**C语言迷宫工程代码报告**

组员：张倬豪，易鸣奇，陈诗翰。

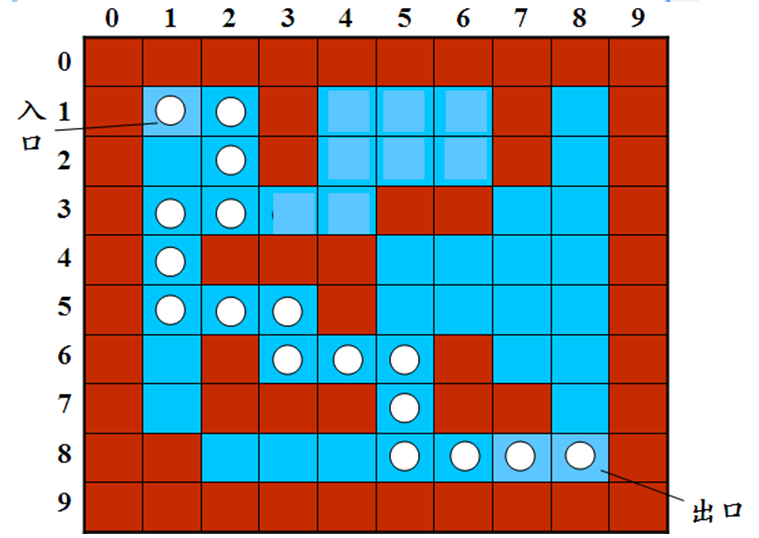
1. **总体功能描述**

迷宫问题是解决从布置了许多障碍的通道中寻找出路的问题。迷宫四周设为墙；无填充处，为可通处。设每个点有四个可通方向，分别为东、南、西、北(为了清晰，以下称“上下左右”)。左上角为入口。右下角为出口。迷宫有一个入口，一个出口。设计程序求解迷宫的一条通路。

要求：

1）按照模块化设计方法实现迷宫。

2）求解最短路径。

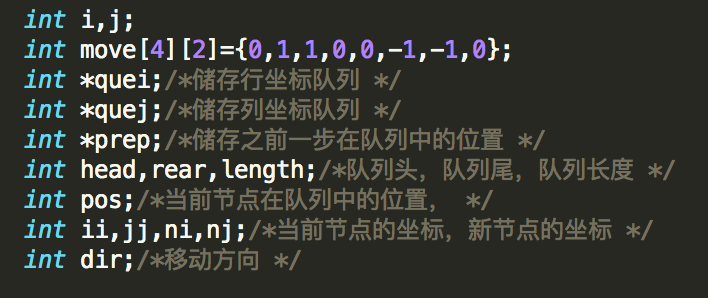


（此图中不是最短路径）

1. **关键设计思路方法**
2. 数据结构分析

../../../Users/Icarus/Desktop/屏幕快照%202016-04-22%20下

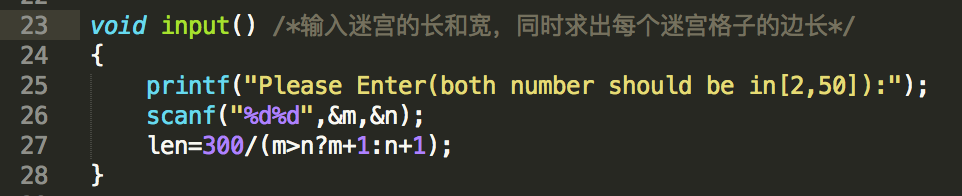
m，n为迷宫的长和宽，为贯穿全局的变量，maze是一个二维数组，最多50\*50，也是贯穿全局的变量，len为每个迷宫格子的边长。



此为这次工程最重要的数据结构：队列，为下面的广度优先搜索打下基础（若采取深度优先搜索和栈的数据结构，没有办法很好地找到最短路径）具体见图中注释。



函数initGraph，实现图形界面的初始化，较为简单机械化，不做过多描述。



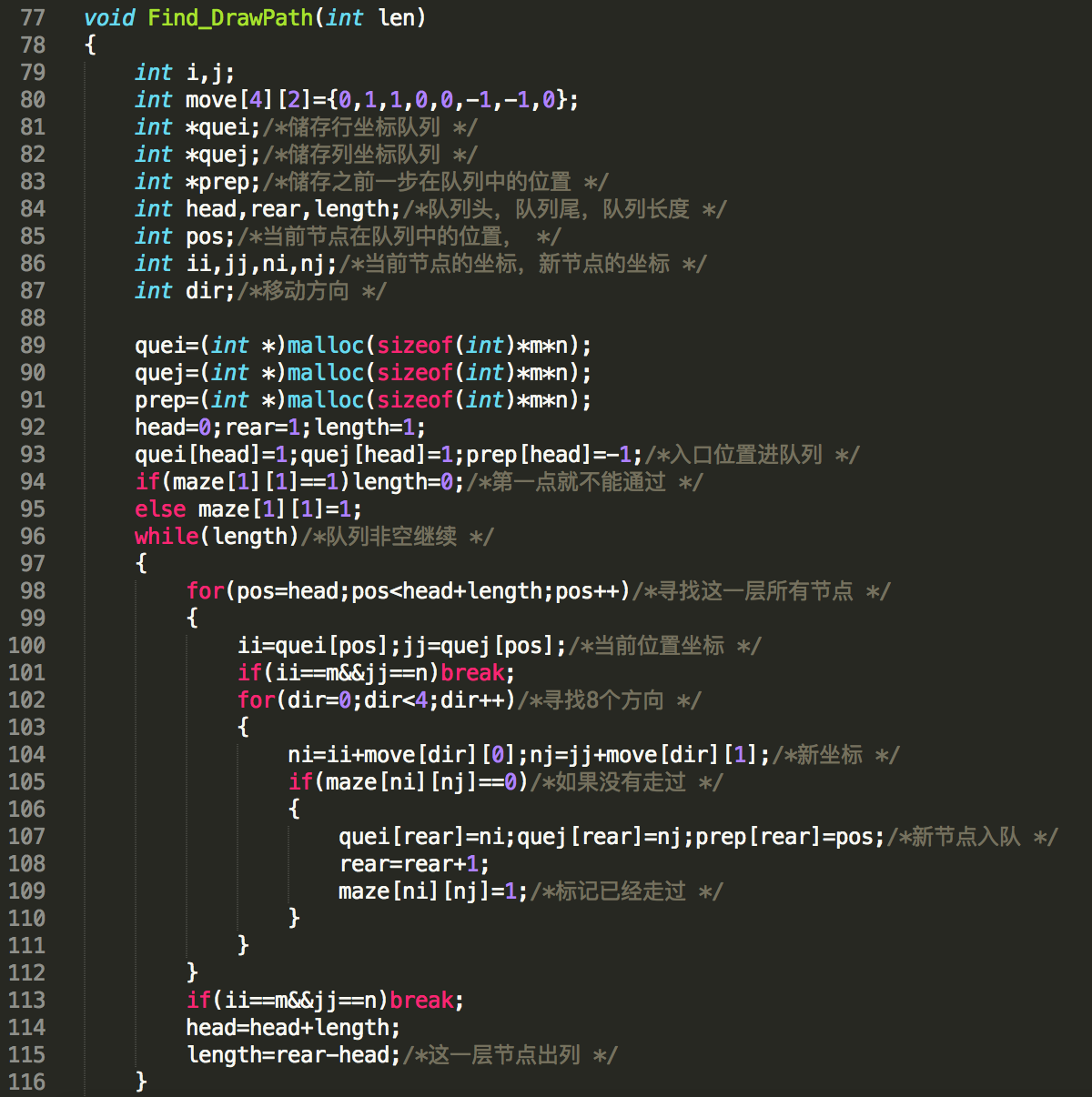
函数input，实现输入迷宫的长和宽，同时求出每个迷宫格子的边长。因为DosBox的屏幕大小有限，若每个迷宫格子大小一样会造成屏幕放不下或者太空的情况，不美观。

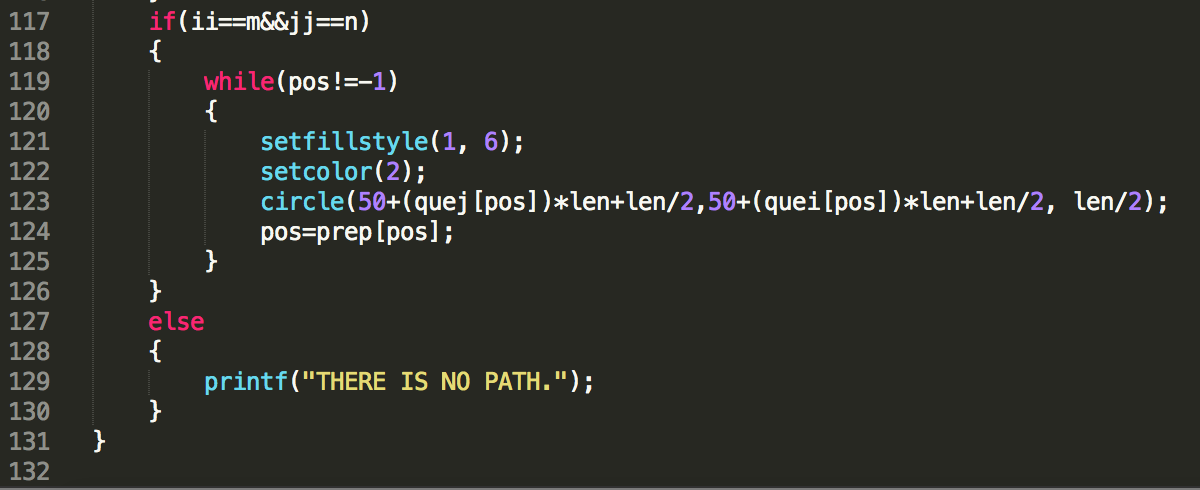


函数GenerateMaze，实现随机生成迷宫，使用rand()函数。并且使用if(x>700){maze[i][j]=1;}来控制矩阵中1的个数，太多1迷宫很容易走不通。这个函数是接下来有迷宫可走的前提基础。



函数DrawMaze，实现最初迷宫图形的绘制，先调用initGraph函数，进入图形界面，然后做一次二重循环，当maze[i][j]，也就是储存迷宫信息的二维数组，为0的时候，代表该地方可走，输出白色方块，为1的时候，输出红色方块，用于区别。

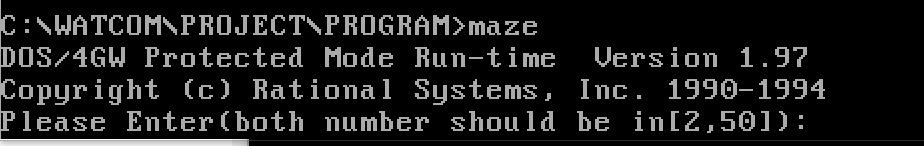




函数Find\_DrawPath，为这次工程最主要算法，找到并打印最短路径，采取广度优先算法。具体每一步作用图中已注释。如果没有路，则关闭图形界面，输出THERE IS NO PATH.

1. **测试报告**

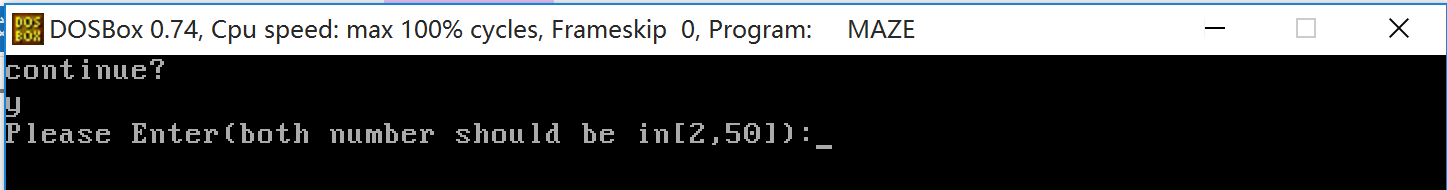
如图所示：



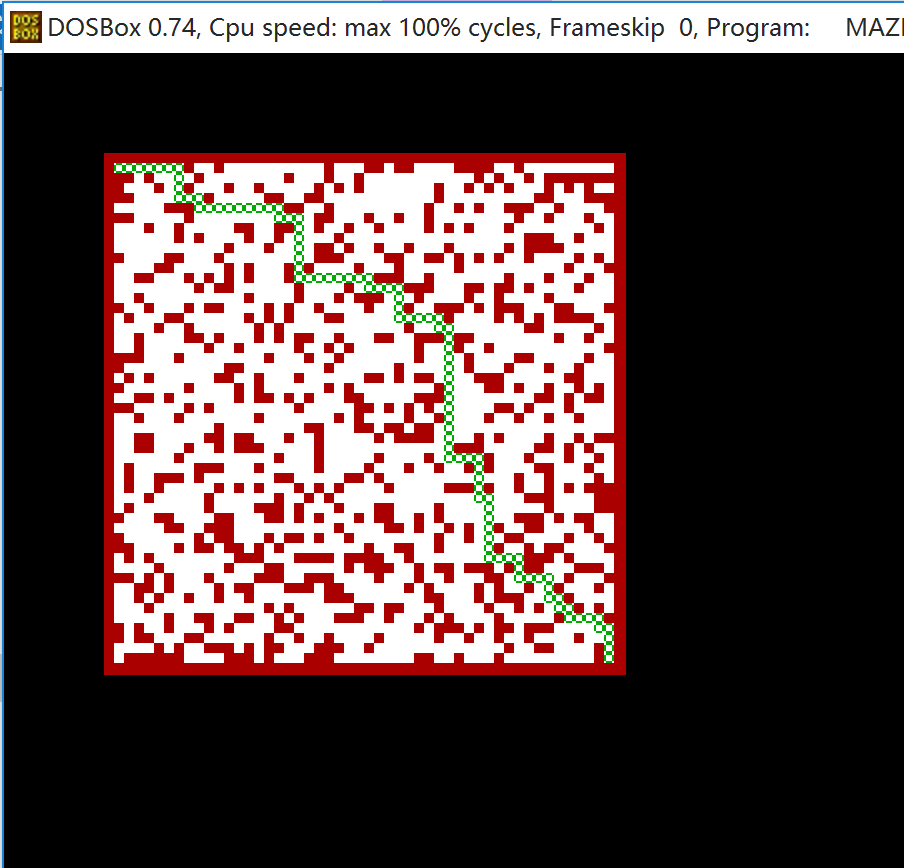
要求输出迷宫的长和宽。

****

**如：此为15\*15的迷宫**

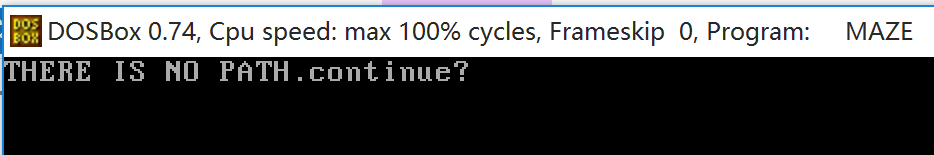
****

**然后按任意键继续，屏幕提示是否继续？按下y或者Y可继续生成迷宫，按下n或者N退出程序，按下其他键，提示输入错误，要求继续输入。**

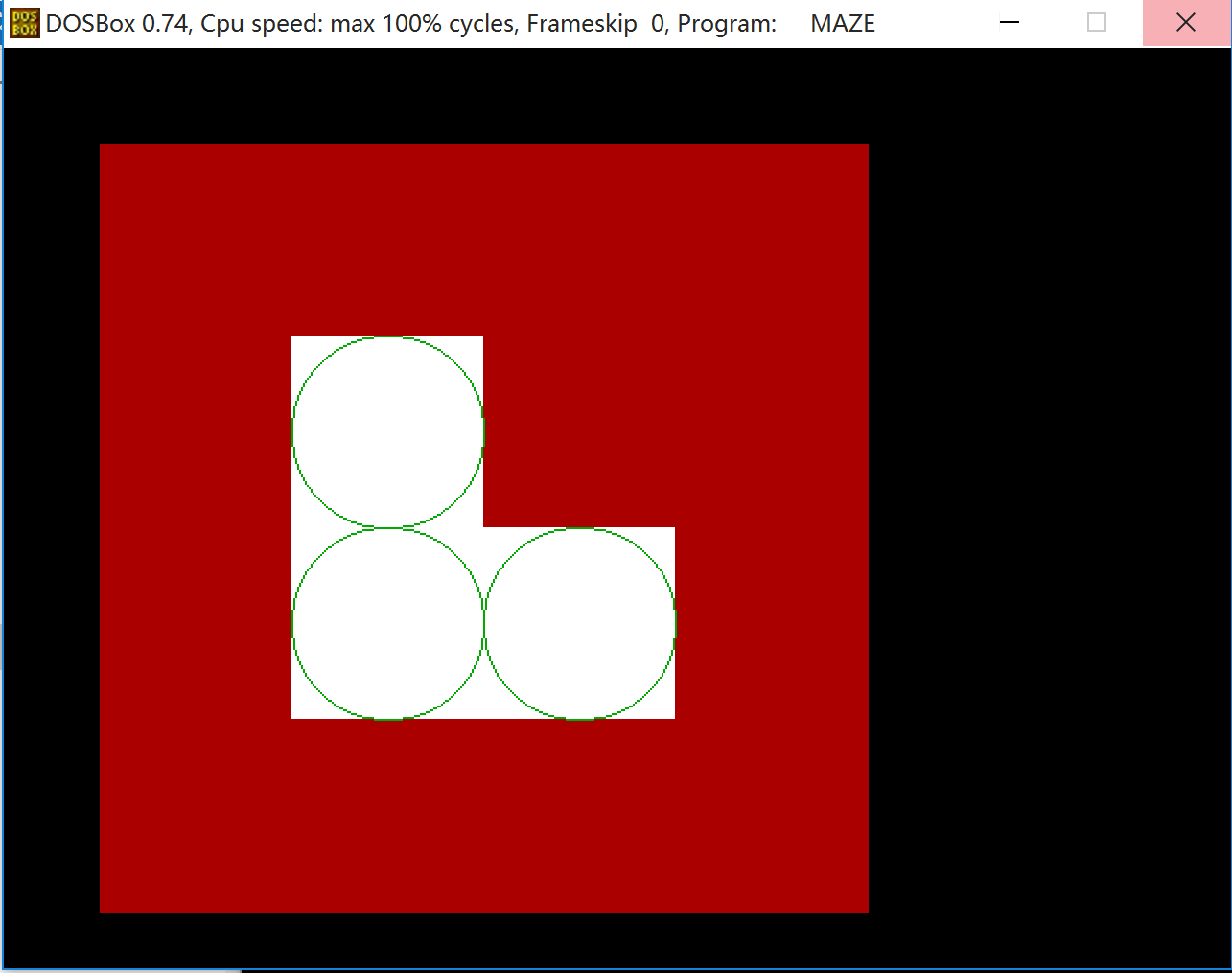
****

**此为50\*50的最大迷宫。**

****

****

**此为无路径的迷宫，按下任意键后提示没有路，询问是否继续**

****

**此为最小的情况2\*2。**

**4、开发体会**

感受到了真正做一个工程的困难，做完这一个工程不知道有多少个大大小小的bug，这么多个日日夜夜慢慢调试，最后慢慢成型，我们的不管是代码能力还是团队协作能力都得到了很大的提高。

**5、人员职责及互评**

指出小组相关人员的角色和职责，包括名称、对应人员姓名以及职责。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 角色 | 姓名 | 学号 | 职责 | 打分 |
| 组长 | *张倬豪* | 3150102418 | * 总体策划，负责协调活动 * 组内成员任务分工 * 最短路径算法等 * 测试 * 撰写《project报告》 | 3 |
| 组员 | *易鸣奇* | 3150102390 | * 信息收集 * 随机生成迷宫算法代码实现 * 测试 * 撰写《project报告》 | 2 |
| 组员 | *陈诗翰* | 3150103690 | * 信息收集 * 整合代码 * 测试 | 1 |

**6、附录**

源代码：

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include "graphics.h"

#define MAXNUM 55

int m,n,maze[MAXNUM][MAXNUM],len;

void initGraph();

void input();

void GenerateMaze();

void DrawMaze(int len);

void Find\_DrawPath(int len);

int main()

{

int flag1=1,flag2=1,flag3=1;

char ch;

while(flag1)

{

flag2=1;

input();

GenerateMaze();

DrawMaze(len);

Find\_DrawPath(len);

getch();

closegraph();

printf("continue?\n");

fflush(stdin);

scanf("%c",&ch);

while(flag2)

{

if(ch=='n'||ch=='N') {

flag1=0;

flag2=0;

}

else if(ch=='y'||ch=='Y') {

flag1=1;

flag2=0;

}

else

{

printf("enter again\n");

fflush(stdin);

scanf("%c",&ch);

}

}

}

return 0;

}

void initGraph()

{

/\* 要求自动检测 \*/

int gdriver = DETECT, gmode, errorcode;

/\* 初始化图形模式 \*/

initgraph(&gdriver, &gmode, "");

/\* 读取初始化结果 \*/

errorcode = graphresult();

if (errorcode != grOk) /\* 发生错误 \*/

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1); /\* 具有错误代码的返回 \*/

}

}

void input() /\*输入迷宫的长和宽，同时求出每个迷宫格子的边长\*/

{

printf("Please Enter(both number should be in[2,50]):");

scanf("%d%d",&m,&n);

len=300/(m>n?m+1:n+1);

}

void GenerateMaze()

{

int i,j,x;

for(i=0;i<=m+1;i++){

for(j=0;j<=n+1;j++){

if(i==0||j==0||i==m+1||j==n+1)

maze[i][j]=1;

else

{

x=rand()%1000;

if(x>700){maze[i][j]=1;} /\*控制矩阵中1的个数，太多1迷宫很容易走不通 \*/

else{maze[i][j]=0;}

}

if (i==m&&j==n) maze[i][j]=0;

if (i==1&&j==1) maze[i][j]=0;

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

}

printf("\n");

}

}

void DrawMaze(int len)

{

int i,j;

initGraph(); /\*初始化图形模式\*/

cleardevice();

setbkcolor(10);

setcolor(4);

setfillstyle(1, 4);

for (i = 0; i < m+2; i++)

for (j = 0; j < n+2; j++)

{

if (1 == maze[i][j])

{

setfillstyle(1, 4);

/\*设定每个格子宽为20个单位，高为20个单位\*/

bar(50+j\*len,50+i\*len,50+len+j\*len,50+len+i\*len);/\*sleep(1);\*/

}

else

{

setfillstyle(1, 15);

bar(50+j\*len,50+i\*len,50+len+j\*len,50+len+i\*len);/\*sleep(1);\*/

}

}

}

void Find\_DrawPath(int len)

{

int i,j;

int move[4][2]={0,1,1,0,0,-1,-1,0};

int \*quei;/\*储存行坐标队列 \*/

int \*quej;/\*储存列坐标队列 \*/

int \*prep;/\*储存之前一步在队列中的位置 \*/

int head,rear,length;/\*队列头，队列尾，队列长度 \*/

int pos;/\*当前节点在队列中的位置， \*/

int ii,jj,ni,nj;/\*当前节点的坐标，新节点的坐标 \*/

int dir;/\*移动方向 \*/

quei=(int \*)malloc(sizeof(int)\*m\*n);

quej=(int \*)malloc(sizeof(int)\*m\*n);

prep=(int \*)malloc(sizeof(int)\*m\*n);

head=0;rear=1;length=1;

quei[head]=1;quej[head]=1;prep[head]=-1;/\*入口位置进队列 \*/

if(maze[1][1]==1)length=0;/\*第一点就不能通过 \*/

else maze[1][1]=1;

while(length)/\*队列非空继续 \*/

{

for(pos=head;pos<head+length;pos++)/\*寻找这一层所有节点 \*/

{

ii=quei[pos];jj=quej[pos];/\*当前位置坐标 \*/

if(ii==m&&jj==n)break;

for(dir=0;dir<4;dir++)/\*寻找8个方向 \*/

{

ni=ii+move[dir][0];nj=jj+move[dir][1];/\*新坐标 \*/

if(maze[ni][nj]==0)/\*如果没有走过 \*/

{

quei[rear]=ni;quej[rear]=nj;prep[rear]=pos;/\*新节点入队 \*/

rear=rear+1;

maze[ni][nj]=1;/\*标记已经走过 \*/

}

}

}

if(ii==m&&jj==n)break;

head=head+length;

length=rear-head;/\*这一层节点出列 \*/

}

if(ii==m&&jj==n)

{

while(pos!=-1)

{

setfillstyle(1, 6);

setcolor(2);

circle(50+(quej[pos])\*len+len/2,50+(quei[pos])\*len+len/2, len/2);

pos=prep[pos];

}

}

else

{

printf("THERE IS NO PATH.");

}

}

讨论聊天记录

