# 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。入口进入迷宫，迷宫设置很多障碍，需要在迷宫中寻找通路以到达出口。

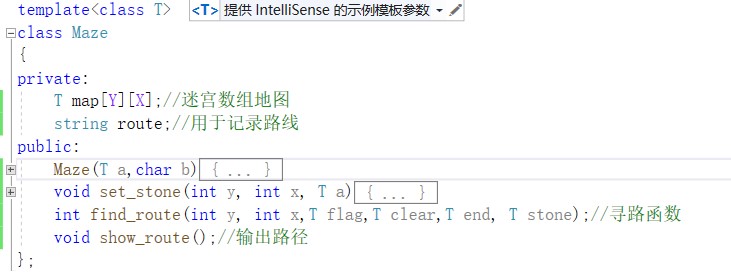
迷宫问题的求解过程可以采用回溯法即在一定的约束条件下试探地搜索前进，若前进中受阻，则及时回头纠正错误另择通路继续搜索的方法。从入口出发，按某一方向向前探索，若能走通，即某处可达，则到达新点，否则探索下一个方向；若所有的方向均没有通路，则沿原路返回前一点，换下一个方向再继续试探，直到所有可能的道路都探索到，或找到一条通路，或无路可走又返回入口点。在求解过程中，为了保证在达到某一个点后不能向前继续行走时，能正确返回前一个以便从下一个方向向前试探，则需要在试探过程中保存所能够达到的每个点的下标以及该点前进的方向，当找到出口时试探过程就结束了。

## 2 程序说明

## 2.1 数据结构设计：

结合老师课堂要求，此题的重点在于练习递归算法，而为确保迷宫存在可行路径，迷宫地图已在程序内部设定成固定模式。但是，为了尽可能大地保留程序的灵活性，本程序尽量采用了宏定义与类模板来提升程序的灵活性。具体说明如下：

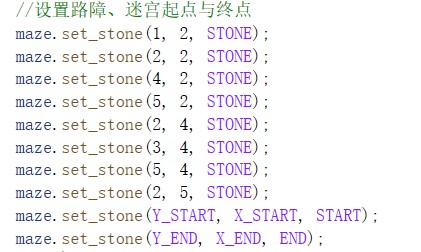
①迷宫类模板：



其中：

二维数组map用于存储迷宫地图，而string字符串用于存储路线，其格式为：第一个字符是 \* ，用于标识路线开头，接着两两一组，分别是路线每一步的y坐标与x坐标。

构造函数仅仅是利用传入的参数初始化类的数据成员，set\_stone函数用于在(x,y)坐标设定路障，将这一功能封装在类中，若有修改迷宫地图的需要，只要调用此函数，传入对应坐标与该坐标的标识符即可。如：main.cpp中的此段程序设定了本程序中的迷宫地图。



## 屏幕截图 2021-12-17 213115

对应效果

## show\_route函数只是根据当前的map数组与string字符串输出相应的迷宫地图与路线

此外，程序中尽可能采用了宏定义以增加程序中迷宫的各种元素的可修改性与灵活性

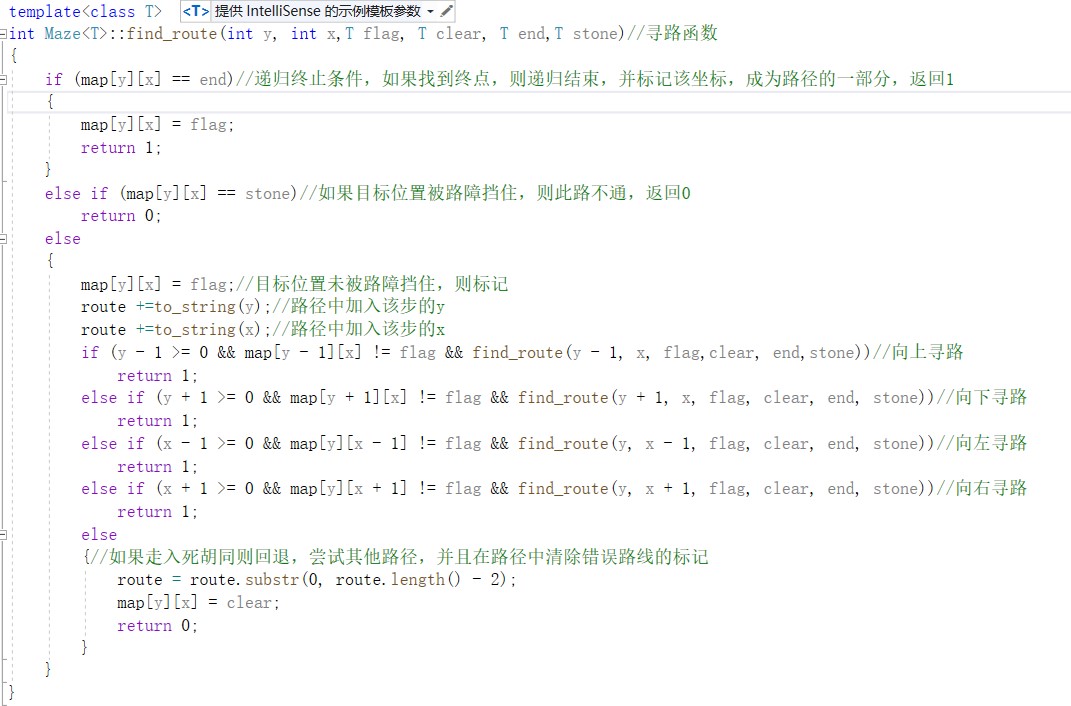


## 2.2 寻路函数算法设计：

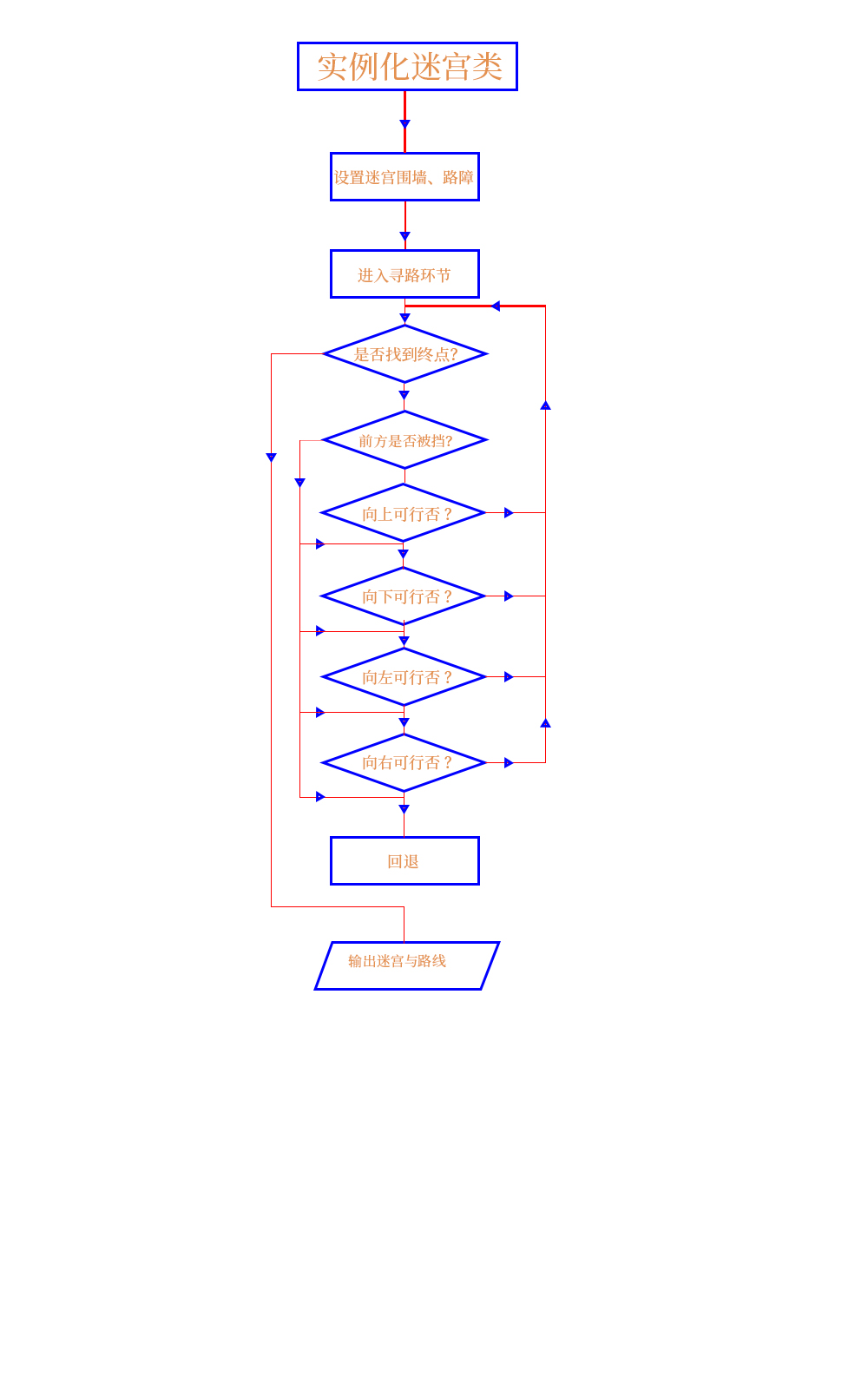
寻路算法是本程序中的**核心代码**，采用递归方式实现：

基本思路是：首先明确递归终止条件：map[y][x]==end,如果找到终点，则递归结束，并标记该坐标，成为路径的一部分，返回1；其次，明确路径回退条件：map[y][x]==stone,如果目标位置被路障挡住，则此路不通，返回0。那么接下来，详细的递归流程是：目标位置未被路障挡住，则标记，接着路径中加入该步的y坐标，加入该步的x坐标。然后分别向上、下、左、右四个方向寻路，需要注意的是，一旦其中一个方向走得通，就会一直走该方向，而不是顺序执行其他方向，这是递归过程中需要注意的代码执行逻辑顺序问题。但如果在某一坐标遍历了四个方向，那就说明走入了死胡同，需要执行回退操作：首先在存储路径的string数组中清楚该位置的坐标，接着在迷宫数组中取消该位置的路线标识符。那么在这样的递归逻辑下，最终会在迷宫中找到可以通行的路线中的其中一条。

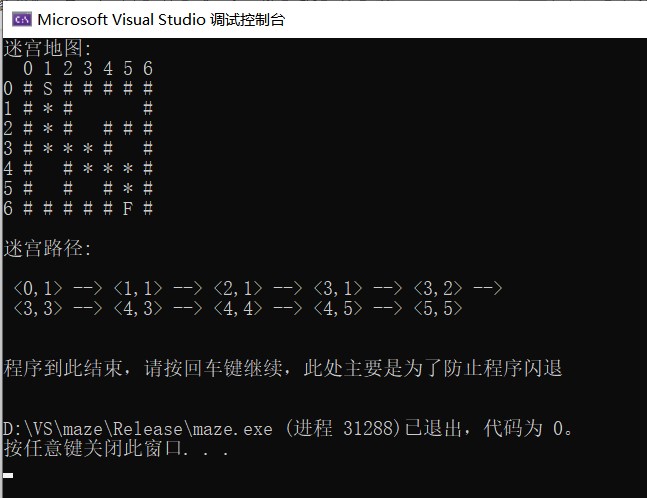
此外，需要注意的是：每个方向的递归之前务必要先判断若考察下一个坐标，那么下一个坐标是否越界数组，接着再判断下一个坐标是否已经走过，最后在调用递归函数进行考察，这三者的顺序在if条件中不可颠倒，这是出于短路运算的原因，一旦前面的条件不满足，就不会执行后面的判断。因此，在本程序中，应先判断是否越界，若先判断是否走过，会产生数组越界的错误，若直接上来就首先利用递归判断，则同样也会产生数组越界问题。



## 主程序流程图如下：

****

**4.测试**



说明：

# 表示的是路障

\* 表示的是路线

S 表示的是路线起点

F 表示的是路线终点

另外，此程序中已加入防闪退功能。