**实验报告3 迷宫求解问题**

班级：09091501 姓名（学号）：段林（2015302280） 郭炜（2015302283）

1. **问题分析**
2. 以二维数组a[MAXSIZE][MAXSIZE]加上行数m,列数n构成matrix迷宫图，根据m,n在迷宫的四周构建一圈障碍，数组中以0代表障碍，以1代表通路但未走过，以2表示从起点到终点的路径，以3表示通路走过但是是死胡同。
3. 迷宫的m、n、迷宫内部构造以及起点和终点由用户决定。
4. 若设定的迷宫有通路以（x,y,z）的形式给出并在迷宫矩阵中进行修改成上述格式,x表示第x+1行，y表示第y+1列，z是1,2,3,4（1：东，2：西，3：南，4：北）。
5. 本程序只求出一条成功通路
6. 测试数据如P50课本3.4所示

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

1. 程序执行的命令为：
2. 创建迷宫
3. 求解迷宫
4. 输出迷宫的解
5. **设计框架**

依照题目要求，用结构体表示迷宫

typedef struct

{

int x, y;

}postype;

typedef struct

{

postype seat;

int di;

}position;

typedef struct

{

int m, n;

int a[MAXSIZE][MAXSIZE];

}matrix;

用栈存储找到的路径

typedef struct stacknode

{

position data;

stacknode \*next;

}\*stacklink;

typedef struct

{

stacklink top;

int length;

}stack;

函数调用关系图

pass

(判断节点是通路1还是障碍0)

footprint

（走过的路径将通路1变为通路2）

Nextpos

（根据当前位置和方向确定下一位置）

markprint

（将死胡同的标志记为通路3）

mazepath

（求解路径）

void mazepath(matrix &M, postype start, postype end)//求解路径算法

int pass(matrix M, postype curpos)//判断节点是通路1还是障碍0

void footprint(matrix &M, postype curpos)//走过的路径将通路1变为通路2

postype Nextpos(postype p, int a) //根据当前位置和方向确定下一位置

void markprint(matrix &M, postype curpos)//将死胡同的标志记为通路3

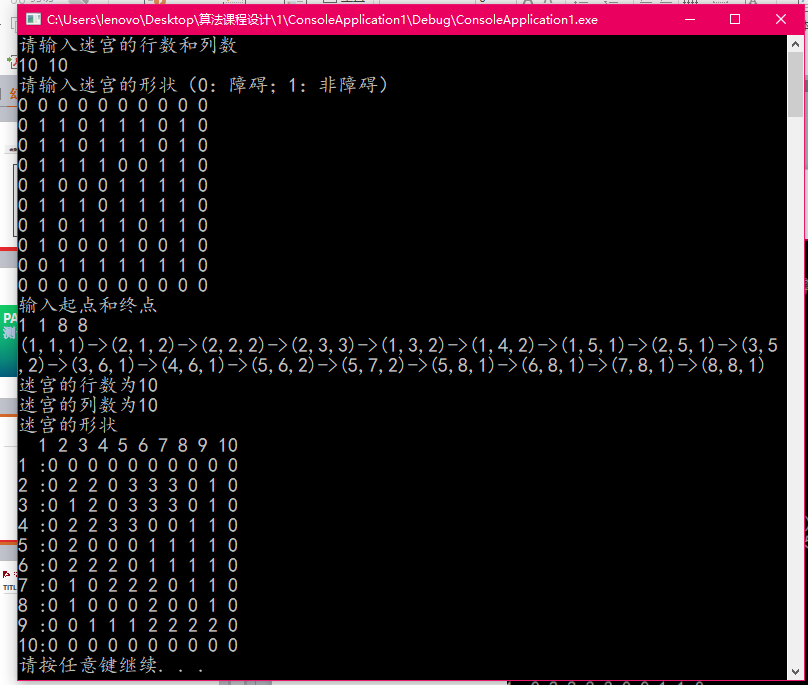
void initstack(stack &S)//初始化栈

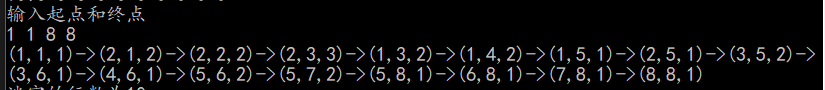
int stackempty(stack S)//判断栈是否为空

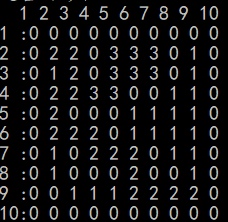
void push(stack &S, position e)//压栈

bool pop(stack &S, position &e)//出栈

1. **测试结果**

****





1. **调试及性能分析**
2. 本次作业比较简单，只有一个核心算法，即求解迷宫的路径，所以总的调试比较顺利。
3. 问题只出现了一次，即为postype Nextpos(postype p, int a) //根据当前位置和方向确定下一位置，开始是根据笛卡尔坐标系的理解进行坐标变换，求解出来的迷宫与最初想法相悖。后经思考，二维数组的行代表x，列代表y，得到正确结果。

int pass(matrix M, postype curpos)

{

if (M.a[curpos.y][curpos.x] == 1)

{

return 1;

}

else

{

return 0;

}

}

void footprint(matrix &M, postype curpos)

{

M.a[curpos.y][curpos.x] = 2;

}

void markprint(matrix &M, postype curpos)

{

M.a[curpos.y][curpos.x] = 3;

}

二维数组先由y确定再由x确定。

1. **特别之处**
2. 用栈存储路径，再由另一个栈接受路径，进而输出从起点到终点的过程。
3. 通过二维数组构造矩阵，进行输入输出。
4. **主要代码**

/\*

0 0 0 0 0 0 0 0 0 0

0 1 1 0 1 1 1 0 1 0

0 1 1 0 1 1 1 0 1 0

0 1 1 1 1 0 0 1 1 0

0 1 0 0 0 1 1 1 1 0

0 1 1 1 0 1 1 1 1 0

0 1 0 1 1 1 0 1 1 0

0 1 0 0 0 1 0 0 1 0

0 0 1 1 1 1 1 1 1 0

0 0 0 0 0 0 0 0 0 0

\*/

#include <iostream>

using namespace std;

#define MAXSIZE 100

typedef struct

{

int x, y;

}postype;

typedef struct

{

postype seat;

int di;

}position;

typedef struct

{

int m, n;

int a[MAXSIZE][MAXSIZE];

}matrix;

typedef struct stacknode

{

position data;

stacknode \*next;

}\*stacklink;

typedef struct

{

stacklink top;

int length;

}stack;

void initstack(stack &S)

{

S.top = NULL;

S.length = 0;

}

int stackempty(stack S)

{

if (S.top == NULL)

return 1;

else

return 0;

}

int pass(matrix M, postype curpos)

{

if (M.a[curpos.y][curpos.x] == 1)

return 1;

else

return 0;

}

void footprint(matrix &M, postype curpos)

{

M.a[curpos.y][curpos.x] = 2;

}

void markprint(matrix &M, postype curpos)

{

M.a[curpos.y][curpos.x] = 3;

}

postype Nextpos(postype p, int a)

{

postype q;

if (a == 1)

{

q.x = p.x + 1;

q.y = p.y;

}

else

{

if (a == 2)

{

q.x = p.x;

q.y = p.y + 1;

}

else

{

if (a == 3)

{

q.x = p.x - 1;

q.y = p.y;

}

else

{

if (a == 4)

{

q.x = p.x;

q.y = p.y - 1;

}

}

}

}

return q;

}

void push(stack &S, position e)

{

stacklink p;

p = new stacknode;

p->data = e;

p->next = S.top;

S.top = p;

++S.length;

}

bool pop(stack &S, position &e)

{

if (!S.top)

{

return false;

}

else

{

stacklink p;

e = S.top->data;

p = S.top;

S.top = S.top->next;

delete p;

--S.length;

return true;

}

}

void creatmatrix(matrix &M)

{

int i, j;

cout << "请输入迷宫的行数和列数" << endl;

cin >> M.m >> M.n;

cout << "请输入迷宫的形状（0：障碍；1：非障碍）" << endl;

for (i = 0; i < M.m; i++)

{

for (j = 0; j < M.n; j++)

{

cin >> M.a[i][j];

}

}

}

void mazepath(matrix &M, postype start, postype end)

{

stack S;

initstack(S);

position e;

postype curpos;

curpos = start;

do {

if (pass(M, curpos))

{

footprint(M, curpos);

e.di = 1, e.seat = curpos;

push(S, e);

if (curpos.x == end.x&&curpos.y == end.y)

{

break;

}

curpos = Nextpos(curpos, e.di);

}

else

{

if (!stackempty(S))

{

pop(S, e);

while (e.di == 4 && !stackempty(S))

{

markprint(M, e.seat);

pop(S, e);

}

if (e.di < 4)

{

e.di++;

push(S, e);

curpos = Nextpos(e.seat, e.di);

}

}

}

} while (!stackempty(S));

if (!stackempty(S))

{

stack A;

initstack(A);

while (!stackempty(S))

{

pop(S, e);

push(A, e);

}

while (!stackempty(A))

{

pop(A, e);

cout << "(" << e.seat.x << "," << e.seat.y << "," << e.di << ")" << "->";

}

cout << '\b' << '\b' << " " << " ";

cout << endl;

}

}

void output(matrix M)

{

int i, j;

cout << "迷宫的行数为" << M.m << endl;

cout << "迷宫的列数为" << M.n << endl;

cout << "迷宫的形状" << endl;

cout << " ";

for (i = 0; i < M.n; i++)

{

cout<< i+1 << " ";

}

cout << endl;

for (i = 0; i < M.m; i++)

{

if (i < 9)

{

cout << i + 1 << " :";

}

else

{

cout << i + 1 << ":";

}

for (j = 0; j < M.n; j++)

{

cout << M.a[i][j] << " ";

}

cout << endl;

}

}

int main()

{

matrix M;

postype start, end;

creatmatrix(M);

output(M);

cout << "输入起点和终点" << endl;

cin >> start.x >> start.y >> end.x >> end.y;

mazepath(M, start, end);

output(M);

system("pause");

}