**湖南工业大学**

**课程设计（实训）**

**说明：红色标记的地方，有的是说明文字，看完请删除，有的是需要根据自己课题情况进行修改的。所有内容定稿后，应全部改成黑色字体。**

**资 料**

|  |  |  |
| --- | --- | --- |
| 学年学期 | ： | 2019-2020-1 |
| 课程名称 | ： | 面向过程程序设计（C语言）课程设计 |
| 题目 | ： | 迷宫 |
| 起止日期 | ： | 2019年12月28日 ～ 2020年01月03日 |
| 学院 | ： | 计算机学院 |
| 班级 | ： | 计算机类1907 |
| 学生姓名 | ： | 陈树雄 |
| 学号 | ： | 19408000250 |

目 录 清 单

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 材 料 名 称 | 资料数量 | 备 注 |
| 1 | 课程设计任务书 | 1 |  |
| 2 | 课程设计（实训）报告 | 1 |  |
| 3 | 设计文档和源程序（电子文档） | 1 |  |

课程设计（实训）任务书

学 院： 计算机学院 学生姓名： 陈树雄 专业班级：计算机类1907

课程名称： 面向过程程序设计（C语言）课程设计 设计题目： 迷宫

完成期限：自 2019 年 12 月 28 日至 2020 年 01 月 03 日共 1 周

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 内  容  及  任  务 | **一、课题的主要内容**  综合运用C语言程序设计课程的主要知识，设计并实现一个迷宫软件。主要内容包括：   1. 在游戏开始的时候，显示迷宫的画面，入口和出口有标识。 2. 有两种选择，一种由人来寻找出路，一种由计算机来寻找出路。 3. 选择人找出路时，显示每一步的结果，到边了和遇上障碍，发出“嘟”的叫声。走到出口处，应给出“胜利”的字样。 4. 选择计算机找出路时，用一条有颜色的线画出路径，若找不出出口就显示“无出路”的字样。   **二、主要任务及说明**   1. 人机交互友好，页面简单，合理。 2. 设置难度，可由玩家来选择，在不同的难度下生成的地图复杂程度会有区别。 3. 生成的地图应该具有有出路和没有出路这两种情况。 4. 由人来寻找出路时，玩家操纵的“人”具有标识度，每一步都同步于玩家的操作，如果触碰入口，入口将会变成蓝色0.3秒并有系统响应音效，如果触碰墙壁，墙壁入口将会变成红色0.3秒并有系统响应音效，在走到出口的时候，出口将会变换多种不同颜色并播放胜利音效，在一定的延迟之后会清除屏幕，显示“恭喜通关”字样，并且询问玩家是否再来一局。 5. 由计算机来寻找出路时，墙壁是用白色的“口”组成，路径是用红色的“国”组成，会显示当前迷宫下的通关路径。 | | | | |
| 进  度  安  排 | 起止日期 | | 工作内容 | | |
| 2019.12.28~2019.12.29 | | 问题分析和设计； | | |
| 2019.12.30~2020.1.1 | | 熟悉软件开发工具、编码系统、系统测试； | | |
| 2020.1.2~2020.1.3 | | 答辩，撰写并提交课程设计报告 (含电子文档)、源程序等。 | | |
| 主  要  参  考  资  料 | [1]谭浩强.C程序设计设计题解与上机指导(第5版) [M].清华大学出版社，2017.08  [2]颜晖,张泳.C语言程序设计实验与习题指导(第3版) [M].高等教育出版社，2015.08  [3]刘强，童启.C语言程序设计实验教程[M].电子工业出版社，2018.02  [4]刘博.C语言课程设计[M].机械工业出版社，2013.04  [5]明日科技. C语言函数参考手册[M].清华大学出版社，2012.02 | | | | |
| 同组信息 | 学号 | 姓名 | | 分工负责的内容 | 备注 |
| 无 |  | |  |  |

签名——刘强-1指导教师（签字）： 年 月 日

系（教研室）主任（签字）： 年 月 日



课程设计（实训）报告

|  |  |  |
| --- | --- | --- |
| 学年学期 | ： | 2019-2020-1 |
| 课程名称 | ： | 面向过程程序设计（C语言）课程设计 |
| 题目 | ： | 迷宫 |
| 起止日期 | ： | 2019年12月28日 ～ 2020年01月03日 |
| 学院 | ： | 计算机学院 |
| 班级 | ： | 计算机类1907 |
| 学生姓名 | ： | 陈树雄 |
| 学号 | ： | 19408000250 |

|  |  |  |  |
| --- | --- | --- | --- |
| 成绩评定 | ： |  |  |
| 课程目标 | | 成绩 | 评定人签名 |
| 目标1 | |  |  |
| 目标2 | |  |  |
| 目标3 | |  |  |

**2020年 01 月**

目 录

1、“目录”两字要求是 二号 黑体

2、目录要求自动生成，页码右对齐，字体为小四号、宋体、1.5倍行距

3、目录页的编码要求使用罗马数字编码，即I、II

看完后请删除此框和后面红色字体内容。

[1 课题简介 1](#_Toc29041139)

[1.1 系统简介 1](#_Toc29041140)

[1.2 开发环境及开发工具 1](#_Toc29041141)

[2 问题分析 2](#_Toc29041142)

[2.1 需求调查 2](#_Toc29041143)

[2.1.1 功能要求 2](#_Toc29041144)

[2.1.1 技术要求 2](#_Toc29041145)

[2.2 数据分析 2](#_Toc29041146)

[3 系统设计 3](#_Toc29041147)

[3.1 系统功能设计 3](#_Toc29041148)

[3.2 系统的功能分析 3](#_Toc29041149)

[3.3 系统的数据结构设计 4](#_Toc29041150)

[4 功能模块的设计与实现 5](#_Toc29041151)

[4.1 系统主函数设计与实现 5](#_Toc29041152)

[4.1.1主函数功能说明 5](#_Toc29041153)

[4.1.2实现流程 5](#_Toc29041154)

[4.1.3函数功能描述 5](#_Toc29041155)

[4.1.4 函数的调用关系 6](#_Toc29041156)

[4.1.5 关键源代码 7](#_Toc29041157)

[4.2创建地图模块设计与实现 8](#_Toc29041158)

[4.2.1 功能说明 8](#_Toc29041159)

[4.2.2 实现流程 8](#_Toc29041160)

[4.2.3关键源代码 9](#_Toc29041161)

[4.3玩家移动模块设计与实现 15](#_Toc29041162)

[4.3.1功能说明 15](#_Toc29041163)

[4.3.2实现流程 16](#_Toc29041164)

[4.3.3关键源代码 16](#_Toc29041165)

[4.4自动寻路模块设计与实现 27](#_Toc29041166)

[4.4.1功能说明 27](#_Toc29041167)

[4.4.2实现流程 27](#_Toc29041168)

[4.4.3关键源代码 28](#_Toc29041169)

[5 系统测试 32](#_Toc29041170)

[5.1 软件测试 32](#_Toc29041171)

[5.1.1 单元测试 32](#_Toc29041172)

[5.1.2 集成测试 38](#_Toc29041173)

[5.2 系统有效性 41](#_Toc29041174)

[6 总结 42](#_Toc29041175)

[7 参考文献 43](#_Toc29041176)

# 课题简介

## 系统简介

我所完成的课程设计是一个游戏类的系统，是要实现用户可以借助这个项目玩迷宫的，我利用“口”来组成我的墙壁，用“入”和“出”分别代表入口和出口，用“人”来做玩家所操纵移动的人物，玩家在每一步操作之后都会有判断，如果触碰墙壁会有反馈。

开发了这个项目之后，我学到了很多新的算法和知识，这个项目开发出来是可以用来进行日常生活中没事的时候进行消遣娱乐的工具，也可以用来测试编译器是否在Debug方面存在不可以调试之类的问题。

我现在开发出来的这个迷宫系统，在游戏开始界面已经实现了很好的人机交互的功能，对于玩家给出的每一个指令，无论是正确的还是错误的，都可以给出相关的对应的响应，如果是正确的指令，系统就会给出相应的操作，要是是错误的指令，系统就会给出提示，并让玩家重新输入。

在游戏过程中，如果是由玩家来操控在触碰了墙壁时，墙壁会显示红色并且会有系统提示音，在达到了出口的时候，会恭喜玩家获得胜利，并且播放胜利音效。

## 开发环境及开发工具

开发环境：Win10下装有C语言编译器的PC机一台

开发工具：Microsoft Visual Studio 2019

我使用的是由微软公司开发的Microsoft Visual Studio 2019来开发的项目，我选择它的原因有几点，一是它的错误提示，在没有编译之前它就会提示你你错误的地方，非常方便；二是它的代码中，不同的属性会有不同的颜色，就比如宏定义是紫色的，函数是黄色的，很有区分度；三是它使用起来也比较舒服，不像其它编译器那样频繁出错。

# 问题分析

## 需求调查

### 功能要求

功能要求如下：

1. 迷宫生成功能：要生成一个玩家可以行走于其中的迷宫，墙壁是不可穿过的，并且起点终点要有标识度。
2. 玩家移动功能：在玩家模式下，玩家要可以按下指定键进行移动。
3. 自动寻路功能：在自动寻路的模式下，要在迷宫中画出一条有颜色的线路来告诉玩家这个迷宫的通路，要是没有路应当提示没有出路。

### 2.1.1 技术要求

技术要求如下：

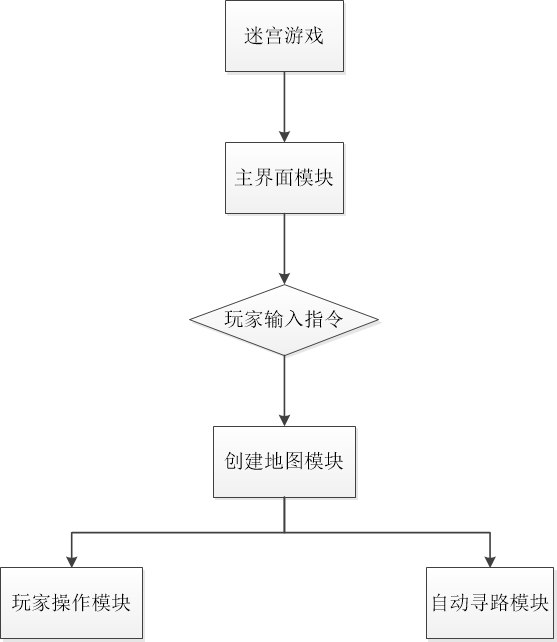
1. 迷宫生成方面：要掌握一定的算法，由于打算使用三种算法结合，是打算利用随机数来产生概率，以此来生成迷宫，需要学习深度优先算法、递归分割算法，同时需要熟练运用随机数。然后，需要掌握动态内存分配，利用指向指针的指针来创建动态二维数组储存迷宫。
2. 玩家移动方面：需要对控制台句柄有一定的了解，要掌握光标的移动和隐藏的指令，同时要对光标的输出与移动有正确的理解，并且要在移动之前判断是否为墙或者终点，并进行相对应的操作。
3. 自动寻路方面：需要熟练掌握DFS（深度优先算法）或者BFS（广度优先算法）才可以在迷宫之中寻找出出路，在寻找到出路的时候将其在数组中的代表Road（路）的值转变成ANS（通路）的值，然后打印即可。

## 数据分析

特殊情况介绍：由于迷宫游戏不需要用到文件，而且也不需要用到数据处理与分析，所以在这里没有数据流图和数据字典

# 系统设计

## 系统功能设计

迷宫游戏总体上才用了模块化程序设计思想，将功能细化，便于使用。从不同功能上分主要可以分为主界面模块、创建地图模块、玩家操作模块、自动寻路模块。如图3-1所示。

3 - 1

## 系统的功能分析

主界面模块是为了打印开始界面，显示人工寻路和电脑寻路两个不同的操作选项，并对玩家输入的指令进行判断。

创建地图模块是为了产生玩家游戏所需要的迷宫地图。

玩家操作模块是在玩家选择了玩家寻路之后，玩家将会在迷宫之中寻找一条出路，这个模块将会对玩家的操作进行分析并给出相对应的响应。

自动寻路模块是在玩家选择了自动寻路之后，电脑将会对迷宫进行分析，最后打印出一条迷宫的通路。

## 系统的数据结构设计

系统的数据结构如下：

1. 储存玩家当前所在的坐标，定义了一个结构体，结构体中有两个数据项：横坐标和纵坐标。

|  |
| --- |
| struct Position  {  int x; //横坐标  int y; //纵坐标  }Player; |

2. 定义了一个全局二维数组来储存在自动寻路中DFS算法所需要的方向

|  |
| --- |
| int Dst[4][2] = { {-1,0}, {1,0}, {0,-1}, {0,1} }; |

3. 定义全局二维数组来储存在调用函数中用动态内存分配临时产生的地图

|  |
| --- |
| int Dot[L + 5][L + 5]; |

# 功能模块的设计与实现

## 系统主函数设计与实现

### 4.1.1主函数功能说明

输出界面，并对判断玩家输入，如果是“1”，则跳至玩家操作主体函数，如果是“2”，则跳至自动寻路主体函数，如果输入有误，则会进行提示并让玩家重新输入。

### 4.1.2实现流程

4 - 1

### 4.1.3函数功能描述

main()函数用以生成主体，并让程序得以展开操作。

HideCursor()函数的原型为void HideCursor(int i)，用以根据 i 的值来判断隐藏还是恢复光标

turnto()函数的原型为void turnto(int x, int y)，用以控制光标的移动

move\_LR()函数的原型是void move\_LR(int x, int y)，用以控制人物在迷宫中向左右移动

move\_UD()函数的原型是void move\_UD(int x, int y)，用以控制人物在迷宫中向上下移动

Color()函数的原型是void Color(const unsigned short color)，用以调整控制台打印的颜色

Create\_main\_DF()函数的原型是void Create\_main\_DF()，是用深度优先创造地图这个方法的主干

Create\_Maze\_DF()函数的原型是void Create\_Maze\_DF(int\*\* maze, int x, int y)，是深度优先生成迷宫的算法实现

Create\_main\_RS()函数的原型是void Create\_main\_RS()，是用递归分割创造地图这个方法的主干

Create\_Maze\_RS()函数的原型是void Create\_Maze\_RS(int\*\* maze, int x1, int y1, int x2, int y2)，是递归分割生成迷宫的算法实现

Create\_main\_Random()函数的原型是void Create\_main\_Random()，是用随机法这个方法的主干

Create\_Maze\_Random()函数的原型是void Create\_Maze\_Random(int\*\* maze)，是随机法生成迷宫的算法实现

Create\_Paint()函数的原型是void Create\_Paint(int\*\* maze)，是用来画出迷宫并将动态二维数组中的迷宫转移到全局二维数组中的

Game\_1\_main()函数的原型是void Game\_1\_main()，是玩家操作主体函数

Game\_1\_Control\_S0()函数的原型是void Game\_1\_Control\_S0()，是玩家在用深度优先和随机法生成的地图下进行操作的函数

Game\_1\_Control\_S1()函数的原型是void Game\_1\_Control\_S1()，是玩家在用递归分割生产的地图下进行操作的函数

Game\_2\_main()函数的原型是void Game\_2\_main()，是自动寻路主体函数

DFS\_main()函数的原型是void DFS\_main()，是深度优先搜索的主体函数

DFS\_Make()函数的原型是void DFS\_Make(int x, int y)，是深度优先算法实现部分

DFS\_main\_Instance()函数的原型是void DFS\_main\_Instance()，是深度优先搜索特殊判断递归分割的主体函数

DFS\_Make\_Instance()函数的原型是void DFS\_Make\_Instance(int x, int y)，是深度优先搜索特殊判断递归分割的实现部分

Music\_win()函数的原型是void Music\_win(int n)，实现播放胜利音效

### 4.1.4 函数的调用关系

main()直接调用的函数：HideCusor()、Game\_1\_main()、Game\_2\_main()

HideCursor()直接调用的函数：无

turnto()直接调用的函数：无

move\_LR()直接调用的函数：turnto()

move\_UD()直接调用的函数：turnto()

Color()直接调用的函数：无

Create\_main\_DF()直接调用的函数：Create\_Maze\_DF()、Create\_Paint()

Create\_Maze\_DF()递归调用的函数：Create\_Maze\_DF()

Create\_main\_RS()直接调用的函数：Create\_Maze\_RS()

Create\_Maze\_RS()递归调用的函数：Create\_Maze\_RS()

Create\_main\_Random()直接调用的函数：Create\_Maze\_Random()、Create\_Paint()

Create\_Maze\_Random()直接调用的函数：无

Create\_Paint()直接调用的函数：无

Game\_1\_main()直接调用的函数：Game\_main\_DF()、Game\_main\_RS()、Game\_main\_Random()、Game\_1\_Control\_S0()、Game\_1\_Control\_S1()

Game\_1\_Control\_S0()直接调用的函数：Music\_win()

Game\_1\_Control\_S1()直接调用的函数：Music\_win()

Game\_2\_main()直接调用的函数：Game\_main\_DF()、Game\_main\_RS()、Game\_main\_Random()、DFS\_main()、DFS\_main\_Instance()

DFS\_main()直接调用的函数：DFS\_Make()

DFS\_Make()直接调用的函数：无

DFS\_main\_Instance()直接调用的函数：DFS\_Make\_Instance()

DFS\_Make\_Instance()直接调用的函数：无

Music\_win()直接调用的函数：无

### 4.1.5 关键源代码

|  |
| --- |
| int main(void) //主函数  {  Rp:  int e = 0; //e 是用来判断是否输入错误并对界面进行调整的变量  Re:  /\*这是后面输入错误的情况\*/  if (e == 1)  {  Sleep(700);  e = 2;  }  system("cls");  Color(9);  printf("\n\n\n\n\t\t\t\t\tWelcome to duidui's Game --- Maze !\n\n");  /\*这个是刚进来时的操作\*/  if (e != 2)  Sleep(500); //延时0.5秒  else  e = 0;  Color(7);  printf("\t\t\t\t\t 请选择你想要进行的游戏模式:\n\n");  printf("\t\t\t\t\t 1.人工摸索探路\n\n");  printf("\t\t\t\t\t 2.缺德地图导航\n\n");  HideCursor(0); //隐藏光标  char f = \_getch(); //f 用来做输入的那个字符，判断进行哪种操作  system("cls");  printf("\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t 请输入一个数值，值越小难度越大\n\n\n\t\t\t\t\t\t\t ");  HideCursor(1);  scanf\_s("%d", &Rank);  HideCursor(0);  if (f == '1') //人工  Game\_1\_main();  else if (f == '2') //智能  Game\_2\_main();  else //如果输入的字符与操作字符无关，则重新输入  {  Color(4);  printf("\t\t\t\t\t 他娘的输错了嗷，给老子重新输入！\n");  e = 1;  goto Re;  }  if (T == 1)  goto Rp;  /\*等待退出\*/  system("cls");  printf("\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t 按任意键退出！\n\n\n\n\n\n\n\n");  int z = \_getch();  return 0;  } |

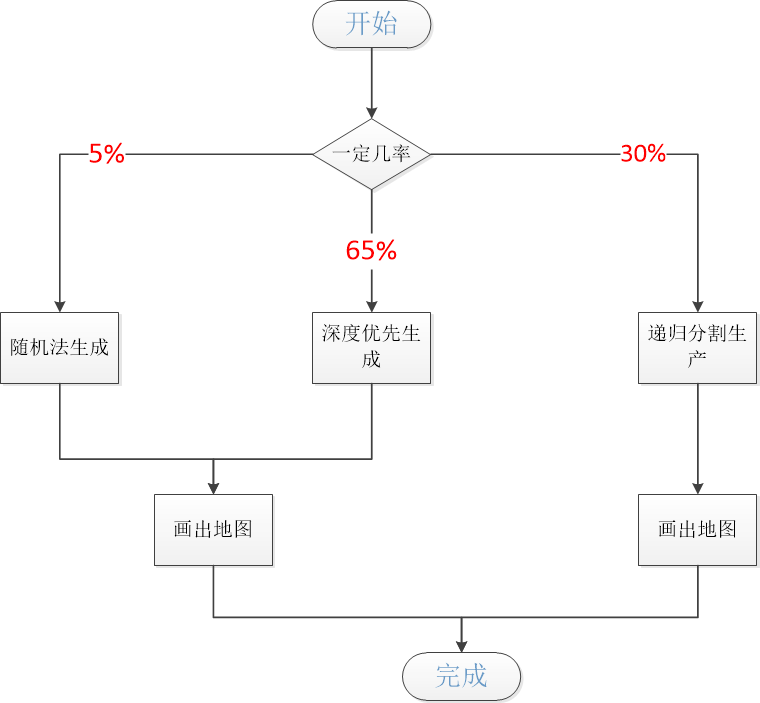
## 4.2创建地图模块设计与实现

### 4.2.1 功能说明

此模块用Game\_main\_DF()、Game\_main\_RS()、Game\_main\_Random()这三个函数实现，可以创造每次都不同的随机地图

### 4.2.2 实现流程

创建地图模块流程图如4 – 2所示

* 1. - 2

### 4.2.3关键源代码

1. 利用随机数产生概率，以此选择不同算法生成迷宫

|  |
| --- |
| /\*利用随机数来产生概率\*/  int random = rand() % 99 + 1;  if (random <= 5)  {  S = 0;  Create\_main\_Random();  }  else if (random >= 80)  {  S = 1;  Create\_main\_RS();  }  else  {  S = 0;  Create\_main\_DF();  } |

2. 随机法生成迷宫

|  |
| --- |
| void Create\_main\_Random() //创造地图——主干——随机创造  {  /\*给二维数组分配动态内存\*/  int\*\* Maze = (int\*\*)malloc(L \* sizeof(int\*));  for (int i = 0; i < L; i++)  Maze[i] = (int\*)calloc(L, sizeof(int));  /\*生成迷宫\*/  Create\_Maze\_Random(Maze);  /\*画迷宫\*/  Create\_Paint(Maze);  /\*释放内存\*/  for (int i = 0; i < L; i++)  free(Maze[i]);  free(Maze);  }  void Create\_Maze\_Random(int\*\* maze) //创造地图——生成迷宫——随机创造  {  srand((unsigned)time(NULL));  /\*使用随机数来生成迷宫中的墙\*/  for (int i = 0; i < L; i++)  for (int j = 0; j < L; j++)  if (i == 1 || i == L - 2 || j == 1 || j == L - 2)  maze[i][j] = Wall;  else if (i == 0 || i == L - 1 || j == 0 || j == L - 1)  maze[i][j] = Road;  else if (((i == 0 || i == 1) && j == 1) || ((i == L - 1 || i == L - 2) && j == L - 2)))  maze[i][j] = Road;  else  maze[i][j] = rand() % 2;  } |

3. 递归分割生成并画出迷宫

|  |
| --- |
| void Create\_main\_RS() //创造地图——主干——递归分割  {  srand((unsigned)time(NULL)); //随机数种子  /\*给二维数组分配动态内存\*/  int\*\* Maze = (int\*\*)malloc(L \* sizeof(int\*));  for (int i = 0; i < L; i++)  Maze[i] = (int\*)calloc(L, sizeof(int));  /\*最外侧是墙\*/  for (int i = 0; i < L; i++)  {  Maze[0][i] = WALL;  Maze[i][0] = WALL;  Maze[L - 1][i] = WALL;  Maze[i][L - 1] = WALL;  }  /\*生成迷宫\*/  Create\_Maze\_RS(Maze, 1, 1, L - 2, L - 2);  /\*起点与终点\*/  Maze[2][0] = START;  Maze[L - 2][L - 1] = END;  /\*画迷宫\*/  for (int i = 0; i < L; i++)  {  for (int j = 0; j < L; j++)  if (Maze[i][j] == WALL)  printf("口");  else  printf(" ");  printf("\n");  }  turnto(0, 2);  printf("入");  turnto(2 \* (L - 1), L - 2);  printf("出");  /\*把临时储存的地图转入全局数组中\*/  for (int i = 0; i < L; i++)  for (int j = 0; j < L; j++)  Dot[i][j] = Maze[i][j];  /\*释放内存\*/  for (int i = 0; i < L; i++)  free(Maze[i]);  free(Maze);  }  void Create\_Maze\_RS(int\*\* maze, int x1, int y1, int x2, int y2) //创造地图——生成迷宫——递归分割  {  /\*判断能否继续分割\*/  if (x2 - x1 < 2 || y2 - y1 < 2)  return;  /\*随机取点\*/  int x = rand() % (x2 - x1 - 1) + x1 + 1;  int y = rand() % (y2 - y1 - 1) + y1 + 1;  /\*划墙\*/  for (int i = x1; i <= x2; i++)  maze[i][y] = WALL;  for (int i = y1; i <= y2; i++)  maze[x][i] = WALL;  /\*递归分割，继续划分区域\*/  Create\_Maze\_RS(maze, x1, y1, x - 1, y - 1);  Create\_Maze\_RS(maze, x + 1, y + 1, x2, y2);  Create\_Maze\_RS(maze, x + 1, y1, x2, y - 1);  Create\_Maze\_RS(maze, x1, y + 1, x - 1, y2);  /\*随机取其中三面墙\*/  int r[4] = { 0 };  r[rand() % 4] = 1;  /\*在墙上随机取点开孔\*/  for (int i = 0; i < 4; i++)  if (r[i] == 0)  {  int rx = x;  int ry = y;  switch (i)  {  case 0:  /\*判断该位置是否能确保打通相邻两块区域，判断依据，上下左右位置最多只有两面墙，下面一样\*/  do  {  rx = rand() % (x - x1) + x1;  } while (maze[rx - 1][ry] + maze[rx + 1][ry] + maze[rx][ry - 1] + maze[rx][ry + 1] > 2 \* WALL);  break;  case 1:  do  {  ry = rand() % (y2 - y) + y + 1;  } while (maze[rx - 1][ry] + maze[rx + 1][ry] + maze[rx][ry - 1] + maze[rx][ry + 1] > 2 \* WALL);  break;  case 2:  do  {  rx = rand() % (x2 - x) + x + 1;  } while (maze[rx - 1][ry] + maze[rx + 1][ry] + maze[rx][ry - 1] + maze[rx][ry + 1] > 2 \* WALL);  break;  case 3:  do  {  ry = rand() % (y - y1) + y1;  } while (maze[rx - 1][ry] + maze[rx + 1][ry] + maze[rx][ry - 1] + maze[rx][ry + 1] > 2 \* WALL);  break;  default:  break;  }  maze[rx][ry] = ROUTE;  }  } |

4. 深度优先生成地图

|  |
| --- |
| oid Create\_main\_DF() //创造地图——主干——深度优先  {  srand((unsigned)time(NULL)); //随机数种子  /\*给二维数组分配动态内存\*/  int\*\* Maze = (int\*\*)malloc(L \* sizeof(int));  for (int i = 0; i < L; i++)  Maze[i] = (int\*)calloc(L, sizeof(int));  /\*最外围层设为路径的原因，为了防止挖路时挖出边界，同时为了保护迷宫主体外的一圈墙体被挖穿\*/  for (int i = 0; i < L; i++)  {  Maze[i][0] = Road;  Maze[0][i] = Road;  Maze[i][L - 1] = Road;  Maze[L - 1][i] = Road;  }  /\*以（2，2）为起点创造迷宫\*/  Create\_Maze\_DF(Maze, 2, 2);  /\*迷宫入口及出口\*/  Maze[2][1] = Road;  /\*由于算法随机性，出口有一定概率不在（L-3,L-2）处，此时需要寻找出口\*/  for(int i = L - 3; i >= 0; i--)  if (Maze[i][L - 3] == Road)  {  Maze[i][L - 2] = Road;  break;  }  Create\_Paint(Maze); //画出地图  /\*释放内存\*/  for (int i = 0; i < L; i++)  free(Maze[i]);  free(Maze);  }  void Create\_Maze\_DF(int\*\* maze, int x, int y) //创造地图——生成迷宫——深度优先  {  maze[x][y] = Road;  /\*确保四个方向随机，即打乱方向\*/  int direction[4][2] = { { 1,0 },{ -1,0 },{ 0,1 },{ 0,-1 } };  for (int i = 0; i < 4; i++)  {  int r = rand() % 4;  int temp = direction[0][0];  direction[0][0] = direction[r][0];  direction[r][0] = temp;  temp = direction[0][1];  direction[0][1] = direction[r][1];  direction[r][1] = temp;  }  /\*朝四个方向开挖\*/  for (int i = 0; i < 4; i++)  {  int dx = x;  int dy = y;  /\*控制挖的距离，由Rank来调整\*/  int range = (Rank == 0 ? 0 : rand() % Rank) + 1;  while (range > 0)  {  dx += direction[i][0];  dy += direction[i][1];  if (maze[dx][dy] == Road) //排除回头路  break;  /\*判断是否挖穿路径\*/  int count = 0;  for (int j = dx - 1; j < dx + 2; j++)  for (int k = dy - 1; k < dy + 2; k++)  /\* abs(j - dx) + abs(k - dy) == 1 确保只判断九宫格内四个特定位置 即上下左右\*/  if (abs(j - dx) + abs(k - dy) == 1 && maze[j][k] == Road)  count++;  if (count > 1)  break;  /\*确保不会挖穿，前进\*/  range--;  maze[dx][dy] = Road;  }  /\*没有挖穿危险，以此节点递归\*/  if (range <= 0)  Create\_Maze\_DF(maze, dx, dy);  }  } |

5. 画出随机法和深度优先创造的地图

|  |
| --- |
| void Create\_Paint(int\*\* maze) //创造地图——画地图  {  /\*将地图画出\*/  for (int i = 0; i < L; i++)  {  for (int j = 0; j < L; j++)  if (maze[i][j] == Wall)  printf("口");  else if (i == 2 && j == 0)  printf("入");  else if (i == L - 3 && j == L - 1)  printf("出");  else  printf(" ");  printf("\n");  }  /\*把临时储存的地图转入全局数组中\*/  for (int i = 0; i < L; i++)  for (int j = 0; j < L; j++)  Dot[i][j] = maze[i][j];  Dot[2][0] = Start;  Dot[L - 3][L - 1] = End;  } |

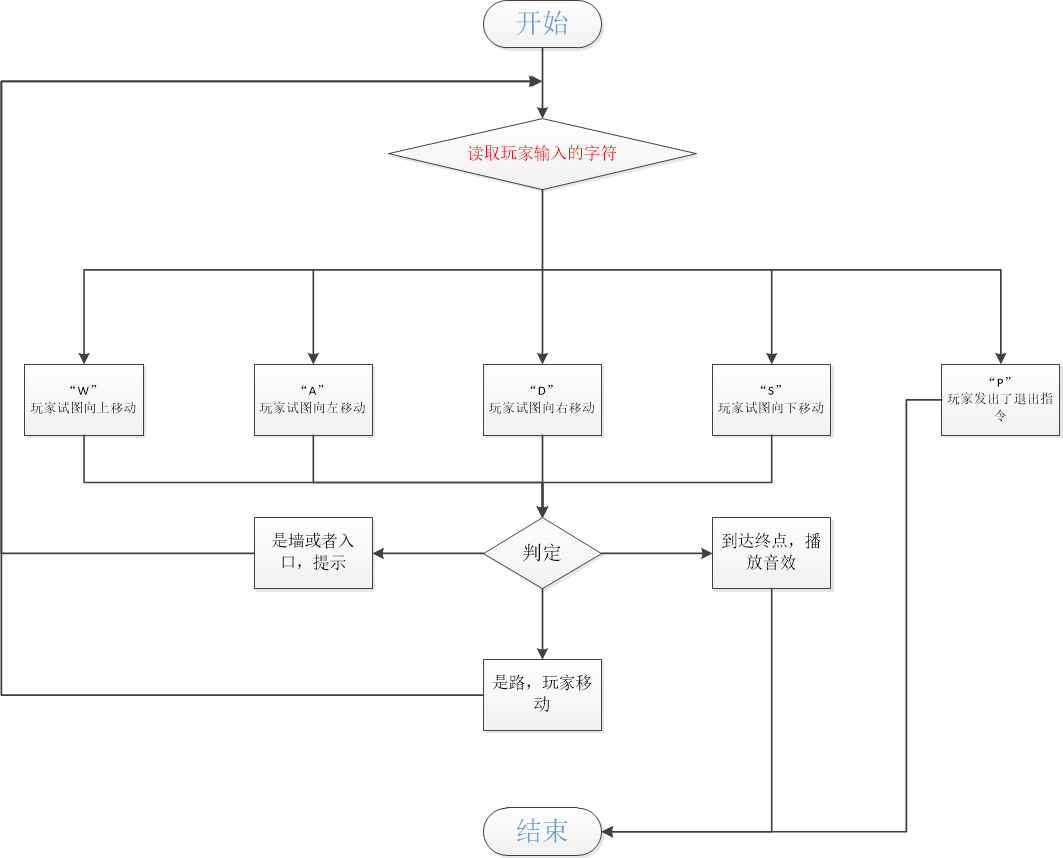
## 4.3玩家移动模块设计与实现

### 4.3.1功能说明

这方面是为了实现玩家在地图中移动的操作，包括碰到墙、终点之后的判断

### 4.3.2实现流程

玩家移动模块流程图如4 – 3所示

4 – 3

### 4.3.3关键源代码

1. 玩家操作 – 在深度优先和随机法生成的迷宫中的操作

|  |
| --- |
| void Game\_1\_Control\_S0(int a, int b) //人工操作\_1  {  char c;  Player.x = b; //这个是纵坐标  Player.y = a - 1; //这个才是横坐标  while ((c = \_getch()) != 'p') //字符的移动  {  if (c == (int)'w')  {  if (Dot[Player.x - 1][Player.y] == Wall)  {  turnto(a, b - 1);  Color(4);  printf("口");  turnto(a, b - 1);  Sleep(300);  Color(7);  printf("口");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x - 1][Player.y] == Start)  {  turnto(a, b - 1);  Color(3);  printf("入");  turnto(a, b - 1);  Sleep(400);  Color(7);  printf("入");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x - 1][Player.y] == End)  {  for (int i = 0; i <= 15; i++)  {  turnto(a, b - 1);  Color(i);  printf("出");  Sleep(100);  Music\_win(i);  }  system("cls");  printf("\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t哈哈哈哈恭喜你通关！\n\n\n\n\n\n\n\n");  break;  }  else  {  turnto(a, b - 1);  move\_UD(a, b);  turnto(a, b - 1);  b = b - 1;  Player.x = Player.x - 1;  }  }  if (c == (int)'a')  {  if (Dot[Player.x][Player.y - 1] == Wall)  {  turnto(a - 2, b);  Color(4);  printf("口");  turnto(a - 2, b );  Sleep(300);  Color(7);  printf("口");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x][Player.y - 1] == Start)  {  turnto(a - 2, b);  Color(3);  printf("入");  turnto(a - 2, b);  Sleep(400);  Color(7);  printf("入");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x][Player.y - 1] == End)  {  for (int i = 0; i <= 15; i++)  {  turnto(a - 2, b);  Color(i);  printf("出");  Sleep(100);  Music\_win(i);  }  system("cls");  printf("\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t哈哈哈哈恭喜你通关！\n\n\n\n\n\n\n\n");  break;  }  else  {  turnto(a - 2, b);  move\_LR(a, b);  turnto(a - 2, b);  a = a - 2;  Player.y = Player.y - 1;  }  }  if (c == (int)'s')  {  if (Dot[Player.x + 1][Player.y] == Wall)  {  turnto(a, b + 1);  Color(4);  printf("口");  turnto(a, b + 1);  Sleep(300);  Color(7);  printf("口");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x + 1][Player.y] == Start)  {  turnto(a, b + 1);  Color(3);  printf("入");  turnto(a, b + 1);  Sleep(400);  Color(7);  printf("入");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x + 1][Player.y] == End)  {  for (int i = 0; i <= 15; i++)  {  turnto(a, b + 1);  Color(i);  printf("出");  Sleep(100);  Music\_win(i);  }  system("cls");  printf("\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t哈哈哈哈恭喜你通关！\n\n\n\n\n\n\n\n");  break;  }  else  {  turnto(a, b + 1);  move\_UD(a, b);  turnto(a, b + 1);  b = b + 1;  Player.x = Player.x + 1;  }  }  if (c == (int)'d')  {  if (Dot[Player.x][Player.y + 1] == Wall)  {  turnto(a + 2, b);  Color(4);  printf("口");  turnto(a + 2, b);  Sleep(300);  Color(7);  printf("口");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x][Player.y + 1] == Start)  {  turnto(a + 2, b);  Color(3);  printf("入");  turnto(a + 2, b);  Sleep(400);  Color(7);  printf("入");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x][Player.y + 1] == End)  {  for (int i = 0; i <= 15; i++)  {  turnto(a + 2, b);  Color(i);  printf("出");  Sleep(100);  Music\_win(i);  }  system("cls");  printf("\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t哈哈哈哈恭喜你通关！\n\n\n\n\n\n\n\n");  break;  }  else  {  turnto(a + 2, b);  move\_LR(a, b);  turnto(a + 2, b);  a = a + 2;  Player.y = Player.y + 1;  }  }  }  } |

2. 玩家操作 – 在递归分割生成的迷宫中的操作

|  |
| --- |
| void Game\_1\_Control\_S1(int a, int b) //人工操作\_2  {  char c;  Player.x = b; //这个是纵坐标  Player.y = a - 1; //这个才是横坐标  while ((c = \_getch()) != 'p') //字符的移动  {  if (c == (int)'w')  {  if (Dot[Player.x - 1][Player.y] == WALL)  {  turnto(a, b - 1);  Color(4);  printf("口");  turnto(a, b - 1);  Sleep(300);  Color(7);  printf("口");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x - 1][Player.y] == START)  {  turnto(a, b - 1);  Color(3);  printf("入");  turnto(a, b - 1);  Sleep(400);  Color(7);  printf("入");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x - 1][Player.y] == END)  {  for (int i = 0; i <= 15; i++)  {  turnto(a, b - 1);  Color(i);  printf("出");  Sleep(100);  Music\_win(i);  }  system("cls");  printf("\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t哈哈哈哈恭喜你通关！\n\n\n\n\n\n\n\n");  break;  }  else  {  turnto(a, b - 1);  move\_UD(a, b);  turnto(a, b - 1);  b = b - 1;  Player.x = Player.x - 1;  }  }  if (c == (int)'a')  {  if (Dot[Player.x][Player.y - 1] == WALL)  {  turnto(a - 2, b);  Color(4);  printf("口");  turnto(a - 2, b);  Sleep(300);  Color(7);  printf("口");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x][Player.y - 1] == START)  {  turnto(a - 2, b);  Color(3);  printf("入");  turnto(a - 2, b);  Sleep(400);  Color(7);  printf("入");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x][Player.y - 1] == END)  {  for (int i = 0; i <= 15; i++)  {  turnto(a - 2, b);  Color(i);  printf("出");  Sleep(100);  Music\_win(i);  }  system("cls");  printf("\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t哈哈哈哈恭喜你通关！\n\n\n\n\n\n\n\n");  break;  }  else  {  turnto(a - 2, b);  move\_LR(a, b);  turnto(a - 2, b);  a = a - 2;  Player.y = Player.y - 1;  }  }  if (c == (int)'s')  {  if (Dot[Player.x + 1][Player.y] == WALL)  {  turnto(a, b + 1);  Color(4);  printf("口");  turnto(a, b + 1);  Sleep(300);  Color(7);  printf("口");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x + 1][Player.y] == START)  {  turnto(a, b + 1);  Color(3);  printf("入");  turnto(a, b + 1);  Sleep(400);  Color(7);  printf("入");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x + 1][Player.y] == END)  {  for (int i = 0; i <= 15; i++)  {  turnto(a, b + 1);  Color(i);  printf("出");  Sleep(100);  Music\_win(i);  }  system("cls");  printf("\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t哈哈哈哈恭喜你通关！\n\n\n\n\n\n\n\n");  break;  }  else  {  turnto(a, b + 1);  move\_UD(a, b);  turnto(a, b + 1);  b = b + 1;  Player.x = Player.x + 1;  }  }  if (c == (int)'d')  {  if (Dot[Player.x][Player.y + 1] == WALL)  {  turnto(a + 2, b);  Color(4);  printf("口");  turnto(a + 2, b);  Sleep(300);  Color(7);  printf("口");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x][Player.y + 1] == START)  {  turnto(a + 2, b);  Color(3);  printf("入");  turnto(a + 2, b);  Sleep(400);  Color(7);  printf("入");  turnto(a, b);  printf("\a");  }  else if (Dot[Player.x][Player.y + 1] == END)  {  for (int i = 0; i <= 15; i++)  {  turnto(a + 2, b);  Color(i);  printf("出");  Sleep(100);  Music\_win(i);  }  system("cls");  printf("\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t哈哈哈哈恭喜你通关！\n\n\n\n\n\n\n\n");  break;  }  else  {  turnto(a + 2, b);  move\_LR(a, b);  turnto(a + 2, b);  a = a + 2;  Player.y = Player.y + 1;  }  }  }  } |

3. 胜利音效

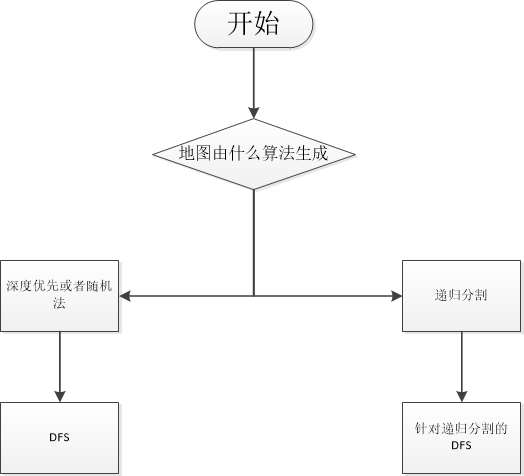
|  |
| --- |
| void Music\_win(int n) //胜利音效  {  switch (n)  {  case 0:  Beep(494, 100);  break;  case 1:  Beep(523, 100);  break;  case 2:  Beep(578, 100);  break;  case 3:  Beep(659, 100);  break;  case 4:  Beep(698, 100);  break;  case 5:  Beep(784, 100);  break;  case 6:  Beep(880, 100);  break;  case 7:  Beep(932, 100);  break;  case 8:  Beep(1046, 100);  break;  case 9:  Beep(1175, 100);  break;  case 10:  Beep(1381, 100);  break;  case 11:  Beep(1480, 100);  break;  case 12:  Beep(1661, 100);  break;  default:  break;  }  } |

## 4.4自动寻路模块设计与实现

### 4.4.1功能说明

这方面是为了实现在迷宫中生成有颜色的路径的功能

### 4.4.2实现流程

自动寻路模块流程图如4 – 4所示

4 – 4

### 4.4.3关键源代码

1. DFS

|  |
| --- |
| void DFS\_main() //深度优先搜索——主体  {  int x = 2;  int y = 1;  DFS\_Make(x, y);  if (u == 0)  {    Color(4);  printf("\a\n\n\n\n\t\t\t\t\t\t 抱歉哇，没得出口嗷！\n\n\n\n\n\n\n\n");  Color(7);  Sleep(300);  }  }  void DFS\_Make(int x, int y) //深度优先搜索——实现  {  if (v == 1)  return;  if (Dot[x][y] == End)  {  system("cls");  for (int i = 0; i < L; i++)  {  for (int j = 0; j < L; j++)  if (Dot[i][j] == Wall)  printf("口");  else if (Dot[i][j] == Ans)  {  Color(4);  printf("国");  Color(7);  }  else if (i == 2 && j == 0)  {  Color(3);  printf("入");  Color(7);  }  else if (i == L - 3 && j == L - 1)  {  Color(2);  printf("出");  Color(7);  }  else  printf(" ");  printf("\n");  }  u = 1;  v = 1;  return;  }    for (int i = 0; i < 4; i++)  {  int next\_x = x + Dst[i][0];  int next\_y = y + Dst[i][1];  if (next\_x >= 0 && next\_x <= L - 1 && next\_y >= 0 && next\_y <= L - 1 && (Dot[next\_x][next\_y] != Wall) && (Dot[next\_x][next\_y] != Ans) && (Dot[next\_x][next\_y] != Start))  {  Dot[x][y] = Ans;  DFS\_Make(next\_x, next\_y);  Dot[x][y] = Road;  }  }  return;  } |

2. 特判递归分割的DFS

|  |
| --- |
| void DFS\_main\_Instance() //深度优先搜索——主体——特判递归分割  {  int x = 2;  int y = 1;  DFS\_Make\_Instance(x, y);  }  void DFS\_Make\_Instance(int x, int y) //深度优先搜索——实现——特判递归分割  {  if (v == 1)  return;  if (Dot[x][y] == END)  {  system("cls");  for (int i = 0; i < L; i++)  {  for (int j = 0; j < L; j++)  if (Dot[i][j] == WALL)  printf("口");  else if (Dot[i][j] == ANS)  {  Color(4);  printf("国");  Color(7);  }  else if (i == 2 && j == 0)  {  Color(3);  printf("入");  Color(7);  }  else if (Dot[i][j] == END)  {  Color(2);  printf("出");  Color(7);  }  else  printf(" ");  printf("\n");  }  u = 1;  v = 1;  return;  }  for (int i = 0; i < 4; i++)  {  int next\_x = x + Dst[i][0];  int next\_y = y + Dst[i][1];  if (next\_x >= 0 && next\_x <= L - 1 && next\_y >= 0 && next\_y <= L - 1 && (Dot[next\_x][next\_y] != WALL) && (Dot[next\_x][next\_y] != ANS) && (Dot[next\_x][next\_y] != START))  {  Dot[x][y] = ANS;  DFS\_Make\_Instance(next\_x, next\_y);  Dot[x][y] = ROUTE;  }  }  return;  } |

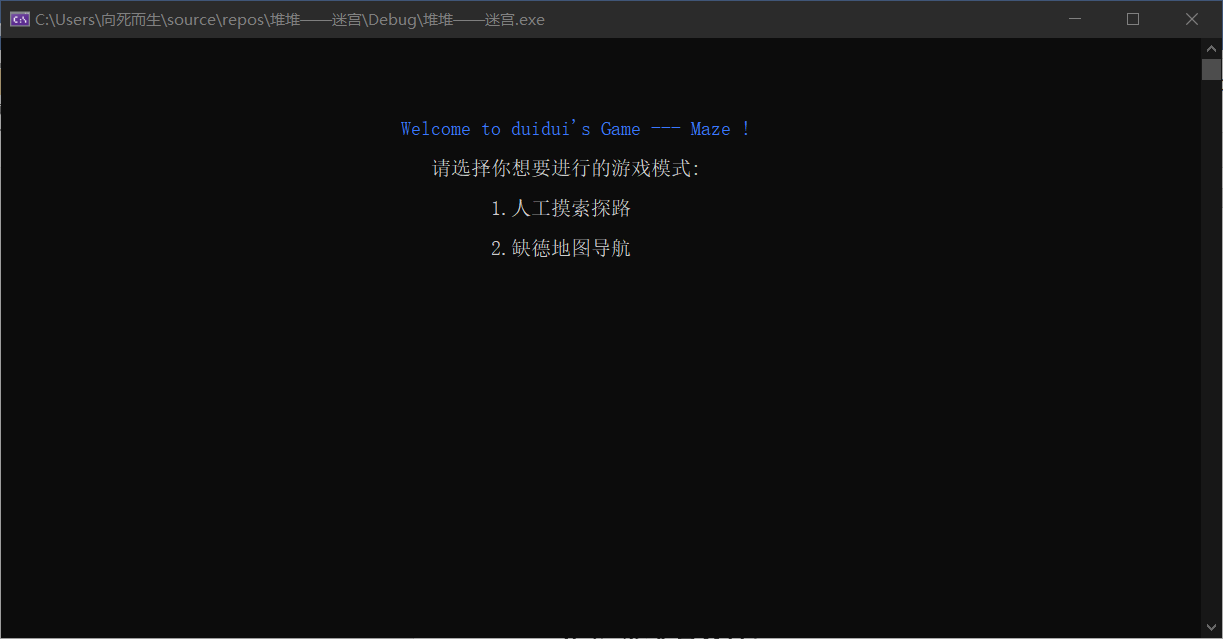
# 系统测试

## 软件测试

### 单元测试

（1）主界面模块测试：

1.查看界面



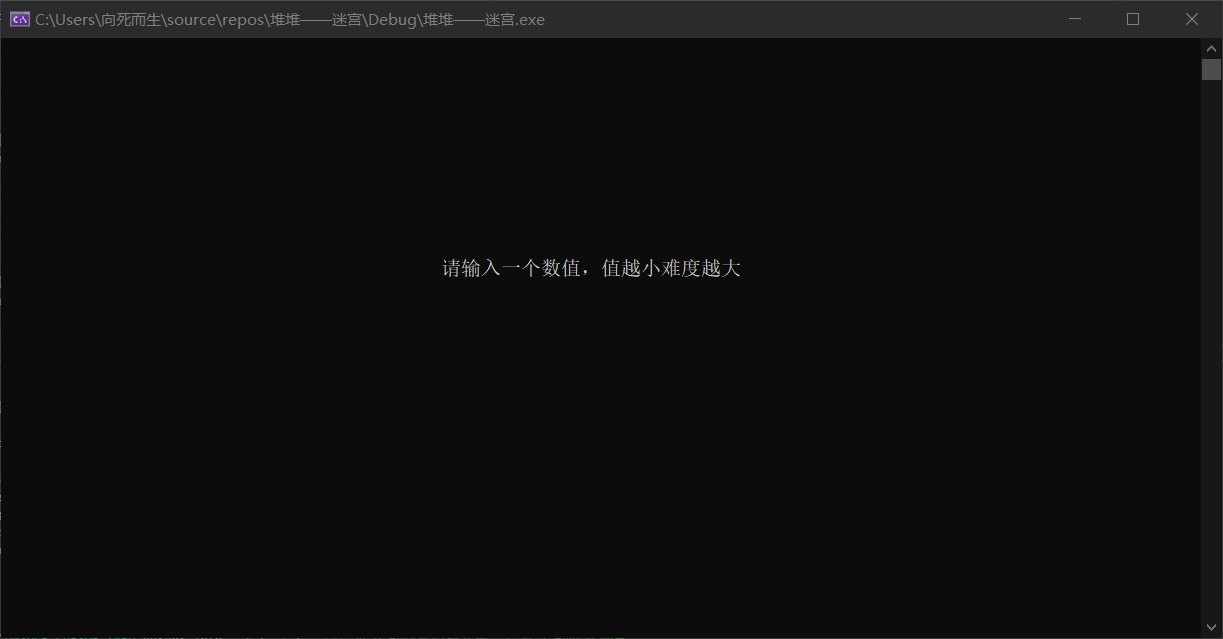
5-1

2.输入错误指令

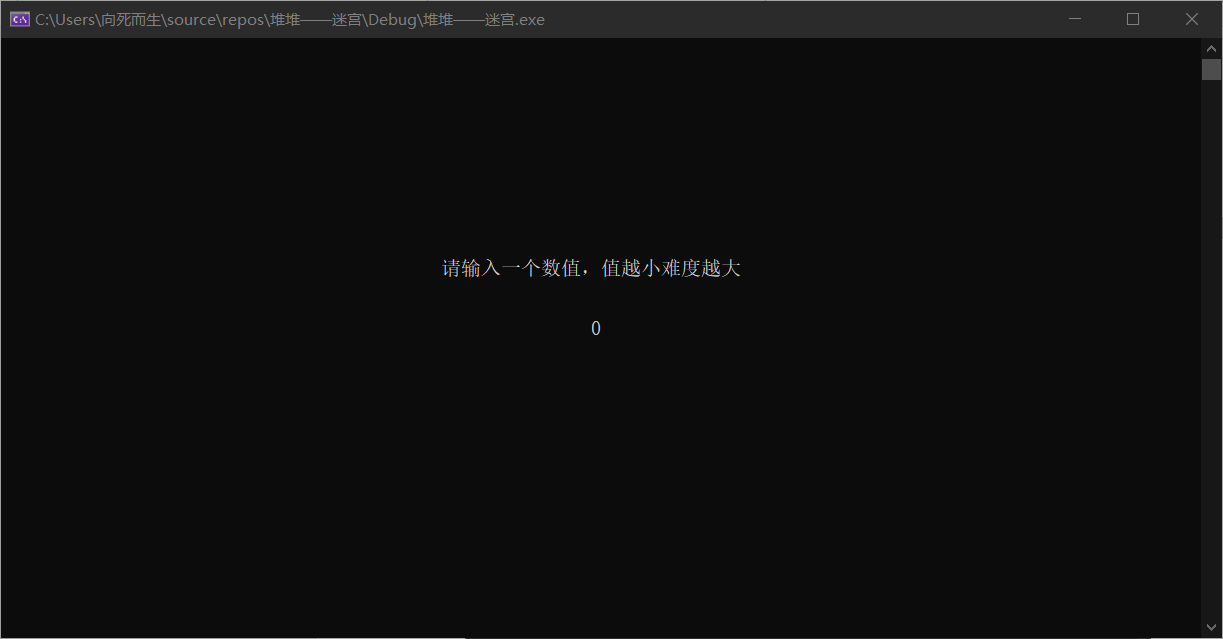


5-2

3.输入指令正确，人机交互界面



5-3



5-4

4.开始游戏界面



5-5

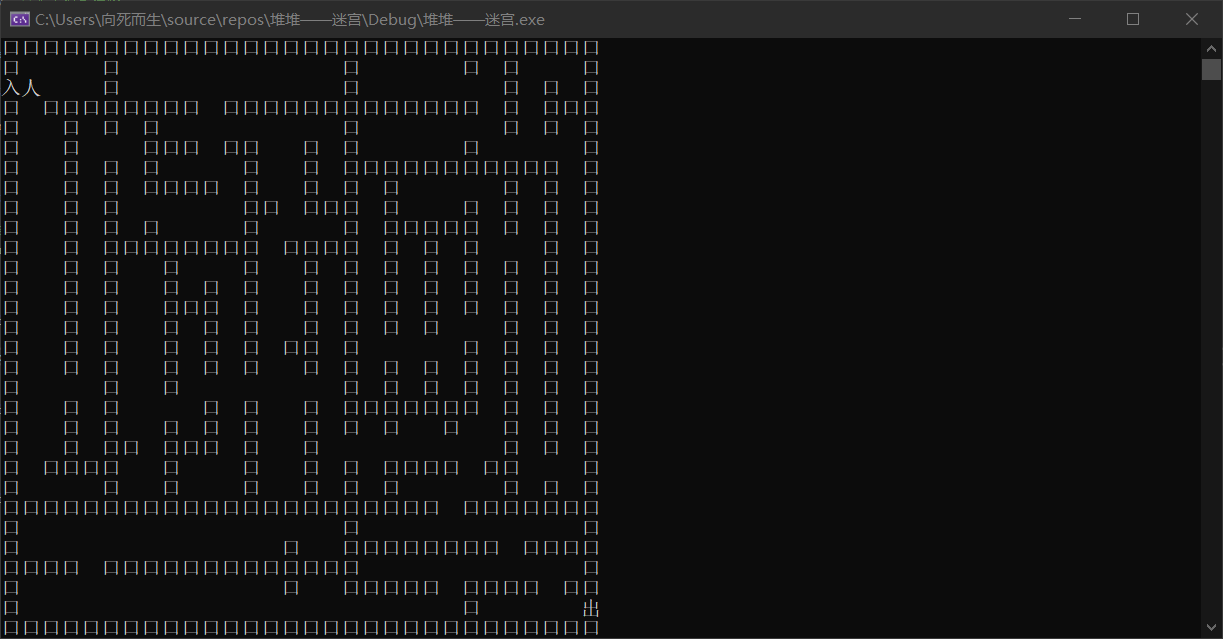
（2）创建地图模块测试：

1.随机法



5-6

2.递归分割



5-7

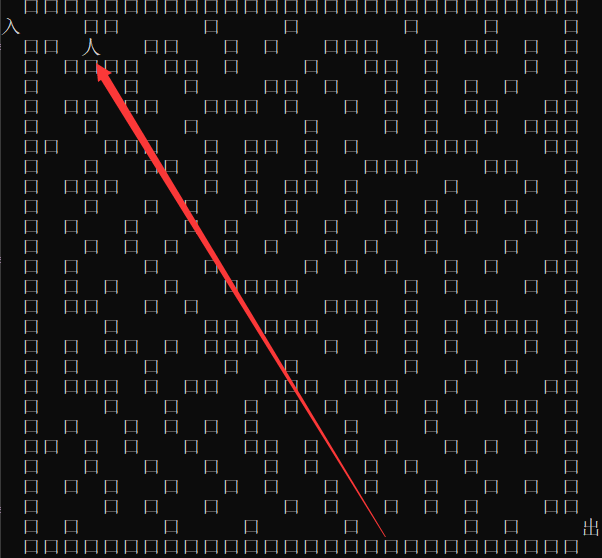
3.深度优先



5-8

（3）玩家操作模块测试：

1.正常行走



5-9 5-10

2.碰墙



5-11

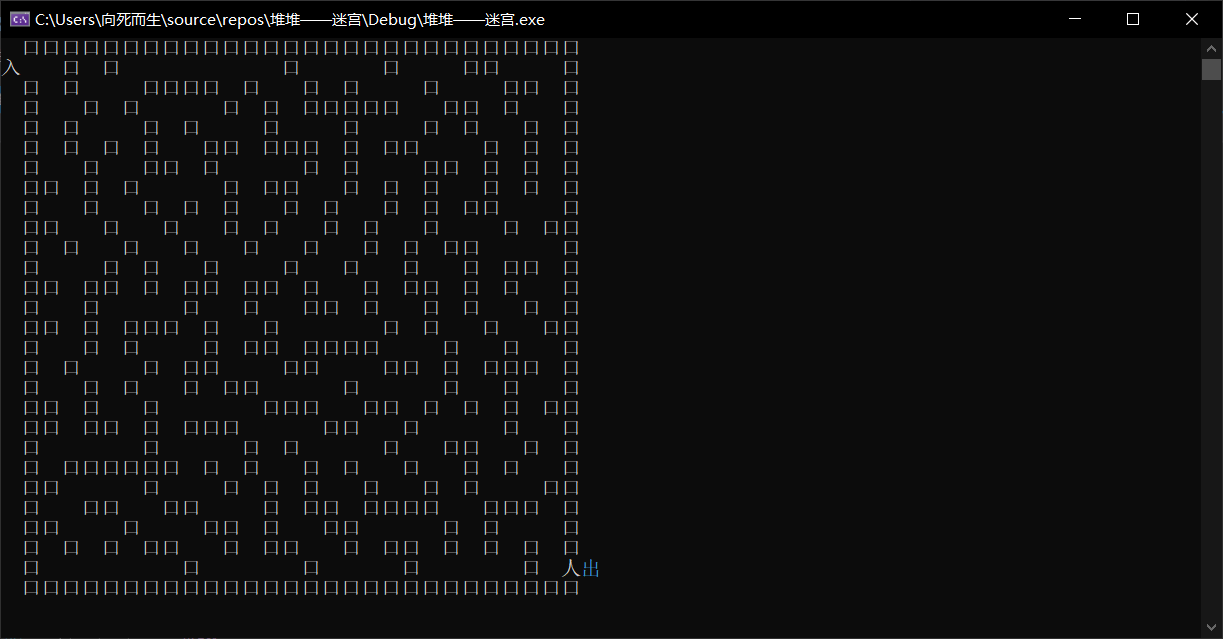
3.碰入口



5-12

4.到达出口

由于一共有十五种颜色变换，不便一一截图，此处截一张图作为测试用例



5-13

5.通关界面



5-14

（4）自动寻路模块测试

1.进入界面



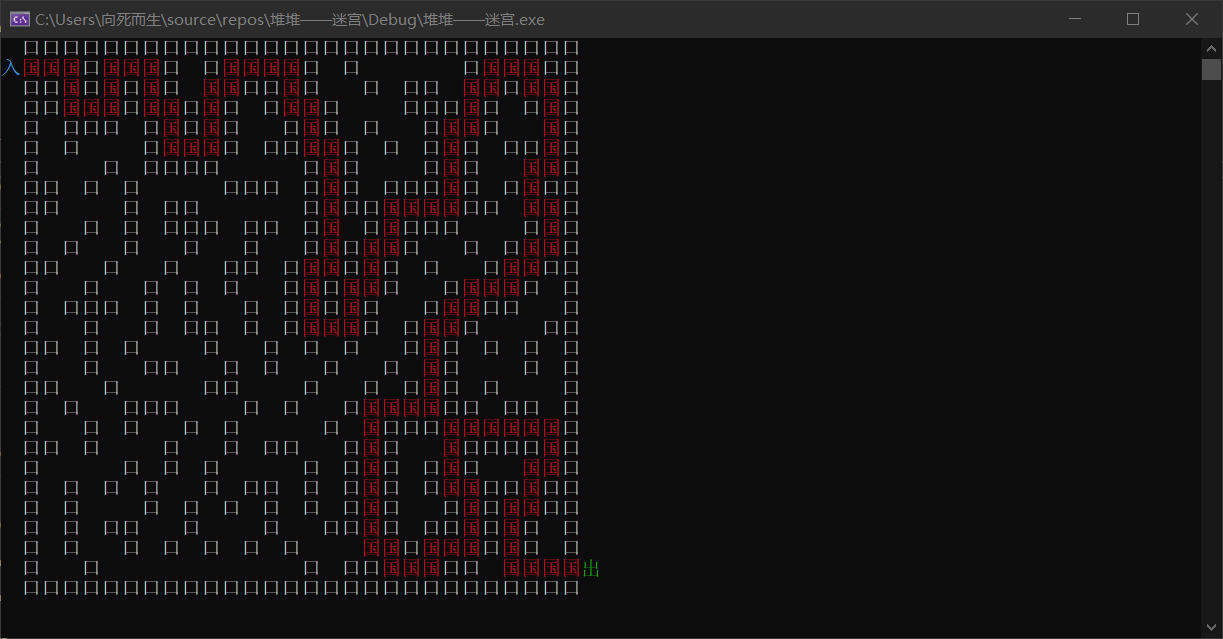
5-15

2.给玩家查看地图



5-16

3.将路径标出



5-17

### 集成测试

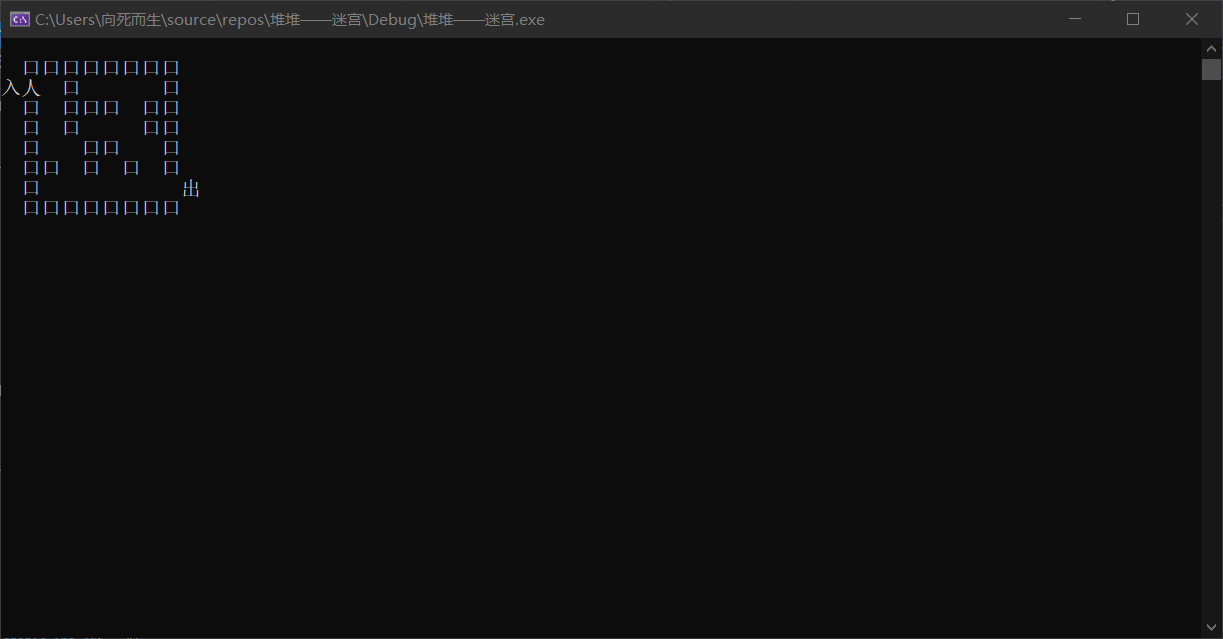
（1）测试说明：

将三种算法生成的迷宫进行测试，并调整长宽进行测试，改变难度

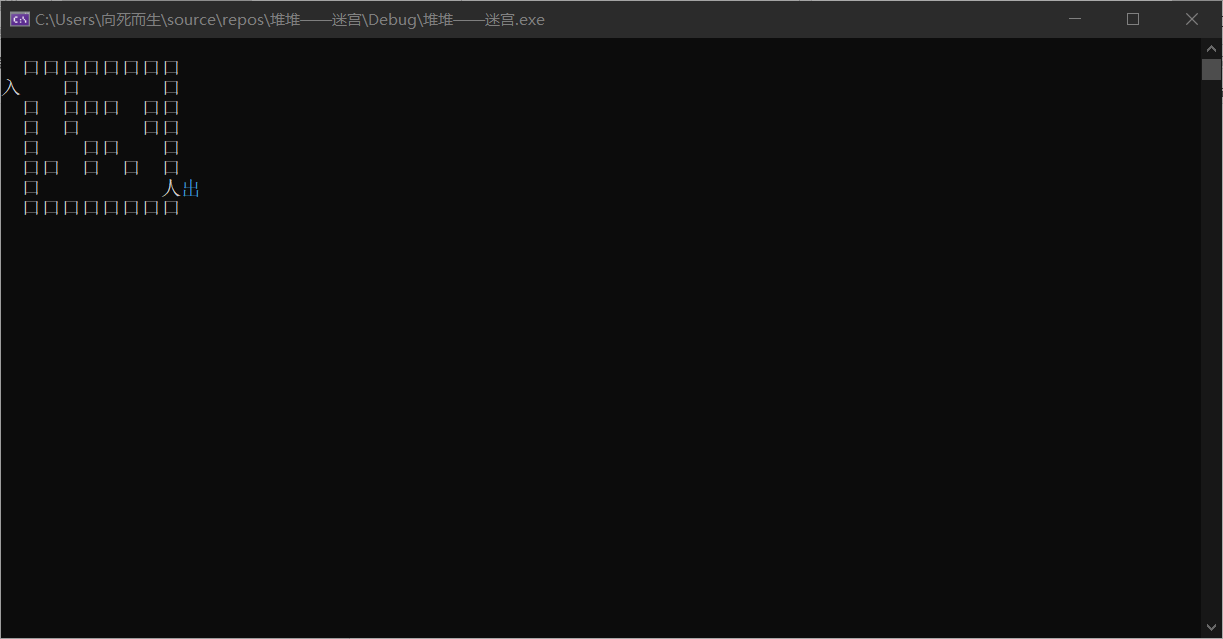
使用人工操作和自动寻路多次测试，看是否存在致命BUG并进行相应调整

（2）测试用例：

1.长宽：10，难度：3，人工操作——>（地图生成合理，成功走出）



5-18



5-19

2.长宽：25，难度：1，人工操作——>（地图生成合理，成功走出）



5-20



5-21

长宽：50，难度：0，人工操作——>（地图生成合理，成功走出)



5-22



5-23

长宽：40，难度：0，自动寻路——>（地图生成合理，路径生成合理）



5-24

（3）程序运行情况：

程序运行非常好，没有出现像错位一类的BUG，但是在按下“P”键的时候没有相应的人机对话，而是直接判断胜利。在自动寻路下，不可以重新开始游戏。

（4）错误修正方案：

在按下“P”之后多进行一个判断，添加人机对话界面，自动寻路每进行完一次迷宫寻路，将弹出人机对话界面，询问是否重新开始。

## 系统有效性

该游戏基本能满足游戏需求，界面简洁、不杂乱，加入了额外的音效以及自动寻路的功能，目前没有其余的BUG发现，基本每个方面的有效性都满足实际需求。

# 总结

本系统基本实现了迷宫基础功能，并且添加了自动寻路的功能，使游戏的可玩性得到了一定的提升，但是在算法和图形化的方面还有优化空间。

后续打算将自动寻路与玩家移动整合为一起，即玩家在玩游戏的时候可以按下某一个键，然后系统将会自动检索路径，然后玩家由电脑操纵移动，玩家可以随时退出电脑操纵模式，或者玩家可以按下某一个键，然后使地图通路显示出来，玩家可以随时看地图指示，这些在我的代码之中已经有了体现，只是可惜时间不太足，没能成功添加进来，还有一个很好的趣味性设计，可以按下一个键，然后全屏变黑，只有入口和出口显示出来，然后玩家在移动的时候会闪一下，让玩家知道自己的位置，然后碰到墙还是会提示，这个设计我已经在另一份代码中将其实现，但是由于有一些小BUG，就未予以其加入本系统的权利。

通过这一次的课程设计使我明白了理论必须和实际相结合，不然空洞洞的理论知识只是空中楼阁，不能为社会服务，只有在实际中融入理论知识，才可以真正去为社会服务，从而提高自己独立思考、自主学习、实际动手的能力。

在课程设计的过程中真的是遇到了重重困难，毕竟是第一次接受这样的考验，同时也在此中发现了自己的不足之处，就比如有些基础知识还是掌握得不够深入，还有有些知识连听都没听说过，就像控制台句柄，还是临时学会。在这次课程设计之后，还是好好温习了一下以前学的基础知识。

然后在程序的编写过程中还是遇到了很多小难点，就是各种各样的编译错误，细节问题层出不穷，由于自己采用的是逐层编写下去的办法，所以最后还是没有隐藏的大BUG问题，因为我在编写过程中就已经排除了那些大的问题所在。

在这次编程的过程中收获最大的恐怕就是在思考能力这一方面了，我学了很多很多算法，创造迷宫的三大算法：Prime算法、递归分割算法、深度优先算法，甚至我自己想到了一种非常独特的——随机数算法，并且在一次次改进之中，我的随机数算法竟和三大算法有得一拼，然后还有就是学了迷宫求解的三大算法：DFS（深度优先搜索）、BFS（广度优先搜索）、A\*（A - star），还有一些小技巧，比如利用随机数来产生概率，还学了蛮多数据结构的相关知识，就像队列和栈，还有动态二维数组的分配与使用.

这些东西都让我受益匪浅，虽然学的过程很累很累，想的过程特别辛苦，但是在学会了之后，那种茅塞顿开的感觉，投入实际运用中之后的那种欣喜和成就感真的非常爽，在这次课设之后，我对编程世界更加感兴趣了。以后一定会更加努力，多多练习，尽力写出更加好的代码。

# 参考文献

1. 何钦铭.C语言程序设计 (第3版) [M].高等教育出版社，2015.08
2. 谭浩强.C程序设计(第5版) [M].清华大学出版社，2017.08
3. ]颜晖,张泳.C语言程序设计实验与习题指导(第3版) [M].高等教育出版社，2015.08
4. 刘强，童启.C语言程序设计实验教程[M].电子工业出版社，2018.02
5. 刘博.C语言课程设计[M].机械工业出版社，2013.04
6. 明日科技. C语言函数参考手册[M].清华大学出版社，2012.02
7. 韩先峰.脑动力：C语言函数速查效率手册[M]. 电子工业出版社，2012.07