实验二 迷宫问题

班级:软件工程一班 姓名:秦源 学号:1525161007

1. 需求分析

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。首先实现一个以链表作存储结构的栈类型，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。

1. 概要设计

计算机解迷宫通常用的是“穷举求解”方法，即从入口出发，顺着某一个方向进行探索，若能走通，则继续往前进；否则沿着原路退回，换一个方向继续探索，直至出口位置，求得一条通路。假如所有可能的通路都探索到而未能到达出口，则所设定的迷宫没有通路。

可以二维数组存储迷宫数据，通常设定入口点的下标为（1，1），出口点的下标为（m,n）。为处理方便起见，可在迷宫的四周加一圈障碍。对于迷宫中任一位置，均可约定有东、南、西、北四个方向可通。

三、 详细设计

//

// main.cpp

// Maze

//

// Created by 秦源.

// Copyright © 2016年 QinYuan. All rights reserved.

//

#include <iostream>

using namespace std;

const int STACK\_INIT\_SIZE=64 ;//定义堆栈的初始大小

const int STACKINCREMENT=32;//栈满后扩充大小

struct PosType{ //定义点的坐标

int x;//点的横坐标

int y;//点的纵坐标

int d=1;//走到下一坐标的方向（默认向上为1，向右为2，向下为3，向左为4）

};

struct SElemType{

int ord; //通道块在路径上的“序号”

PosType seat;//通道块在迷宫中的“坐标位置”

int di;//从此通道块走向下一通道块的“方向”

};

struct MazeType{//定义储存迷宫的数组（最外侧需要封闭上）

int L\_x;//实际迷宫横为8

int L\_y;//实际迷宫纵为9

int map[10][11];//将迷宫包围后横为10，纵为11

};

MazeType InitMazeMap(){//初始化迷宫

MazeType M;

M.L\_x=8;

M.L\_y=9;

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

for(int y=0;y<11;y++)

M.map[x][y]=0;

//包围迷宫外侧

for(int x=0;x<10;x++){

M.map[x][0]=1;

M.map[x][10]=1;

}

for(int y=1;y<11;y++){

M.map[0][y]=1;

M.map[9][y]=1;

}

//绘制迷宫

M.map[3][1]=1;

M.map[7][1]=1;

M.map[3][2]=1;

M.map[7][2]=1;

M.map[5][3]=1;

M.map[6][3]=1;

M.map[8][3]=1;

M.map[2][4]=1;

M.map[3][4]=1;

M.map[4][4]=1;

M.map[7][4]=1;

M.map[4][5]=1;

M.map[2][6]=1;

M.map[6][6]=1;

M.map[8][6]=1;

M.map[2][7]=1;

M.map[3][7]=1;

M.map[4][7]=1;

M.map[5][7]=1;

M.map[8][7]=1;

M.map[1][8]=1;

M.map[2][8]=1;

M.map[6][8]=1;

M.map[8][8]=1;

M.map[1][9]=1;

M.map[2][9]=1;

//输出迷宫

cout<<"迷宫实例：（在迷宫基础上，最外层用1包围）"<<endl;

for(int y=0;y<11;y++){

for(int x=0;x<10;x++){

cout<< M.map[x][y]<<" ";

}

cout<<endl;

}

return M;

}

struct SqStack{ // 定义顺序栈结构体

SElemType \* base;

SElemType \* top;

int stacksize;

};

SqStack InitStack(){// 初始化顺序栈

SqStack S;

S.base=new SElemType[STACK\_INIT\_SIZE];

if(!S.base)

exit(1);

S.top=S.base;

S.stacksize=STACK\_INIT\_SIZE;

return S;

}

void Push(SqStack & S,SElemType &e){//入栈

//栈满情况下扩充栈的大小

if(S.top-S.base>=S.stacksize){

SElemType \* temp;

temp=new SElemType[S.stacksize+STACKINCREMENT];

if(!temp)

exit(1);

for(int i=0;i<S.stacksize;i++)

temp[i]=S.base[i];

delete []S.base;

S.base=temp;

S.top=S.base+S.stacksize;

S.stacksize+=STACKINCREMENT;

}

//入栈

\*S.top++=e;

}

bool StackEmpty(SqStack &S){//判断栈是否为空

if(S.base==S.top)

return true;

return false;

}

SElemType Pop(SqStack & S){//出栈

//判断是否栈为空

if(S.base==S.top)

exit(1);

//出栈

SElemType e;

e=\*--S.top;

return e;

}

bool Pass(MazeType & maze,PosType pos){ //判断当前点是否可通

if(maze.map[pos.x][pos.y]==0)

return true;

return false;

}

PosType NextPos(PosType now,int w){ //根据上一个点的信息，定义下一个点的位置

PosType next;

switch(w){

case 1:next.x=now.x;next.y=now.y-1;break;

case 2:next.x=now.x+1;next.y=now.y;break;

case 3:next.x=now.x;next.y=now.y+1;break;

case 4:next.x=now.x-1;next.y=now.y;break;

default:cout<<"Error!";break;

}

return next;

}

void printPos(PosType e){//输出点坐标

cout<<"("<<e.x<<","<<e.y<<","<<e.d<<")"<<endl;

}

void printStack(SqStack s1){//输出结果路径

cout<<"已找到路径："<<endl;

//将栈转置

SqStack s2=InitStack();

while(!StackEmpty(s1)){

SElemType e1= Pop(s1);

Push(s2,e1);

}

//输出结果

while(!StackEmpty(s2)){

SElemType e2= Pop(s2);

cout<<"第";

printf("%3d",e2.ord);

cout<<" 步"<<"：";

e2.seat.d=e2.di;

printPos(e2.seat);

}

}

bool MazePath(MazeType &maze,PosType start,PosType end){//迷宫解法

SqStack S=InitStack();

PosType curPos=start;//设”当前位置“为”入口位置“

int curstep=1;//探索第一步

do{

if(Pass(maze,curPos)){//当前位置可以通过

SElemType e;

e.ord=curstep;

e.seat=curPos;

e.di=1;

Push(S, e);

maze.map[curPos.x][curPos.y]=1;//将走过的点标记为不可通

if(curPos.x==end.x&&curPos.y==end.y){//找到出路

printStack(S);

return true;

}

curPos=NextPos(curPos,e.di);

curstep++;

}else{

if(!StackEmpty(S)){

SElemType e= Pop(S);

curstep--;

while(e.di==4&&!StackEmpty(S)){ //遇到死胡同时将当前点提到可以改变方向的通点

e= Pop(S);

curstep--;

}

if(e.di<4){//当下个点不可通时，改变方向寻找其他通路

e.di++;

Push(S, e);

curstep++;

curPos=NextPos(e.seat, e.di);

}

}

}

}while(!StackEmpty(S));

return false;

}

int main(int argc, const char \* argv[]) {

//实例化堆栈

MazeType M;

M=InitMazeMap();

//确认迷宫入口

PosType start;

start.x=1;

start.y=1;

//确认迷宫出口

PosType end;

end.x=8;

end.y=9;

//迷宫求解计算

MazePath(M, start, end);

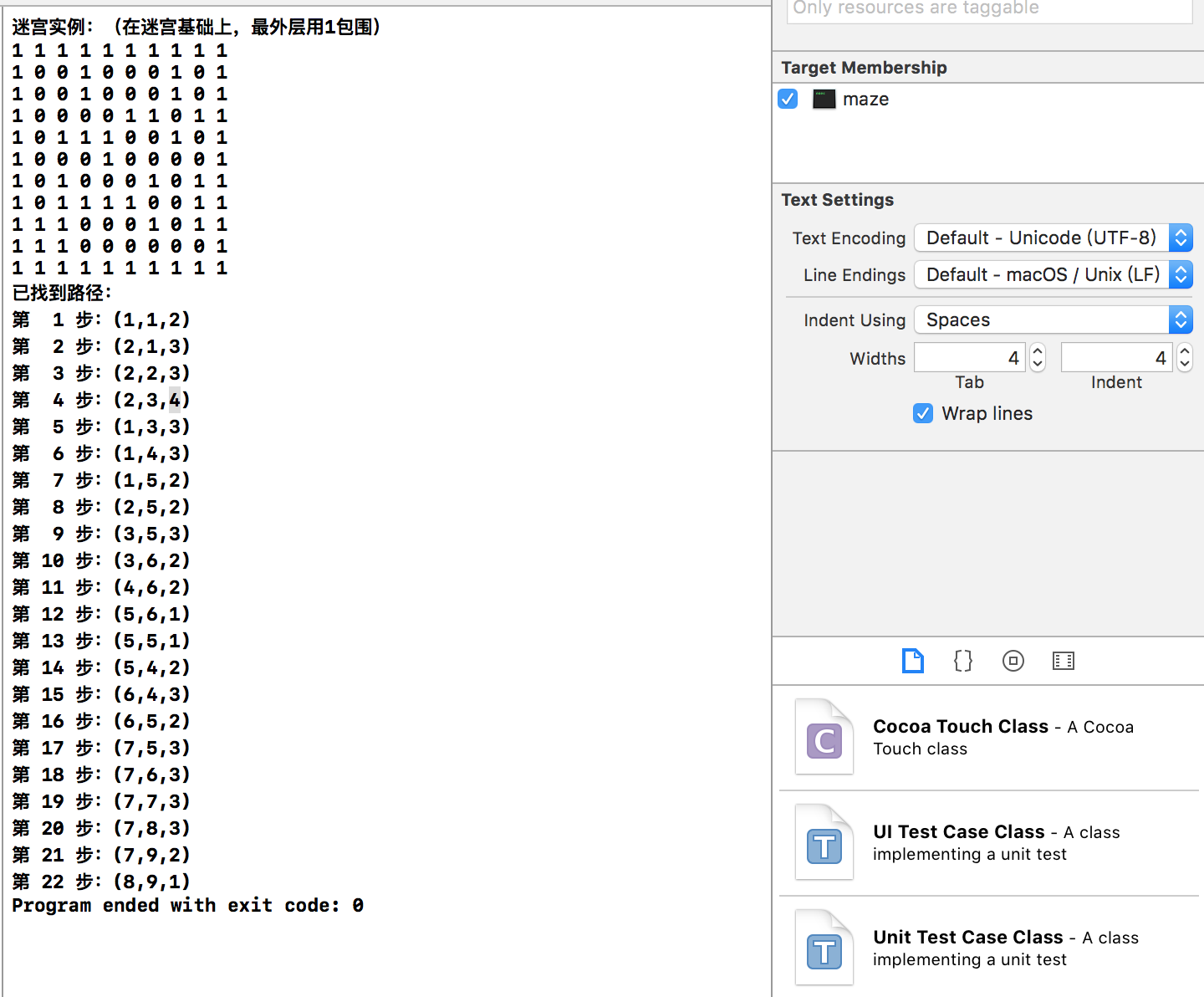
return 0;

}

四、用户使用说明

测试数据：数据结构实验书P105页

五、 测试结果



六、 附录

1. 迷宫起点为（1，1）

2. 迷宫外侧由1包围一圈

3.（x，y，d）横轴为x，纵轴为y，方向为d

3. 关于d：向上为1，向右为2，向下为3，向左为4