XXXXXX·XXXXXX学院

**《数据结构程序开发课程设计》报告**

迷宫问题求解的设计与实现

专 业： 软件工程

班 级： XXXXXXXX

学 号： XXXXXXXXXX

姓 名： rv

指导老师： XX

报告日期： 2022年06月

目录

[一、概述 3](#_Toc2042)

[二、系统分析 3](#_Toc17898)

[1、基本原理分析 3](#_Toc27024)

[2、功能设计分析 4](#_Toc14630)

[三、概要设计 4](#_Toc25084)

[1、整体框架 4](#_Toc2851)

[2、数据结构的选择 6](#_Toc9575)

[四、详细设计 6](#_Toc21350)

[1、设计每个成员函数： 6](#_Toc22838)

[（1）生成迷宫 7](#_Toc17873)

[（2）dfs寻找最短路函数 7](#_Toc15761)

[（3）bfs寻找最短路函数 8](#_Toc13962)

[（4）bfs记录迷宫路径函数 9](#_Toc8221)

[2、设计主函数 10](#_Toc18013)

[五、运行与测试 11](#_Toc18789)

[1、调试分析 11](#_Toc17124)

[2、使用说明和运行结果 11](#_Toc25298)

[3、 缺点与改进 15](#_Toc24888)

[六、总结与心得 16](#_Toc290)

**一、概述**

课程设计是实践性教学中的一个重要环节，它以某一课程为基础，可以涉及和课程相关的各个方面，是一门独立于课程之外的特殊课程。课程设计是让同学们对所学的课程更全面的学习和应用，理解和掌握课程的相关知识。《数据结构》是一门重要的专业基础课，是计算机理论和应用的核心基础课程。

数据结构课程设计，要求学生在数据结构的逻辑特性和物理表示、数据结构的选择和应用、算法的设计及其实现等方面，加深对课程基本内容的理解。同时，在程序设计方法以及上机操作等基本技能和科学作风方面受到比较系统和严格的训练。

本次课设我选择了迷宫问题，迷宫求解是数据结构课程的一个经典问题。

迷宫问题要求寻找一条从入口到出口的路径。即从入口出发，顺着某一个方向进行探索，若能走通，则继续往前走；否则沿着原路退回，换一个方向继续探索，直至出口位置，求得一条通路。假如所有可能的通路都探索到而未能到达出口，则所设定的迷宫没有通路，通常用的是“穷举求解”的方法。为了保证在任何位置上都能原路退回，显然需要用一个先进先出的结构来保存从入口到当前位置的路径。因此，在求解迷宫通路的算法中要应用“队列”的思想。对于队列的内容在整个学期的学习中我也有了一定的了解，所以选择了迷宫这一经典问题作为本次课设的内容。

**二、系统分析**

求迷宫中从一个入口到出口的所有路径是一个经典的程序设计问题。由于计算机解迷宫时，通常用的是“穷举求解”的方法，即从入口出发，顺某一方向向前探索，若能走通，则继续往前走；否则沿原路返回，换一个方向再继续探索，直到所有可能的通路都探索到为止；假如所有可能的通路都探索到而未能到达出口，则所假定的迷宫没有解。

迷宫问题要求寻找一条从入口到出口的路径。通常用的是“穷举求解”的方法。为了保证在任何位置上都能原路退回，显然需要用递归来探寻迷宫的路径。因此，在求解迷宫通路的第一个方法中我选择了dfs算法，即深度优先搜素算法。

而为了保证在任何位置上都能找全其东、西、南、北四个方向的路径，我选择用“先进先出”的队列结构来保存从入口到当前位置的路径。因此，在求解通路的第二种方法中我应用“队列”的思想，选择了bfs算法，即广度优先搜索算法求解。

**1、基本原理分析**

迷宫问题通常用的是“穷举求解”的方法。为了保证在任何位置上都能原路退回，显然需要用递归来探寻迷宫的路径。因此，在求解迷宫通路的第一个方法中我选择了dfs算法，即深度优先搜素算法。

而为了保证在任何位置上都能找全其东、西、南、北四个方向的路径，我选择用“先进先出”的队列结构来保存从入口到当前位置的路径。因此，在求解通路的第二种方法中我应用“队列”的思想，选择了bfs算法，即广度优先搜索算法求解。

以二维数组存储迷宫数据，通常设定入口点的下标为（1，1），出口点的下标为（n,n）。对于迷宫任何一个位置，均约定东、南、西、北四个方向可通。

**2、功能设计分析**

1: 以二维数组存储迷宫数据，通常设定入口点的下标为（1，1），出口点的下标为（n,n）。对于迷宫任何一个位置，均约定东、南、西、北四个方向可通。

2: 以一个二维数组MAP[n][n]表示迷宫，数组中以O表示通路，X表示障碍,S表示入口，E表示出口。

3：用户需用输入迷宫的数据：输入的数据为迷宫的边长n。

4: 程序随机生成以n为边长的正方体迷宫，迷宫的入口位置为（1，1），出口位置为（n，n）。

5: 程序执行的命令为：

1：输入迷宫的边长；2：创建迷宫；3 ：迷宫寻路；4：输出迷宫的路径。

6: bfs算法寻路：用队列存储路径，当队列中还有值时，继续寻路，从此时的队头开始，寻找其所有方向可走的方块并将可走方块入队后，将其出队。**一层层向外扩展查找可走的方块，直到找到出口为止，最先找到的这个答案就必然是最短的。**

7:dfs算法寻路：采用回溯法实现该问题的求解。从入口出发，按某一方向向前探索，若能走通(未走过的)，即某处可以到达，则到达新点，否则试探下一方向；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，直到所有可能的通路都搜索到，或找到一条通路，或无路可走又返回到入口点。

8：路径打印。将用算法寻得的通路打印出来，实现的迷宫寻路可视化。

**三、概要设计**

**1、整体框架**

本程序包含五个模块：

1. 迷宫模块——实现迷宫抽象数据类型即输入边长，最后依赖随机数种子随机建立迷宫，并检验迷宫是否有通路，最后在屏幕上输出迷宫的形式。
2. bfs寻路模块——实现队列数据类型,以队列存储结构为基础设计的，因为本设计常做查找路径，所以采用了栈的链式存储结构，其中包括队列的初始化，建队，入队，出队，判队列是否为空等操作。本模块主要实现探索有无通路时的先进先出的操作。
3. dfs寻路模块——实现迷宫寻路。采用回溯法实现该问题的求解

从入口出发，按某一方向向前探索，若能走通(未走过的)，即某处可以到达，则到达新点，否则试探下一方向；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，直到所有可能的通路都搜索到，或找到一条通路，或无路可走又返回到入口点。

1. 路径打印模块——其中又分为dfs算法寻路的路径打印和bfs算法寻路的路径打印。本模块主要实现经过算法寻路后，迷宫路径探寻的可视化。
2. 主程序模块。
3. 各模块之间的大致调用关系图：

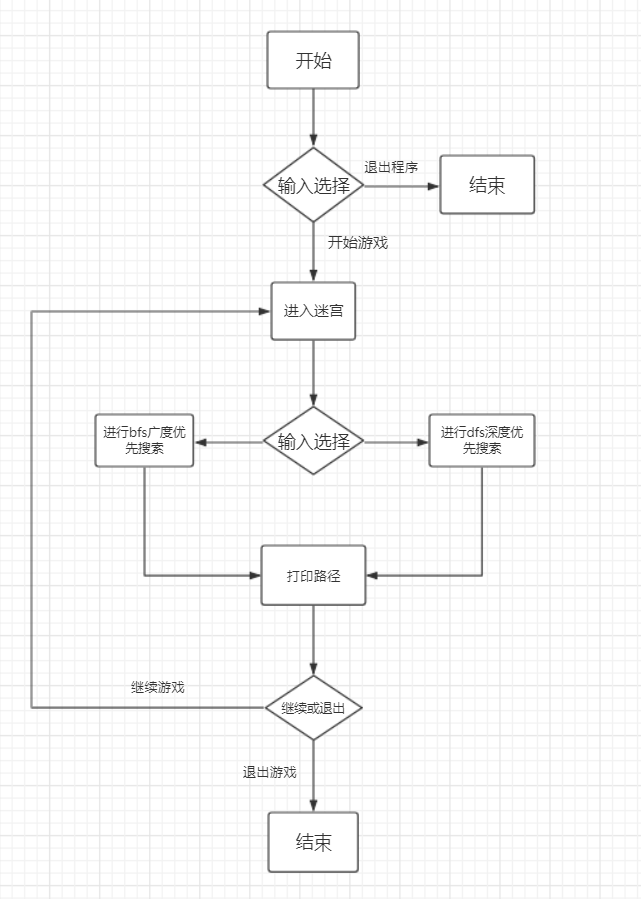


图 1

1. ：程序的实现流程图：

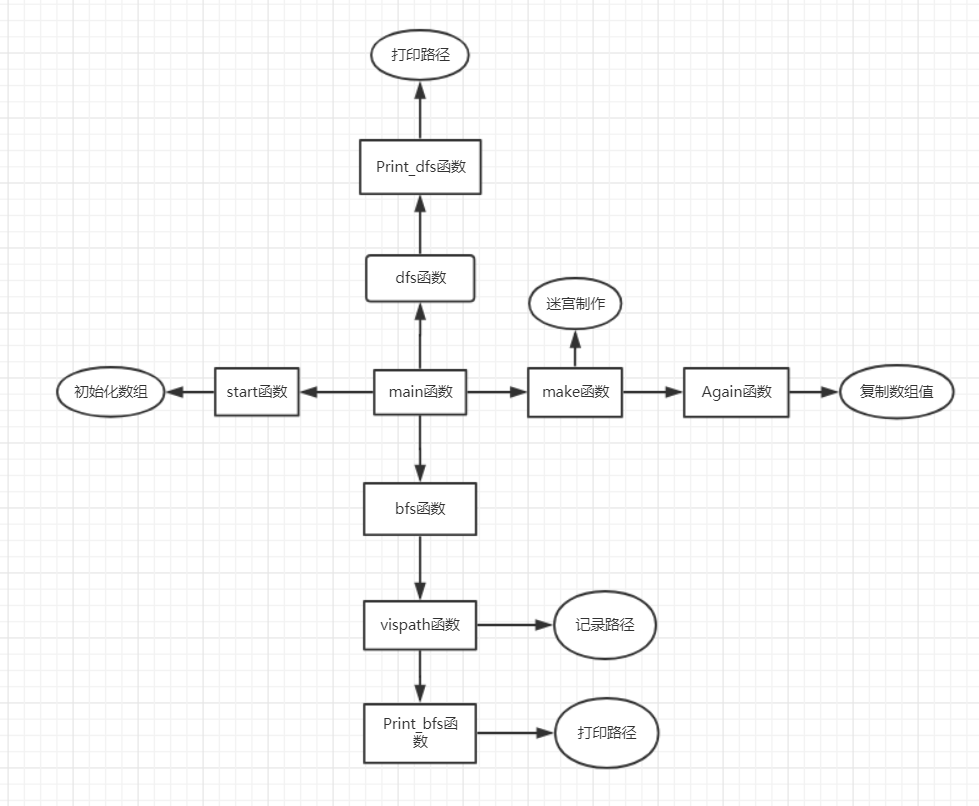


图 2

**2、数据结构的选择**

本设计部分采用的队列的存储结构，因为在进行探索是否有通路的过程中，常进行查找，比较，为了保证在任何位置上都能找全其东、西、南、北四个方向的路径，我选择用“先进先出”的队列结构来保存从入口到当前位置的路径。同时对于此次课程设计，队列的存入、弹出数据的方式比数组更为方便，且不用限定存储空间的大小，这个优点对设计很有利，所以采用了队列的存储结构。

**四、详细设计**

**1、设计每个成员函数：**

struct node { //用结构体同时存储该点坐标和父结点坐标

int x,y; //当前位置的坐标

int fromx,fromy; //当前位置坐标由上一个坐标的转移

}

（1）生成迷宫

void Make() {

system("cls");

cout<<"请输入制作随机迷宫边长：\n";

cin>>n;

system("cls"); //清屏

printf("迷宫制作中请等待1~3秒");

while (1) {

srand(time(NULL)); //以当前时间为种子，生成随机数

for (int i=1;i<=n;i++) {

for (int j=1;j<=n;j++) {

int num=rand()%3; //根据生成的随机数设置墙/路

if (num==0) {

MAP[i][j]='X'; //墙

} else {

MAP[i][j]='O'; //路

}

}

}

Again(); //把 MAP中的值赋给 MAPbfs和 MAPdfs中

bfs(); //判断是否通路，是则退出函数，否则继续生成迷宫

if (flag==1) break;

}

}

（2）dfs寻找最短路函数

void dfs(int x,int y,int step) {

if (flag) return;

if (x==n&&y==n) { //找到终点

flag=1;

Sleep(3000);

Print\_dfs(step); //打印路径

Again(); //把 MAP中的值赋给 MAPbfs和 MAPdfs中

return;

}

for (int i=0;i<4;i++) { //遍历该点的东西南北四个方向

int nextx=x+turnx[i];

int nexty=y+turny[i];

if (nextx>=1&&nextx<=n&&nexty>=1&&nexty<=n&&(MAPdfs[nextx][nexty]=='S'||MAPdfs[nextx][nexty]=='O'||MAPdfs[nextx][nexty]=='E')) { //边界判断与通路判断

MAPdfs[nextx][nexty]='o';

if (nextx!=n||nexty!=n) Print\_dfs(step); //打印路径

dfs(nextx,nexty,step+1); //从该点继续寻找路径

if (flag) return; //若找到通路，则直接回溯

MAPdfs[nextx][nexty]='x'; //否则 将该店设为未访问状态

Print\_dfs(step); //打印路径

}

}

}

（3）bfs寻找最短路函数

void bfs() {

int nx,ny; //下一个坐标的 x，y值

path[1][1].x=path[1][1].y=1;

Q.push(path[1][1]);

while (!Q.empty()) { //队列中还有数时，继续寻路

NOW=Q.front(); //将此时的队头（即当前位置）取出

Q.pop(); //队头出队

if (NOW.x ==n && NOW.y ==n) { //找到出口

flag=1; //做个标记，方便后续判断该迷宫是否有通路

return;

}

for (int i=0;i<4;i++) { //遍历该点的东西南北四个方向

nx=NOW.x+turnx[i];

ny=NOW.y+turny[i];

if (nx>=1 && nx<=n && ny>=1 && ny<=n && MAPbfs[nx][ny]!='X') { //越界和通路判定

path[nx][ny].x =nx;

path[nx][ny].y =ny;

path[nx][ny].fromx =NOW.x ;

path[nx][ny].fromy =NOW.y ; //记录路径

MAPbfs[nx][ny]='X'; //标记已找过的路

Q.push(path[nx][ny]) ; //将该点入队

}

}

}

}

（4）bfs记录迷宫路径函数

void vispath(int x,int y) { //运用用递归、回溯，做到寻找、标记、打印路径

memset(vis,0,sizeof(vis)); //初始化

if (x==1&&y==1) {

vis[1][1]=1;

Print\_bfs(); //从起点开始打印路径

return; //找到起点后，开始回溯，输出最终路径

}

int fx=path[x][y].fromx ;

int fy=path[x][y].fromy ;

vispath(fx,fy); //不断找该点的父结点

vis[path[x][y].x ][path[x][y].y ]=1; //回溯后将该点做上标记

Print\_bfs(); //打印路径

}

**2、设计主函数**

int main() {

while (1) {

init();

flag=0;

string choice;

{

system("cls");

system("color 0F");

cout<<"请选择您的操作：\n";

cout<<"1.开始游戏\n";

cout<<"2.退出程序\n";

cin>>choice; //用户选择操作

while (choice!="1"&&choice!="2") { //错误特判

cout<<"操作有误，请重新选择！\n";

cin>>choice;

}

if (choice == "2") return 0;

Make(); //制作迷宫

}

while (1) {

Print\_maze();

cout<<"\n请选择寻找出口的方式：\n";

cout<<"1.dfs深度优先搜索\n";

cout<<"2.bfs广度优先搜索\n";

cout<<"3.返回上一界面\n";

flag=0;

cin>>choice; //用户选择操作

while (choice!="1"&&choice!="2"&&choice!="3") { //错误特判

cout<<"操作有误，请重新选择！\n";

cin>>choice;

}

if (choice == "1") flag=0, dfs(1,1,0);

else if (choice == "2") vispath(n,n), Print\_bfs();

else if (choice == "3") break;

cout<<"\n按 1 返回上一界面（ ￣ ▽ ￣ ）\n";

cin>>choice;

while (choice!="1") { //错误特判

cout<<"输错了哦，请重新输入（ ￣ ▽ ￣ ）\n";

cin>>choice;

}

system("cls");

}

}

return 0;

}

**五、运行与测试**

**1、调试分析**

本程序的核心算法，即求迷宫的路径，在调试的时候，出现的问题是：无法设计遍历是的四个方向。

方法：用数组存储坐标，实现四个方向的遍历：

turnx[4] = {0, 1, 0, -1}; //x坐标的移动数组

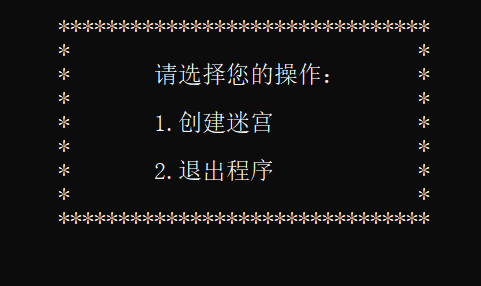
turny[4] = {1, 0, -1, 0}; //y坐标的移动数组

**2、使用说明和运行结果**

程序运行后进入初始页面：



按任意键后进入到操作页面：



选择1.创建迷宫后程序会提示出入迷宫的边长:



输入后，程序开始制作迷宫：





1. 选择dfs迷宫寻路：

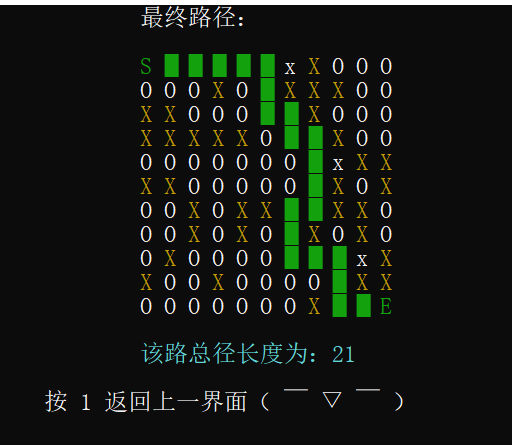
程序会打印出寻路过程，正确的路径用绿色方块表示，错误的路径用红色x表示，

且在迷宫下方有此时的路径长度。



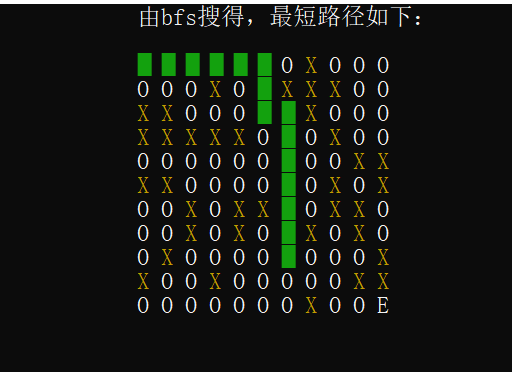


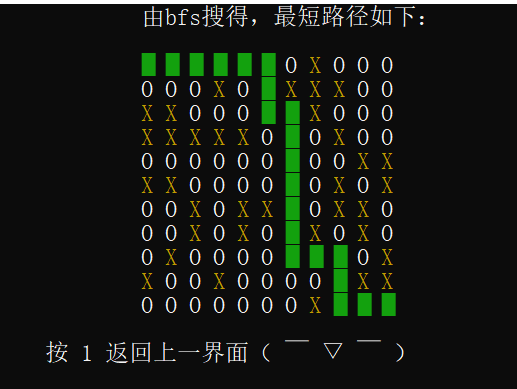
探寻到终点，停留3秒后输出最终结果：



1. 选择bfs迷宫寻路

程序会打印出最短路径探寻过程，最短路径用绿色方块表示。





1. **缺点与改进**

缺点1：程序最初设计的迷宫是有一个固定的模板、或让用户以手动的形式输入迷宫。前者感觉太单一，只用一个样例检验迷宫寻路算法的实现是否成功不具备普遍性，检验结果不科学，一般来说,样本的容量大的话,样本的误差就小。 后者以手动的形式输入迷宫，如果迷宫数据量大时，1.用户体验感不好；2.先建好文件很浪费时间。

改进：以随机产生函数自动产生迷宫会更好，既能产生足够的样本来检验算法设计是否成功，又能解放用户双手。所以我设计了自定义函数Make()函数来实现迷宫的随机生成。

缺点2：在运行时，我发现在进行用户输入数字，来选择所要进行的操作时，如果此时输入的是字母，程序则会陷入死循环。

改进：将存储用户做出选择时在控制台输入的字符的变量类型，由int整形变为string字符型后，这一错误得到解决。

**六、总结与心得**

数据结构是信息的一种组织方式，其目的是为了提高算法的效率，它通常与一组算法的集合相对应，通过这组算法集合可以对数据结构中的数据进行某种操作。数据结构课程的主要目的是介绍一些常用的数据结构，阐明数据结构内在的逻辑关系，讨论它们在计算机中的存储表示，并结合各种数据结构，讨论对它们实行的各种运算的实现算法。

通过这次数据结构课程设计，让我学到了好多东西。在实际操作过程中犯的一些错误却让我有了意外的收获，所学数据结构理论知识得到了巩固。通过实际操作，学会数据结构程序编程的基本步骤、基本方法，开发了自己的逻辑思维能力，培养了分析问题、解决问题的能力。

在优化课设时，想写出迷宫的随机生成功能，但是完全想不到怎么用现有的知识来实现。向acm集训队的学长请教后才知道是自己想的太过复杂。学习了这种方法的同时也教育了我要灵活、多角度分析问题。

在实现如何打印迷宫路径这一模块，通过网络探寻方法后，发现和自己曾经做过的一道用bfs打印路径的online judge题如出一辙。但是我在写本次课设之初完全没印象，这警醒了我学习是一个不断探索的过程，但在探索过程中千万不要忘了曾经被自己挖掘出来的宝藏。学习要时常温习，在学科竞赛这样的激烈环境下更甚，这样才能收获的更多，才能了解到以前别人所不知道的知识。即——温故而知新，可以为师矣。

**参考文献:**

[1] 刘汝佳著 《算法竞赛入门经典（第2版）》 清华大学出版社

[2] 殷人昆等著《数据结构》（C++版）清华大学出版社 2001