**实验报告：****迷宫路径搜索**

**实验题目：**

迷宫路径搜索

**实验要求：**

1. 以[n+2][n+2]数组表示迷宫，0为可通行，1为障碍
2. 走迷宫
3. 打印迷宫路径

**编程思路：**

1. 以curpos结构体储存坐标，以STACK栈储存每一次走路的坐标
2. 每一次循环往四个方向探索可走路径，若可走，则移动，压栈；若不可走，则回退，弹栈
3. 当走到终点坐标或者空栈时停止循环，打印栈中的坐标元素，完成走迷宫

**核心代码：**

1. 数据结构定义

int Block[N+2][N+2]={//棋盘

1,1,1,1,1,1,1,1,1,1,

1,0,1,0,0,1,0,0,0,1,

1,0,1,0,1,0,0,1,1,1,

1,0,1,0,1,0,0,0,0,1,

1,0,1,0,0,0,0,1,0,1,

1,0,1,0,1,1,1,0,0,1,

1,0,1,0,0,1,0,0,0,1,

1,0,1,1,0,1,1,1,0,1,

1,0,0,0,0,1,0,0,0,1,

1,1,1,1,1,1,1,1,1,1

};

int Find[N+2][N+2]={0};//用于标记，防止走回头路

typedef struct{ //坐标

int x;

int y;

}CURPOS;

typedef struct{ //栈，储存每一步的坐标

CURPOS e[MAXSIZE];

int top;

}STACK;

1. 主要逻辑

int main(){

CURPOS p,end,start;//p为移动坐标，start、end为起点、终点坐标

STACK sta;//步骤栈

start.x=1;start.y=1;end.x=N;end.y=N;

Initial(p,start,sta);//初始化，输入起点坐标，同时初始化步骤栈

do{

Search(p,sta);//搜索每一个落脚点的位置，搜索到，则移动，没搜到，则回退

}while(Ifend(p,end)==0&&sta.top>=0);//p不在终点，且不空栈

StepPrint(sta);//打印步骤坐标

}

1. 移动

int IfMove(CURPOS p,int i,STACK sta){//搜索落脚点位置

if(i==0){//以i定义四个方向分别搜索

p.x++;

}

else if(i==1){

p.y++;

}

else if(i==2){

p.x--;

}

else if(i==3){

p.y--;

}

for(int j=0;j<=sta.top;j++){//与栈中步骤坐标重合，不能走回头路

if(p.x==sta.e[j].x&&p.y==sta.e[j].y) return 0;

}

if(Block[p.x][p.y]==0&&Find[p.x][p.y]==0) return 1;//没有碰到障碍或者重复的路径，可以走

else return 0;

}

int Move(CURPOS &p,int i,STACK &sta){//移动p,压栈

if(i==0){//以i定义四个方向分别移动

p.x++;

}

else if(i==1){

p.y++;

}

else if(i==2){

p.x--;

}

else if(i==3){

p.y--;

}

Push(sta,p);//压栈

return 1;

}

**实验结果：**

输入：

int Block[N+2][N+2]={//棋盘

1,1,1,1,1,1,1,1,1,1,

1,0,1,0,0,1,0,0,0,1,

1,0,1,0,1,0,0,1,1,1,

1,0,1,0,1,0,0,0,0,1,

1,0,1,0,0,0,0,1,0,1,

1,0,1,0,1,1,1,0,0,1,

1,0,1,0,0,1,0,0,0,1,

1,0,1,1,0,1,1,1,0,1,

1,0,0,0,0,1,0,0,0,1,

1,1,1,1,1,1,1,1,1,1

};

起点：(1,1)

终点：(8,8)

**结果**：

第0步:

(2,1)

第1步:

(3,1)

第2步:

(4,1)

第3步:

(5,1)

第4步:

(6,1)

第5步:

(7,1)

第6步:

(8,1)

第7步:

(8,2)

第8步:

(8,3)

第9步:

(8,4)

第10步:

(7,4)

第11步:

(6,4)

第12步:

(6,3)

第13步:

(5,3)

第14步:

(4,3)

第15步:

(4,4)

第16步:

(4,5)

第17步:

(4,6)

第18步:

(3,6)

第19步:

(3,7)

第20步:

(3,8)

第21步:

(4,8)

第22步:

(5,8)

第23步:

(6,8)

第24步:

(7,8)

第25步:

(8,8)

**源码全文：**

#include<stdio.h>

#define N 8//迷宫大小

#define MAXSIZE 64//栈的最大大小

int Block[N+2][N+2]={//棋盘

1,1,1,1,1,1,1,1,1,1,

1,0,1,0,0,1,0,0,0,1,

1,0,1,0,1,0,0,1,1,1,

1,0,1,0,1,0,0,0,0,1,

1,0,1,0,0,0,0,1,0,1,

1,0,1,0,1,1,1,0,0,1,

1,0,1,0,0,1,0,0,0,1,

1,0,1,1,0,1,1,1,0,1,

1,0,0,0,0,1,0,0,0,1,

1,1,1,1,1,1,1,1,1,1

};

int Find[N+2][N+2]={0};//用于标记，防止走回头路

typedef struct{//坐标

int x;

int y;

}CURPOS;

typedef struct{//栈，储存每一步的坐标

CURPOS e[MAXSIZE];

int top;

}STACK;

int Initial(CURPOS &p,CURPOS start,STACK &sta);

int Ifend(CURPOS &p,CURPOS end);

int Search(CURPOS &p,STACK &sta);

int IfMove(CURPOS p,int i,STACK sta);

int Move(CURPOS &p,int i,STACK &sta);

int Back(CURPOS &p,STACK &sta);

int Push(STACK &sta,CURPOS p);

CURPOS Pop(STACK &sta);

int StepPrint(STACK &sta);

int main(){

CURPOS p,end,start;//p为移动坐标，start、end为起点、终点坐标

STACK sta;//步骤栈

start.x=1;start.y=1;end.x=N;end.y=N;

Initial(p,start,sta);//初始化，输入起点坐标，同时初始化步骤栈

do{

Search(p,sta);//搜索每一个落脚点的位置，搜索到，则移动，没搜到，则回退

}while(Ifend(p,end)==0&&sta.top>=0);//p不在终点，且不空栈

StepPrint(sta);//打印步骤坐标

}

int Initial(CURPOS &p,CURPOS start,STACK &sta){//初始化，输入起点坐标，同时初始化步骤栈

p.x=start.x;p.y=start.y;

sta.top=(-1);

return 1;

}

int Ifend(CURPOS &p,CURPOS end){//输入终点坐标，判断p是否在终点

if(p.x==end.x&&p.y==end.y) return 1;

else return 0;

}

int Search(CURPOS &p,STACK &sta){//搜索每一个落脚点的位置，搜索到，则移动，没搜到，则回退

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

if(IfMove(p,i,sta)){

Move(p,i,sta);

return 1;

}

}

Back(p,sta);

return 0;

}

int IfMove(CURPOS p,int i,STACK sta){//搜索落脚点位置

if(i==0){//以i定义四个方向分别搜索

p.x++;

}

else if(i==1){

p.y++;

}

else if(i==2){

p.x--;

}

else if(i==3){

p.y--;

}

for(int j=0;j<=sta.top;j++){//与栈中步骤坐标重合，不能走回头路

if(p.x==sta.e[j].x&&p.y==sta.e[j].y) return 0;

}

if(Block[p.x][p.y]==0&&Find[p.x][p.y]==0) return 1;//没有碰到障碍或者重复的路径，可以走

else return 0;

}

int Move(CURPOS &p,int i,STACK &sta){//移动p,压栈

if(i==0){//以i定义四个方向分别移动

p.x++;

}

else if(i==1){

p.y++;

}

else if(i==2){

p.x--;

}

else if(i==3){

p.y--;

}

Push(sta,p);//压栈

return 1;

}

int Back(CURPOS &p,STACK &sta){//返回上一步，弹栈，标记已走过的路径

Find[p.x][p.y]=1;

Pop(sta);

p=sta.e[sta.top];

return 1;

}

int Push(STACK &sta,CURPOS p){//压栈

if(sta.top>=MAXSIZE) return 0;//合法性检查

sta.top++;

sta.e[sta.top]=p;

return 1;

}

CURPOS Pop(STACK &sta){//弹栈

if(sta.top<0) return sta.e[0];//合法性检查

sta.top--;

return sta.e[sta.top+1];

}

int StepPrint(STACK &sta){//打印步骤坐标

if(sta.top<0){

printf("无解！");

return 0;

}

for(int i=0;i<=sta.top;i++){

printf("第%d步:\n",i);

printf("(%d,%d)\n",sta.e[i].x,sta.e[i].y);

}

return 1;

}