**数据结构课程设计《勇闯迷宫游戏》项目说明文档**

2253893 苗君文 软件工程

**1. 项目简介**

**1.1. 项目背景**

迷宫只有两个门，一个门叫入口，另一个门叫出口。一个骑士骑马从入口进入迷宫，迷宫设置很多障碍，骑士需要在迷宫中寻找通路以到达出口。

**1.2. 项目要求**

迷宫问题的求解过程可以采用回溯法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。在求解过程中，为了保证在达到某一个点后不能向前继续行走时，能正确返回前一个以便从下一个方向向前试探，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。

**1.3. 输入格式**

本项目首先需要输入迷宫的阶数以及迷宫地图数组，并根据给出的可视化迷宫地图输入迷宫的入口与出口坐标。

**1.4. 输出格式**

本项目针对输入有详细的输入错误处理，若用户输入错误，则程序会输出相关提示信息，并要求重新输入。除此之外，本项目会根据用户输入的迷宫数组输出初始的迷宫地图。并在用户输入入口与出口坐标后，若无解则会输出提示信息；若有解，则会输出带有路径的可视化的迷宫地图，并输出含坐标的迷宫路径。

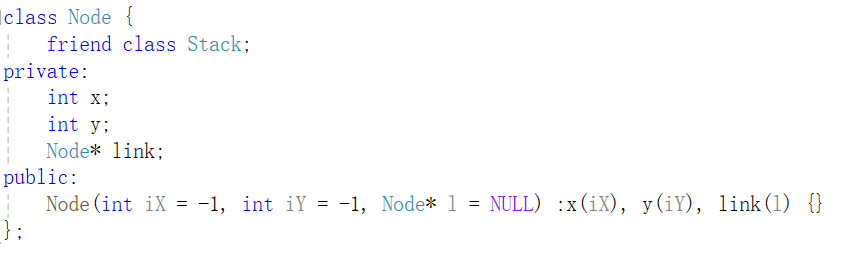
**2. 设计思路**

**2.1. 数据结构设计**

由于最后需要输出走过的迷宫路径（从入口坐标开始，到出口坐标结束），因此选用栈（Stack）作为存储迷宫路径的数据结构。具体原因如下：在迷宫搜索过程中，需要记录从入口到出口的路径。路径记录是一种动态的过程，因为在不断尝试不同的路径，而且在找到有效路径后需要回溯到上一个状态。栈是一种后进先出（LIFO）的数据结构，非常适合用于记录路径。每当我们探索一步新的路径时，将该位置入栈，而当需要回溯时，只需从栈顶取出最近的一个位置即可，符合迷宫搜索中的动态路径记录需求。栈的使用保证了输出迷宫路径时，不会从终点到起点输出，而是借助栈的后进先出的特性从起点到终点依次输出，从而符合项目的需求。

**2.2. 类设计**

**2.2.1. Node（结点类）**



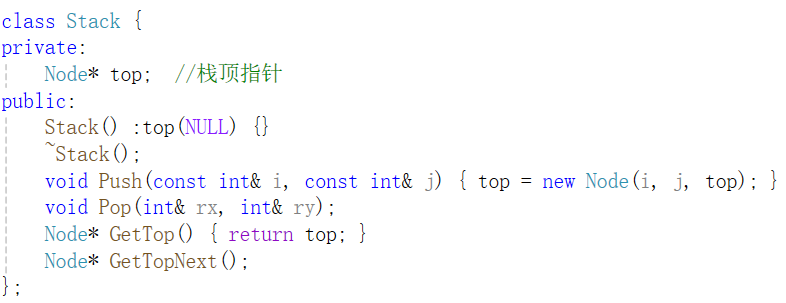
成员变量：

* int x, y: 保存节点的横纵坐标。
* Node\* link: 保存节点的下一个节点。

成员函数：

* Node(int iX = -1, int iY = -1, Node\* l = NULL): 构造函数，初始化节点的坐标和该结点的link。

**2.2.2. Stack（栈类）**



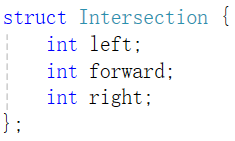
成员变量：

* Node\* top: 栈顶指针。

成员函数：

* Stack(): 构造函数，初始化栈为空。
* ~Stack(): 析构函数，释放栈的内存空间。
* void Push(const int& i, const int& j): 入栈操作，将坐标 (i, j) 入栈。
* void Pop(int& rx, int& ry): 出栈操作，将栈顶元素出栈，并返回其坐标。
* Node\* GetTop(): 获取栈顶指针。
* Node\* GetTopNext(): 获取栈顶的下一个节点指针。

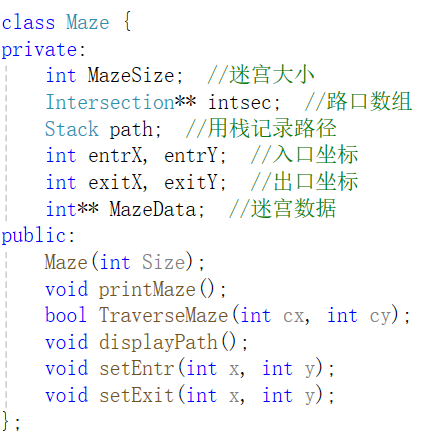
**2.2.3. Intersection（路口结构体）**



成员变量：

* int left, forward, right: 分别表示路口的左、前、右是否可通。

**2.2.4. Maze（迷宫类）**



成员变量：

* int MazeSize: 迷宫大小。
* Intersection\*\* intsec: 路口数组。
* Stack path: 记录路径的栈。
* int entrX, entrY: 入口坐标。
* int exitX, exitY: 出口坐标。
* int\*\* MazeData: 迷宫数据。

成员函数：

* Maze(int Size): 构造函数，初始化迷宫。
* void setEntr(int x, int y): 设置入口坐标。
* void setExit(int x, int y): 设置出口坐标。
* void printMaze(): 打印迷宫地图。
* bool TraverseMaze(int cx, int cy): 递归遍历迷宫，找到路径。
* void displayPath(): 打印迷宫路径。

**2.3. 主程序设计**

首先输出欢迎信息和测试数据提示，调用dealInputError函数获取迷宫的阶数，创建Maze类对象myMaze，传入迷宫的阶数，并通过用户输入迷宫数组以初始化迷宫。调用printMaze函数，打印初始迷宫地图。然后，调用deal2Error函数获取入口和出口坐标，并调用setEntr和setExit函数设置入口和出口坐标。之后，调用TraverseMaze函数进行迷宫遍历，判断是否有可行路径，如果有可行路径，继续执行后续步骤和，否则输出提示信息并结束程序。有可行路径则调用printMaze函数打印勇闯迷宫后的地图，再调用displayPath函数，打印迷宫路径。最后输出结束提示，等待用户按回车键结束程序。

**2.4. 输入错误处理设计**

**2.4.1. 单个参数的输入错误处理**

函数dealInputError使用 while 循环不断尝试获取用户输入，直到输入满足要求。使用 cin.fail() 及min、max来判断输入是否出错，检查输入是否在有效范围内。如果输入无效，则输出错误信息，清除输入缓冲区并忽略之后的字符，并重新输出输入的提示信息。通过一个字符来获取输入一个数之后的字符以检查输入字符的个数，若不正确，也输出错误信息，并重新输出输入的提示信息，执行相应清除操作。如果输入有效，则跳出循环。

此错误处理可以用在输入迷宫的阶数。

**2.4.2. 两个参数的输入错误处理**

函数deal2Error使用 while 循环不断尝试获取用户输入的两个参数，直到输入满足要求。使用 cin.fail() 及max来判断输入是否出错，检查输入是否在有效范围内。如果输入无效，则输出错误信息，清除输入缓冲区并忽略之后的字符，并重新输出输入的提示信息。通过一个字符来获取输入两个数之后的字符以检查输入字符的个数，若不正确，也输出错误信息，并重新输出输入的提示信息，执行相应清除操作。如果输入有效，则跳出循环。

此错误处理可以用在输入迷宫的入口坐标和出口坐标。

**2.4.3. 迷宫数组的输入错误处理**

与前面两种类似，要求用户只能输入0或1，使用循环嵌套以获取迷宫数组的具体数据，输入内容错误和输入个数超出都会要求用户重新输入迷宫数字。

**3. 功能实现**

**3.1. 主体功能实现（TraverseMaze）**

本项目的关键在于TraverseMaze函数，函数开始时首先检查当前位置 (cx, cy)是否在迷宫范围内。如果当前位置是出口，它标记出口位置，将其推入路径栈，并返回true。这是递归函数的基本情况。如果当前位置是“0”，则将该单元格标记为已访问。然后在四个方向上执行递归DFS：上、左、下、右。它对每个相邻单元格递归调用自身。如果递归调用返回true，表示找到了解路径，将当前位置推入路径栈。如果当前位置不在解路径上，则恢复标记，以允许在递归调用期间进行回溯。如果从当前位置找不到路径，函数返回false。这种DFS方法通过递归遍历每条可能的路径，直到找到解决方案或耗尽所有可能性。在遍历过程中，路径记录在path栈中。

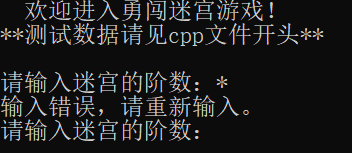
**3.2. Maze初始化函数**

此函数首先初始化迷宫的入口、出口坐标为 -1，迷宫大小为传入的Size。然后分配路口数组和迷宫数据数组，使用new操作符为路口数组intsec和迷宫数据数组MazeData分配内存。然后，使用循环接受用户输入的迷宫数据，同时检查输入的有效性。根据迷宫数据初始化路口数组intsec。左、前、右三个方向的值分别取决于迷宫数据和边界情况。

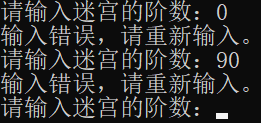
**4. 测试结果（包含边界测试）**

**4.1. 输入迷宫的阶数**

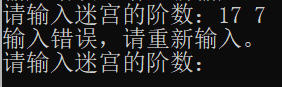
（1）输入的类型非法



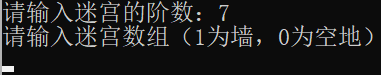
（2）输入的范围非法



（3）输入的个数超过一个

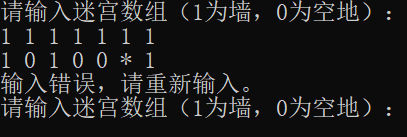


（4）输入正确即可进入下一环节

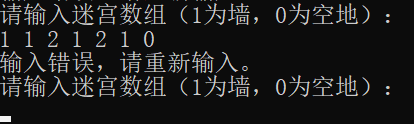


**4.2. 输入迷宫数组**

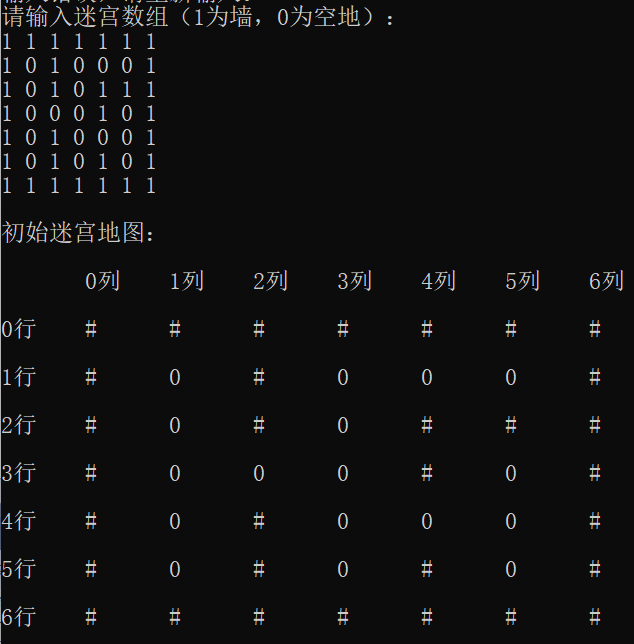
（1）输入的类型错误



（2）输入的内容错误

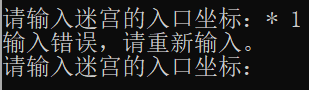


（3）输入正确即可输出初始迷宫地图

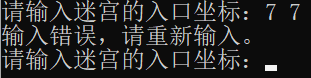


**4.3. 输入迷宫的入口、出口坐标**

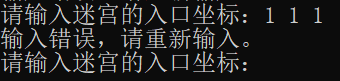
（1）输入的类型错误



（2）输入的范围错误（不在迷宫内）

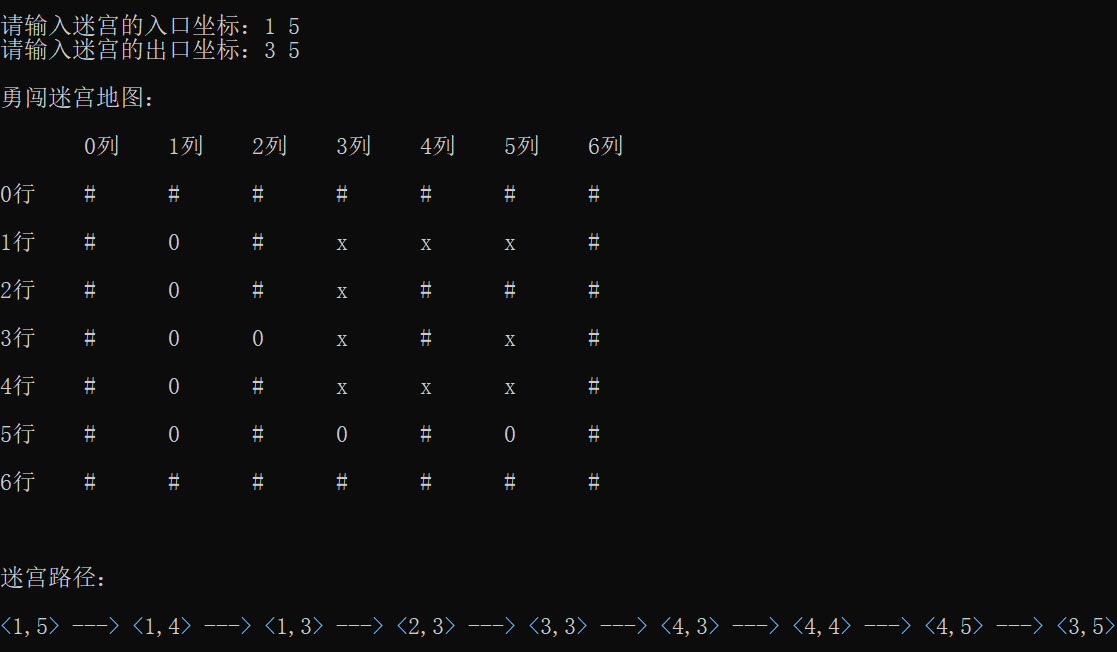


（3）输入的个数超过2个

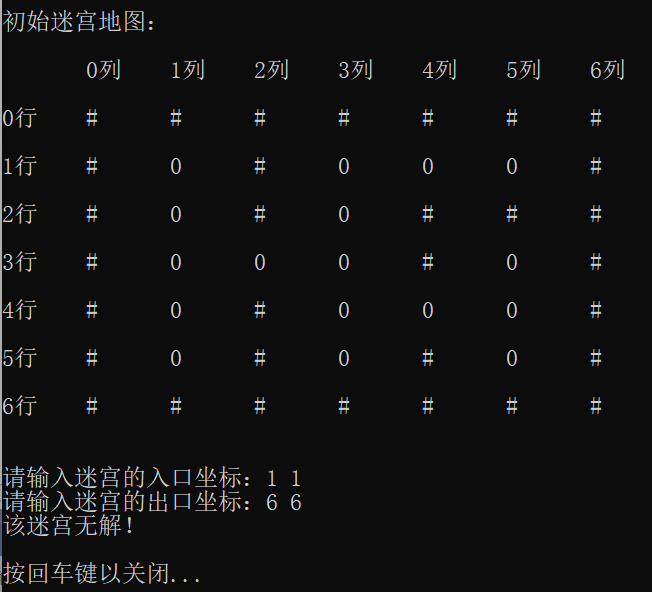


（4）输入正确即可输出勇闯迷宫地图及经过的迷宫路径

a. 迷宫有可以走通的路径



b. 迷宫无可行的路径，则会输出提示信息



**4.4. 程序的测试结果总览（仅展示输入正确有效的情况）**

