**电子科技大学**

**计算机科学与工程学院**

**标 准 实 验 报 告**

**（实验）课程名称 软件开发综合实验**

**电子科技大学教务处制表**

**电 子 科 技 大 学**

# 实 验 报 告

**学生姓名： 学 号： 指导教师：**

**实验地点： 实验时间：**

**一、实验室名称：**

**二、实验项目名称：迷宫游戏**

**三、实验学时：8学时**

**四、实验原理：**

1.从文件读取数据，转化为地图。或者利用并查集即，随机生成地图。

2.利用类似于图的深入优先搜索算法，寻找入口和出口之间的路。

3.输出路径坐标，并用字符画出地图和路径。

**五、实验目的：**

1.熟悉基本数据结构（数组等）的使用。

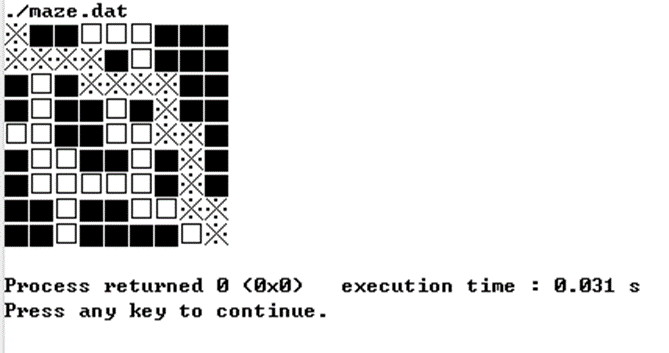
2.掌握文件操作。

3.掌握并查集生成地图的算法和搜索路径的算法。

4.学会利用数据结构和算法编写简单程序。

**六、实验内容：**

写一个终端迷宫游戏，能够从文件中或者利用程序生成迷宫地图，并且至少找出一条可行的通路。如下图所示：



要求如下：

1.能够从文件中读取地图数据，并生成地图。

2.利用算法生成一条通路。

3.修改地图数据，利用算法生成多条通路。（可选）

4.自动生成迷宫地图，并自动寻路。（可选）

1. **实验步骤：**

由于实验需要分别实现从文件中读取地图和随机生成地图两个功能，故为方便展示，分别编写两个程序，一个用于随机生成并找路，一个用于读取地图并找路。

（1）随机生成地图并找路：

1. 首先定义map结构体，用于存储各种信息，部分代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct map {

int s[N\*N]; //每个格子的根

int wall\_row[N][N];//行墙

int wall\_col[N][N];//列墙

int stack[N\*N];

int visit[N\*N]; //每个元素访问状态

int maxsize;

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

为了方便，这里设置了右墙和下墙。另外定义了栈用于处理搜索算法。

1. 对结构体定义操作，部分函数头代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool empty(map \*a) //判空

bool full(map \*a) //判满

void push(map \*a, int num) //入栈

void pop(map \*a, int num) //出栈

bool instack(map\*a ,int num) //遍历 判断是否在栈 用于打印在栈元素

bool left(map \*a, int num) //判断是否能往左走

bool right(map \*a, int num) //判断是否能往右走

bool up(map \*a, int num) //判断是否能往上走

bool down(map \*a, int num) //判断是否能往下走

bool isvalid(map \*a, int num) //判断是否可走 必须同时满足没走过和没墙 只要上下左右有一个可以就可取

void findnext(map \*m, int num) //找路过程

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. 部分核心代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void findnext(map \*m, int num) { //找路过程

if (m->visit[num] == 0) {

m->visit[num] = 1;

cout << "(" << num / N << "," << num%N << ")";

if (num == (N\*N - 1)) {

cout << "successful"<<endl;

return;

}

if (isvalid(m, num)) {

if (left(m, num) && (m->visit[num - 1] == 0)) {

push(m, num - 1);

}

if (right(m, num) && (m->visit[num + 1] == 0)) {

push(m, num + 1);

}

if (up(m, num) && (m->visit[num - N] == 0)) {

push(m, num - N);

}

if (down(m, num) && (m->visit[num + N] == 0)) {

push(m, num + N);

}

}

else pop(m, num);

}

else pop(m, num);

findnext(m, m->stack[top]);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

srand((unsigned)time(NULL)); //随机生成过程

//初始化map

for (int i = 0; i < N\*N; i++) {

map1.s[i] = -1;

}

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

map1.wall\_row[i][j] = -1;

}

}

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

map1.wall\_col[i][j] = -1;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

4.运行截图如下：

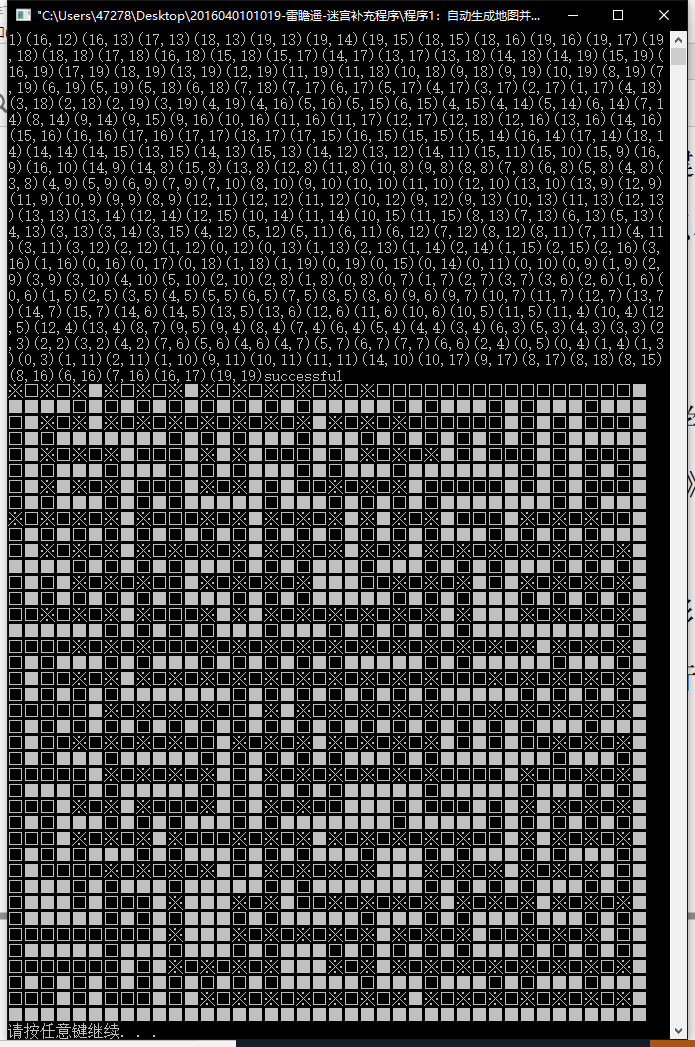


图1.自动生成地图程序

（2）读文件生成地图并找路：

1. 首先定义map结构体，用于存储各种信息，部分代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef struct map {

int s[N\*N]; //每个格子的根

int wall\_row[N][N];//行墙

int wall\_col[N][N];//列墙

int stack[N\*N];

int visit[N\*N]; //每个元素访问状态

int maxsize;

}MAP;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

为了方便，这里设置了右墙和下墙。另外定义了栈用于处理搜索算法。

1. 对结构体定义操作，部分函数头代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReadA(FILE \*pf, MAP \*pm)

bool empty(map \*a) //判空

bool full(map \*a) //判满

void push(map \*a, int num) //入栈

void pop(map \*a, int num) //出栈

bool instack(map\*a ,int num) //遍历 判断是否在栈 用于打印在栈元素

bool left(map \*a, int num) //判断是否能往左走

bool right(map \*a, int num) //判断是否能往右走

bool up(map \*a, int num) //判断是否能往上走

bool down(map \*a, int num) //判断是否能往下走

bool isvalid(map \*a, int num) //判断是否可走 必须同时满足没走过和没墙 只要上下左右有一个可以就可取

void findnext(map \*m, int num) //找路过程

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. 部分核心代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void findnext(map \*m, int num) { //找路过程

if (m->visit[num] == 0) {

m->visit[num] = 1;

cout << "(" << num / N << "," << num%N << ")";

if (num == (N\*N - 1)) {

cout << "successful"<<endl;

return;

}

if (isvalid(m, num)) {

if (left(m, num) && (m->visit[num - 1] == 0)) {

push(m, num - 1);

}

if (right(m, num) && (m->visit[num + 1] == 0)) {

push(m, num + 1);

}

if (up(m, num) && (m->visit[num - N] == 0)) {

push(m, num - N);

}

if (down(m, num) && (m->visit[num + N] == 0)) {

push(m, num + N);

}

}

else pop(m, num);

}

else pop(m, num);

findnext(m, m->stack[top]);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReadA(FILE \*pf, MAP \*pm) //读文件过程

{

pf = fopen("Maze.txt","rt");

int count=0;

char temp;

for(;;)

{

temp = fgetc(pf);

if(temp == EOF)

{

break;

}

if(temp!='#')

{

int a=count/(2\*N);

int b=count%(2\*N);

int c=a/2;

int d=b/2;

if(a%2==0&&b%2==0){ //JJ

if(temp=='0'){

pm->s[N\*c+d]=0;

}

else

pm->s[N\*c+d]=-1;

}

}

if(temp!='#') //JJ

{

int a=count/(2\*N);

int b=count%(2\*N);

int c=a/2;

int d=b/2;

if(a%2==0&&b%2!=0){

if(temp=='0'){

pm->wall\_row[c][d]=0;

}

else

pm->wall\_row[c][d]=-1;

}

}

if(temp!='#') //JJ

{

int a=count/(2\*N);

int b=count%(2\*N);

int c=a/2;

int d=b/2;

if(a%2!=0&&b%2==0){

if(temp=='0'){

pm->wall\_col[c][d]=0;

}

else

pm->wall\_col[c][d]=-1;

}

}

if(temp=='#')

{

count--;

}

count++;

}

fclose(pf);

return;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. 运行截图如下：

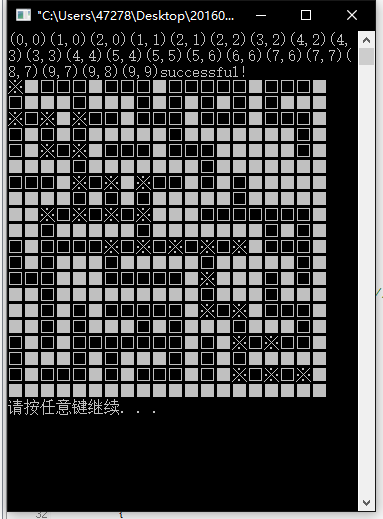


图2.读取文件地图并找路程序

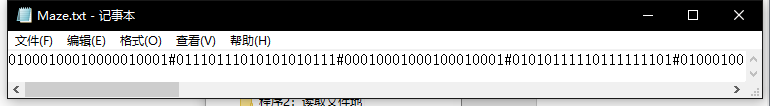


图3.地图数据文件

**八、总结及心得体会：**

通过本实验的练习，学会了通过并查集生成迷宫算法和迷路找路算法，更深层地理解了图的深度优先遍历的思想，特别是递归和栈的思想。

进一步加强了对C语言文件操作的使用，了解了面向对象编程的思想。

理解了迷宫问题的解决思路，体会了把理论运用到实际的具体方法。

**九、对本实验过程及方法、手段的改进建议：**

可以制作进一步制作程序，变为可以人为操作光标走迷宫的一个小游戏，这样可以更深刻体会迷宫找路的算法。

**报告评分：**

**指导教师签字：**