项目说明文档

数据结构课程设计

——勇闯迷宫游戏

作 者 姓 名： 陈 磊

学 号： 1750981

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](#_Toc495668153)

[1.1 背景分析 1](#_Toc495668154)

[1.2 功能分析 1](#_Toc495668155)

[2 设计 2](#_Toc495668156)

[2.1 数据结构设计 2](#_Toc495668157)

[2.2 类结构设计 2](#_Toc495668158)

[2.3 成员与操作设计 2](#_Toc495668159)

[2.4 系统设计 3](#_Toc495668160)

[3 实现 4](#_Toc495668161)

[3.1 绘制迷宫的实现 4](#_Toc495668162)

[3.1.1 绘制迷宫核心代码 4](#_Toc495668164)

[3.1.2 绘制迷宫截屏示例 6](#_Toc495668165)

[3.2 寻找路径实现 7](#_Toc495668166)

[3.2.1 寻找路径流程图 7](#_Toc495668167)

[3.2.2 寻找路径核心代码 8](#_Toc495668168)

[3.2.3 寻找路径截屏示例 8](#_3.2.2_寻找路径截屏实例)

[3.3 总体系统的实现 9](#_Toc495668182)

[3.3.1 总体系统流程图 9](#_Toc495668183)

[3.3.2 总体系统核心代码 9](#_Toc495668184)

[4 测试 10](#_Toc495668186)

[4.1 功能测试 10](#_Toc495668187)

[4.2 边界测试 11](#_Toc495668193)

[4.3 出错测试 12](#_Toc495668197)

# 1 分析

## 1.1 背景分析

人类建造迷宫已有5000年的历史。在世界的不同文化发展时期，这些奇特的建筑物始终吸引人们沿着弯弯曲曲、困难重重的小路吃力地行走，寻找真相。迷宫类小游戏应运而生。在游戏中，迷宫被表现为冒险舞台里，藏有各式各样奇妙与谜题或宝藏的危险区域。型态有洞窟、人工建筑物、怪物巢穴、密林或山路等。迷宫内有恶徒或凶猛的生物（真实存在或想像物体都有）徘徊，其中可能会有陷阱、不明设施、遗迹等。

求解迷宫游戏有很多方法，回溯法是其中一种。回溯法是一种选优搜索法，又称为试探法，按选优条件向前搜索，以达到目标。但当探索到某一步时，发现原先选择并不优或达不到目标，就退回一步重新选择，这种走不通就退回再走的技术为回溯法，而满足回溯条件的某个状态的点称为“回溯点”。

## 1.2 功能分析

迷宫游戏首先是要绘制迷宫，绘制迷宫这个功能应该要有。然后还要给迷宫一个起点和终点的位置。这个时候再让程序去求解起点和终点之间是否有路径，有几条路，每条路分别是什么。找到路了要输出路径，找不到路就应该给出相应的提示。

# 2 设计

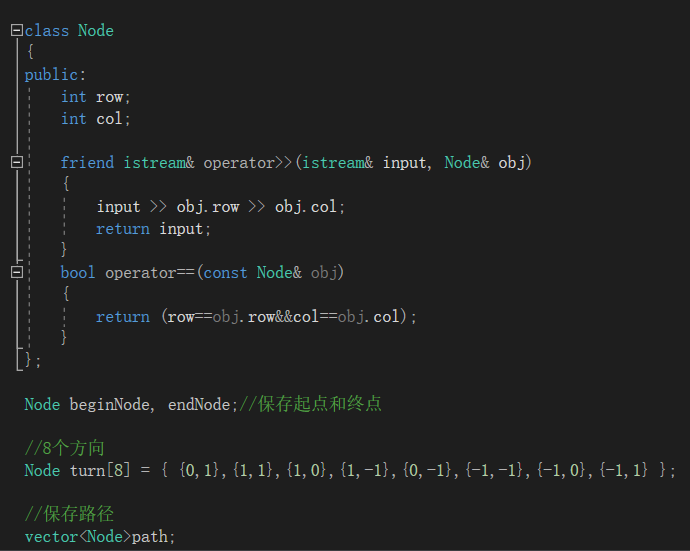
## 2.1 数据结构设计

我们存储迷宫的方式以二维数组来存储，不同的内容代表着墙和路。线性数组能非常方便的随机访问，并且这个程序不需要对数组进行插入和删除，反而是需要大量的随机访问，线性数组无疑是最适合这个程序去存储迷宫。

## 2.2 类结构设计

为了操作方便，定义了一个Node类去表示结点，这样在做判断结点相同时能在一定程度上简化代码，增加其可读性。Node类还可以表示偏转的八个方向。

## 2.3 成员与操作设计



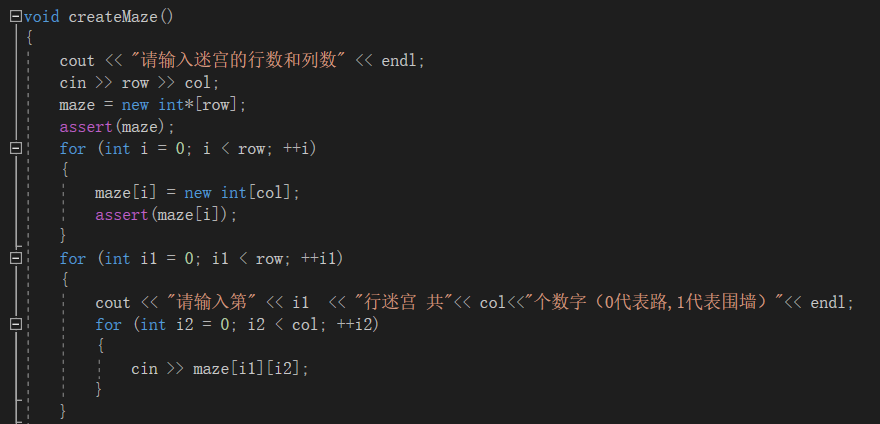
## 2.4 系统设计

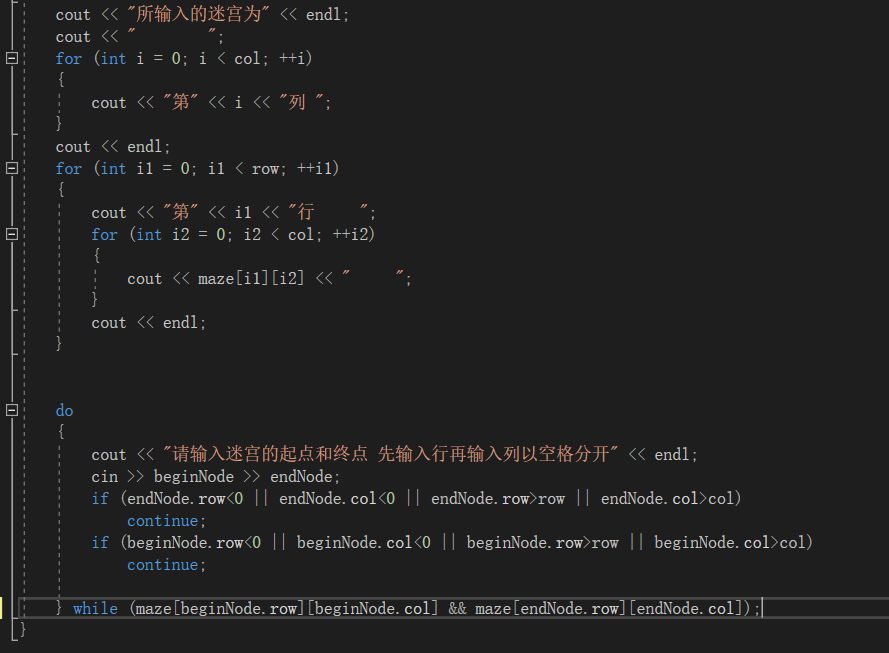
系统首先调用createMaze()函数迷宫的绘制。然后输入起点和终点。最后调用算法去寻找迷宫的道路。

# 3 实现

## 3.1 绘制迷宫的实现

### 3.1.1 绘制迷宫的核心代码



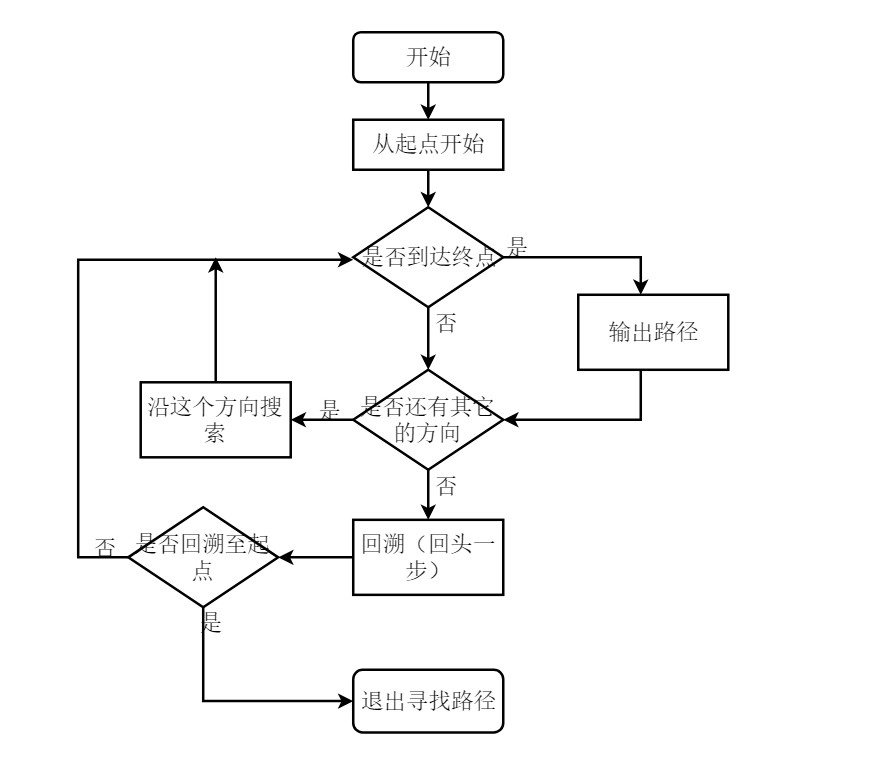


### 3.1.2 绘制迷宫截屏示例



## 3.2 寻找路劲的实现

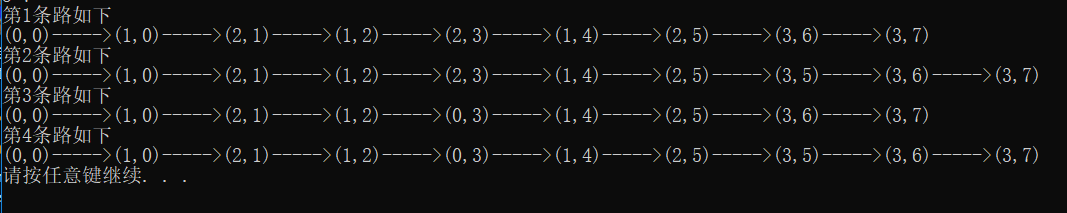
### 3.2.1 寻找路径流程图



### 3.2.2 寻找路径核心代码

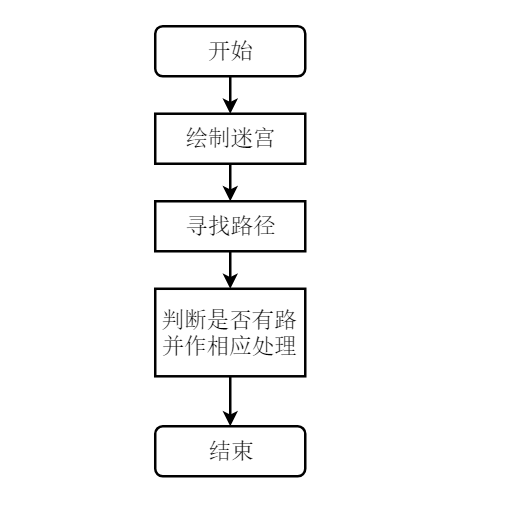
### 

### 3.2.2 寻找路径截屏实例

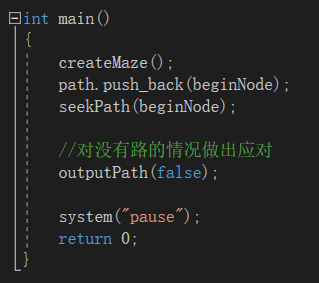


## 3.3 总体系统的实现

### 3.3.1 总体系统流程图



### 3.3.2 总体系统核心代码



# 4 测试

## 4.1 功能测试

**测试用例**：

4 8

0 1 1 0 1 1 1 1

0 1 0 1 0 1 1 1

1 0 1 0 1 0 1 1

1 1 1 0 1 0 0 0

0 0

3 7

**预期结果**：

第1条路如下

(0,0)----->(1,0)----->(2,1)----->(1,2)----->(2,3)----->(1,4)----->(2,5)----->(3,6)----->(3,7)

第2条路如下

(0,0)----->(1,0)----->(2,1)----->(1,2)----->(2,3)----->(1,4)----->(2,5)----->(3,5)----->(3,6)----->(3,7)

第3条路如下

(0,0)----->(1,0)----->(2,1)----->(1,2)----->(0,3)----->(1,4)----->(2,5)----->(3,6)----->(3,7)

第4条路如下

(0,0)----->(1,0)----->(2,1)----->(1,2)----->(0,3)----->(1,4)----->(2,5)----->(3,5)----->(3,6)----->(3,7)

**实验结果**



## 

## 4.2 边界测试

### 没有道路到达终点

**测试用例：**

4 8

0 1 1 0 1 1 1 1

0 1 1 1 0 1 1 1

1 0 1 0 1 0 1 1

1 1 1 0 1 0 0 0

0 0

3 7

**预期结果：**给出没有路的信息，程序运行正常不崩溃。

**实验结果：**



## 4.3 出错测试

### 输入的起点终点超出两边边界或者是墙壁

**测试用例：**输入起点终点位置不再迷宫内

**预期结果：**程序给出提示信息，重新输入起点和终点。

**实验结果：**

