数据结构：

数组：数组类使用动态分配维度、大小的形式，保存模板化的数据类型，拥有成员函数：

Status InitArray(int nDim, ...); ///初始化数组

Status DestroyArray(); ///销毁数组

Status Locate(int nDim, va\_list ap); ///定位下标指向的元素在数组中的位置

Status Assign(ElemType \*elm, ...); ///数组赋值

Status Value(ElemType \*elm, ...); ///数据取值

该类可通过可变数量参数调用，实现最大八维数组的按列优先存储与引用，所存取数据类型可根据模板自行选择。

本次实验中采用了二维整型数组，用于存储迷宫地图。

栈：栈类使用动态分配存储空间大小的形式，保存模板化的数据类型，拥有成员函数：

Status InitStack(); ///初始化栈

Status DestoryStack(); ///销毁栈

Status Push(ElemType e); ///元素e入栈

Status Pop(ElemType &e); ///栈顶元素出栈，保存在e中

Status Top(ElemType &e); ///获取栈顶元素，保存在e中

Status Empty(); ///检查栈是否为空

该类可以动态分配栈的大小，既能保存大量数据，也不会浪费内存空间，所存取数据类型可通过模板自行选择。

本次实验中所使用栈的初始大小设置为100\*数据类型大小，存满时每次增加10\*数据类型大小，栈中存储的数据类型为自定义结构体 stackpoint，存储迷宫点的行列坐标与路径中前往下一个节点的方向。

队列：队列类使用循环队列的形式，保存模板化的数据类型，拥有成员函数：

Status InitQueue(); ///初始化队列

Status DestoryQueue(); ///销毁数组

Status Push(ElemType e); ///元素e入队

Status Pop(ElemType &e); ///队首元素出队，保存在元素e中

Status Empty(); ///检查队是否为空

该类通过循环使用数组空间的形式充分利用空间，保存最大不超过数组大小的数据，所存取数据类型可通过模板自行选择。

本次实验中所使用的队列保存数据的最大空间设定为1000\*数据类型大小，队列中所存取的数据类型为自定义结构体 queuepoint，存储迷宫点的行列坐标，走到目前为止的路径长度，路径中每个点的哈希值。

算法：//算法默认输入数据起点和终点不是同一点且起点可走

DFS：利用栈模拟递归DFS，找到一条迷宫路径。算法开始创建保存数据类型为stackpoint的栈，保存目前所走的路径中的节点，每个节点保存迷宫点的行列坐标与路径在该点前进的方向。栈初始存入迷宫起点，进入whie循环开始寻找路径，若栈中没有剩余节点结束循环，若此时还是没有任何路径，输出找不到路径。每次循环开始时出栈栈顶元素保存为当前节点，修改前进方向，若有剩余可走的方向，则重新将该节点入栈，修改迷宫上的该节点为不可走，并替换当前节点为前进方向的下一个节点，重新选择方向，否则进入下一次循环；如果找到终点，则逆序输出栈中所有元素，通过保存的方向确定路径轨迹，销毁栈，算法结束。

BFS：通过使用循环队列找到所有合法的迷宫路径，并按由短到长的顺序输出。算法开始时创造保存数据类型为 quuepoint的节点，每一轮循环入队队首元素可以到达的所有节点，每个节点保存了迷宫点的行列坐标，该节点所在路径的长度，与所有该路径走过节点的哈希值。队列初始入队起点，然后进入while循环寻找所有可能的路径。每次循环出队队首元素保存为当前节点，若当前节点为终点，找到的路径值加一，并根据节点中保存的路径哈希值解码出路径的每个节点，根据节点坐标反推出前进方向，输出路径，进入下一次循环；否则考察当前节点的所有前进方向，若方向可走，则创建节点保存可走节点，计算坐标哈希值，并将路径长度设置为目标节点路径长度加一，入队该节点。直到队列中没有任何节点时退出循环，销毁队列，算法结束。