《数据结构》实验报告三

|  |  |  |
| --- | --- | --- |
| 题目： 迷宫问题 | | 班级： 通信1601 |
| 学号： 29 |  | 姓名： 郑祖闯 |
| 日期： |  | 程序名： |

### 一、上机实验的问题和要求：

迷宫问题（书P50 3.2.4和《数据结构题集》P105 2.9）：以一个m×n的长方阵表示迷宫，0和1分别表示迷宫中的通路和障碍。设计一个程序，对任意设定的迷宫，求出一条从入口到出口的通路，或得出没有通路的结论。

利用链栈结构，然后编写一个求解迷宫的非递归程序。求得的通路以三元组(i,j,d)的形式输出，其中：(i,j)指示迷宫中的一个坐标，d表示走到下一坐标的方向。如：对于下列数据的迷宫，输出的一条通路为：(1，1，1)，（1，2，2），（2，2，2），（3，2，3），….。

**测试数据**：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

★鼓励图形化的输入输出界面设计

★设计递归算法，求出迷宫中所有可能的通路

### 二、程序设计的基本思想，原理和算法描述：

（包括程序的结构，数据结构，输入/输出设计，符号名说明等）

迷宫求解问题主要运用了堆栈的性质

求迷宫中一条从入口到出口的路径的算法描述：

do

{

若当前位置可通

则 {

将当前位置插入栈顶;

若该位置时出口位置，则结束 ；

否则切换当前位置为东邻方块为新的当前位置;

}

否则

{

若栈不空且栈顶位置尚有其他方向未经探索，

则设定新的当前位置为沿顺时针方向旋转的栈顶位置的下一相邻模块

若栈不空但栈顶位置的四周均不可通

则 {

删去栈顶位置;

若栈不空，则重新测试新的栈顶位置

直至找到一个可通的相邻模块或出栈至栈空 ；

}

}

}while(栈不空)

实现的函数为

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

若迷宫maze中从入口 start到出口 end的通道，则求得一条存放在栈中

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Status MazePath(int maze[12][12],SqStack &S, PosType start, PosType end)

{

PosType curpos;

int curstep;

SElemType e;

InitStack(S);

curpos = start; // 设定"当前位置"为"入口位置

curstep = 1; // 探索第一步

do

{

if (Pass(maze,curpos)) // 当前位置可通过，即是未曾走到过的通道块

{

Markfoot(maze,curpos); // 留下足迹

e.di =1;

e.ord = curstep;

e.seat= curpos;

Push(S,e); // 加入路径

if (curpos.row==end.row && curpos.line==end.line)

return OK; // 到达终点（出口）

curpos = NextPos(curpos, 1); // 下一位置是当前位置的东邻

curstep++; // 探索下一步

}

else // 当前位置不能通过

{

if (!StackEmpty(S))

{

Pop(S,e);

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

{

Markfoot(maze,e.seat); // 留下不能通过的标记，并退回一步

Pop(S,e);

}

if (e.di<4)

{

e.di++; // 换下一个方向探索

Push(S, e);

curpos = NextPos(e.seat, e.di); //当前位置设为新方向的 //相邻块

}

}

}

}while (!StackEmpty(S));

return ERROR;

}

围绕这个函数需要定义一些相关的函数操作，由以下函数实现

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数原型说明

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Status InitStack(SqStack &S); //创建一个空栈S

Status Push(SqStack &S,SElemType &a); //插入新元素a

Status Pop(SqStack &S,SElemType &a); //删除栈顶元素，a返回其值

Status StackEmpty(SqStack S); //检查是否空栈

Status MazePath(int maze[12][12],SqStack &S, PosType start, PosType end); //找通路

void Initmaze(int maze[12][12],int size); //初始化迷宫

void printmaze(int maze[12][12],int size); //显示迷宫

Status Pass(int maze[12][12],PosType CurPos); //判断当前位置是否可通

void Markfoot(int maze[12][12], PosType CurPos); //标记当前位置不可通

PosType NextPos(PosType CurPos, int Dir); //进入下一位置

void printpath(int maze[12][12],SqStack S,int size); //显示通路

算法中我用正方形迷宫，即行数等于列数。

迷宫的存储我用了一个整形二维数组表示,

int size; //正方形迷宫尺寸

int maze[12][12]; //存储迷宫内路径可通情况

二维数组存储的数字表示对应迷宫位置处可通与否，0表示可通，1表示不可通。

尺寸大小size可以设置，但是不能超过10，因为二维数组第一行，最后一行，第一列，最后一列一定要是不可通的，这是算法中用到的一个技巧。

迷宫内通道块位置变量类型定义为PosType

typedef struct

{

int row; //row表示“行”号

int line; //line表示“列”号

}PosType; //位置的元素类型

这样判断其可通与否的语句为

if (maze[CurPos.row][CurPos.line]==0)

1.迷宫的初始化

void Initmaze(int maze[12][12],int size); //初始化迷宫

迷宫的初始化有两种方法，一是随即生成，一是手动设置，由用户选择。随即生成的方法是程序生成随机数，除以2取余

maze[i][j]=rand()%2;

手动设置是用户输入0,1由程序读取

scanf("%d",&maze[i][j]);

2.显示迷宫

void printmaze(int maze[12][12],int size); //显示迷宫

只需要整齐打印出 0，1即可，可以看到很好的效果

显示初始化的迷宫。

3.显示通路

void printpath(int maze[12][12],SqStack S,int size); //显示通路

用到了一个技巧，只要是纳入堆栈的位置元素即为通路上的路径，将其迷宫对应位置

值变为2，

while(p!=S.top)

{

maze[p->seat.row][p->seat.line]=2; //标记为路径中的点

p++;

}

然后显示通路时只要等于2 的地方就打印一个0，否则打印空格。

if(maze[i][j]==2) printf("%c ",'0');

else printf(" ");

程序见第三部分。

4.进入下一位置

PosType NextPos(PosType CurPos, int Dir); //进入下一位置时按顺时针方向

//向下一位置探索

5.堆栈操作，包括创建，入栈，出栈，判空。

Status InitStack(SqStack &S); //创建一个空栈S

Status Push(SqStack &S,SElemType &a); //插入新元素a

Status Pop(SqStack &S,SElemType &a); //删除栈顶元素，a返回其值

Status StackEmpty(SqStack S); //检查是否空栈

程序见第三部分。

### 三、源程序及注释：

#include<stdlib.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

数据定义

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef enum { ERROR, OK } Status;

typedef struct

{

int row; //row表示“行”号

int line; //line表示“列”号

}PosType; //位置的元素类型

typedef struct

{

int ord; //该通道在路径上的“序号”

PosType seat; //通道块在迷宫中的“坐标位置”

int di; //从此通道走向下以通道块的“方向”

}SElemType; //栈的元素类型

typedef struct

{

SElemType \* base;

SElemType \* top;

int stacksize;

}SqStack;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数原型说明

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Status InitStack(SqStack &S); //创建一个空栈S

Status Push(SqStack &S,SElemType &a); //插入新元素a

Status Pop(SqStack &S,SElemType &a); //删除栈顶元素，a返回其值

Status StackEmpty(SqStack S); //检查是否空栈

Status MazePath(int maze[12][12],SqStack &S, PosType start, PosType end); //找通路

void Initmaze(int maze[12][12],int size); //初始化迷宫

void printmaze(int maze[12][12],int size); //显示迷宫

Status Pass(int maze[12][12],PosType CurPos); //判断当前位置是否可通

void Markfoot(int maze[12][12], PosType CurPos); //标记当前位置不可通

PosType NextPos(PosType CurPos, int Dir); //进入下一位置

void printpath(int maze[12][12],SqStack S,int size); //显示通路

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

主函数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main (void)

{

SqStack S;

int size; //正方形迷宫尺寸

int maze[12][12]; //存储迷宫内路径可通情况

for(int n=0;n<10;n++)

{

printf("创建一个正方形迷宫，请输入迷宫尺寸(注意不要大于10):\n"); //设置迷宫大小

scanf("%d",&size);if(size<1 || size>10){printf("输入错误!");return;}

Initmaze(maze,size); //初始化迷宫

printmaze(maze,size); //显示所创建的迷宫

PosType start,end; //设置入口和出口

printf("输入入口行坐标和列坐标:");

scanf("%d",&start.row);

scanf("%d",&start.line);

printf("输入出口行坐标和列坐标:");

scanf("%d",&end.row);

scanf("%d",&end.line);

if(MazePath(maze,S,start,end)) //若有通路，显示通路

printpath(maze,S,size);

else printf("找不到通路!\n\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

若迷宫maze中从入口 start到出口 end的通道，则求得一条存放在栈中

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Status MazePath(int maze[12][12],SqStack &S, PosType start, PosType end)

{

PosType curpos;

int curstep;

SElemType e;

InitStack(S);

curpos = start; // 设定"当前位置"为"入口位置

curstep = 1; // 探索第一步

do

{

if (Pass(maze,curpos)) // 当前位置可通过，即是未曾走到过的通道块

{

Markfoot(maze,curpos); // 留下足迹

e.di =1;

e.ord = curstep;

e.seat= curpos;

Push(S,e); // 加入路径

if (curpos.row==end.row && curpos.line==end.line)

return OK; // 到达终点（出口）

curpos = NextPos(curpos, 1); // 下一位置是当前位置的东邻

curstep++; // 探索下一步

}

else // 当前位置不能通过

{

if (!StackEmpty(S))

{

Pop(S,e);

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

{

Markfoot(maze,e.seat); // 留下不能通过的标记，并退回一步

Pop(S,e);

}

if (e.di<4)

{

e.di++; // 换下一个方向探索

Push(S, e);

curpos = NextPos(e.seat, e.di); // 当前位置设为新方向的相邻块

}

}

}

} while (!StackEmpty(S));

return ERROR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

初始化迷宫

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initmaze(int maze[12][12],int size)

{

char select;

printf("选择创建方式 A:自动生成 B:手动创建\n");

label:scanf("%c",&select);

if(select=='a'||select=='A') //自动生成

{

for(int i=0;i<size+2;i++)maze[0][i]=1;

for( i=1;i<size+1;i++)

{

maze[i][0]=1;

for(int j=1;j<size+1;j++)

maze[i][j]=rand()%2;

maze[i][size+1]=1;

}

for(i=0;i<size+2;i++)maze[size+1][i]=1;

}

else if (select=='b'||select=='B') //手动设置

{

printf("按行输入%d\*%d数据,0代表可通,1代表不可通(每行以Enter结束):\n",size,size);

for(int i=0;i<size+2;i++)maze[0][i]=1;

for( i=1;i<size+1;i++)

{

maze[i][0]=1;

for(int j=1;j<size+1;j++)

scanf("%d",&maze[i][j]);

maze[i][size+1]=1;

}

for(i=0;i<size+2;i++)maze[size+1][i]=1;

}

else if(select=='\n')goto label; //排除Enter键的影响

else printf("输入错误!");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

显示迷宫

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void printmaze(int maze[12][12],int size)

{

printf("\n\n");

printf("显示所建的迷宫(#表示外面的墙):\n");

for(int i=0;i<size+2;i++)printf("%c ",'#');printf("\n");

for(i=1;i<size+1;i++)

{

printf("%c ",'#');

for(int j=1;j<size+1;j++)

{

printf("%d ",maze[i][j]);

}

printf("%c",'#');

printf("\n");

}

for(i=0;i<size+2;i++)printf("%c ",'#');printf("\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

输出路径

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void printpath(int maze[12][12],SqStack S,int size)

{

printf("\n\n通路路径为:\n");

SElemType \* p=S.base;

while(p!=S.top)

{

maze[p->seat.row][p->seat.line]=2; //标记为路径中的点

p++;

}

for(int i=0;i<size+2;i++)printf("%c ",'#');printf("\n");

for(i=1;i<size+1;i++)

{

printf("%c ",'#');

for(int j=1;j<size+1;j++)

{

if(maze[i][j]==2) printf("%c ",'0');

else printf(" ");

}

printf("%c",'#');

printf("\n");

}

for(i=0;i<size+2;i++)printf("%c ",'#');printf("\n\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

判断当前位置是否可通

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Status Pass(int maze[12][12],PosType CurPos)

{

if (maze[CurPos.row][CurPos.line]==0)

return OK; // 如果当前位置是可以通过，返回1

else return ERROR; // 其它情况返回0

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

标记当前位置不可通

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Markfoot(int maze[12][12],PosType CurPos)

{

maze[CurPos.row][CurPos.line]=1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

进入下一位置

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PosType NextPos(PosType CurPos, int Dir)

{

PosType ReturnPos;

switch (Dir)

{

case 1: //下一模块为东临模块

ReturnPos.row=CurPos.row;

ReturnPos.line=CurPos.line+1;

break;

case 2: //下一模块为南临模块

ReturnPos.row=CurPos.row+1;

ReturnPos.line=CurPos.line;

break;

case 3: //下一模块为西临模块

ReturnPos.row=CurPos.row;

ReturnPos.line=CurPos.line-1;

break;

case 4: //下一模块为北临模块

ReturnPos.row=CurPos.row-1;

ReturnPos.line=CurPos.line;

break;

}

return ReturnPos;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

创建一个空栈S

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Status InitStack(SqStack &S)

{

S.base=(SElemType \*) malloc(100\*sizeof(SElemType));

if(!S.base) return ERROR;

S.top=S.base;

S.stacksize=100;

return OK;

}

### 四、运行输出结果：

自动生成迷宫和手动生成：





### 五、调试和运行程序过程中产生的问题及采取的措施：

在自动生成的迷宫中会可能产生通路很短或没有的情况，加入手动添加迷宫，可以自行设置路径以解决路径短或没有的问题。

2、当模块到了边界，则无法向下一位置探索，比如到了最东边时，再往东探索就出现了问题。解决方法是将二维数组扩大两行两列，将边界位置自动存为不可通。 比如在随即生成时：

for(int i=0;i<size+2;i++)maze[0][i]=1; //第一行为边界，不可通 for( i=1;i<size+1;i++)

{

maze[i][0]=1; //该行第一个为边界，不可通

for(int j=1;j<size+1;j++)

maze[i][j]=rand()%2;

maze[i][size+1]=1; //该行最后一个为边界，不可通

}

for(i=0;i<size+2;i++)maze[size+1][i]=1; //最后一行为边界，不可通 在实现图形化输出时遇到了很多问题，最后用了一个方法：

void printpath(int maze[12][12],SqStack S,int size); //显示通路

用到了一个技巧，只要是纳入堆栈的位置元素即为通路上的路径，将其迷宫对应位置 值变为2，

while(p!=S.top)

{

maze[p->seat.row][p->seat.line]=2; //标记为路径中的点

p++;

}

然后显示通路时只要等于2 的地方就打印一个0，否则打印空格，这样能看到不错的效果。 if(maze[i][j]==2) printf("%c ",'0'); else printf(" ");

### 六、对算法的程序的讨论、分析，改进设想，其它经验教训：

1、程序的设计很有兴趣，但整体实现起来比较困难，需要解决很多问题。程序还可以改进，因为通路可能不止一条，可以求出所有通路后，选取一条最短的路劲。

2、由实验结果可看出程序在选择路径时没有将相邻的路径做出适当的选择，因为只记录当和前一个位置，所以程序会将相邻的通路都会显示出来，并没能做到合理的比较，然后做取舍

### 七、对实验方式、组织、设备、题目的意见和建议：

实验内容很是有趣，但是实现的过程比较痛苦，不过做这样的实验内容很兴趣，老师的选题都是学生日常感兴趣的，应该继续保持。