**数据结构课程设计**

**-勇闯迷宫游戏**

# 题目描述

1. 项目简介

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. 项目功能要求

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。

# 总体思路

对于迷宫问题有深度优先搜索DFS和广度优先搜索BFS两种寻路算法。根据题意需要用回溯法试探地搜索前进，若前进中受阻，则及时回头纠正错误另择通路继续搜索，这正是DFS的算法思路。由于DFS可能会找出不止一条的线路，因此还需要判断那一条是最短路径，即搜索次数最少的那一条路线输出。

# 代码类设计

源文件还包含了maze类的定义，其中有地图的数组定义，DFS的算法、地图的打印和路径的打印。

Mystack.hpp中包含了自定义的栈容器，和STL标准库的vector类似。

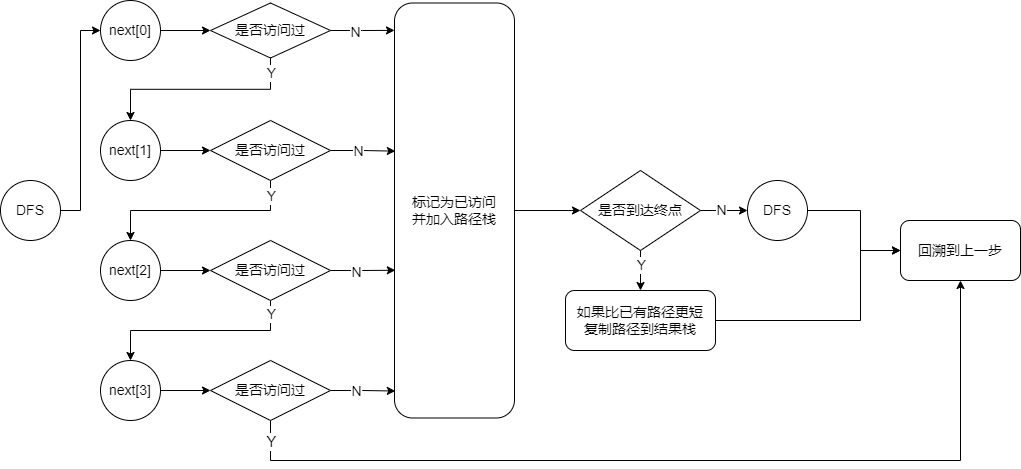
Stack类定义如下：

# 具体实现

## DFS算法

要完成深度优先的搜索，需要一个标记访问记录的数组，一个记录每条路径的栈和一个记录最短路径的栈。其中标记访问的数组可与存储地图的数组并用，我们规定数值为0代表不可访问，即地图中的障碍；数值不为0代表可访问，即地图的道路，其中数值为1代表未访问过，而数值2代表已经访问。

具体步骤为：从当前点出发，向上下左右四个方向分别试探，在确保这一步不会超出地图范围或者是不可通行的单元的前提下，标记该位置为2，并将此点加入到路径栈中，步数计数器加一。如果该点到达终点，则判断当前步数是否更小，如果更小则更新步数最小值并将该路径栈中元素复制到结果路径栈中；如果未到达终点，则以此位置作为当前点进行DFS搜索。对于当前点来说，做完四个方向的遍历，标记此点的mark为1，并从路径栈中弹出此点。以此步骤的流程图如下：



DFS代码如下：

## 打印路径

在DFS过程中，标记为0的是不可通行的墙，而非零则是可通行的道路。而这个非零数可以表示很多信息，我们人为地认定标记值为1代表未访问过的位置，而标记值为2的点代表已访问。而最终由于每个位置都经历了回溯，所以标志值信息都会丢失，因此在某位置访问到了终点，需要把本次记录的所有路径复制到结果栈中，所经过的步数也需要记录，以保证最终结果栈中是步数最少的路线，以便取得最优结果。

在打印有最短路径的地图前，需要将结果栈中的所有步骤在标记数组中展现出来，做法就是遍历结果栈，把其中的某步骤在mark数组中标记为某个统一的非零值，我们可以设定它为2。如此打印地图时，可以根据标记数组不同的取值打印不同的图形，数值为0代表墙体就输出“#”，数值为1就输出未经过的道路“0”，而作为最短路径的标记数2相应输出“x”。而文字线路只需要对结果栈进行顺次输出即可。注意结果栈为空时代表此迷宫无解，需要特判输出。

## k类的实现

栈的数据存储有数组和链表两种形式，这里综合了多种情况选择了由数组作为栈的存储结构。由主程序的需求可知，stack类需要在出入栈操作基础上，额外实现对拷贝构造函数和等于号、输出运算符、下标运算符的重载。如果栈满时需要对栈数组进行扩容，具体实现由函数实现，代码如下：

# 项目小结

1. 性能分析

由于我们并不知道迷宫的求解路线到底有多长，因此采用单一数组来存储可能会浪费大量空间，因此需要一个可扩展的存储空间，但固定尺寸的迷宫所求路径的长度是有个理论范围的，比如m\*n的迷宫，起点在左上角，终点在右下角，最短可以有步的线路，最长可以有大约步的线路，因为每个位置最多访问一次。因此对数组扩展要求不是那么频繁，链表结构的栈对元素一个个的扩增有很好的优势，但线路步数的取值的范围较大，一次扩增更多的空间会使得效率更高，因此数组型栈结构更佳。对此，空间复杂度为，k为步数在浮动，虽然会空间有所浪费，但优于链表的两倍存储。时间复杂度方面比较难以确定，原因在于递归的次数完全取决于迷宫的构造。但内部循环的次数是一定的，因为路径上的每一个位置不会重复访问，时间复杂度为。假设迷宫中的道路数量为p，那么最多p个位置可以标记，而，因此内部循环的计算时间为。

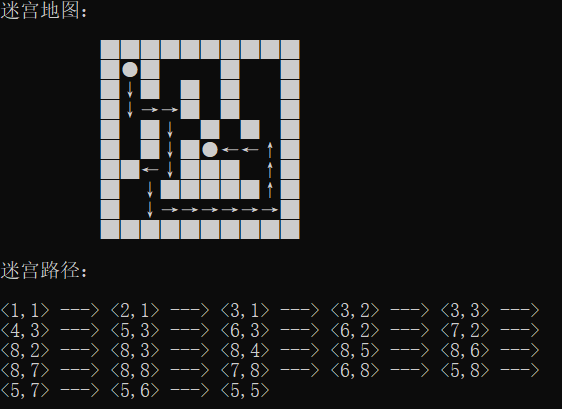
1. 代码优化

一般来说，递归的算法都可以写成非递归的形式。而DFS非递归算法本质是使用栈来模拟递归的实现过程。一旦需要回退，可以从栈中取得刚才走过位置的坐标和前进方向。当在迷宫中向前试探时，可能同时存在几个允许的前进行方向。利用point结构体存储了当前位置和上一步前进方向，根据next数组前进，并将活动记录退栈，以在前进路上回退一步在尝试其他允许的方向。如果栈为空则表示已经回溯到开始位置。

1. 其他方法BFS

除了深度优先搜索，还可以用广度优先搜索。广度优先搜索由于每一步都把所有方向全部进行，一级一级向下推进，因此找到终点时所得路径就是最短路径。由于BFS搜索时所有点都会经过，在到达终点前无法知道哪条路径是答案，因此可以从终点往起点再进行一次BFS，如果遇到上一次搜索标记的路径则记录下来。或者每个位置多存一个维度，表示当前状态的上一个状态，最后从终点一步一步回溯上去即可得到路径。

# 程序功能测试



更改地图为非通

