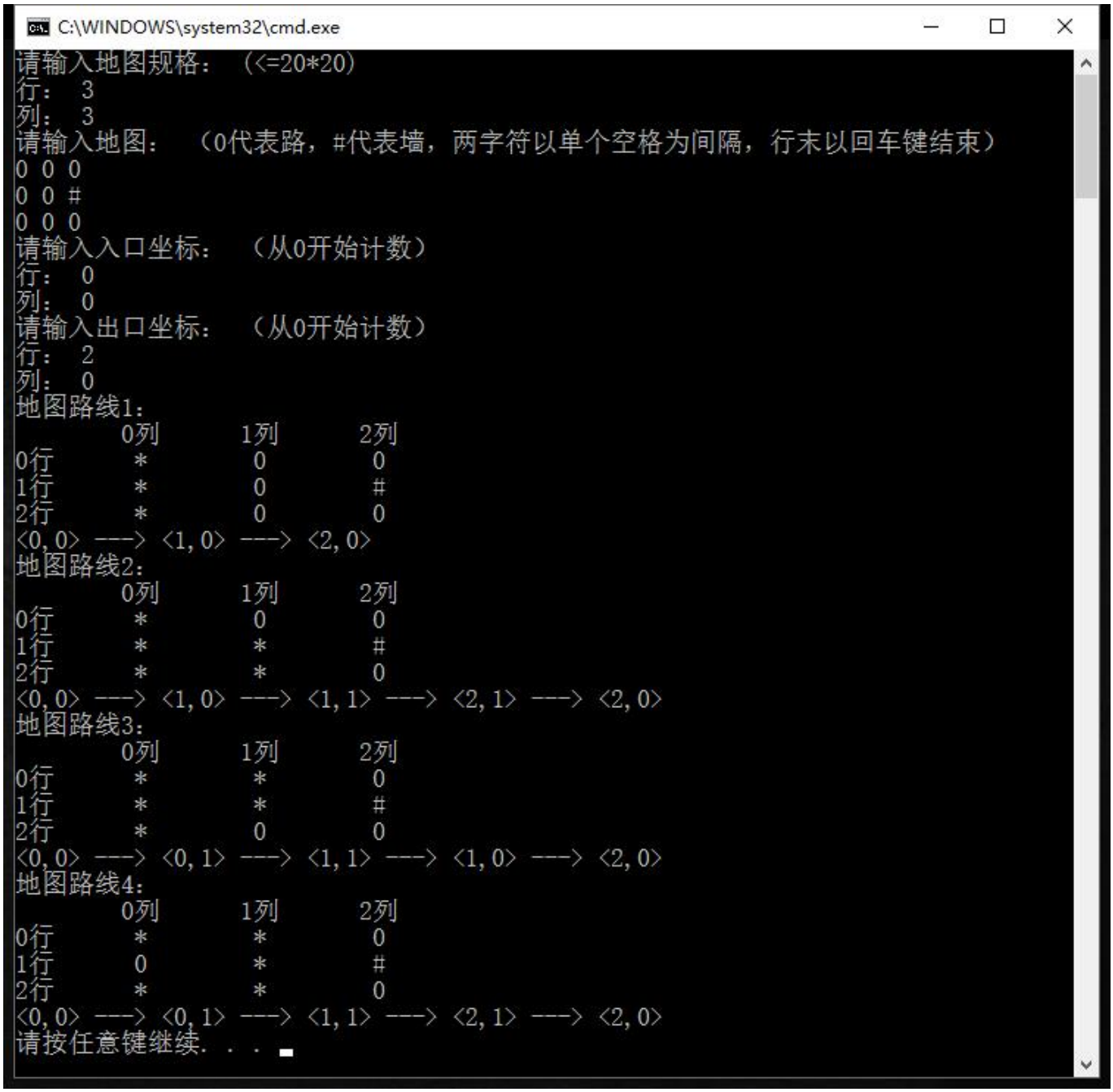
- 输出所有地图路线。

```
地图路线1：
        0列     1列     2列     3列     4列     5列     6列
0行      #       #       #       #       #       #       #
1行      #       *       #       0       0       0       #
2行      #       *       #       0       #       #       #
3行      #       *       *       *       #       0       #
4行      #       0       #       *       *       *       #
5行      #       0       #       0       #       *       #
6行      #       #       #       #       #       #       #
<1,1> ---> <2,1> ---> <3,1> ---> <3,2> ---> <3,3> ---> <4,3> ---> <4,4> ---> <4,
5> ---> <5,5>
```

- 结束程序。

注意事项

- 当输入的地图规格过大时，程序运行时间较长。
- 当地图规格达到一定大小时，由于深度优先搜索的函数递归调用会导致函数栈溢出。
- 若输入地图规格行/列的数字非法时，程序会输出“输入非法数字，请重新输入:”字样，并要求重新输入。
- 当输入地图未按照要求时，程序将无法正确执行。
- 当输入的入口坐标和出口坐标不符合逻辑要求时，程序会输出“此位置不能作为入口/出口，请重新输入：”字样，并要求重新输入。
- 程序可以实现输出地所有无返回的地图路线。如图：



概述

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。

项目功能要求

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。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。

程序设计目的

练习掌握DFS及回溯算法，实现一个根据输入地图产生所有无返回的地图路线输出的程序。

算法思路

- 利用深度优先搜索算法。
- 利用回溯遍历所有路线。
- 利用一个栈存放路线的位置坐标。

数据结构

- 利用一个栈存放路线的位置坐标。（手造）

文件目录

- 6_1552651_wangyirui.cpp（主文件）
- posstack.h（多叉树类）
- 6_1552651_wangyirui.exe（可执行文件）
- 6_1552651_wangyirui.pdf（项目文档）

函数接口

路线的位置坐标栈（PosStack）接口

成员函数名	公有/私有性	功能	参数	返回值类型
top()	公有	返回top_	/	栈顶指针int
PAIR route(int n)	公有	返回route_[n]，即第n+1步的位置坐标	n	第n+1步的位置坐标route_[n]
void push(PAIR pos)	公有	入栈操作	待入栈的位置坐标元素pos	/
void pop()	公有	出栈操作	/	/

主文件代码实现

```
#include"posstack_class.h"

enum Type { kMapLn, kMapCol, kLn, kCol };
enum LnMove { kLeft = -1, kStill = 0, kRight = 1 };
enum ColMove { kUp = -1, kDown = 1 };

int ln_move[] = { kLeft, kStill, kRight, kStill };
int col_move[] = { kStill, kUp, kStill, kDown };

char input_map[20][20];//存储玩家输入的地图
bool isvisited[20][20];//存储坐标是否访问过

int map_ln, map_col;//地图的行列
int entrance_ln, entrance_col;//入口坐标
int exit_ln, exit_col;//出口坐标
int route_count = 0;//通路数
PAIR pos;

PosStack posstack;

void DFS(int cur_ln, int cur_col) {

    //若访问的位置是终点，输出路线图和路线经过的坐标
    if (cur_ln == exit_ln && cur_col == exit_col) {

        ++route_count;

        //输出路线图
        cout << "地图路线" << route_count << "：" << endl;
```

```
        for (int j = 0; j < map_col; ++j) {
            if (j == 0)
                cout << "         " << j << "列";
            else
                cout << "       " << j << "列";
        }
        cout << endl;

        for (int i = 0; i < map_ln; ++i) {
            cout << i << "行";
            for (int j = 0; j < map_col; ++j) {
                if (j == 0)
                    cout << "       ";
                else
                    cout << "         ";
                if (isvisited[i][j])
                    cout << '*';
                else
                    cout << input_map[i][j];
            }
            cout << endl;
        }


        //输出路线经过的坐标
        for (int i = 0; i <= posstack.top(); ++i) {

            if (i == 0)
                cout << "<" << posstack.route(i).first << "," << posstack.route(i)
.second << ">";
            else
                cout << " ---> " << "<" << posstack.route(i).first << "," << posst
ack.route(i).second << ">";
        }
        cout << endl;

        return;
    }

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

        //访问下一位置
        int next_ln = cur_ln + ln_move[i];
        int next_col = cur_col + col_move[i];

        //若访问的位置不是终点，标记后进行递归
        if (next_ln > -1 && next_ln < map_ln && next_col > -1 && next_col < map_co
l) {
```

```
            if (input_map[next_ln][next_col] == '0'&&isvisited[next_ln][next_col]
== false) {

                isvisited[next_ln][next_col] = true;
                //压栈
                pos = make_pair(next_ln, next_col);
                posstack.push(pos);
                //递归
                DFS(next_ln, next_col);
                //回溯、出栈
                isvisited[next_ln][next_col] = false;
                posstack.pop();
            }
        }
    }
    return;
}

//读入函数，用第二个参数判断第一个参数，输出相应的输入提示文字
void readin(int &n, int type) {

    cin >> n;
    if(type == kMapLn)
    while (n <= 0) {
        cout << "输入非法数字，请重新输入：  " << endl
            << "行：  " << flush;
        cin >> n;
    }
    if (type == kMapCol)
        while (n <= 0) {
            cout << "输入非法数字，请重新输入：  " << endl
                << "列：  " << flush;
            cin >> n;
        }
    if (type == kLn)
        while (n < 0 || n >= map_ln) {
            cout << "输入非法数字，请重新输入：  " << endl
                << "行：  " << flush;
            cin >> n;
        }
    if (type == kCol)
        while (n < 0 || n >= map_col) {
            cout << "输入非法数字，请重新输入：  " << endl
                << "列：  " << flush;
            cin >> n;
        }
}
```

```
int main() {

    cout << "请输入地图规格：  (<=20*20)" << endl;

    cout << "行：  " << flush;
    readin(map_ln, kMapLn);
    cout << "列：  " << flush;
    readin(map_col, kMapCol);

    cout << "请输入地图：  （0代表路，#代表墙，两字符以单个空格为间隔，行末以回车键结束）" << e
ndl;

    for (int i = 0; i < map_ln; ++i)
        for (int j = 0; j < map_col; ++j) {
            cin >> input_map[i][j];
            getchar();
        }

    cout << "请输入入口坐标：  （从0开始计数）" << endl;

    cout << "行：  " << flush;
    readin(entrance_ln, kLn);

    cout << "列：  " << flush;
    readin(entrance_col, kCol);

    while (input_map[entrance_ln][entrance_col] != '0') {

        cout << "此位置不能作为入口，请重新输入：  " << endl;

        cout << "行：  " << flush;
        readin(entrance_ln, kLn);
        cout << "列：  " << flush;
        readin(entrance_col, kCol);
    }

    cout << "请输入出口坐标：  （从0开始计数）" << endl;

    cout << "行：  " << flush;
    readin(exit_ln, kLn);
    cout << "列：  " << flush;
    readin(exit_col, kCol);

    while (input_map[exit_ln][exit_col] != '0') {

        cout << "此位置不能作为出口，请重新输入：  " << endl;

        cout << "行：  " << flush;
```

```
        readin(exit_ln, kLn);
        cout << "列：  " << flush;
        readin(exit_ln, kCol);

    }

    //入口
    isvisited[entrance_ln][entrance_col] = true;
    pos = make_pair(entrance_ln, entrance_col);
    posstack.push(pos);
    DFS(entrance_ln, entrance_col);

    //若不存在通路数的输出
    if (route_count == 0) {
        cout << "不存在从出口到达入口的通路" << endl;
    }

    return 0;

}
```