项目说明文档

数据结构课程设计

——勇闯迷宫游戏

作 者 姓 名： 张诚睿

学 号： 2150998

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](#_Toc122644213)

[1.1 项目背景分析 1](#_Toc122644214)

[1.2 功能分析 1](#_Toc122644215)

[2 设计 1](#_Toc122644216)

[2.1 数据结构设计 1](#_Toc122644217)

[2.2 类结构设计 1](#_Toc122644218)

[2.3 类具体成员与操作设计 2](#_Toc122644219)

[2.4 系统设计 3](#_Toc122644220)

[3 实现 3](#_Toc122644221)

[3.1 链式栈-进栈功能的实现 3](#_Toc122644222)

[3.1.1 功能流程图 3](#_Toc122644223)

[3.1.2 核心代码 4](#_Toc122644224)

[3.2 链式栈-出栈功能实现 5](#_Toc122644225)

[3.2.1 功能流程图 5](#_Toc122644226)

[3.2.2 核心代码 5](#_Toc122644227)

[3.3 回溯法求迷宫解功能的实现 6](#_Toc122644228)

[3.3.1 功能流程图 6](#_Toc122644229)

[3.3.2 核心代码 6](#_Toc122644230)

[3.4 输出路径功能的实现 7](#_Toc122644231)

[3.4.1 功能流程图 7](#_Toc122644232)

[3.4.2 核心代码 8](#_Toc122644233)

[3.4.3 效果展示 8](#_Toc122644234)

[3.5 总体功能的实现 8](#_Toc122644235)

[3.5.1 总体流程 8](#_Toc122644236)

[3.5.2 总体功能核心代码 8](#_Toc122644237)

[3.6.3 总体功能截屏示例 9](#_Toc122644238)

[4 测试 9](#_Toc122644239)

[4.1 功能测试 9](#_Toc122644240)

[4.1.1 求解一般迷宫可行路径功能测试 9](#_Toc122644241)

[4.2 边界测试 10](#_Toc122644242)

[4.2.1迷宫有多个解 10](#_Toc122644243)

[4.2.2 迷宫无解 11](#_Toc122644244)

# 1 分析

## 1.1 项目背景分析

迷宫游戏是常见的一种益智游戏，但对于人脑来说，只能较快地解决小型迷宫问题，对于地图庞大、线路多的迷宫地图则很难找到可行路径，需要计算机快速分析。

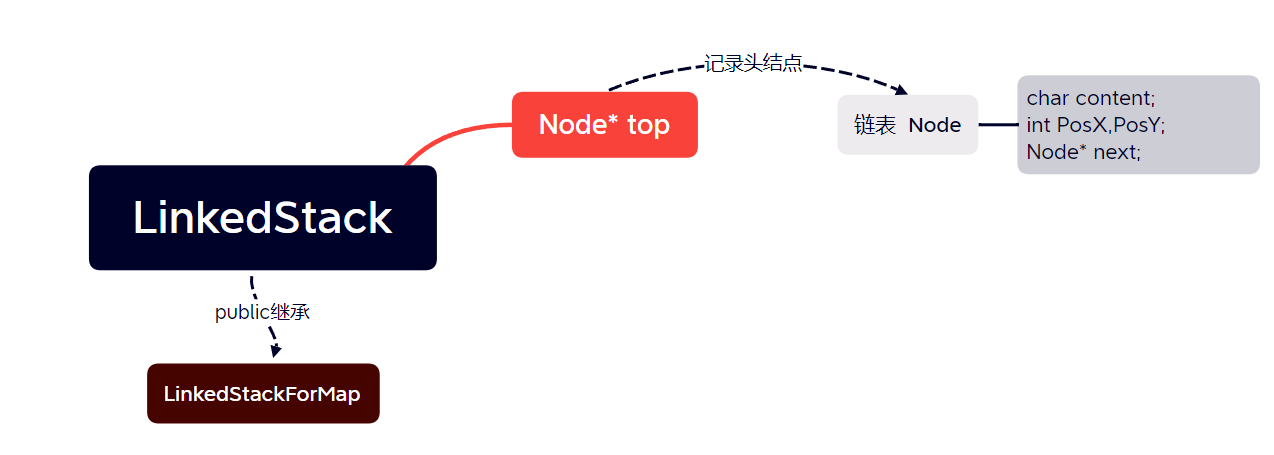
## 1.2 功能分析

本项的目的在于实现对给定迷宫地图求解一条可行路径并打印出来，对于求解路径是否为最短路径不做要求。

在本项目中，迷宫的起点被设定为<1,1>,终点被设置为<5,5>，地图被静态存入，运行过程中不做更改。

# 2 设计

## 2.1 数据结构设计



## 2.2 类结构设计

如上所述，本项目采用struct Node描述链表结点类（LNode），这样使得链表结点类LinkedStack（LinkList）可以访问链表结点。

同时，将链表实现的栈式结构以public方式派生出LinkedStackForMap专门用于求解迷宫路径问题。这样，既可以用链式栈-回溯法求解迷宫路径，也可以用链式栈将结果路径倒置，打印出正确顺序的答案。

除此以外，为了能够打印出完整地图，还设计了Map类对地图做操作，在此程序中专门用于打印完整迷宫地图。

## 2.3 类具体成员与操作设计

**地图操作类**

class Map {

public:

void PrintMap();

};

**链表结点类**

struct Node{

char content;

int PosX, PosY; // 坐标

Node\* next;

};

**链式栈类**

class LinkedStack {

protected:

Node\* top; // 头结点指针

bool Pop(); // 出栈

void makeEmpty(); // 释放内存

public:

LinkedStack() :top(NULL){};

~LinkedStack() { makeEmpty(); };

void Push(int Posx, int Posy); // 进栈

friend ostream& operator<<(ostream& os, LinkedStack& target);

};

**迷宫求路径类**

class LinkedStackForMap:public LinkedStack {

public:

LinkedStackForMap() :LinkedStack() {};

~LinkedStackForMap();

void FindPath(); // 寻找出口路径

void PrintPath(); // 打印结果

private:

bool Step(const int direction, const int NowX, const int NowY, bool(\*IsDetected)[7]);

};

## 2.4 系统设计

地图用全局变量const char[MAP\_ROWS][ MAP\_COLS]存储，主函数首先建立Map对象，打印完整地图，然后建立LinkedStackForMap对象，求解路径答案。

# 3 实现

此处仅介绍核心功能的实现

## 3.1 链式栈-进栈功能的实现

### 3.1.1 功能流程图

### 3.1.2 核心代码

if (top == NULL) // 空栈

{

top = new(nothrow) Node;

if (top == NULL)

{

cout << "Memory Wrong";

exit(EXIT\_FAILURE);

}

top->content = MAZE\_MAP[Posy][Posx];

top->PosX = Posx;

top->PosY = Posy;

top->next = NULL;

}

else // 非空栈，头结点处进栈

{

Node\* tmp= new(nothrow) Node;

if (tmp == NULL)

{

cout << "Memory Wrong";

exit(EXIT\_FAILURE);

}

tmp->content = MAZE\_MAP[Posy][Posx];

tmp->PosX = Posx;

tmp->PosY = Posy;

tmp->next = top;

top = tmp;

}

}

## 3.2 链式栈-出栈功能实现

### 3.2.1 功能流程图

### 3.2.2 核心代码

if (top == NULL) // 空栈，无法出栈

{

return false;

}

else // 从头结点出栈

{

Node\* tmp = top;

top = top->next;

delete tmp;

return true;

}

## 3.3 回溯法求迷宫解功能的实现

### 3.3.1 功能流程图

### 3.3.2 核心代码

while (top!=NULL && (NowX != 5 || NowY != 5))

{

NowX = top->PosX;

NowY = top->PosY;

if (Step(LEFT, NowX, NowY, IsDetected)) // 可向左探索

{

NowX -= 1;

Push(NowX, NowY);

}

else if (Step(RIGHT, NowX, NowY, IsDetected)) // 可向右探索

{

NowX += 1;

Push(NowX, NowY);

}

else if (Step(UP, NowX, NowY, IsDetected)) // 可向上探索

{

NowY -= 1;

Push(NowX, NowY);

}

else if (Step(DOWN, NowX, NowY, IsDetected)) // 可向下探索

{

NowY += 1;

Push(NowX, NowY);

}

else // 无路可走

{

Pop(); // 出栈，回溯

}

}

## 3.4 输出路径功能的实现

### 3.4.1 功能流程图

### 3.4.2 核心代码

void LinkedStackForMap::PrintPath()

{

if (top == NULL) // 没有路可走

{

cout << "\nNo Way Out!" << endl;

return;

}

Node\* pnode = top;

LinkedStack output; // 倒序输出栈

while (pnode != NULL)

{

output.Push(pnode->PosX, pnode->PosY);

pnode = pnode->next;

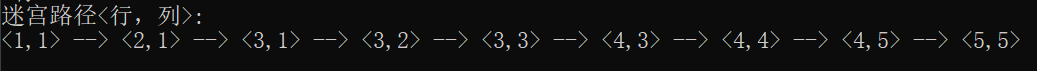
}

cout << "迷宫路径<行，列>:" << endl;

cout << output;

}

### 3.4.3 效果展示



## 3.5 总体功能的实现

### 3.5.1 总体流程

地图用全局变量const char[MAP\_ROWS][ MAP\_COLS]存储，主函数首先建立Map对象，打印完整地图，然后建立LinkedStackForMap对象，求解路径答案。

### 3.5.2 总体功能核心代码

#define MAP\_ROWS 7

#define MAP\_COLS 7

const char MAZE\_MAP[7][7]

int main()

{

Map mainMap;

mainMap.PrintMap(); // 打印地图

LinkedStackForMap MapResult;

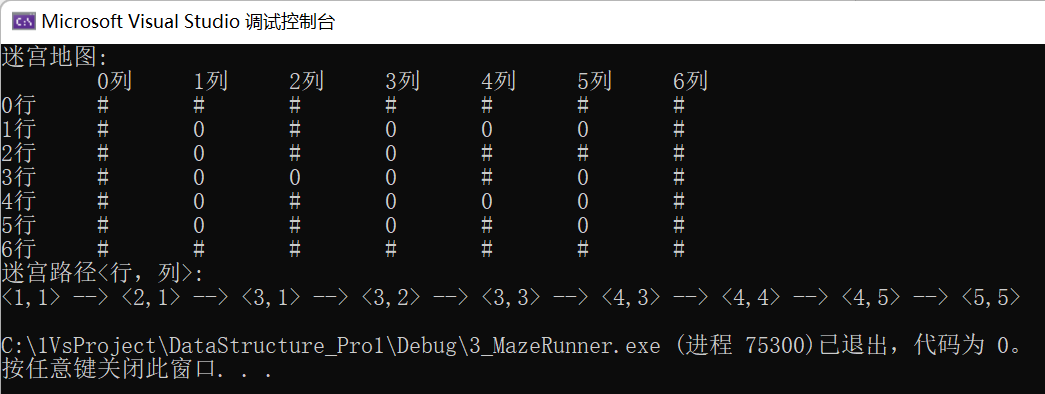
MapResult.FindPath();

MapResult.PrintPath();

return 0;

}

### 3.6.3 总体功能截屏示例



# 4 测试

## 4.1 功能测试

### 4.1.1 求解一般迷宫可行路径功能测试

**测试用例**：

const char MAZE\_MAP[7][7]{

'#','#','#','#','#','#','#',

'#','0','#','0','0','0','#',

'#','0','#','0','#','#','#',

'#','0','0','0','#','0','#',

'#','0','#','0','0','0','#',

'#','0','#','0','#','0','#',

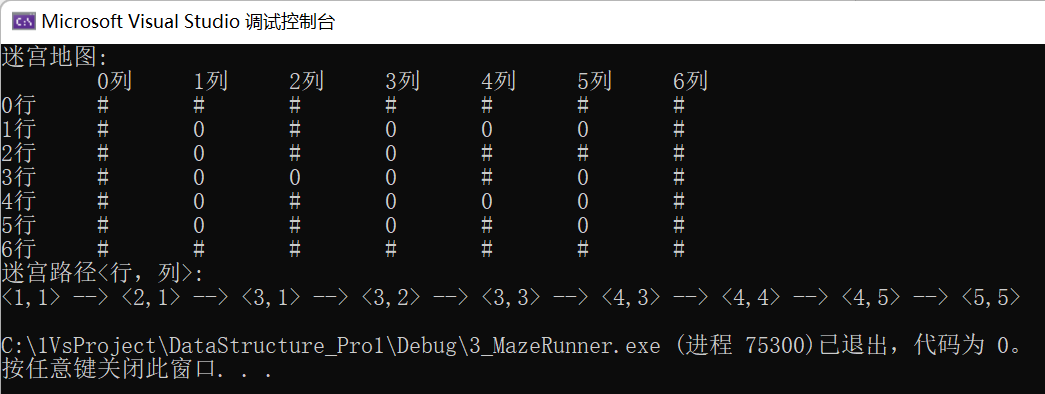
'#','#','#','#','#','#','#',

};

**预期结果**：

<1,1> --> <2,1> --> <3,1> --> <3,2> --> <3,3> --> <4,3> --> <4,4> --> <4,5> --> <5,5>

**实验结果**



## 4.2 边界测试

### 4.2.1迷宫有多个解

**测试用例：**

const char MAZE\_MAP[7][7]{

'#','#','#','#','#','#','#',

'#','0','#','0','0','0','#',

'#','0','#','0','#','#','#',

'#','0','0','0','#','0','#',

'#','0','#','0','0','0','#',

'#','0','#','0','0','0','#',

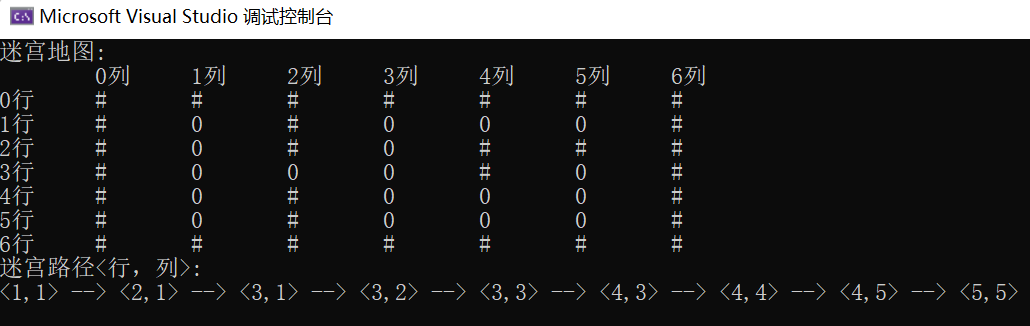
'#','#','#','#','#','#','#',

};

**预期结果：**

<1,1> --> <2,1> --> <3,1> --> <3,2> --> <3,3> --> <4,3> --> <4,4> --> <4,5> --> <5,5>

**实验结果：**



### 4.2.2 迷宫无解

**测试用例：**

const char MAZE\_MAP[7][7]{

'#','#','#','#','#','#','#',

'#','0','#','0','0','0','#',

'#','0','#','0','#','#','#',

'#','0','0','0','#','0','#',

'#','0','#','0','0','#','#',

'#','0','#','0','#','0','#',

'#','#','#','#','#','#','#',

};

**预期结果：**

No Way Out!

**实验结果：**

