数据结构课程设计

项目说明文档

勇闯迷宫游戏

|  |  |
| --- | --- |
| 作者姓名： | 高逸轩 |
| 学 号： | 2053385 |
| 指导教师： | 张 颖 |
| 学院专业： | 软件学院 软件工程 |



同济大学

Tongji University

# 1项目分析

## 1.1 项目需求分析

对于迷宫寻路算法这个问题，我们在实现的时候首先考虑到降低时间复杂度，即在一个迷宫中寻路的时候应当尽可能快速的找出合适的路径。

* 执行效率高

当用户所输入的迷宫数据比较大的情况下，该系统需要能够在比较短的时间内找出所要求解两点间的路径。

* 健壮性

当用户输入的数据不合理时，系统应当给予相应的提示而非直接报错。

* 可视化

该系统应当将迷宫和寻找到的路径通过字符展示出来，以便直观的显示寻找到的路径。

# 1.2 项目要求

### 1.2.1 功能要求

本项目的实质是完成对考生信息的建立，查找，插入，修改，删除等功能。其中考生信息包括准考证号，姓名，性别，年龄和报考类别等信息。同时实现健壮性。并提供了部分测试数据。

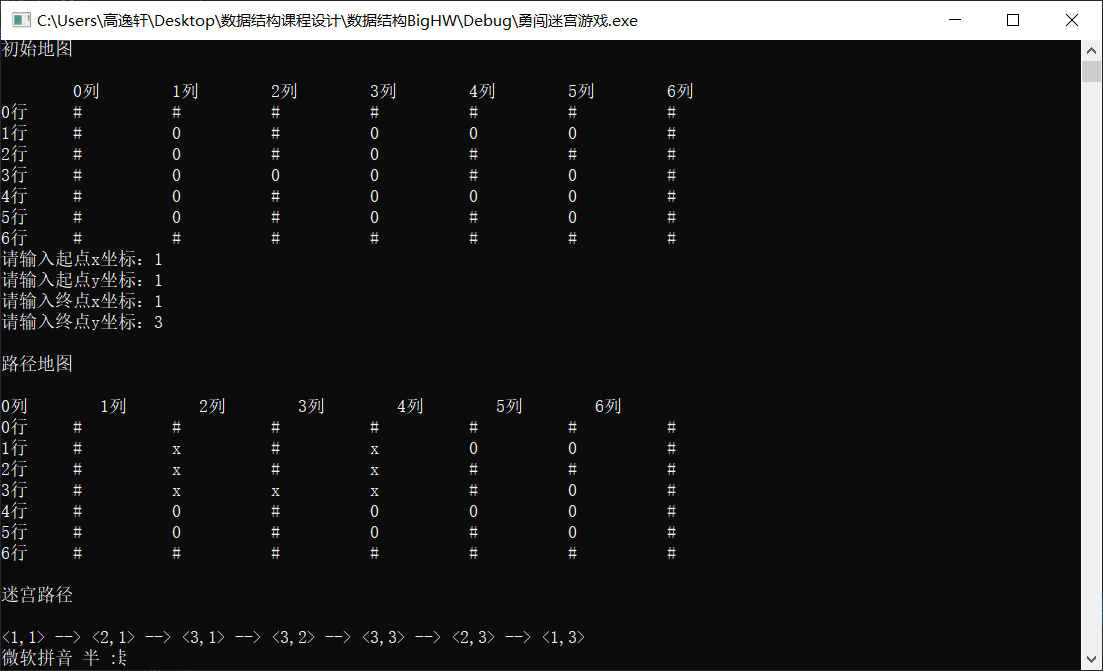
### 1.2.2 输入格式

路径起点、终点的坐标。

### 1.2.3 输出格式

地图状态、搜索路径

### 1.2.4 项目示例



# 2 项目设计

## 2.1 数据结构设计

本项目中使用了多种算法，在算法的使用过程中，为记录迷宫的路径信息，采用了之前题目中已经实现的链表的方式。

## 2.2 类设计

经典的链表一般包括两个抽象数据类型（ADT）——链表结点类（LinkedListNode）与链表类（LinkedList），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。另外，为实现游戏数据的封存，另外多设计了Game类，用以存储游戏数据和其他功能。以下为本程序的设计：

### 2.2.1 链表节点

// 节点

struct LinkedListNode

{

// 构造函数，初始化坐标

LinkedListNode(int xx = 0, int yy = 0) :x(xx), y(yy) { next = NULL; flag = 0; };

// 展示该节点坐标

void Display() { cout << '<' << x << ',' << y << '>'; }

// 记录行坐标、列坐标

int x, y;

// 记录搜索时是否以及被经过

bool flag;

// 记录搜索时的下一个点

LinkedListNode\* next;

};

### 2.2.2 链表

// 链表类

class LinkedList

{

friend Game;

public:

// 构造函数，为head指针分配初始空间

LinkedList()

// 析构函数，释放空间

~LinkedList()

// 计算带附加头结点的单链表的长度

int Length()

// 定位函数，返回表中第i个元素的地址

LinkedListNode\* Locate(int i)

// 将新元素x插入到第i个节点后

bool Insert(int i, LinkedListNode& x)

// 输出链表信息

void Display()

private:

LinkedListNode\* head; // 链表头指针

};

### 2.2.3 游戏功能

// 游戏主功能

class Game

{

public:

// 构造函数，初始化坐标信息

Game()

// 深度优先搜索，寻路，若可达返回true并记录路径，不可达返回false

bool FindPath(int nowX, int nowY, int dstX, int dstY)

// 输出迷宫路径

void Display()

// 得到某个节点的访问情况

bool GetNodeFlag(int x, int y) { return node[x][y].flag; }

private:

LinkedListNode node[c\_xMAX + 1][c\_yMAX + 1]; // 节点

LinkedList path; // 记录下的搜索路径

};

## 2.3 算法设计

本题为经典的深度优先搜索例题。以下为深度优先搜索介绍：

深度优先搜索算法（Depth First Search , DFS）的思想即回溯法（Backtrack），即在一定的约束条件下试探地搜索前进，若前进中受阻，则及时回头纠正错误另择通路继续搜索的方法。从入口出发，按某一方向向前探索，若能走通，即某处可达，则到达新点，否则探索下一个方向；若所有的方向均没有通路，则沿原路返回前一点，换下一个方向再继续试探，直到所有可能的道路都探索到，或找到一条通路，或无路可走又返回入口点。

在本题的求解过程中，我们通过深度优先搜索的方式寻找可达路径。当我们处在某一个点时，我们会先沿着一个方向一直向下搜索，当在某个点搜索失败后，会回溯到上一层的点，换方向继续搜索。为了保证实现这个功能时，能正确返回前一个点以便向下一个方向试探，则需要在试探过程中保存所能够达到的每个点的下标以及该点前进的方向，当找到出口时试探过程就结束了。此时，从路径的一端回溯，并记录下回溯时经过的点，一直到另外一端，便生成了所求的路径信息。

## 2.4 项目流程图

# 3 核心代码介绍

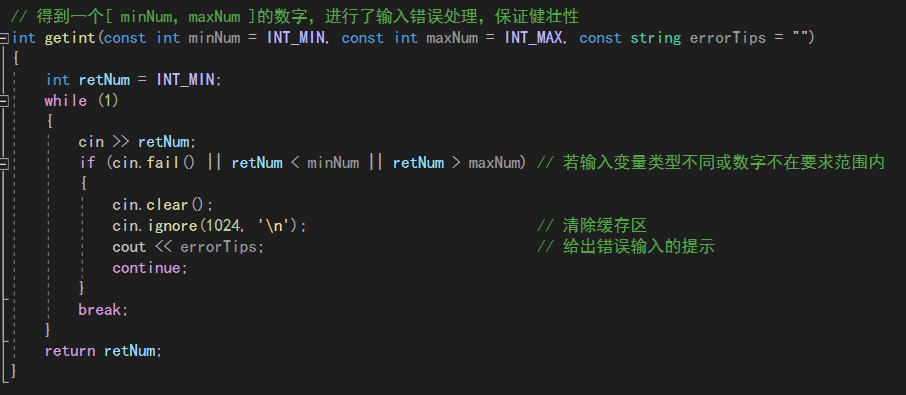
## 3.1 链表功能

本题目为了存储搜索的路径，采用了单链表数据结构。由于在第一题、第二题中，已经详细介绍了其相关信息，在此不再介绍。

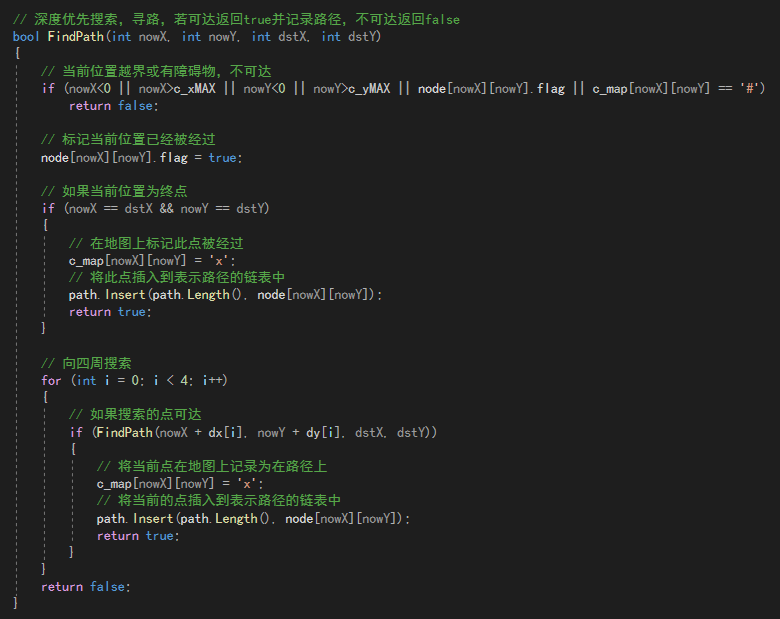
## 3.2 获取输入信息



在展示过地图信息后，需要得到起始点和终点的坐标。在这里需要保证输入坐标的合法性（不能越界）。采取了自己编写的getint()函数，来保证健壮性。另外，在搜索之前，先对起点的信息进行判断，若起点处即为障碍，则直接判断为无法到达。下附getint()代码：



## 3.3 迷宫寻路算法（DFS算法）

 以上为前置信息，包含了地图范围、坐标变化辅助数组、地图信息内容。

对于FindPath() DFS函数，函数参数记录当前的点坐标、终点坐标。

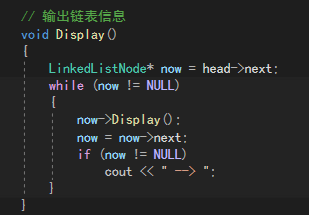
首先，对当前点进行判断：若坐标越界或者当前坐标有障碍物或已经被经过（搜索边界），记录为搜索失败，同时回溯。

若当前点位置合法，记录为已经被经过，避免重复经过。然后判断是否为终点。若为终点，则将地图上标记为被经过，将其插入到记录路径的链表内，然后返回为搜索成功。

若当前点位置合法，且不是终点，则利用坐标变化辅助数组向四个方向扩展搜索。若其向四个方向搜索的结果中有成功到达的路径，则将当前的点记录为在路径上，插入到链表尾端，更改地图信息，并返回为搜索成功。若四个方向都没有搜索成功的，标记为搜索失败。

## 3.4 迷宫路径展示

迷宫路径的展示采取了链表的遍历方式，由于在搜索时，已经将路径上的点依次插入了链表，所以展示路径仅需要采取链表的遍历即可。

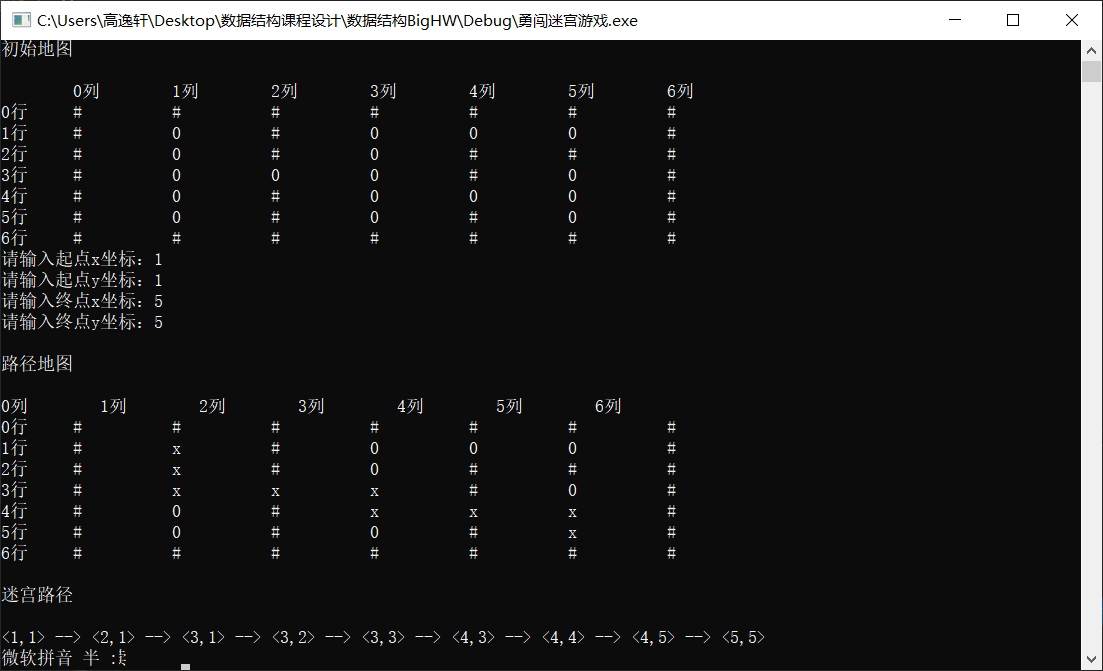


# 4 项目测试

## 4.1 健壮性测试

本次项目错误输入处理共包括两种：一类为输入数据类型与所需数据不符，一类为输入的坐标越界，造成坐标不合法。以上两种错误通过自行编写的getint()函数实现解决，getint()代码上附。

## 4.2 功能测试

为方便老师测试，提供了文件3\_test.txt，内含一组数据，测试了本程序的全部功能。以下为一组测试数据：

# 5 心得体会

本题为经典的深度优先搜索例题。在本次作业的完成中，我采取了DFS搜索的方式，进一步理解了深度的优先搜索思想。同时为达到任务要求，采取了已经熟练掌握的链表结构来进行信息的存储和展示。

另外，在本题采用搜索方法的方面，由于只能向上下左右四个方向扩展，可以采取**以曼哈顿距离为启发函数的A\*搜索算法**，在进行大规模数据时可以拥有更佳的时间复杂度，作为本题的改进。