### 第15题：迷宫问题

#### 数据结构：

用于存储坐标的结构体point

struct point{

int x;

int y;

};

由于存储迷宫地图的结构体 maze

typedef struct maze{

char mazeMap[MAZEMAXSIZE][MAZEMAXSIZE];

}maze,\*mazepoint;

#### 算法设计思想：

由于题目要求中使用的是栈的操作。故本题采用DFS回溯算法。使用字符数组记录迷宫地图。

使用函数CreateMaze（）从文件中读取迷宫，文件中第一行的两个数字分别是迷宫的长度和宽度，然后后面为迷宫，其中0表示通路可走，1表示墙壁不可走。然后获取迷宫入口和出口位置，判断是否输入错误。输入错误情况如下：1，输入的坐标超过迷宫大小； 2.输入的坐标为墙壁。鉴于入口和出口位置在迷宫里面具有一定的合理性，故不判断为错误。

使用函数Maze()进行DFS算法和回溯算法找出路径。并输出路径图和以坐标和箭头表示的路径。

#### 源程序：

#include<iostream>

#include<fstream>

#include<iomanip>

#include<stack>

#include<Windows.h>

using namespace std;

#define red SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), FOREGROUND\_INTENSITY | FOREGROUND\_RED) //红色

#define blue SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), FOREGROUND\_BLUE) //蓝色

#define yellow SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), FOREGROUND\_INTENSITY | FOREGROUND\_RED | FOREGROUND\_GREEN) //黄色

#define green SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), FOREGROUND\_INTENSITY | FOREGROUND\_GREEN) //绿色

#define white SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), FOREGROUND\_INTENSITY | FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE) //白色

#define ching SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), FOREGROUND\_INTENSITY | FOREGROUND\_GREEN |FOREGROUND\_BLUE) //青色

#define MAZEMAXSIZE 200

//1为墙壁，0为通路

//坐标

struct point{

int x;

int y;

};

//迷宫

typedef struct maze{

char mazeMap[MAZEMAXSIZE][MAZEMAXSIZE];

}maze,\*mazepoint;

bool check\_board(int x,int mx){

if (x < 0) {

cout << "您的坐标超出边界"<<endl;

return false;

}

else if (x >= mx) {

cout << "您的坐标超出边界"<<endl;

return false;

}

else return true;

}

//创建迷宫

void CreateMaze(mazepoint p,point &start,point &end ,int &x,int &y)

{

//存储迷宫长和宽

//int x, y;

ifstream in;

in.open("maze.txt");

if (!in)

{

cout << "打开 maze.txt 失败" << endl;

exit(1);

}

//输入迷宫

in >> x >> y;

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

{

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

{

in >>p->mazeMap[i][j];

}

}

in.close();

//输出迷宫

cout << "原迷宫为:\n(用1表示墙壁)\n(用0表示通路)" << endl;

cout<<" ";

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

cout<< setw(3)<<setiosflags(ios::left)<<j;

cout << endl;

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

{

cout << setw(3)<<setiosflags(ios::left) << i;

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

{

if ( p->mazeMap[i][j] == '1'){

//墙壁

red;

cout << p->mazeMap[i][j]<<" ";

white;

}

else if (p->mazeMap[i][j] == '0'){

//通路

yellow;

cout << p->mazeMap[i][j]<<" ";

white;

}

if (j == y - 1)

cout << endl;

}

}

cout << "你可输入的横坐标范围为 0至"<<x-1<<endl;

cout << "你可输入的纵坐标范围为 0至"<<y-1<<endl;

while (1){

cout << "请输入迷宫的入口坐标:";

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

if (check\_board(start.x,x) && check\_board(start.y,y)) {

if (p->mazeMap[start.x][start.y] =='1'){

cout << "输入的入口坐标为墙壁，请重新输入\n"<<endl;

continue;

}

else break;

}

else {

cout <<"输入入口坐标有误，请重新输入\n"<<endl;

continue;

}

}

while (1){

cout << "请输入迷宫的出口坐标:";

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

if (check\_board (end.x,x)&& check\_board(end.y,y)){

if (p->mazeMap[start.x][start.y] =='1'){

cout << "输入的出口坐标为墙壁，请重新输入\n"<<endl;

continue;

}

else break;

}

else {

cout <<"输入出口坐标有误，请重新输入\n"<<endl;

continue;

}

}

}

//走迷宫

void Mazes(mazepoint p,point &start,point &end,int &boardX,int &boardY ){

stack<point> Stack;

Stack.push(start);

//cur表示当前位置

point cur = start,q;

int cur\_x = start.x;

int cur\_y = start.y;

//是否走过标志

bool flag = false;

//依次下左上右表示四个方向的移动，i表示几个方向被尝试过；

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

int i = 0;

while (!Stack.empty() && ((cur.x != end.x)||(cur.y != end.y)) ){

cur = Stack.top();

flag = 0;

i = 0;

//从当前点开始尝试四个方向

while (flag == 0 && i < 4) {

cur\_x = cur.x + moveing[i][0];

cur\_y = cur.y + moveing[i][1];

if (p->mazeMap[cur\_x][cur\_y] == '0'){

cur.x = cur\_x;

cur.y = cur\_y;

Stack.push(cur);

// \*表示通路

p->mazeMap[cur\_x][cur\_y] = '\*';

flag = 1; //该点已经走过

}

i++; //换方向

}

if (i == 4 && flag == 0)

{

p->mazeMap[cur.x][cur.y] = '#'; //那么将当前点置为墙壁,便于处理

Stack.pop(); //将该点出栈回到上一个点

}

}

p -> mazeMap[start.x][start.y] ='\*';

p -> mazeMap[end.x][end.y] ='\*';

if (Stack.empty()){

cout << "没有路径"<<endl;

}

else{

//输出迷宫路径图

cout << "解后迷宫路径如下\n(用1表示墙壁)\n(用0表示通路)\n(用\*表示路径)\n(用#表示用算法回溯时探索过的路径，但不是答案路径)"<<endl;

cout<<" ";

for (int j = 0; j < boardY; j++)

cout<< setw(3)<<setiosflags(ios::left)<<j;

cout << endl;

for (int i = 0; i < boardX; i++)

{

cout << setw(3)<<setiosflags(ios::left) << i;

for (int j = 0; j < boardY; j++)

{

if ( p->mazeMap[i][j] == '1'){

//墙壁

red;

cout << p->mazeMap[i][j]<<" ";

white;

}

else if (p->mazeMap[i][j] == '0'){

//通路

yellow;

cout << p->mazeMap[i][j]<<" ";

white;

}

else if (p->mazeMap[i][j] == '\*'){

green;

cout << p->mazeMap[i][j]<<" ";

white;

}

else if (p->mazeMap[i][j] == '#'){

ching;

cout << p->mazeMap[i][j]<<" ";

white;

}

if (j == boardY - 1)

cout << endl;

}

}

cout << endl;

//输出迷宫路径表

//栈反转

stack<point> temp;

while (!Stack.empty()){

q = Stack.top();

Stack.pop();

temp.push(q);

}

cout << "以坐标显示的迷宫路径如下："<<endl;

int huanhang = 0;

while (!temp.empty()){

q = temp.top();

temp.pop();

huanhang++;

cout << "("<< q.x <<","<<q.y<<")";

if (!temp.empty()) cout << "->";

if (huanhang % 8 == 0) cout<<endl;

}

//StackTraverse(S);

}

}

int main(){

maze map;

mazepoint p = &map;

point start,end;

int x,y;

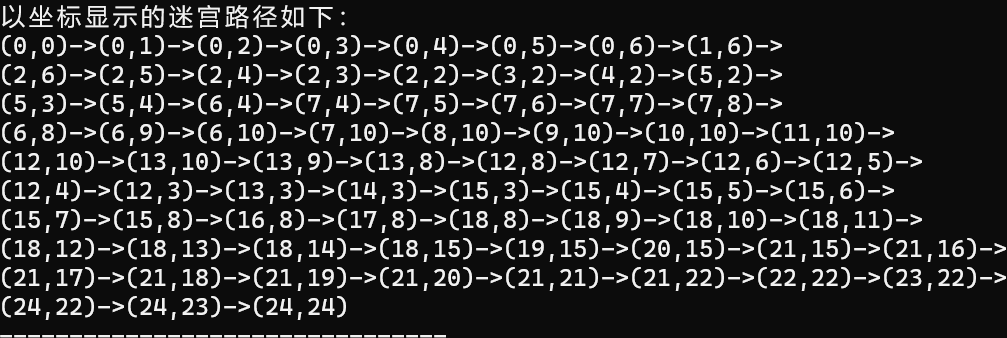
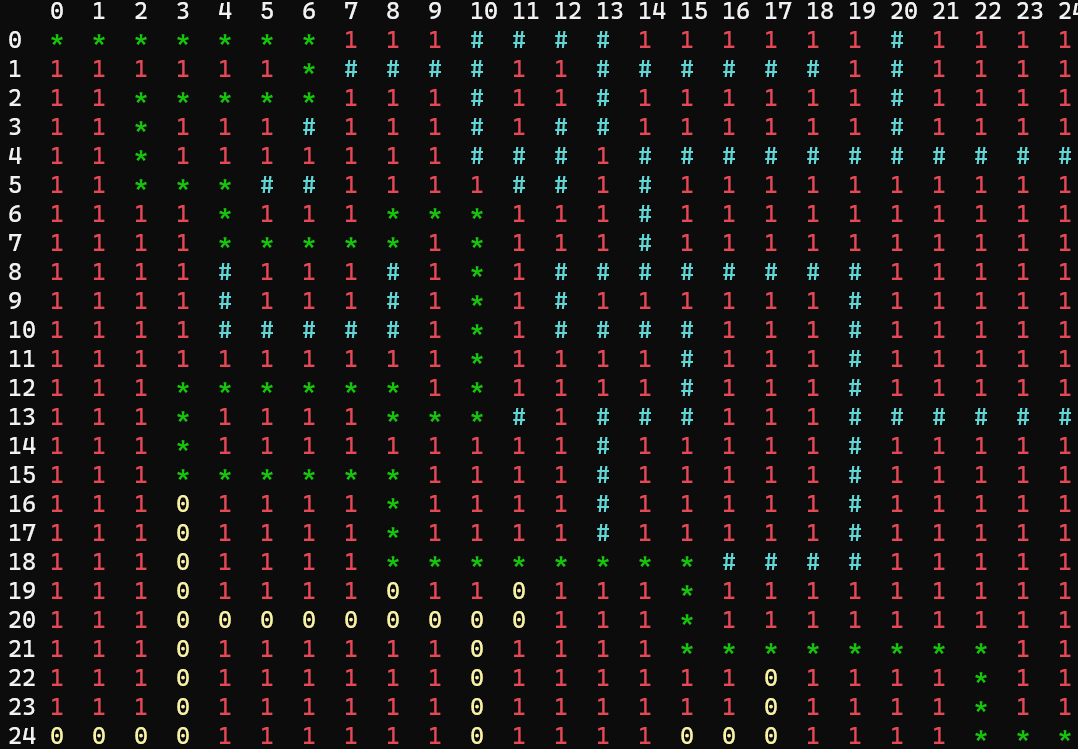
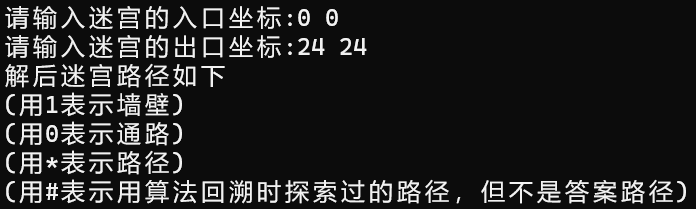
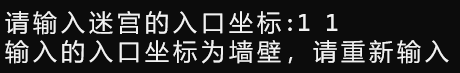
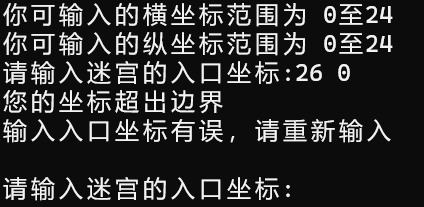
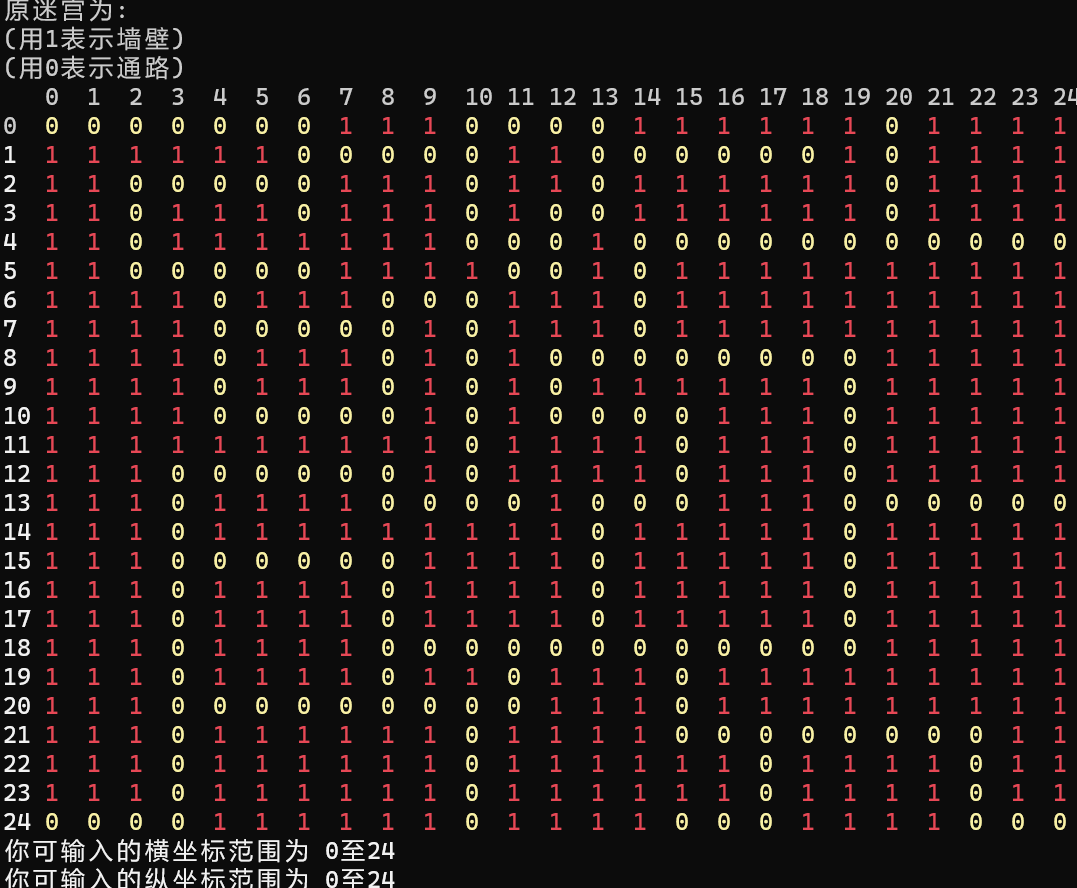
CreateMaze(p,start,end,x,y);

Mazes(p,start,end,x,y);

return 0;

}

#### 测试数据和结果



#### 时间复杂度：

DFS算法O（N）

#### 反思：

还可以将输出功能封装为一个函数，还可以再做出一个功能，利用清屏函数和Sleep函数将程序找路径的过程演示为一个“动画”。

#### 改进：

#### 该题代码行：

234行