项目说明文档

数据结构课程设计

——勇闯迷宫

作 者 姓 名： 李子涵

学 号： 1851892

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目录

[1 分析](#_Toc16774_WPSOffice_Level1) [2](#_Toc16774_WPSOffice_Level1)

[1.1 背景分析](#_Toc31854_WPSOffice_Level2) [2](#_Toc31854_WPSOffice_Level2)

[1.2 功能分析](#_Toc4874_WPSOffice_Level2) [2](#_Toc4874_WPSOffice_Level2)

[2 设计](#_Toc31854_WPSOffice_Level1) [3](#_Toc31854_WPSOffice_Level1)

[2.1 数据结构设计](#_Toc18669_WPSOffice_Level2) [3](#_Toc18669_WPSOffice_Level2)

[2.2 类结构设计](#_Toc13386_WPSOffice_Level2) [3](#_Toc13386_WPSOffice_Level2)

[2.3 成员与操作设计](#_Toc7315_WPSOffice_Level2) [3](#_Toc7315_WPSOffice_Level2)

[前进方向结构（offsets）](#_Toc31854_WPSOffice_Level3) [3](#_Toc31854_WPSOffice_Level3)

[坐标结构（coordinate）](#_Toc4874_WPSOffice_Level3) [3](#_Toc4874_WPSOffice_Level3)

[栈类（path）](#_Toc18669_WPSOffice_Level3) [3](#_Toc18669_WPSOffice_Level3)

[私有成员：](#_Toc13386_WPSOffice_Level3) [3](#_Toc13386_WPSOffice_Level3)

[公有操作：](#_Toc7315_WPSOffice_Level3) [3](#_Toc7315_WPSOffice_Level3)

[2.4 系统设计](#_Toc12363_WPSOffice_Level2) [4](#_Toc12363_WPSOffice_Level2)

[3 实现](#_Toc4874_WPSOffice_Level1) [5](#_Toc4874_WPSOffice_Level1)

[3.1 构造迷宫的实现](#_Toc27738_WPSOffice_Level2) [5](#_Toc27738_WPSOffice_Level2)

[3.1.1 构造迷宫流程图](#_Toc12363_WPSOffice_Level3) [5](#_Toc12363_WPSOffice_Level3)

[3.1.2 构造迷宫核心代码](#_Toc27738_WPSOffice_Level3) [5](#_Toc27738_WPSOffice_Level3)

[3.2 寻路功能的实现](#_Toc31667_WPSOffice_Level2) [6](#_Toc31667_WPSOffice_Level2)

[3.2.1 寻路功能流程图](#_Toc31667_WPSOffice_Level3) [6](#_Toc31667_WPSOffice_Level3)

[3.2.2 寻路功能核心代码](#_Toc18309_WPSOffice_Level3) [6](#_Toc18309_WPSOffice_Level3)

[3.3 打印地图的实现](#_Toc18309_WPSOffice_Level2) [8](#_Toc18309_WPSOffice_Level2)

[3.3.1 打印地图流程图](#_Toc26921_WPSOffice_Level3) [8](#_Toc26921_WPSOffice_Level3)

[3.3.2 打印地图核心代码](#_Toc23777_WPSOffice_Level3) [8](#_Toc23777_WPSOffice_Level3)

[3.4 打印路径的实现](#_Toc26921_WPSOffice_Level2) [9](#_Toc26921_WPSOffice_Level2)

[3.4.1 打印路径流程图](#_Toc32766_WPSOffice_Level3) [9](#_Toc32766_WPSOffice_Level3)

[3.5 总体程序的实现](#_Toc23777_WPSOffice_Level2) [10](#_Toc23777_WPSOffice_Level2)

[3.5.1 总体程序流程图](#_Toc17228_WPSOffice_Level3) [10](#_Toc17228_WPSOffice_Level3)

[3.5.2 总体程序核心代码](#_Toc15648_WPSOffice_Level3) [10](#_Toc15648_WPSOffice_Level3)

[3.5.3 总体程序截屏示例](#_Toc26464_WPSOffice_Level3) [11](#_Toc26464_WPSOffice_Level3)

[4 测试](#_Toc18669_WPSOffice_Level1) [13](#_Toc18669_WPSOffice_Level1)

[4.1 存在路径测试](#_Toc32766_WPSOffice_Level2) [13](#_Toc32766_WPSOffice_Level2)

[测试用例：](#_Toc14490_WPSOffice_Level3) [13](#_Toc14490_WPSOffice_Level3)

[预期结果：](#_Toc20329_WPSOffice_Level3) [13](#_Toc20329_WPSOffice_Level3)

[实验结果：](#_Toc15414_WPSOffice_Level3) [13](#_Toc15414_WPSOffice_Level3)

[测试用例：](#_Toc12303_WPSOffice_Level3) [14](#_Toc12303_WPSOffice_Level3)

[预期结果：](#_Toc31734_WPSOffice_Level3) [15](#_Toc31734_WPSOffice_Level3)

[实验结果：](#_Toc24415_WPSOffice_Level3) [15](#_Toc24415_WPSOffice_Level3)

[4.2 路径不存在测试](#_Toc17228_WPSOffice_Level2) [15](#_Toc17228_WPSOffice_Level2)

[测试用例：](#_Toc26489_WPSOffice_Level3) [15](#_Toc26489_WPSOffice_Level3)

[入口（1,1）出口（5,5）](#_Toc24859_WPSOffice_Level3) [16](#_Toc24859_WPSOffice_Level3)

[预期结果：给出路径不存在提示，程序运行正常不崩溃。](#_Toc918_WPSOffice_Level3) [16](#_Toc918_WPSOffice_Level3)

[实验结果：](#_Toc24178_WPSOffice_Level3) [16](#_Toc24178_WPSOffice_Level3)

# 

# 1 分析

## 1.1 背景分析

迷宫是我们常玩的游戏。然而得到的迷宫存不存在从入口到出口的路径、如何走到出口则是未知的，若想高效准确地知道一个迷宫存不存在通路、如何从入口走到出口，就需要借助计算机来进行计算。因此，可以设计一个闯迷宫的程序，用来寻找迷宫的解法。

## 1.2 功能分析

一个解迷宫的程序，首先需要知道迷宫的结构，这就要求程序有输入功能。解迷宫最重要的就是探索道路，这要求程序有分析计算能力，以找到正确的道路。找到道路或者求得迷宫不存在路径之后，还需要输出到屏幕上，因此迷宫程序需要输出的功能。

综上所述，探索迷宫程序至少需要输入、输出、分析计算能力。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，程序要探索迷宫中的路径，若一个方向行不通还需回转并继续探索，这就适合使用递归算法。使用递归算法则需要从最终成功到达出口的位置逐一退回并记录路径，而输出的路径应当是从入口开始到出口，栈的先进后出储存方式适合这种路径储存，因此使用栈的结构来设计程序。

## 2.2 类结构设计

栈通常的储存结构有链表和数组，本程序使用不到太大的数据量，为了简洁使用数组结构的栈。而为了数据稳定以及便于操作，将栈顶封存在类的私有中。

## 2.3 成员与操作设计

**前进方向结构（offsets）**

int x\_move;//x方向的移动

int y\_move;//y方向的移动

**坐标结构（coordinate）**

int x;

int y;

**栈类（path）**

**私有成员：**

coordinate p[100];//数组结构储存坐标

int top;//栈顶指针

**公有操作：**

path() { top = -1; }//构造函数,默认top为-1

~path(){}//析构函数

bool path::push(int a, int b)//入栈

{

if (top >= 99)//若栈已满返回0

return 0;

top ++;//栈不满,栈顶指针+1

p[top].x = a;//输入横坐标

p[top].y = b;//输入纵坐标

return 1;

}

bool path::pop(int &a, int &b)//出栈

{

if (top == -1)//若栈已空返回0

return 0;

a=p[top].x;//输出横坐标

b=p[top].y;//输出纵坐标

top--;//栈顶指针-1

return 1;

}

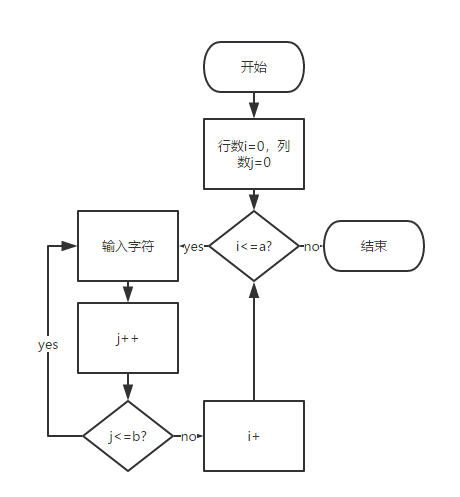
## 2.4 系统设计

首先初始化屏幕，让用户输入迷宫的大小（即行列数），然后输入迷宫具体结构以及出入口位置，开始探索路线，若成功则输出地图和路线，失败则提示无路径。

# 3 实现

## 3.1 构造迷宫的实现

### 3.1.1 构造迷宫流程图



### 3.1.2 构造迷宫核心代码

void build\_maze(int a,int b)//建立迷宫

{

int i,j;//第i列,第j行

for(i=0;i<=a;i++)//循环输入迷宫每一坐标,#为墙壁,0为通路

{

for(j=0;j<=b;j++)

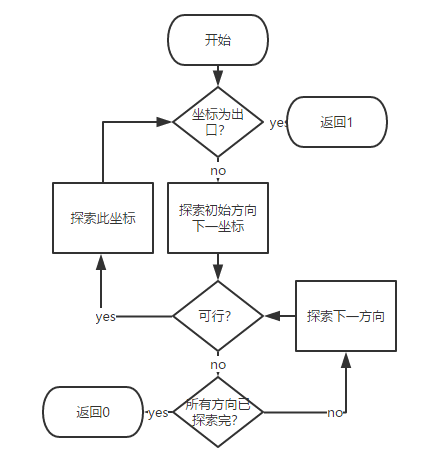
cin>>Maze[i][j];

}

}

## 3.2 寻路功能的实现

### 3.2.1 寻路功能流程图



### 3.2.2 寻路功能核心代码

bool seek\_path(int x,int y)//寻路函数

{

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

//定义方向,0：上；1：左；2：下；3：右；

if(x==ox&&y==oy)//当坐标为出口,返回1

{return 1;}

int i,sx,sy;//i用于记录试探到第几个方向,sx和sy为对应方向下一坐标

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

{

sx=x+move[i].x\_move;

sy=y+move[i].y\_move;

if(Maze[sx][sy]=='0')//若下一位置为0,继续寻路

{

Maze[sx][sy]='1';//标记1,已到达过此位置

if(seek\_path(sx,sy))//递归继续试探

{

if (!pa.push(sx, sy))//到达终点,坐标入栈

return 0;//坐标入队列

Maze[sx][sy]='\*';//更改当前位置为表示路径的\*

return 1;

}

}

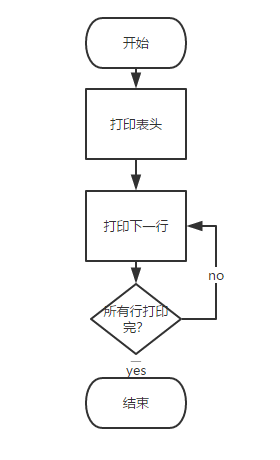
}

return 0;//寻路不成功返回0

}

## 3.3 打印地图的实现

### 3.3.1 打印地图流程图



### 3.3.2 打印地图核心代码

void print\_maze(int a, int b)//打印迷宫

{

int i, j;//第i行第j列

cout << ' ' << '\t';

for (j = 0;j <= b; j++)//先输出表头

cout << "第" << j << "列" << '\t';

cout << endl;

for (i = 0; i <= a; i++)//逐行输出

{

cout<< "第" << i << "行" << '\t';

for (j = 0; j <= b; j++)

{

if (Maze[i][j] == '1')//将标记为1而不是最终路径的1改回0输出

cout << '0'<<'\t';

else

cout << Maze[i][j] << '\t';

}

cout << endl;

}

}

## 3.4 打印路径的实现

### 3.4.1 打印路径流程图

### 3.4.2 打印路径核心代码

void print\_path(path pa)//输出路径

{

int x, y;

while(pa.pop(x,y))//当退栈成功

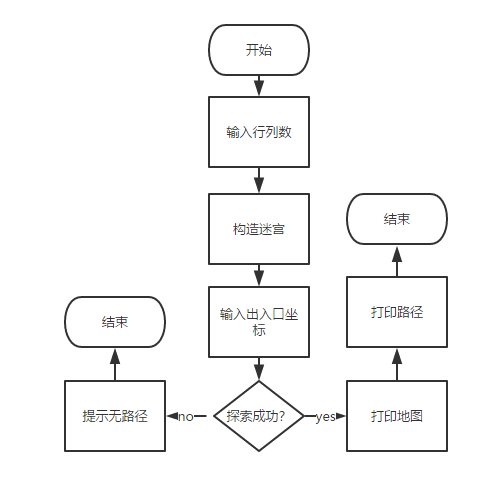
cout << "--->(" << x << "," << y << ")";//输出从起点下一位置开始的坐标

cout << endl;

}

## 3.5 总体程序的实现

### 3.5.1 总体程序流程图



### 3.5.2 总体程序核心代码

int a,b;

cout<<"输入迷宫行数："<<endl;

cin>>a;

cout<<"输入迷宫列数："<<endl;

cin>>b;

cout << "输入迷宫:" << endl;

build\_maze(a,b);

int x,y;

cout<<"输入入口坐标x、y："<<endl;

cin>>x>>y;

Maze[x][y] = '\*';//将起点标记为路径\*

cout<<"输入出口坐标x、y："<<endl;

cin >> ox >> oy;

if (seek\_path(x, y))//若寻路成功

{

cout << "迷宫地图："<<endl;//打印地图

print\_maze(a, b);

cout <<"迷宫路径： "<<endl//打印路径

<<"(" << x << "," << y << ")";//先打印出起点

print\_path(pa);

}

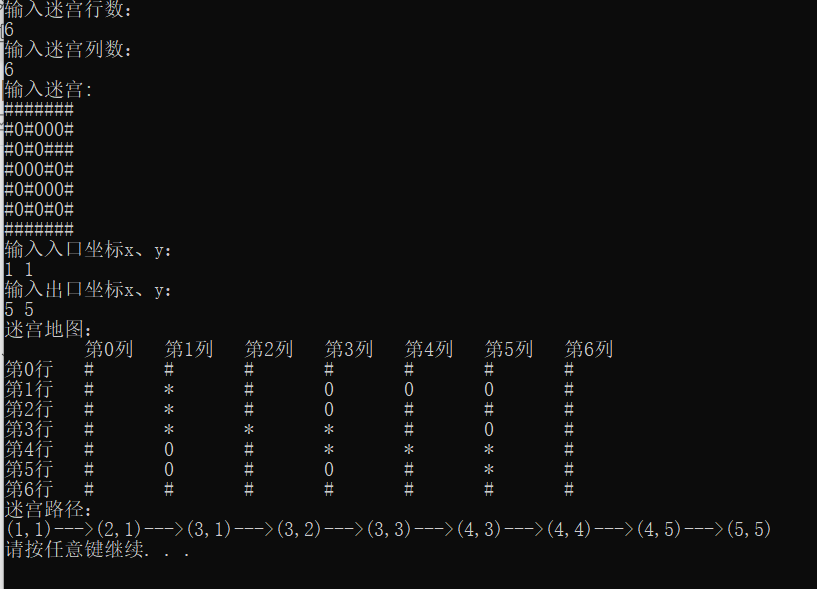
else

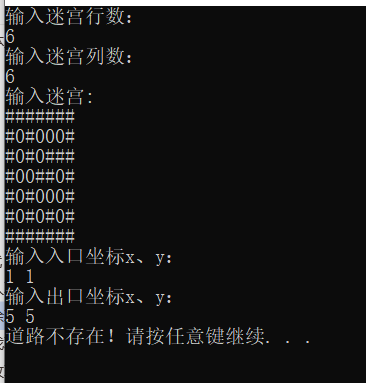
cout << "道路不存在！";//寻路不成功输出无路径提示

system("pause");

return 0;

### 3.5.3 总体程序截屏示例





# 4 测试

## 4.1 存在路径测试

### 4.1.1

**测试用例**：

#######

#0#000#

#0#0###

#000#0#

#0#000#

#0#0#0#

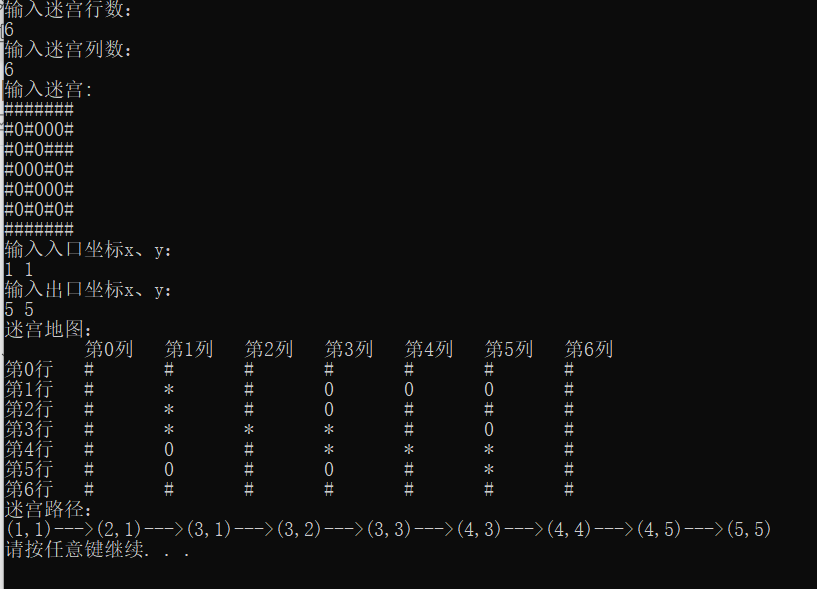
#######

入口（1,1），出口（5,5）

**预期结果**：

寻路成功，输出地图路径。

**实验结果：**



### 4.1.2

**测试用例：**

############

#0#000000###

#0#0##0#0###

#000#0##00##

#0#000#0#00#

#0#0#0#000##

##000##0####

#0##0##0####

##00#0000###

#0##0#0#####

##00##00000#

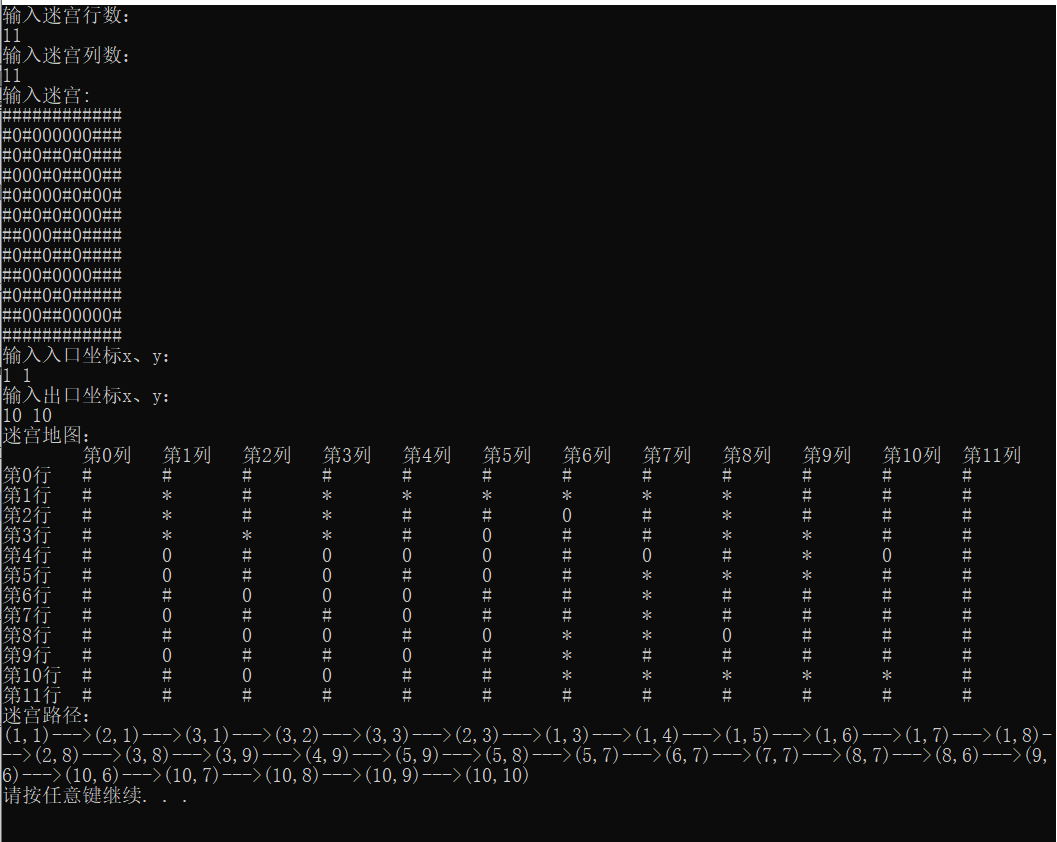
############

入口（1,1）出口（10，10）

**预期结果：**

寻路成功，输出地图和路径。

**实验结果：**



## 4.2 路径不存在测试

### 4.2.1

**测试用例：**

#######

#0#000#

#0#0###

#00##0#

#0#000#

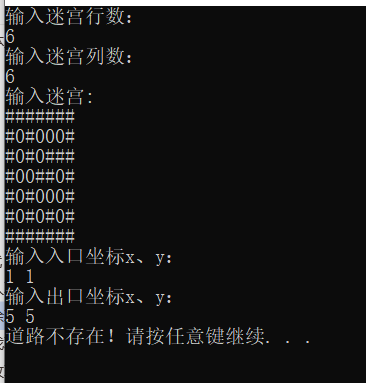
#0#0#0#

#######

**入口（1,1）出口（5,5）**

**预期结果：**给出路径不存在提示，程序运行正常不崩溃。

**实验结果：**



# 