项目说明文档

数据结构课程设计

——勇闯迷宫游戏

作 者 姓 名： 谢宇翔

学 号： 1951708

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 1](#_Toc13361)

[1.1 背景分析 1](#_Toc17973)

[1.2 功能分析 1](#_Toc26702)

[2 设计 1](#_Toc21189)

[2.1 数据结构设计 1](#_Toc21898)

[2.2 类结构设计 1](#_Toc10609)

[2.3 成员与操作设计 1](#_Toc28940)

[2.4 系统设计 2](#_Toc2170)

[3 实现 2](#_Toc3810)

[3.1 总体程序的实现 3](#_Toc3907)

[3.1.1 总体系统流程图 3](#_Toc9800)

[3.1.2深度优先搜索核心代码 4](#_Toc24560)

[3.6.3 总体系统截屏示例 6](#_Toc8512)

[4 测试 6](#_Toc4903)

# 1 分析

## 背景分析

生活中我们经常会遇到迷宫的问题，对于给定一个终点和起点，如何利用计算机快速地找到一条通路，本程序就是用了深度优先算法解决了迷宫问题，并且可在解决迷宫问题的同时，增加其他可操作条件使得玩法多样化。

## 1.2 功能分析

迷宫问题是使用一个二维字符数组进行迷宫地图的装载，node结构体进行迷宫中每个点的取值，是通路还是墙壁，path类进行存储通路，采用dfs深度优先的算法解决从起点到终点的迷宫走法。

为了简便查看，此程序已经固定了迷宫地图，以及迷宫起点以及终点。

# 2 设计

## 2.1 数据结构设计

本程序是在二维字符数组地图内进行遍历，每次遍历一个节点，所以设计node结构体，对每个节点进行处理，同时还包含了visit数组查看某点是否已经遍历过，将成功的路径储存到path类内。

## 2.2 类结构设计

本次程序的类结构只有path类，其实是个链表，链表内储存着从起点到终点的通路，在遍历地图时实时更新，并在最后输出通路，node节点结构体储存每个地图内的一点带有next指针，before指针便于遍历。

## 2.3 成员与操作设计

**结点结构体（node）**

struct node

{

int x = 1;

int y = 1;

node\* next = NULL;

node\* before=NULL;

};

**通路类（path）**

class path

{

node\* head=new node, \* current=head;

friend void dfs(int x, int y);//深度优先算法

void show();//展示通路函数

};

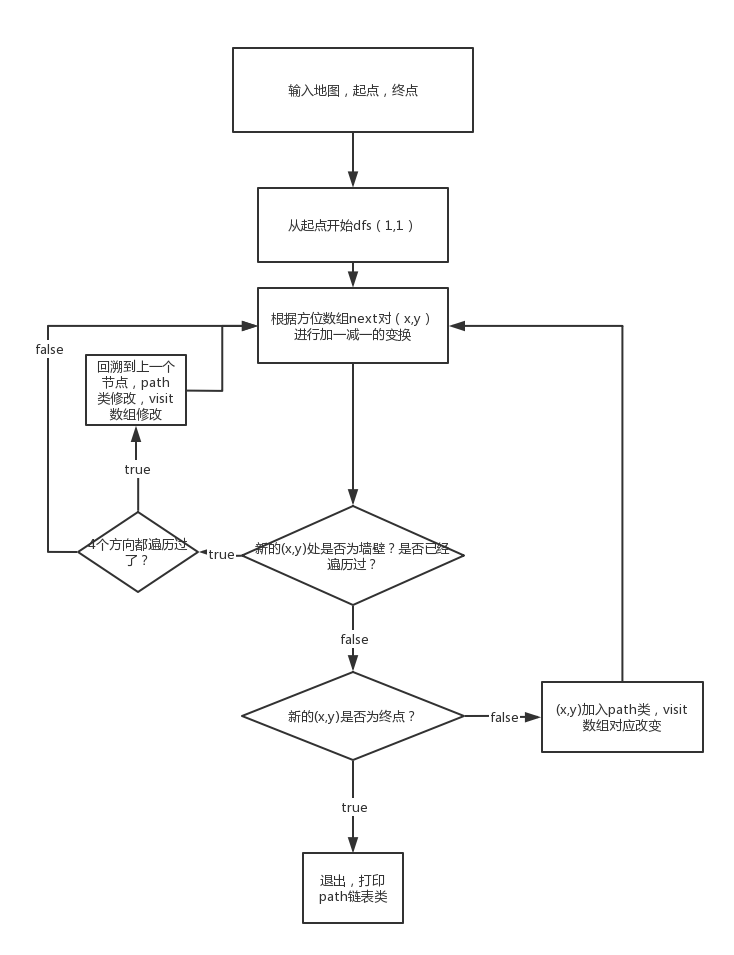
## 2.4 系统设计

首先设计地图，本次程序已经固定了地图，输入好地图后，输入起点终点，从起点开始dfs遍历，dfs里有方位数组二维next来控制往四个方向的前进，visit数组判断某点是否已经遍历过，重复递归，直到判断到终点为止。

# 3 实现

## 3.1 总体程序的实现

### 3.1.1 总体系统流程图



### 3.1.2深度优先搜索核心代码

bool isdone=0；

void dfs(int x, int y)

{

if (isdone == 1)//到了终点了就退出

{

return;

}

int next[4][2] =

{

{0,1},//向右走

{1,0},//向下走

{0,-1},//向左走

{-1,0} //向上走

};//方位数组

int tx, ty, k;//判断下一个方位

if (x == targetx && y == targety)//到了终点

{

isdone = 1;

p1.show();//打印路径

return;

}

for (k = 0; k < 3; k++)

{

//计算下一个点坐标

tx = x + next[k][0];

ty = y + next[k][1];

if (tx < 1 || tx>5 || ty < 1 || ty>5)//越界

{

continue;

}

if (map[tx][ty] == '0' && value[tx][ty] == 0)//到了一个可取的方位

{

p1.current->next = new node;

p1.current->next->before = p1.current;

p1.current = p1.current->next;

p1.current->x = tx;

p1.current->y = ty;

value[tx][ty] = 1;//标记这个点已经走过

dfs(tx, ty);//尝试下一个点

if (isdone == 0)//到这里是删除本个节点，这里是回溯

{

node\* q = p1.current;

p1.current = p1.current->before;

p1.current->next = NULL;

delete q;

}

else

return;

}

}

if (x == 1 && y == 1)//走到这里说明无路可走，无法到达终点，在不确定起点时，可将2个1改为startx,starty即可

{

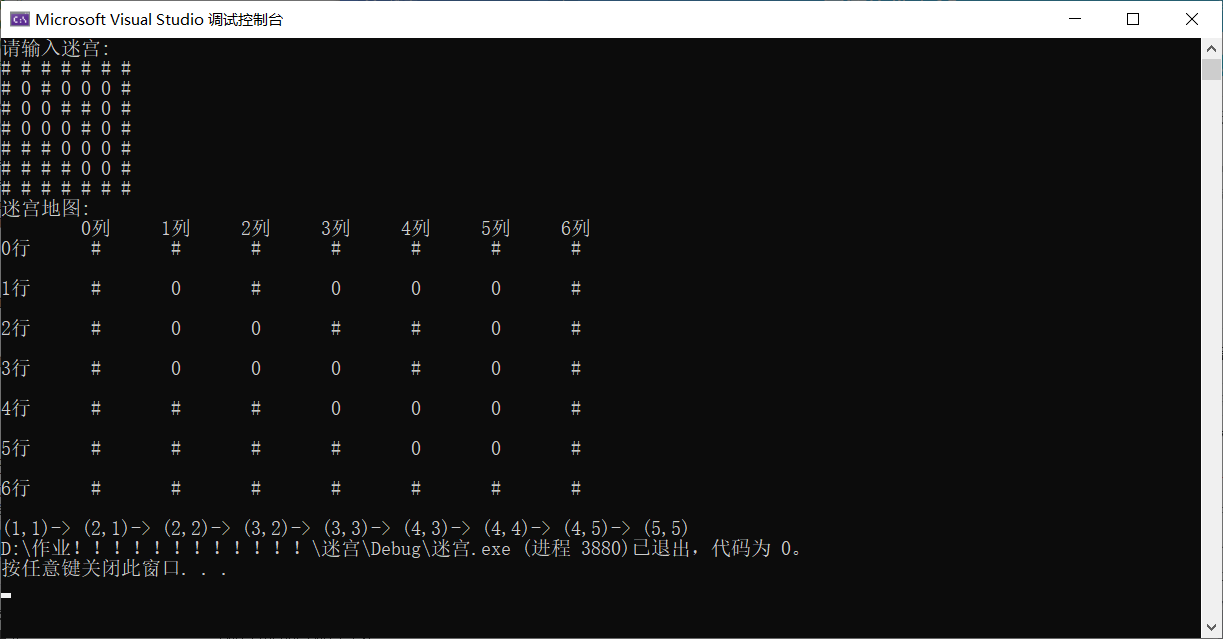
cout << "此起点无法到达终点！" << endl;

return;

}

return;}

### 3.6.3 总体系统截屏示例



# 4 测试

4.1无法到达终点

