**《动感迷宫》实习报告（19151135 周建屹）**

**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**

**姓名：周建屹 学号：19151135**

**完成日期：2020.10.20**

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. 进入主菜单后输入1进入设定迷宫长宽模块
3. 限制长宽的合法输入为[5,10]
4. 若输入非法请重新输入，而非直接退出程序
5. 实现随机生成迷宫以及文件导入
6. 起点规定为（1，0），终点规定为（宽度-2，长度-1）
7. 在输入1设定迷宫的长宽后，用户可以输入2自行生成迷宫
8. 用户在主菜单可以输入3导入根目录下的input.txt迷宫文件
9. 文件中规定 1 为墙 0为路 且输入的长宽均控制在[5,10] 无其他数据

长宽会由程序自动判断

1. 实现走迷宫的动态显示，让走迷宫可视化
2. 在用户输入迷宫后可以选择输入5进入动态显示功能
3. 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
4. 用箭头表示实时走迷宫的方向
5. 最后再输出具体走过的路径供用户参考

（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入6输出所有路径
2. 用户可以在输入迷宫后可以选择输入7输出最短的路径
3. 选择所有路径后 输入地图显示所有的走法 以及相应走法的路径

(只显示10条，过多的省略)

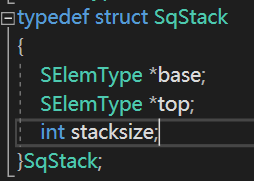
1. 选择最短路径后 输入地图显示所有最短的走法 以及相应走法的路径  
   (只显示10条，过多的省略)

（6） 可以让用户选择遍历序列的顺序，优先向哪个方向走

1. 用户可在主菜单输入4来修改遍历序列
2. 用户只需按顺序输入4个对应方向的编号即可
3. 合法输入为4个方向均有出现且只出现一次
4. 若输入非法则可以重新输入序号重新指定
5. **概要设计**
6. 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)

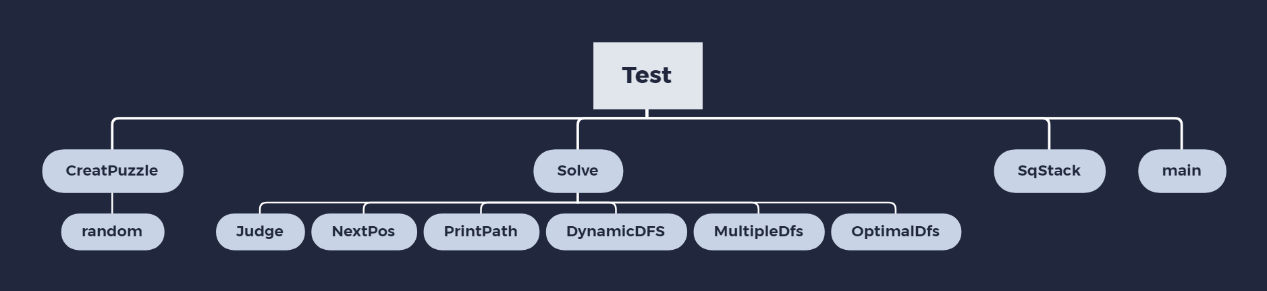
1. 判断栈是否为空

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

start：起点 end：终点 s：用于储存的栈

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

start：起点 end：终点 s：用于储存的栈

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

start：起点 end：终点 s：用于储存的栈

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

start：起点 end：终点 s：用于储存的栈 len:控制模式

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

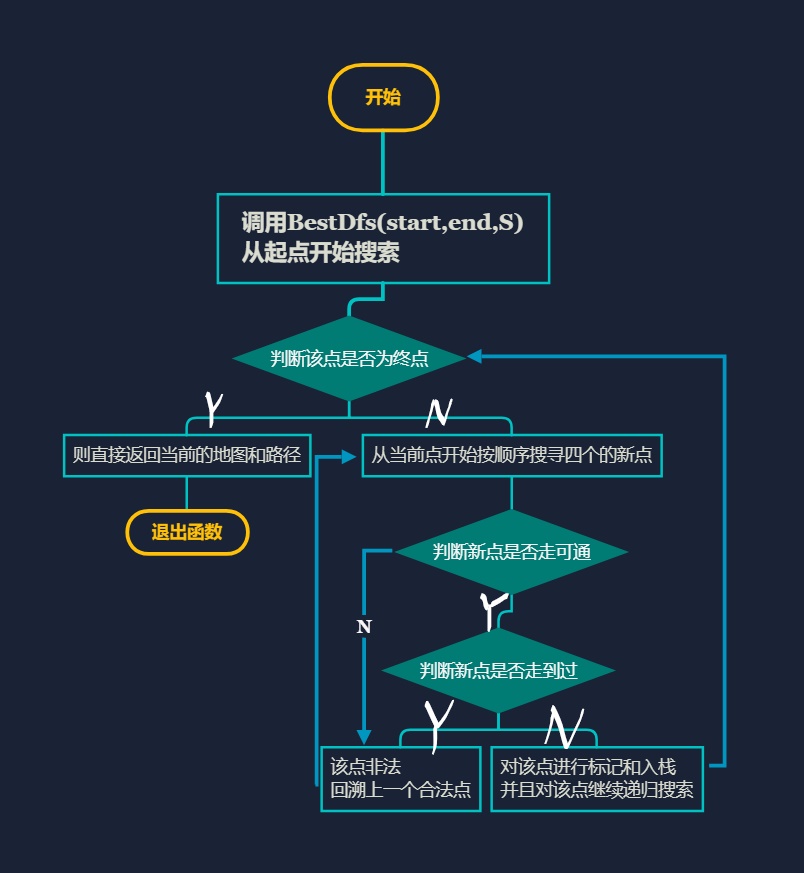
1. void main()

//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**
2. 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);

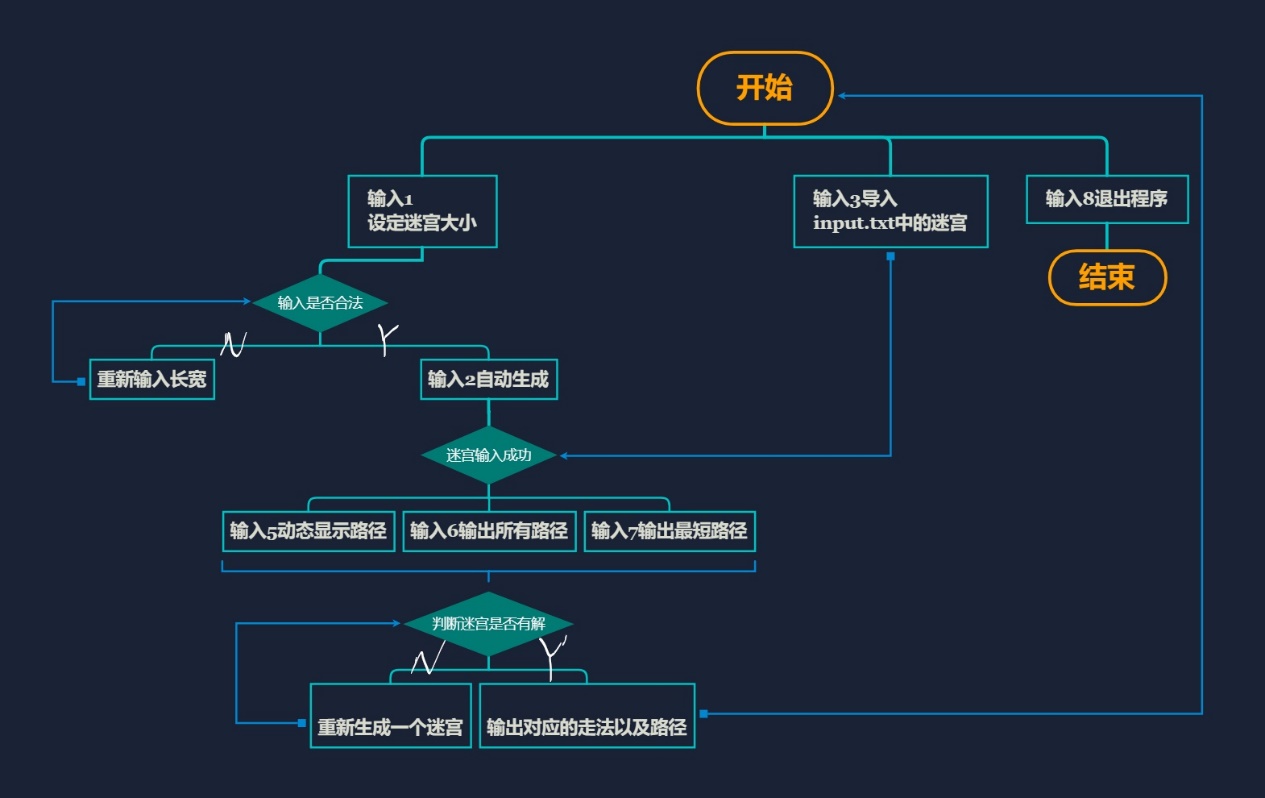
dfs2(nextpos.seat,end,s,len+1);

在搜索结束后进行回溯。

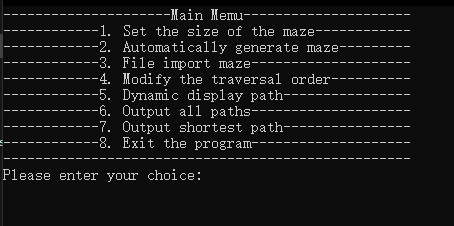
vis[nextpos.seat.x][nextpos.seat.y]=0;

Pop(s,&nextpos);

1. **程序流程**



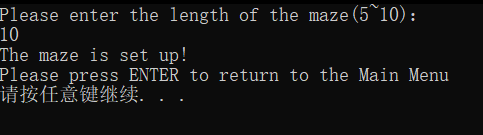
1. **用户手册**



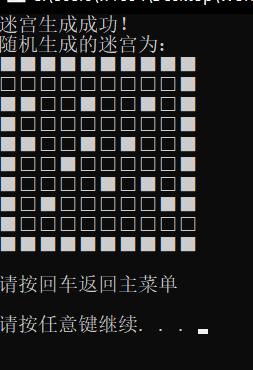
**用户可以输入[1,8]的数字选择相应的功能**

1. 输入1进入设定迷宫长宽模块

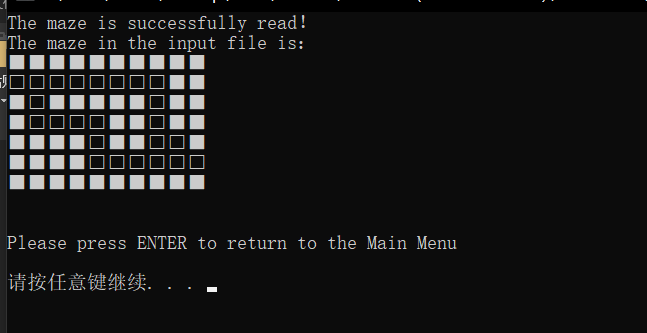
输入合法的宽 长



1. 设定迷宫长宽后输入2随机生成迷宫



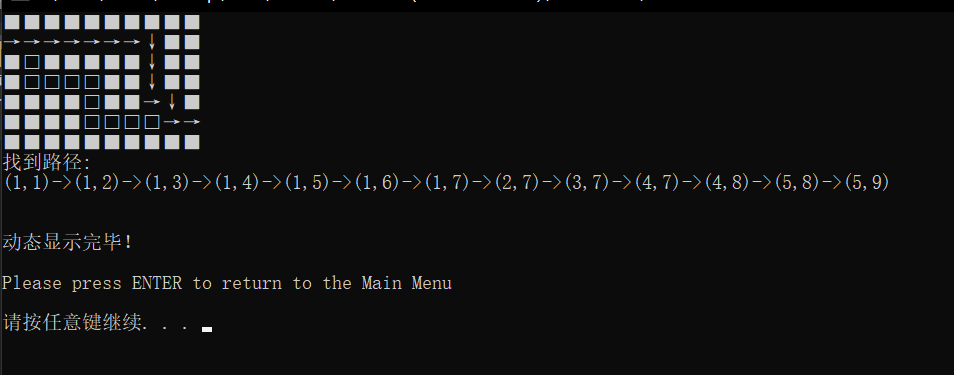
1. 输入3可以导入文件中的迷宫



1. 输入4可以修改遍历序列

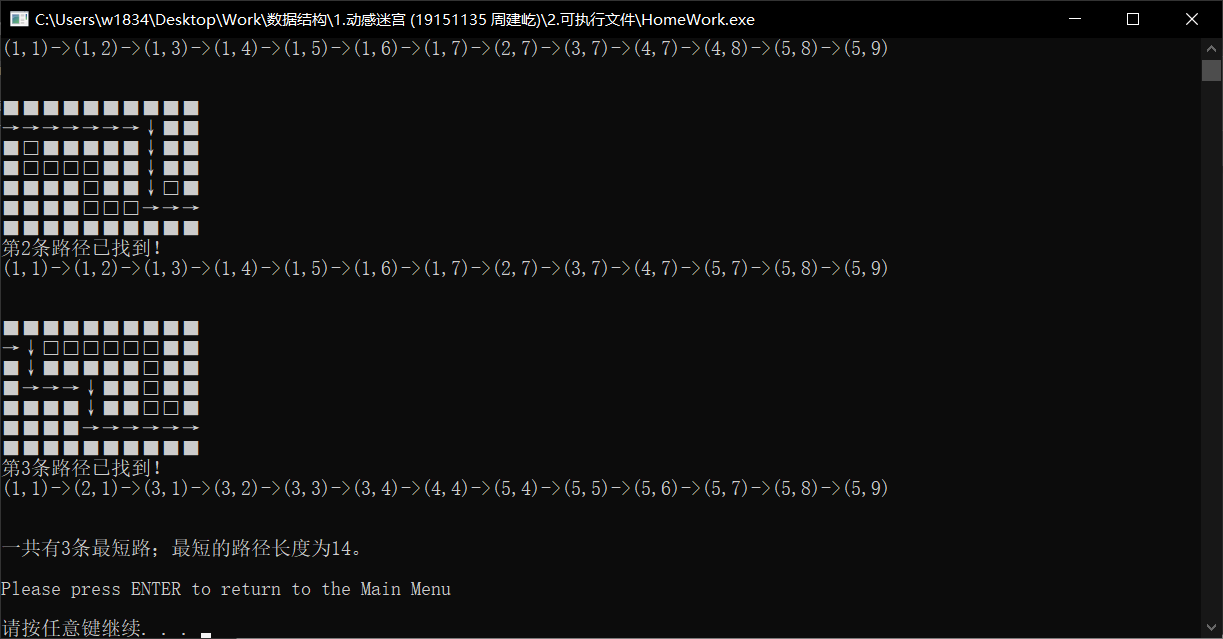


1. 输入5 动态显示迷宫路径



1. 输入6显示全部路径



1. 输入7显示全部路径  
   
2. 输入8退出程序



1. **测试分析**
2. 通过用户规定迷宫的长度与宽度
3. 输入用例：

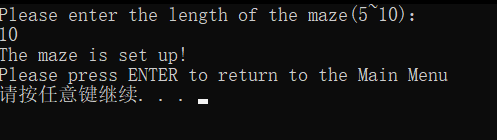
1 10 10

期望结果：

The maze is set up!

Please press ENTER to return to the Main Menu

1. 实测结果



1. 结论：测试通过
2. 随机生成迷宫
3. 输入用例：

2

期望结果：

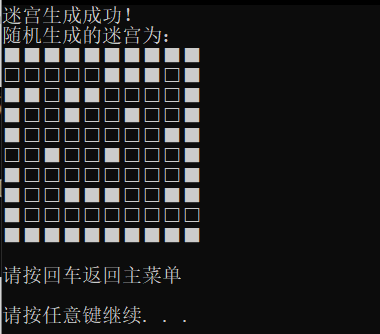
迷宫生成成功！

随机生成的迷宫为：

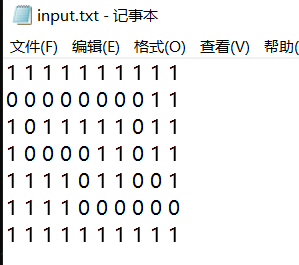
\*\*\*

请按回车键返回

1. 实测结果



1. 结论：测试通过
2. 文件导入
3. 输入用例



期望结果：

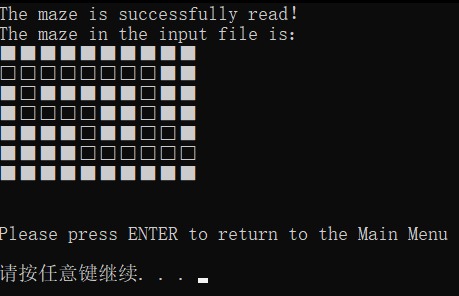
The maze is successfully read！

The maze in the input file is：

\*\*\*

Please press ENTER to return to the Main Menu

1. 实测结果



1. 结论：测试通过
2. 实现走迷宫的动态显示，让走迷宫可视化
3. 输入用例：

5

期望结果：

\*\*\*\*（动态显示走迷宫）

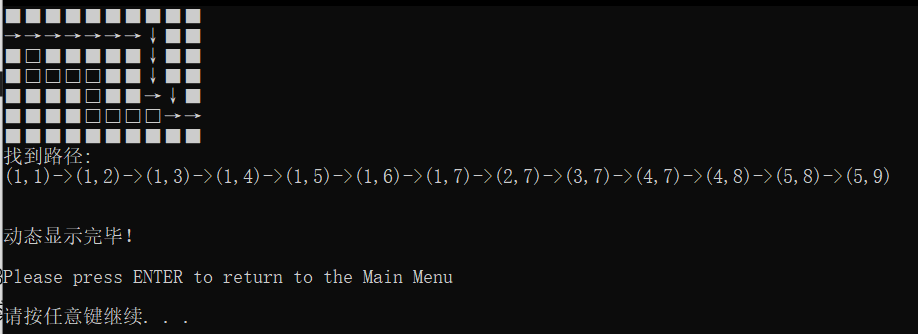
找到路径：

\*\*\*（路径）

动态显示完毕！

Please press ENTER to return to the Main Menu

1. 实测结果



1. 结论：测试通过
2. 当迷宫存在多条路径时输出全部路径路径
3. 输入用例：

6

期望结果：

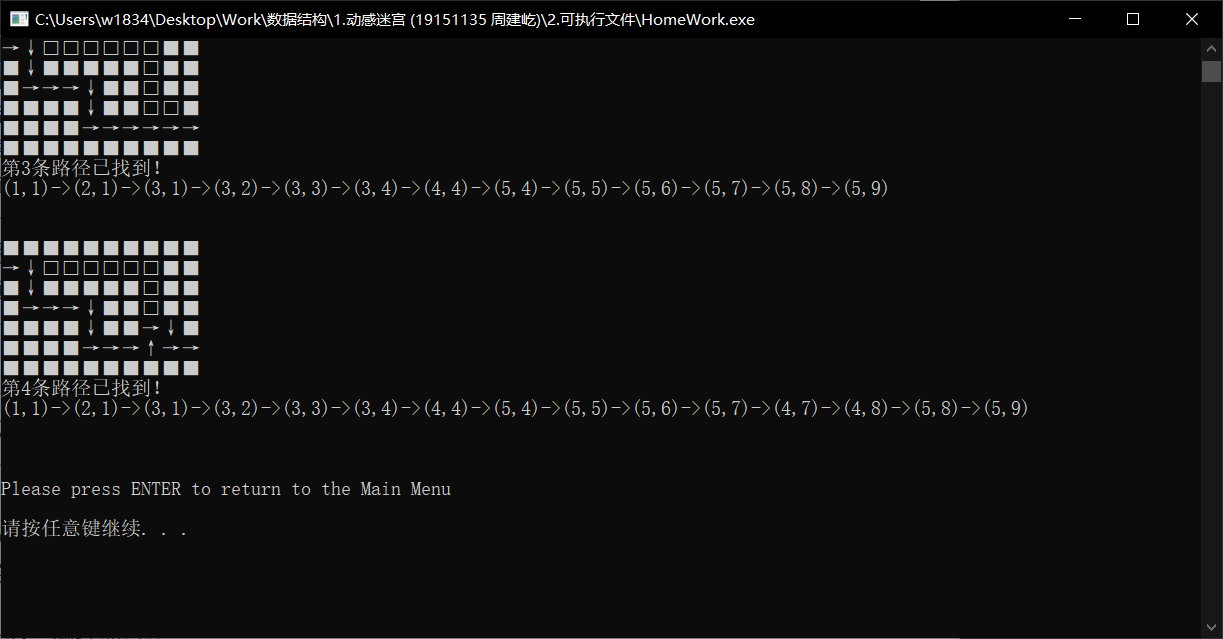
\*\*\*（迷宫）

第\*条路径已找到

\*\*\*（路径）

Please press ENTER to return to the Main Menu

1. 实测结果



1. 结论：测试通过
2. 当迷宫存在多条路径时输出全部最短路径
3. 输入用例：

7

期望结果：

\*\*\*（迷宫）

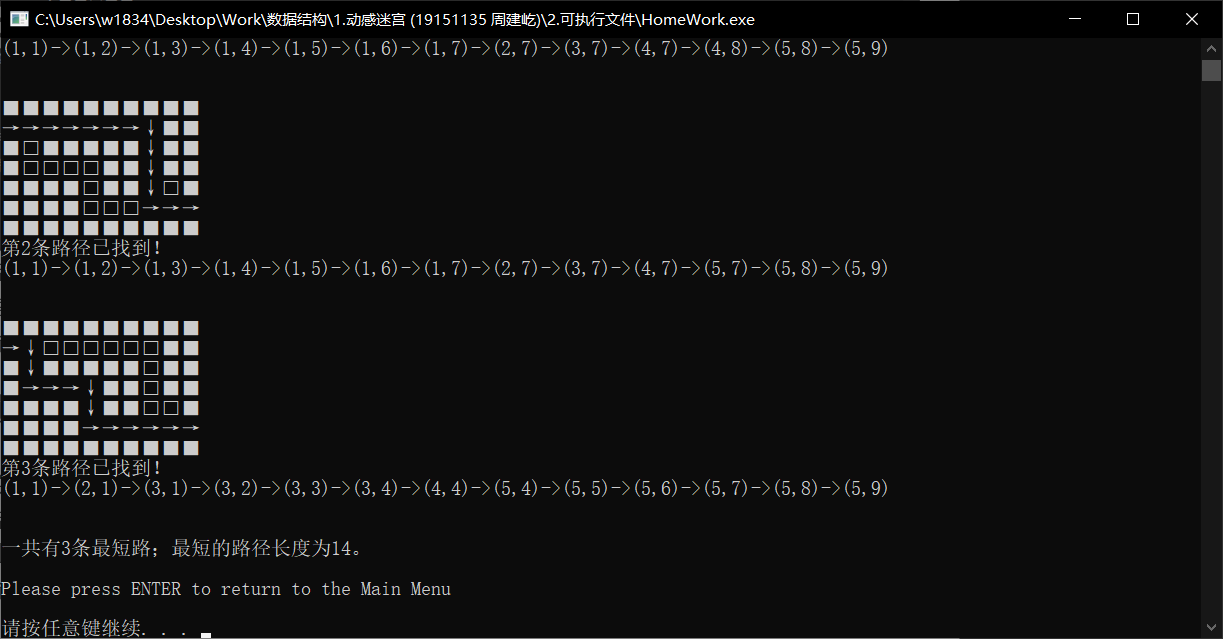
第\*条路径已找到

\*\*\*（路径）

一共有\*条最短路；最短的路径长度为\*\*。

Please press ENTER to return to the Main Menu

1. 实测结果



1. 结论：测试通过
2. 可以让用户选择遍历序列的顺序，优先向哪个方向走
3. 输入用例：

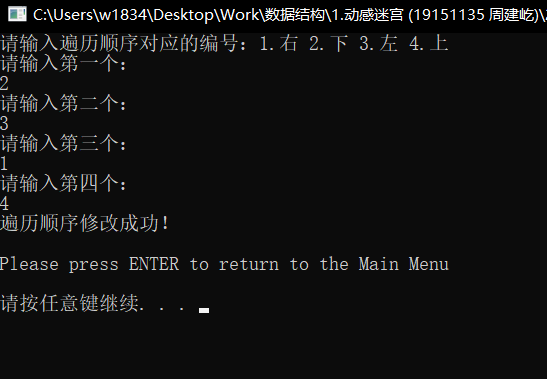
4 2 3 1 4

期望结果：

遍历序列修改成功！

Please press ENTER to return to the Main Menu

1. 实测结果



1. 结论：测试通过
2. **附录**

**文件目录**



**CreatPuzzle.cpp**随机生成迷宫

**Solve.cpp** 迷宫求解以及迷宫和路径的打印

**SqStack.cpp** 提供基础的数据结构栈

**Test.cpp** 包含用户界面供与用户交互