勇闯迷宫游戏 **by 1352847 JoshOY**

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。

解决方案

使用深度优先搜索（DFS）：

深度优先搜索算法（Depth-First-Search），是搜索算法的一种。是沿着树的深度遍历树的节点，尽可能深的搜索树的分支。当节点v的所有边都已被探寻过，搜索将回溯到发现节点v的那条边的起始节点。这一过程一直进行到已发现从源节点可达的所有节点为止。如果还存在未被发现的节点，则选择其中一个作为源节点并重复以上过程，整个进程反复进行直到所有节点都被访问为止。属于盲目搜索。

深度优先搜索是图论中的经典算法，利用深度优先搜索算法可以产生目标图的相应拓扑排序表，利用拓扑排序表可以方便的解决很多相关的图论问题，如最大路径问题等等。

代码如下：

```
bool dfs(int x, int y)
{
    if ((x == endPosX) && (y == endPosY)) {
        steps.push_back({x, y});
        return true;
    }

    bool canGoLeft = true, canGoRight = true, canGoUp = true, canGodown = true;
    if (stepsSet.count({ x - 1, y }) || (maze[x - 1][y] == false) ) {
        canGoUp = false;
    }
    if (stepsSet.count({ x + 1, y }) || (maze[x + 1][y] == false)) {
        canGodown = false;
    }
    if (stepsSet.count({ x, y - 1 }) || (maze[x][y - 1] == false)) {
        canGoLeft = false;
    }
    if (stepsSet.count({ x, y + 1 }) || (maze[x][y + 1] == false)) {
```

```
        canGoRight = false;
    }

    steps.push_back({ x, y });
    stepsSet.insert({ x, y });
    if (canGoRight) {
        if (dfs(x, y + 1)) {
            return true;
        }
    }
    if (canGodown) {
        if (dfs(x + 1, y)) {
            return true;
        }
    }
    if (canGoLeft) {
        if (dfs(x, y - 1)) {
            return true;
        }
    }
    if (canGoUp) {
        if (dfs(x - 1, y)) {
            return true;
        }
    }
    steps.pop_back();
    stepsSet.erase({ x, y });
    return false;
}
```