《 数 据 结 构 》课 程 设 计

3.8 老鼠迷宫

姓 名: XXX

院系: 计算机学院

专 业: 计算机科学与技术

年 级: 13 级

学 号: EXXX

指导教师:XXX

2015 年 10 月 5 日

目录

1.课程设计的目的 3
2.需求分析 3
3 走迷宫游戏的设计 3
3.1 概要设计 3
3.1.1 主界面设计 3
3.1.2 存储结构 3
3.1.3 系统功能设计 4
3.2 详细设计 4
3.2.1 系统子程序及功能设计 4
3.2.2 数据类型定义 5
3.2.3 系统主要子程序详细设计 6
3.3 调试分析 15
3.4 用户手册 16
4 总结 16
5、程序清单:(见附录) 16
7、程序运行结果 16
附录 1 22

1.课程设计的目的

(1) 熟练使用 C 语言编写程序，解决实际问题;

(2) 了解并掌握数据结构与算法的设计方法，具备初步的独立分析和设计能力;

(3) 初步掌握软件开发过程的问题分析、系统设计、程序编码、测试等基本方法和技能;

(4) 提高综合运用所学的理论知识和方法独立分析和解决问题的能力。

2.需求分析

(1) 程序运行时显示一个迷宫地图，迷宫中央有一只老鼠，迷宫的右下方有一个粮仓。

(2) 按动键盘上的方向操作杆控制老鼠在规定的时间内走到粮仓。

(3) 老鼠形象可辨认，迷宫的墙足够结实，老鼠不能穿墙。

(4) 若老鼠在规定时间走到粮仓处则提示成功，否则提示失败。

(5) 添加编辑迷宫功能，可使迷宫墙变路，路变墙。

(6) 找出走出迷宫的所有路径，以及最短路径。

3 走迷宫游戏的设计

3.1 概要设计

3.1.1 主界面设计

图 1

主界面，如图1 所示，包含六个菜单项，分别是开始游戏、编辑迷宫、读入迷宫、生成随机迷宫、最短路径、退出系统。输入数字选择对应菜单，进入子项。

3.1.2 存储结构

本系统用结构体Maze 存储迷宫，该结构体由room[M][N]，status[M][N]组成。room 是 int 型数组，存储迷宫具体内容；status 是 int 型数组，用来存放迷宫求解时通堵状态，迷宫大小 M，N 固定。本系统用结构体 pos 存储迷宫求解时每一步的信息，该结构体由

x,y,parent,self,next[4]组成。x,y 是整型变量，存储坐标，parent 是上一步在队列数组中的位置， self 是本步在队列数组中的位置,next[4]存储四个方向下一步在队列数组中的位置。本系统用结构体 Queue形成队列，该结构体由 p[M*N],front，rear组成。p是pos型变量，存储队列中的内容。本系统用数组实现队列操作，是为了求最短路径时，用 BFS 算法后，还能找到出队列的数据。

3.1.3 系统功能设计

本系统主菜单包含四个选项，功能描述如下：

菜单 1：开始游戏。在菜单1 中开始走迷宫游戏，老鼠用笑脸表示，初始状态老鼠在迷宫中间；通过键盘操纵老鼠上下左右移动，游戏分为6 关，同一张地图，在规定时间到达粮仓可升级，否则提示失败，返回主菜单，随着级别增高最大时间逐渐减少。

菜单 2：编辑迷宫。在菜单2 中编辑迷宫，按上下左右操作光标 。可输入1，控制光标走过的地方为通路，输入 0 控制光标走过的地方为障碍。输入非法字符退出编辑。光标默认在迷宫中间，默认情况下光标走过的地方都为通路。

菜单 3：读入迷宫。在菜单 3 中读入已存在的迷宫文件，并输出读入的迷宫。

菜单 4：随机迷宫。在菜单4 中以系统时间作为种子，随机生成地图，并输出地图，询问是否保存到文件。

菜单 5：最短路径。在菜单5 中进行迷宫求解，若存在走出迷宫的路径，则找出最小路径，并用图形化界面输出，同时输出所有可走出迷宫的路径数目；否则，提示未找到最短路径，即该迷宫无解。

菜单 0：退出系统。

3.2 详细设计

3.2.1 系统子程序及功能设计

本系统设置 19 个子程序，各程序的函数名及功能说明如下：

int maincourse()//主菜单

int inputselection(int n)//输入菜单选择，返回值为输入的选项void

initqueue(Queue &Q)　　//初始化队列

void enqueue(Queue &Q,pos p)//进队void

dequeue(Queue &Q,pos &p)//出队int

emptyqueue(QueueQ)//判队空

void display(Elemtype room[M][N],int degree,int t,int flag)//输出迷宫void

countdown()　　//倒计时进入迷宫游戏

void play(Elemtype room[M][N])　　//迷宫游戏

void filesave(Maze maze)　　//保存地图到文件

```
void fileread(Maze &maze)              //读入地图
void modify(Maze &maze)        //编辑迷宫void
mazerand(Maze &maze)//随机生成迷宫
int pass(Maze maze,pos p)              //是否可通过
void markprint(Maze &maze,pos p)//留下足迹void
stepsdisplay(Queue Q,pos p,Maze maze) void          //输出最短路径
minroad(Maze maze)                     //最短路径
void findallroad(Maze &maze,pos p,int &num)          //找所有路径
void allroad(Maze maze)                //所有路径
```

各个函数间的调用关系如下:

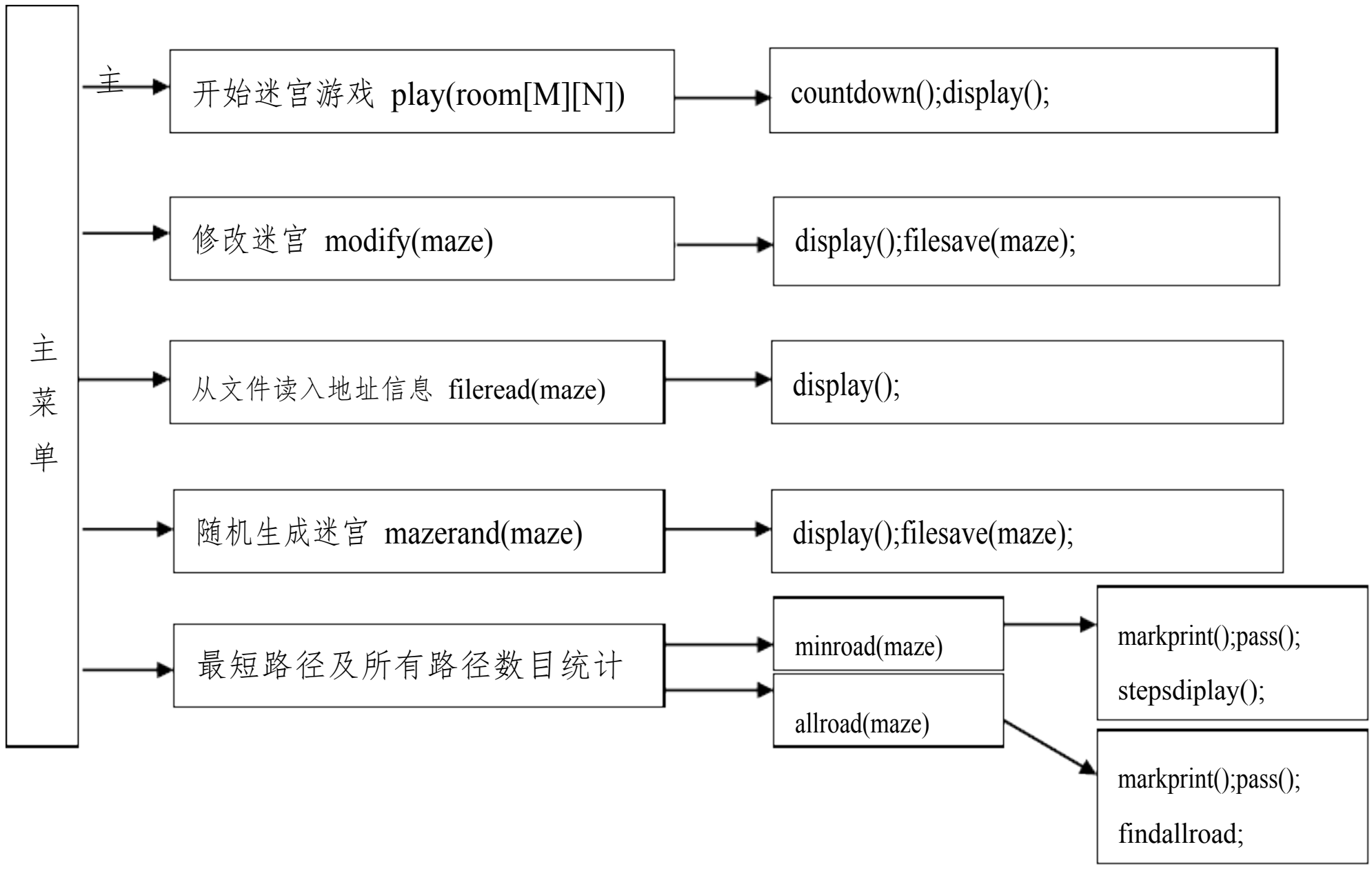


图 3

3.1.1 数据类型定义

```
struct Maze{
    Elemtype room[M][N]; //迷宫Elemtype
    status[M][N]; //迷宫求解记录
};
struct pos{
    intx,y,parent,self;      //x,y是当前坐标,parent是上一步在队列中的位置
```

```
        intnext[4];       //self是本步在队列中的位置,next是下四步在队列中的位置
};
struct Queue{
        posp[M*N];                    //队列中元素
        intfront;                     //队首
        intrear;                      //队尾
};
```

3.2.2 系统主要子程序详细设计

输出迷宫：

```
void display(Elemtype room[M][N],int degree,int t,int flag){//输出迷宫
        inti,j;    //flag为0为游戏模式,flag为1或2为编辑模式,大于2为输出模式
printf("\n\n");
for(i=0;i<M;i++){
        putchar('\t'); for(j=0;j<N;j++)
                switch(room[i][j]){
                case 0:
                        printf("■");
                        break;
                case 1:
                        printf("    ");
                        break;
                case 2:
                        printf("%c ",1);
                        break;
                case 3:
                        printf("∏");
                        break;
        case 4:
                printf("↓");     //上
                break;
        case 5:
                printf("→");     //右
                break;
```

```
        case 6:
            printf("←");     //左
            break;
        case 7:
            printf("↑");     //上
            break;
        default:
            printf("\n 迷宫格式错误!停止运行!\n");
            exit(0);
            break;
    }
    if(flag>=3){                //其他模式
        putchar('\n');
        continue;
    }
    switch(i){
    case 0:
        printf(" --------------------");
        break;
    case 1:
        if(flag)
            printf("          编辑迷宫!       ");//编辑模式
        else
            printf("    欢迎进入迷宫游戏!");//游戏模式
        break;
    case 2:
        printf(" --------------------");
        break;
    case 3:
        printf(" --------------------");
        break;
    case 4:
        printf("       按←↑→↓操纵  ");
        break;
```

```
case 5:
        printf("--------------------");
        break;
case 6:
        printf("--------------------");
        break;
case 7:
        if(flag)
                printf("     0,路边墙;1,墙变路"); //编辑模式

        else
                printf("        当前等级:%d    ",degree);     //游戏模式
        break;
case 8:
        printf("--------------------");
        break;
case 9:
        printf("--------------------");
        break;
case 10:
        if(flag==2)
                printf("          此时墙变路       ");//编辑模式
        else if(flag==1)
                printf("          此时路变墙       ");
        else
                printf("        剩余时间:%d\t",t);//游戏模式
                break;
            case 11:
                printf("--------------------");
                break;
            }
            putchar('\n');
        }
}
```

走出迷宫游戏：

```
void play(Elemtype room[M][N]){                    //迷宫游戏
        countdown();
        system("cls"); clock_t
        start,start2;
        start=clock();
        int x=M/2,y=N/2,x1=1,y1=1;
        int degree=1,lefted_t=30,all_t=30;
        while(1){
                start2=clock();
                while((lefted_t=(start2-start)/second)<all_t&&!kbhit()){//计时及判断是否有输入
        system("cls");
        display(room,degree,all_t-lefted_t,0);
        while(clock()-start2<second&&!kbhit());
        start2=clock();
}
lefted_t=all_t-lefted_t;
if(lefted_t<=0){              //时间用完
        system("cls");
        printf("\t\t\t*****************************\n");
        printf("\t\t\t                游戏结束!挑战失败!          \n");
        printf("\t\t\t*****************************\n");
        room[x][y]=1;
        room[M/2][N/2]=2; return;
}
getch();
switch(getch()){       //操纵
case 72:
        x1=x-1;
        y1=y;
        break;          //  向上
case 80:
        x1=x+1;
```

```
        y1=y;
        break;      //向下
    case 75:
        x1=x;
        y1=y-1;
        break;      //向左
    case 77:
        x1=x;
        y1=y+1;      //向右
        break;
    default:
        continue;
    }
    if(x1==0||x1==M-1||y1==0||y1==N撞边/墙了
        continue;
    else if(room[x1][y1]==0)/撞墙了
        continue;
    else{
        room[x][y]=1;
        x=x1;
        y=y1;
        if(x==N-2&&y==N-2){      /找到出口
            system("cls");
            degree++;
            all_t-=5;
        if(all_t==0){    /随着升级，时间减少
            system("cls");
            printf("¥t¥t¥t*****************************¥n");
            printf("¥t¥t¥t        你 赢 了 !¥n");
            printf("¥t¥t¥t*****************************¥n");
            room[x][y]=3;
            room[M/2][N/2]=2;
            return;
        }
```

```
        printf("\t\t\t*****************************\n");
        printf("\t\t\t                  升级了!等级%d!\n",degree);
        printf("\t\t\t*****************************\n");
        system("pause");
        start=clock();//下一关重新计时
        x=M/2;
        y=N/2;
        room[x][y]=2;//更新坐标
    }
else
        room[x][y]=2;//更新坐标
            }
        }
    }
```

编辑迷宫：

```
void modify(Maze &maze){ //编辑迷宫int
        flag=1;
        int laststep;
        int x1=M/2,y1=N/2,x=M/2,y=N/2; char c;
        while(1){
                system("cls");
                display(maze.room,0,0,flag+1);
                while(!kbhit());
                c=getch();
                if(c==-32)
        c=getch();
    switch(c){ case
    72:
                x1=x-1;
                y1=y;
                break;          // 向上
    case 80:
                x1=x+1;
```

```
        y1=y;
        break;      //向下
    case 75:
        x1=x;
        y1=y-1;
        break;      //向左
    case 77:
        x1=x;
        y1=y+1;      //向右
        break;
    case 48:
        flag=0;      /变墙
        break;
    case 49:
        flag=1;      /变路
        break;
    default:
            system("cls");
            printf("¥t¥t¥t*********************************¥n");
            printf("¥t¥t¥t        退 出 编 辑 !¥n");
            printf("¥t¥t¥t*********************************¥n");
            maze.status[x][y]=maze.room[x][y]=laststep;
            maze.status[M/2][N/2]=maze.room[M/2][N/2]=2;
            filesave(maze);
            system("pause");
        return;
    }
    if(x1==0||x1==M-1||y1==0||y1==N-1撞墙了
            continue;
    else if(x1==M-2&&y1==M-2)
            continue;
    else{
        if(flag)    /变路
            maze.status[x][y]=maze.room[x][y]=1;
```

```
                else                //变墙
                    maze.status[x][y]=maze.room[x][y]=0;
                laststep=maze.room[x1][y1];
                x=x1;
                y=y1;
                maze.status[x][y]=maze.room[x][y]=2; //定位
            }
        }
    }
最短路径：
voidminroad(Mazemaze){              //最短路径
    system("cls");
    printf("------------------------------\n\t 最 短 路 径 \n");
    printf("--------------------------------\n");
    Queue Q; initqueue(Q);
    posp1,p2;       //进队用 p1，出队用 p2
    int next[4][2]={{1,0},{0,1},{0,-1},{-1,0}};
    p1.x=M/2;
    p1.y=N/2;
    p1.parent=-1;
    p1.self=0;
    enqueue(Q,p1);
    do{
        dequeue(Q,p2);
        for(inti=0;i<4;i++){        //四个方向
            p1.x=p2.x+next[i][0]; //下一方向的坐标
            p1.y=p2.y+next[i][1];
            p1.parent=Q.front-1;
            p1.self=Q.rear;
            if(pass(maze,p1)){ //判断是否可通过
                markprint(maze,p1);//留下足迹
                enqueue(Q,p1);
                Q.p[p1.parent].next[i]=Q.rear-1; //父结点指向进队的子结点
```

```
            }
        }
        if(p2.x==M-2&&p2.y==N-2){
                printf("输 出 最 短 路 径 :");
                stepsdisplay(Q,p2,maze); //输出最短路径return;
        }
    }while(!emptyqueue(Q));
    printf("未找到最短路径!\n");
}
```

统计所有路径数目：

```
void findallroad(Maze &maze,pos p,int &num){//找从 p 开始的所有路径int
next[4][2]={{1,0},{0,1},{0,-1},{-1,0}};//移动方向
pos p1;
Elemtype temp;
for(inti=0;i<4;i++){        //四个方向
    p1.x=p.x+next[i][0];        //下一方向的坐标
    p1.y=p.y+next[i][1];
    if(pass(maze,p1)){ temp=maze.status[p1.x][p1.y];//
        保留状态markprint(maze,p1);//留下足迹
        if(p1.x==M-2&&p1.y==N-2)
            num++;
        else
            findallroad(maze,p1,num);//递归调用
        maze.status[p1.x][p1.y]=temp;//恢复状态
    }
}
void allroad(Maze maze){                //所有路径
    int num=0;
    pos p; p.x=M/2;
    p.y=N/2;
```

```
    markprint(maze,p);//留下足迹findallroad(maze,p,num);//
    寻找从 p 开始的所有通路if(num)
        printf("共有%d 条路径\n",num);
}
```

3.3 调试分析

测试数据及测试结果在 7 中有详细给出，这里省略。

时间复杂度分析：

play 函数为走出迷宫游戏，其时间复杂度也不确定，与通路的块数以及通路的布局有关。

display 函数为显示迷宫，被多次调用，因为要将迷宫一个一个地输出，所以时间复杂度为$O(m \times n)$

modify 函数为修改迷宫的函数，其时间复杂度不确定，与要修改的块相关。

fileread 和 filesave 函数保存一个 Maze 型数据，时间复杂度为$O(1)$。mazerand 函数用 rand()生成$m \times n$ 个伪随机数，所以时间复杂度为$O(m \times n)$

minroad 函数对可走块四个方向都要判断是否可通，若可走块的数目为n，则最多需要判断 $4n$ 次，时间复杂度为$O(n)$

allroad 函数对可走块四个方向都要判断是否可通，时间复杂度同样为$O(n)$。

调试思考：

1 注意菜单选择函数的设计，它们的功能和菜单之间的联系。注意各个菜单函数中迷宫是否要初始化。

2 为了让迷宫边界更加明显，默认设置边界部分全部为障碍。

3 用键盘操作老鼠的移动方向，可用 switch 函数，根据输入的按键，进入不同分支，按照不同方式修改当前老鼠坐标。

4 迷宫游戏需通过计时，判断是否在规定时间内完成迷宫。计时可通过不断获取系统时间实现。同时，需在动画界面不断更新计时，每隔一段时间刷新屏幕。

5 迷宫游戏要用动画界面实现，在按动键盘或计时更新时都要刷新屏幕，可用一个 while 循环实现，当时间间隔足够或者键盘有操作时退出while 循环并刷新屏幕。

6 迷宫编辑函数 modify 中为实现图形化的界面，让修改更加的直观，设置每按下一个按键就根据所按下按键修改地图，并刷新屏幕，重新输出迷宫，从而实现可直观编辑迷宫的目的。

7 求最短路径要用 BFS 的方法，从起点即树的根节点按层遍历，第一次到达终点也就是遍历层数最少时到达终点，求得最短路径。

8 求所有可通路径，让方向有优先级，先沿着优先级高的方向走，当走到终点或者没有通路可走时，尝试下一个方向，直至所有可走块的方向都尝试完后即可得到所有

可通路径的数目。

3.4 用户手册

1　本程序执行文件为maze.exe。

2　用户进入系统后根据提示，输入数字进入相应的子菜单，实现相应功能。

3　本系统可实现走出迷宫游戏、编辑迷宫、保存迷宫到文件、从文件读入迷宫、随机生成迷宫、迷宫最短路径求解、统计迷宫所有可通路径数目的功能。

4 总结

本次实验的内容比较难，几乎综合了迷宫求解和动画输出界面，还好我有迷宫求解的题 目，先做了迷宫求解，稍微适应一下使用堆栈求解后，再做的这个题目。事实上，这个题目的最短路径求解和迷宫求解不同，比迷宫求解要难，用 BFS 求最短路径很难想到，看了网上的一些代码才有思路。求所有路径也非常难，也是看了网上的提示才做的。不过这也让我 意识到了从了解 BFS，DFS 到应用它们之间的差距。的确需要较多的练习才能用好BFS 和DFS。除此之外，我也意识到要多看一些算法，如果脑子里全是自己想的不规整的方法，解 决问题时会比较吃力，遇到问题不能得心应手。

5、程序清单:(见附录)

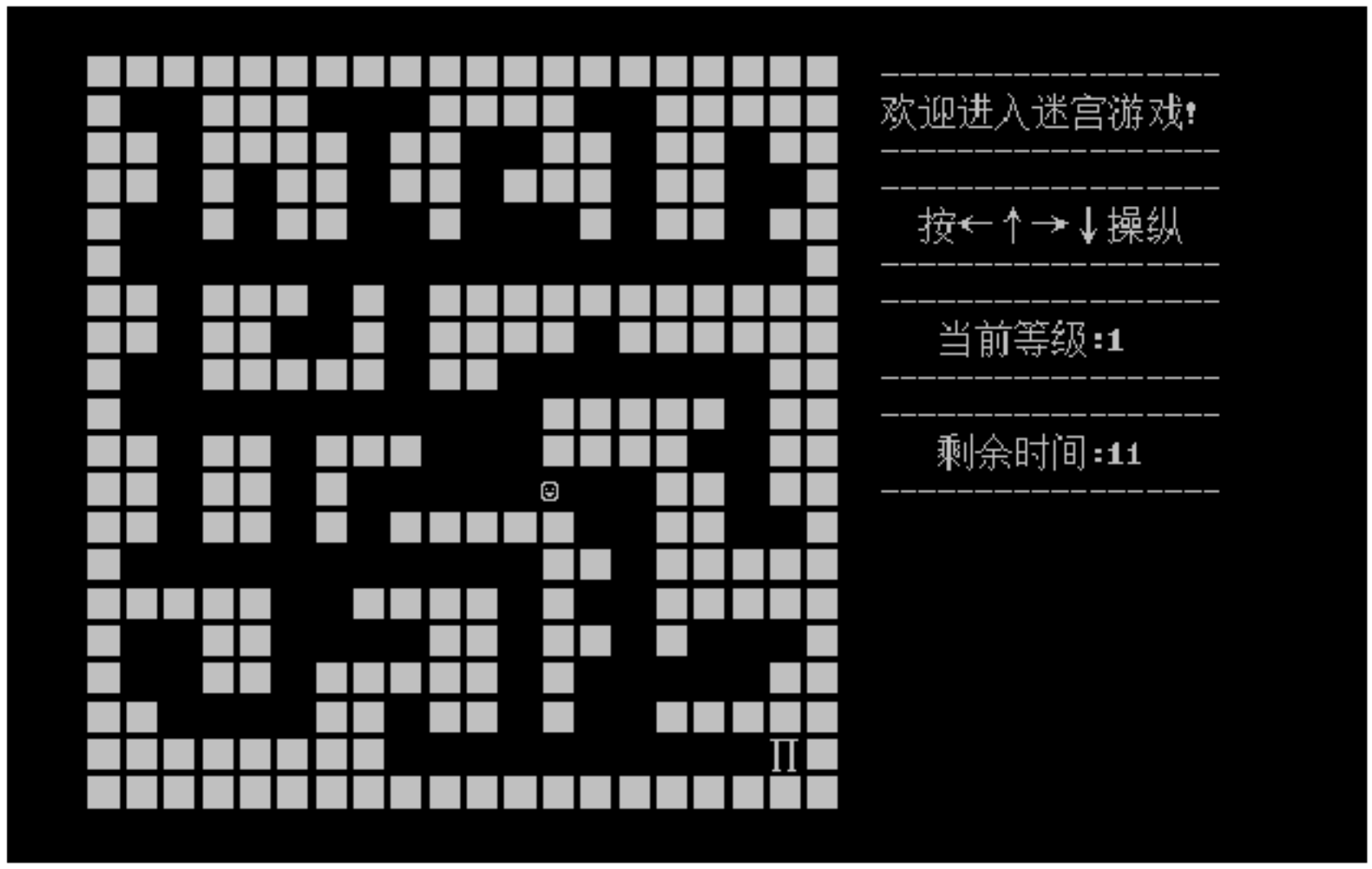
7、程序运行结果

主界面

菜单 1　开始迷宫游戏

倒计时

游戏界面



超时提示失败

升级提示

菜单 2　编辑迷宫

编辑前: