****

**数据结构课程实践1-迷宫设计实习报告**

**题目：编制一个求解迷宫的程序**

**班级：19184115**

**专业：计算机科学与技术（人工智能与大数据）创新实验班**

**代码 姓名：洪智豪 学号：19195216**

**文档 姓名：周建屹 学号：19151135**

**完成日期：2020.10.16**

1. **需求分析**

* **总需求：输入或生成一个由二维数据表示的迷宫，实现自动寻路并且输出对应的路径**

**具体功能如下**

1. 通过用户规定迷宫的长度与宽度
2. 可以自动随机生成迷宫或者选择由用户以TXT文本的形式导入迷宫
3. 实现走迷宫路径的动态显示，实时显示走迷宫的状态，让走迷宫可视化
4. 当迷宫存在多路径时，可以输出全部路径
5. 当存在多条路径时，可以输出同等级的最优解
6. 可以让用户选择遍历序列的顺序，优先向哪个方向走
7. **具体需求实现设计**
8. 通过用户规定迷宫的长度与宽度

1）为防止迷宫过小或过大 规定用户输入的数组范围为（5~10）

2) 若输入非法则可以重新输入而不至于退出程序

1. 实现随机生成迷宫以及文件导入
2. 起点规定为（1，0），终点规定为（M-2，N-1）
3. 在运行时输入迷宫的长宽后显示选择界面，用户可以自行选择自行生成迷宫或者导入根目录下的input.txt文件
4. 若文件导入迷宫与设定的迷宫长宽不一致，则退出程序
5. 实现走迷宫的动态显示，让走迷宫可视化
6. 在用户输入迷宫后可以选择动态显示1条路模式进入动态显示功能
7. 用实心方块■表示墙 空心方块□表示路 X表示已知的不通的路
8. 用箭头表示实时走迷宫的方向
9. 通过延时刷新迷宫来展示人走迷宫的实时状态
10. 最后再输出具体走过的路径供用户参考
11. 检测有无随机生成或者地图导入 若无则返回用户界面

（4&5）当迷宫存在多条路径时输出全部路径或最短路径

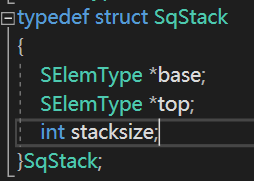
1. 用户可以在输入迷宫后可以选择输出多条路径或者输出最短路径
2. 选择多条路径后 显示所有的走法(只显示10条，过多的省略)
3. 选择最短路径后 显示所有最短的走法
4. 检测有无随机生成或者地图导入 若无则返回用户界面

（6） 可以让用户选择遍历序列的顺序，优先向哪个方向走

1. 在选择解决迷宫的模式后可以自定义遍历顺序
2. 用户只需按顺序输入4个对应方向的编号即可
3. 若输入非法则可以重新输入序号而不至于退出程序
4. **概要设计**
5. 数据结构设计

**这次采用的基本数据结构是顺序栈 实现了以下功能**

1. 基本的结构体设计



1. 获取栈顶元素

Status GetTop(SqStack \*s,SElemType \*e)

1. 构造一个空栈

Status InitStack(SqStack \*s)

1. 元素出栈

Status Pop(SqStack \*s,SElemType \*e)

1. 元素入栈

Status Push(SqStack (\*s),SElemType e)

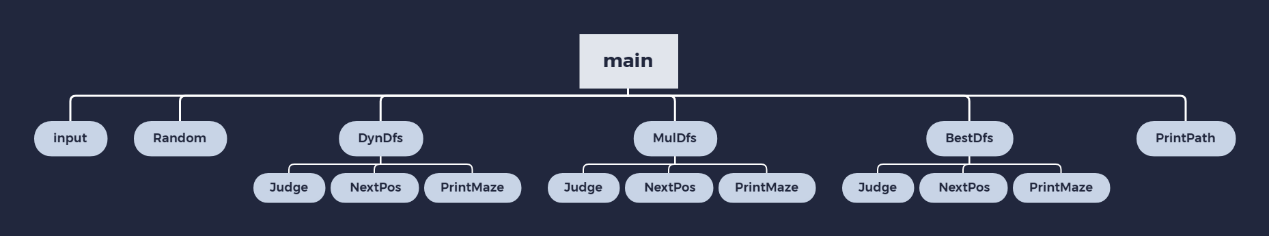
1. 判断栈是否为空

Status StackEmpty(SqStacks)

1. 清空栈中所有元素

Void StackClear(SqStack \*s)

1. 模块设计
2. 模块结构图



1. 函数说明

项目中包含了以下函数

1. void Random()//随机生成迷宫的函数

生成Maze[长][宽]的数组来代表迷宫

1. Status Judge(PosType e) //判断是否可走  
   输入待判定合法性的节点e 若合法返回OK（1）否则返回OVERFLOW(0)
2. PosType NextPos(PosType e,int dir) //确定下一步的节点

在函数中定义了1个新的节点E用于返回，输入当前节点e以及代表方向的标记dir，返回下一个可能合法的节点

1. void PrintPath(SqStack \*s)//输出路径

输入栈s将其中的节点元素以路径的方式打印

1. void PrintMaze()//打印迷宫

调用Maze[][]以迷宫的格式打印

1. void DynDfs(PosType start,PosType end,SqStack \*s) //深搜并动态显示路径

输入起点start以及终点end以及用于储存的栈s

进行寻路并动态显示路径

若寻路成功则将路径压栈

若失败则退出函数

1. void MulDfs(PosType start,PosType end,SqStack \*s) //深度搜索所有路径

输入起点start以及终点end以及用于储存的栈s

进行寻路并记录长度

若失败则退出函数

若寻路成功则将所有路径压栈

最后输出全部路径以及迷宫走法

若失败则退出函数

1. void MulDfs(PosType start,PosType end,SqStack \*s) //深度搜索所有路径

输入起点start以及终点end以及用于储存的栈s

进行寻路并记录长度

若寻路成功则将所有路径压栈

最后输出全部路径以及迷宫走法

若失败则退出函数

1. void BestDfs(PosType start,PosType end,SqStack \*s,int len)

//深度搜索最短路径

输入起点start以及终点end用于储存的栈s且将len当作0输入

进行寻路并记录长度

若寻路成功则将所有路径压栈 并记录其长度

最后输出全部最短路径以及迷宫走法

若失败则退出函数

1. void main()

//主函数 打印用户界面 同时通过调用其他函数完成用户的需求

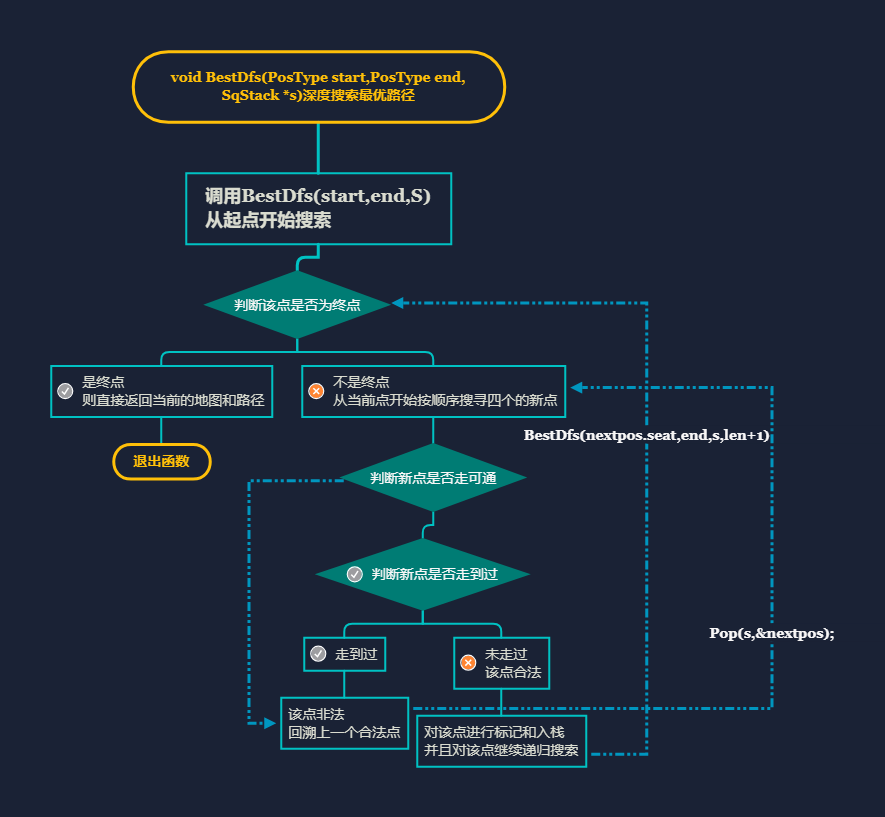
实现用户界面的打印，接受用户输入实现与用户的基本交互

以及调用相应的函数实现用户所需的功能

1. **详细设计**

**（1）程序流程图**

**void BestDfs(PosType start,PosType end,SqStack \*s,int len)**

****

**（2）详细设计**

**BestDfs（PosType start,PosType end,SqStack \*s）**

**用深度优先搜索的方式递归搜寻全部可行且路径最短的路径**

**从起点开始搜索**

**BestDfs(start,end,&S);**

**如果该点是终点，那么直接返回当前的地图和路径。**

**if(start.x==end.x&&start.y==end.y)**

**{**

**PrintMaze();**

**printf("第%d条路径已找到！\n",++count);**

**PrintPath(s);**

**puts("");**

**return;**

**}**

**如果该点不是终点，从当前点开始按顺序搜寻四个的新点。**

**for(int i=1;i<=4;i++) nextpos.seat=NextPos(start,i);**

**判断新点有没有走到过与是否可通。**

**if(!Judge(nextpos.seat)) continue;如果该点是合法的，那么接着对该点进行递归搜索。dfs(nextpos.seat,end,s);**

**如果该点合法，对该点进行标记和入栈，接着递归搜索。**

**vis[start.x][start.y]=i;**

**vis[nextpos.seat.x][nextpos.seat.y]=1;**

**Push(s,nextpos);**

**dfs2(nextpos.seat,end,s,len+1);**

**在搜索结束后进行回溯。**

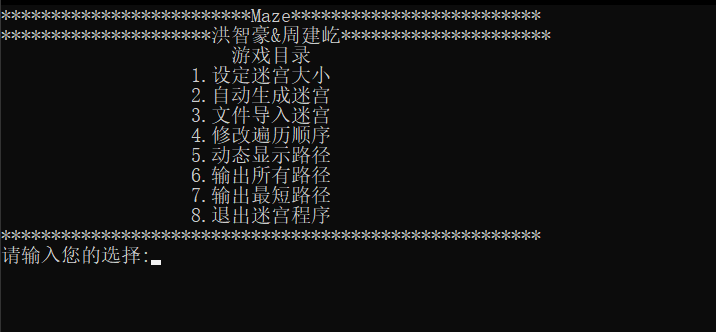
**vis[nextpos.seat.x][nextpos.seat.y]=0;**

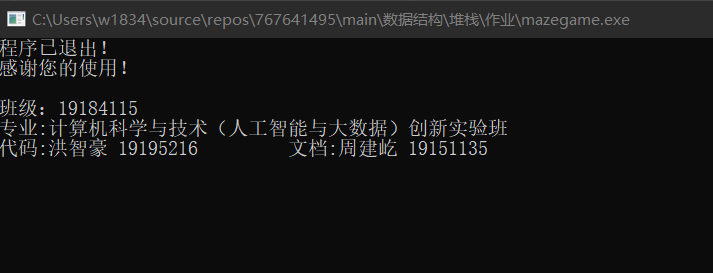
**Pop(s,&nextpos);**

1. **程序流程**

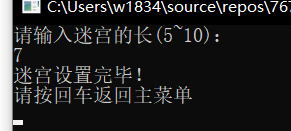
****

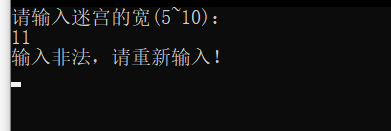
1. **用户手册**

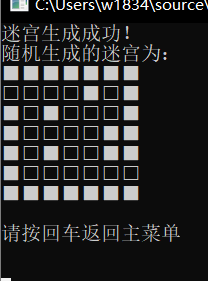




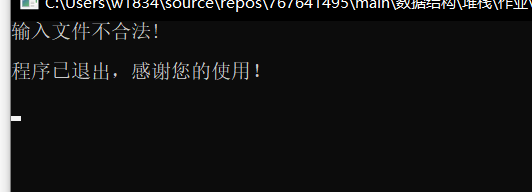
1. **测试结果**



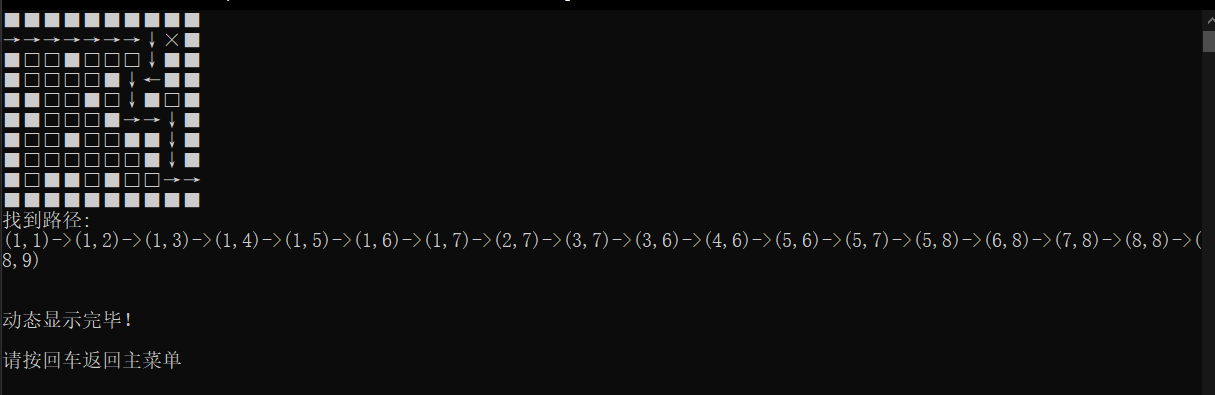


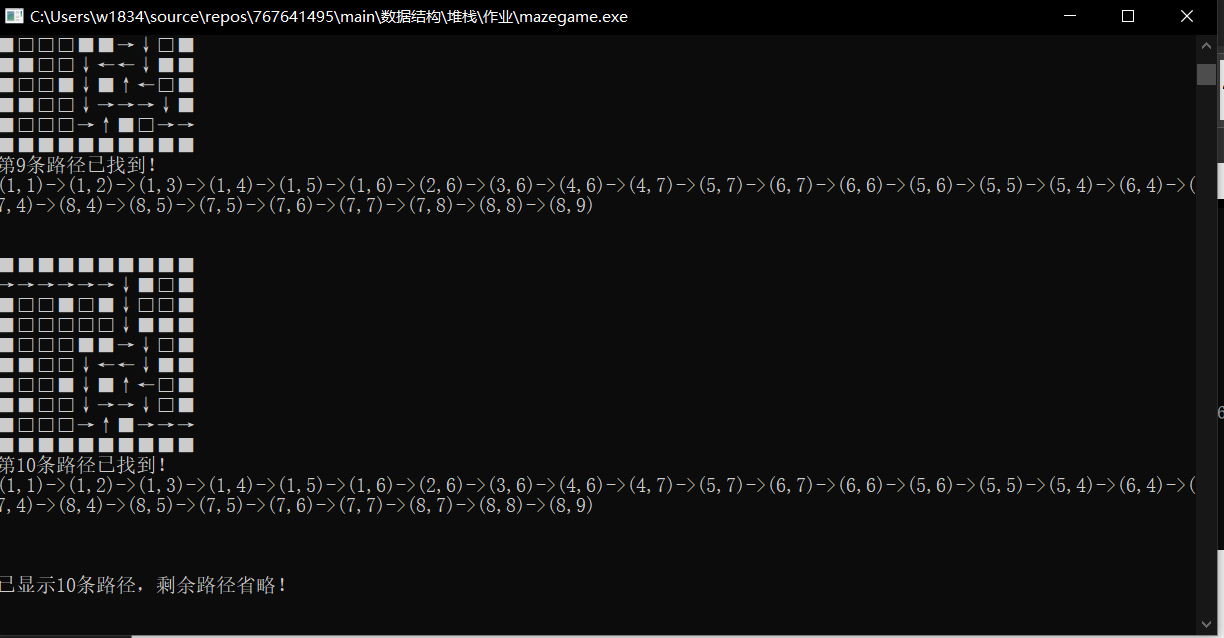
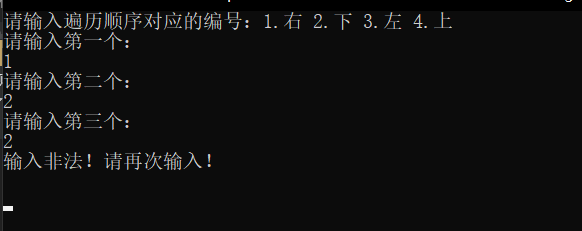


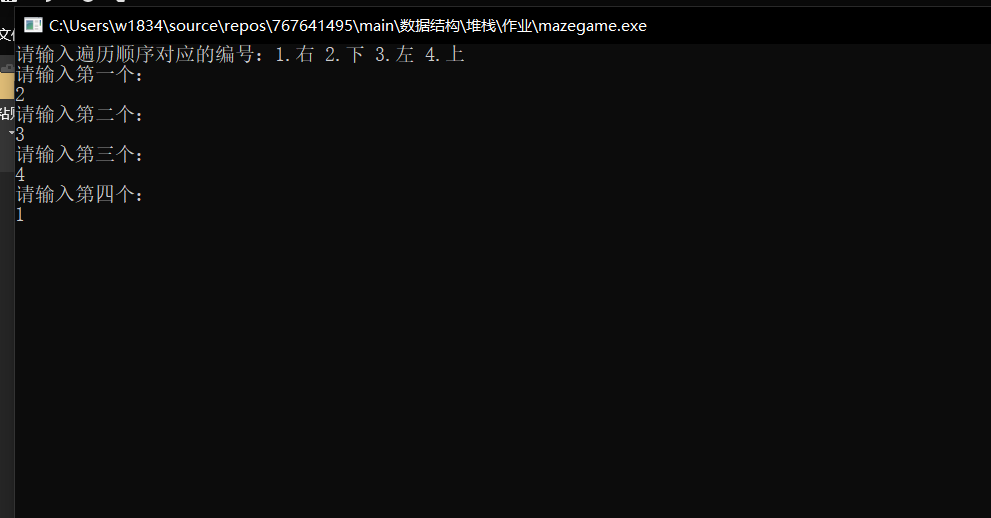




1. 



1.   
   
2. 
3. 



1. **附录**

