

《程序设计基础》课程设计报告

课题名称： **推箱子**

组长：  **蔡雯**

课题组员： **吴佳柳、雷志锋、熊海**

专业：  **软件工程**

班级：  **2019级 软件工程 三、四班**

院系： **信息与工程学院**

指导教师：  **彭琛老师**

时间： 2020.6.25

========== 目录 ==========

[一、 题目概述 1](#_Toc44196025)

[1.0游戏介绍 1](#_Toc44196026)

[1.1游戏玩法 1](#_Toc44196027)

[1.2多平台 1](#_Toc44196028)

[二、 需求分析 2](#_Toc44196029)

[三、 总体设计 3](#_Toc44196030)

[3.0菜单模块 3](#_Toc44196031)

[3.1界面模块 4](#_Toc44196032)

[3.2操作模块 5](#_Toc44196033)

[3.3备份模块 6](#_Toc44196034)

[四、 详细设计 7](#_Toc44196035)

[4.0地图绘制 7](#_Toc44196036)

[4.1移动操作 9](#_Toc44196037)

[4.2备份操作 10](#_Toc44196038)

[4.3地图切换与游戏结束判断 12](#_Toc44196039)

[五、 编程实现 15](#_Toc44196040)

[5.0 mian.cpp 15](#_Toc44196041)

[5.1 base.h 16](#_Toc44196042)

[5.2 map.h 16](#_Toc44196043)

[5.3 opration.h 19](#_Toc44196044)

[5.4 backup.h 26](#_Toc44196045)

[六、 调试与测试 30](#_Toc44196046)

[开始界面 30](#_Toc44196047)

[游戏界面 30](#_Toc44196048)

[结束界面 32](#_Toc44196049)

[七、 结论与心得 33](#_Toc44196050)

[八、 致谢 34](#_Toc44196051)

[附录 34](#_Toc44196052)

1. 题目概述

# 1.0 游戏介绍

经典的推箱子是一个来自日本的古老游戏，1981年由日本人今林宏行首创，1982年12月由Thinking Rabbit 公司首次发行，名“[仓库番](https://baike.baidu.com/item/%E4%BB%93%E5%BA%93%E7%95%AA)”。

箱子只可以推, 不可以拉, 而且一次只能推动一个，胜利条件就是把所有的箱子都推到目的地。

目的是在训练你的逻辑思考能力。在一个狭小的仓库中，要求把木箱放到指定的位置，稍不小心就会出现箱子无法移动或者通道被堵住的情况，所以需要巧妙的利用有限的空间和通道，合理安排移动的次序和位置，才能顺利的完成任务。

# 1.1 游戏玩法

1、游戏同时支持鼠标和键盘操作，鼠标操作时，可点击任意你想去的地方。

2、提供回复上一步的功能，可以回到任何一步走过的路径，可以给用户足够的思考空间，使用户的智力在游戏的乐趣中得到充分的锻炼。也可以一退到底。

3、屏幕上方显示当前的关数，您可以方便地选关进行游戏。

# 1.2 多平台

作为一个风靡全球的游戏作品，推箱子除了传统的PC端游戏之外，也开发出了手机、Web、小程序等多种平台。

以推出高品质Symbian游戏闻名的[唐图](https://baike.baidu.com/item/%E5%94%90%E5%9B%BE" \t "_blank)科技，这次是将经典的《推箱子》游戏搬上了手机。在这款已经被多次翻新的经典益智游戏中，唐图科技的开发人员仍然没有让我们失望。独特的创意，精美的画面，丰富的效果，以及方便的操作，处处体现出唐图科技对于玩家的了解和独到的设计。

这款基于X-Factory[游戏引擎](https://baike.baidu.com/item/%E6%B8%B8%E6%88%8F%E5%BC%95%E6%93%8E)开发的《推箱子》游戏，首先将玩家熟悉的主角换成了一艘[宇宙飞船](https://baike.baidu.com/item/%E5%AE%87%E5%AE%99%E9%A3%9E%E8%88%B9)，游戏的场景也相应地移到了浩瀚的宇宙空间中，而被推动的箱子则变成了不断闪烁的能量块。游戏场景层次非常丰富，令人眼前一亮，可以说是一个完全不一样的[推箱子游戏](https://baike.baidu.com/item/%E6%8E%A8%E7%AE%B1%E5%AD%90%E6%B8%B8%E6%88%8F)。玩家操纵的飞船，在移动时会从尾部喷出动态的火焰，飞船的飞行动作平滑流畅，操作非常爽快。当能量块被推到正确的位置时，会不断地发出绿光，提示已经完成一部分。这些丰富的效果，都依赖于X-Factory的强大功能和唐图科技开发人员的精彩创意。

1. 需求分析

在传统游戏中，一般最最基础的移动操作与其配套的推箱子逻辑，这里我们增加上一步与重新开始的操作。这样我们可以来详细的总结下我们可能需要的游戏主体和功能

项目

│

├── 游戏主体/

│ ├── 边框

│ ├── 地图

│ ├── 人物

│ ├── 墙

│ ├── 箱子

│ └── 结果点

│

└── 功能/

├── 上下左右移动

├── 重新开始

├── 上一步

└── 关卡记录

由此我们可以程序分为以下几个方面：

1. 构建游戏界面的基础元素（如：游戏界面、记分牌、操作区等等）
2. 游戏的地图产生
3. 游戏操作逻辑的处理
4. 撞墙算法（是否移动、是否撞墙、是否推箱子等等行为的判断）
5. 回退算法（上一步、重新开始等操作的记录）

蔡雯20199458：录制视频、地图、界面、菜单程序的部分  
吴佳柳20177774：核心算法、人物的移动逻辑与操作列表逻辑的判断与算法设计  
雷志锋20199456：报告的编写与程序备份重置模块的编写  
熊海20199466：程序可行性分析与代码测试调试审计  
全部函数参与，作审核和逻辑处理，与大家一起讨论

1. 总体设计

根据项目的需求分析，我们来进行进一步的总体设计。

## 3.0 菜单模块

作为一个成熟的游戏，我们应该拥有至少三个部分：开始界面，游戏过程界面，结束界面。并且我们也可以从百度中找到数以千计的页面设计。但是这里我们仅使用最简单字符串打印来构成游戏界面。





## 3.1 界面模块

由于我们需要实时绘制地图和其他信息，来构成游戏的过程界面。我们采用类枚举的方式，来输出不同的字符。

这里我们做如下约定：

0：代表着空白的地方；

1：代表墙；

2：代表人；

3：代表箱子；

4：代表着箱子的终点位置

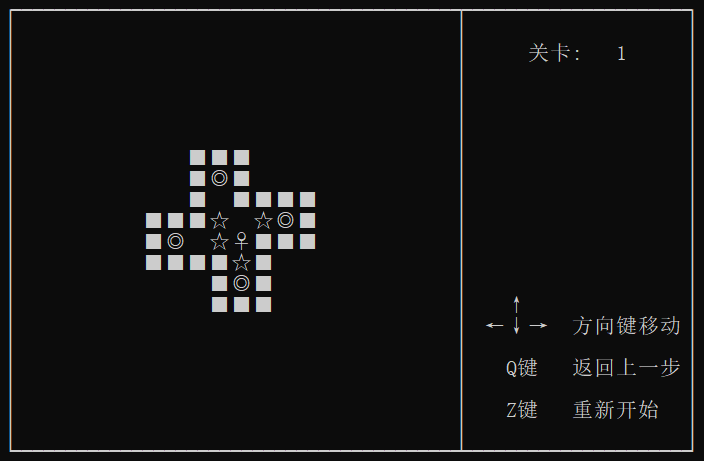
我们还需要绘制界面的信息，比如这里的键位提示信息

我们可以采用联系的 **if–else if–else** 来判断输出的行数根据行数来输出对应的语句

除此之外，我们需要根据游戏的进程，改变关卡的等级，所以我们创建一个全局变量，放置于 **base.h** 来实现整个程序的使用

通过上面所有的方法，就可以来完整的输出整个游戏界面，我们所作的，只不过用约定的枚举来输出地图元素，用行数的判断来输出键位提示信息，用全局变量来实现关卡等级的改变。

如此用一个**for**循环进行输出，只需要实时判断**i**变量的改变，选择需要输出的信息，就可以完美的输出整个界面



## 3.2 操作模块

根据我们之前学习的知识，操作模块其实很好完成，只需要用一个**switch**来判断键入的字符然后来进行操作即可

我们可以来模拟一个列表：

游戏

├── “w、W、[↑](https://baike.baidu.com/item/%E2%86%91" \t "_blank)” 角色向上行走

├── “a、A、[←](https://baike.baidu.com/item/%E2%86%90" \t "_blank)” 角色向左行走

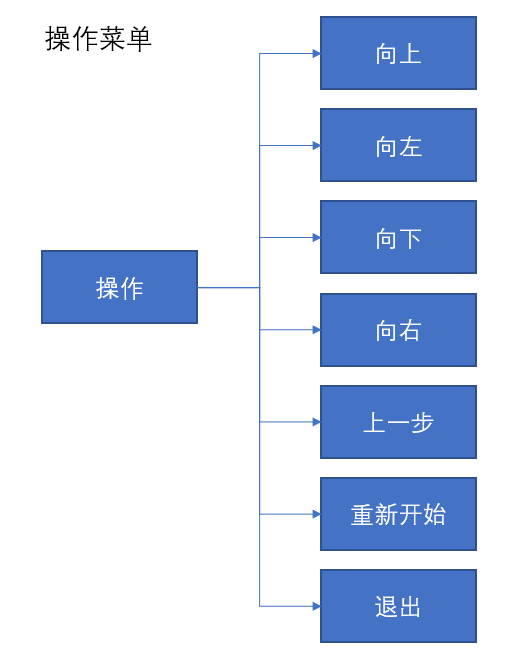
├── “s、S、[↓](https://baike.baidu.com/item/%E2%86%93" \t "_blank)” 角色向下行走

├── “d、D、[→](https://baike.baidu.com/item/%E2%86%92" \t "_blank)” 角色向右行走

├── “q、Q” 返回上一步操作

├── “z、Z” 重新开始操作

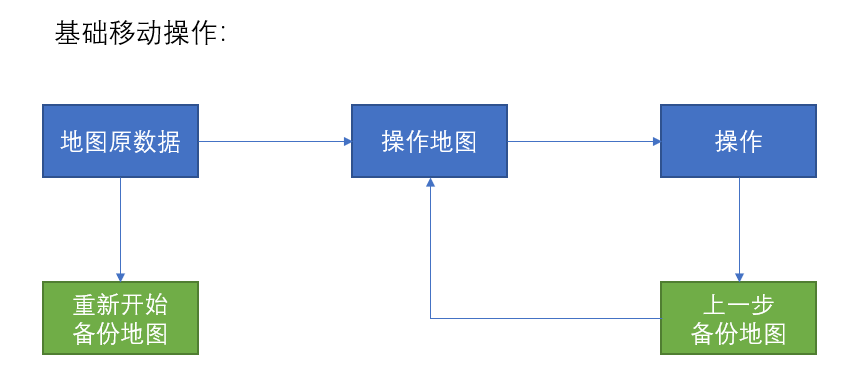
└── “e、E” 退出游戏操作

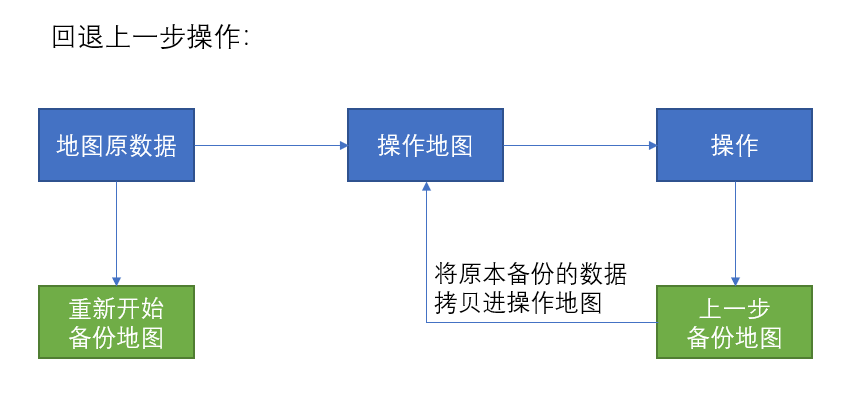


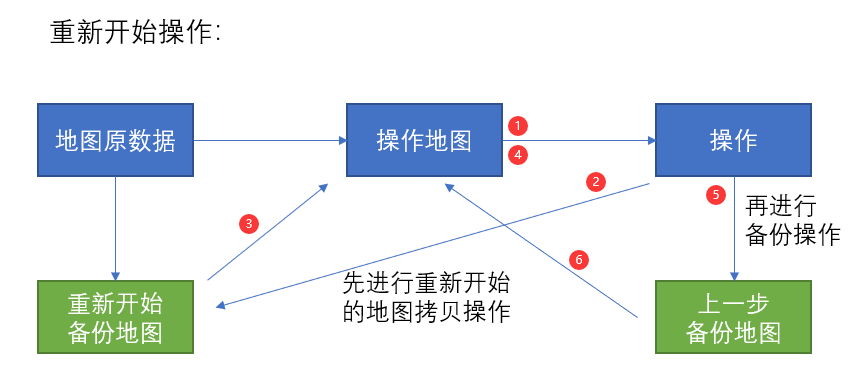
## 3.3 备份模块

为了实现游戏进程的备份，我们将引入一个备份模块，此模块的作用是：记录每次操作前的地图，在每次操作完成后，进行更新，当需要回退上一步操作，或者重新开始之时，从备份地图中拷贝需要的地图数据至进行实时使用的地图库中

这里我们可以使用如下流程图，来详细解释其作用







1. 详细设计

之前我们在总体设计中，比较系统的将游戏分出了四个模块：菜单模块、界面模块、操作模块、备份模块。这里我们进一步就四个模块中所要求的技术，进行更深一步的探讨与论证

## 4.0 地图绘制

根据之前的论证，我们使用二维数组来储存地图的数据信息，用字符约定枚举来进行地图字符输出，并且用其来打印出地图。

随后我们使用来展示游戏的一些信息，这里我们需要来判断行数来决定每一行我们需要来输出什么信息。

如下是地图数据、绘制函数和信息函数：

1. **int** map[1][20][20] = {
2. {
3. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
4. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
5. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
6. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
7. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
8. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
9. {0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0},
10. {0,0,0,0,0,0,0,0,1,4,1,0,0,0,0,0,0,0,0,0},
11. {0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,0,0,0,0},
12. {0,0,0,0,0,0,1,1,1,3,0,3,4,1,0,0,0,0,0,0}, // 10
13. {0,0,0,0,0,0,1,4,0,3,2,1,1,1,0,0,0,0,0,0},
14. {0,0,0,0,0,0,1,1,1,1,3,1,0,0,0,0,0,0,0,0},
15. {0,0,0,0,0,0,0,0,0,1,4,1,0,0,0,0,0,0,0,0},
16. {0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0},
17. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
18. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
19. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
20. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
21. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
22. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
23. }
24. }
26. **void** draws()
27. {
28. printf("┌────────────────────────────────────────┬────────────────────┐\n");
29. **for** (**int** i = 0; i < 20; i++)
30. {
31. printf("│");
32. **for** (**int** j = 0; j < 20; j++)
33. {
34. **switch** (map[level][i][j])
35. {
36. **case** 0:
37. printf("  ");       // 空白的地方
38. **break**;
39. **case** 1:
40. printf("■");        // 墙
41. **break**;
42. **case** 2:
43. printf("♀");        // 人
44. **break**;
45. **case** 3:
46. printf("☆");        // 箱子
47. **break**;
48. **case** 4:
49. printf("◎");        // 终点地方
50. **break**;
51. **case** 6:
52. printf("♀");        // 人在终点位置
53. **break**;
54. **case** 7:
55. printf("★");        // 箱子在终点位置
56. **break**;
57. }
58. }
59. data(i);
60. }
61. printf("└────────────────────────────────────────┴────────────────────┘\n");
62. }
64. **void** data(**int** row)
65. {
66. **switch** (row)
67. {
68. **case** 1:
69. printf("│      关卡: %3d     │\n", level + 1);
70. **break**;
71. **case** 4:
72. **if** (flag == 1)
73. printf("│      恭喜通关!     │\n");
74. **else**
75. printf("│                    │\n");
76. **break**;
77. **case** 13:
78. printf("│    ↑              │\n");
79. **break**;
80. **case** 14:
81. printf("│  ←↓→  方向键移动│\n");
82. **break**;
83. **case** 16:
84. printf("│    Q键   返回上一步│\n");
85. **break**;
86. **case** 18:
87. printf("│    Z键   重新开始  │\n");
88. **break**;
89. **default**:
90. printf("│                    │\n");
91. **break**;
92. }
93. }

## 4.1 移动操作

在确立了地图信息之后，我们才可以进行移动操作的编写。

移动操作的本质其实是对于地图约定枚举的判断，通过地图数据中点值的不同，我们以此来判断移动前方的物件，是墙？是箱子？还是空地

我们首先需要确定玩家所在的位置，这里可以用**for**遍历来判断

1. **int** count, caw;                                            // 行和列
2. **for** (**int** i = 0; i < 20; i++)                               // 确定人的位置
3. {
4. **for** (**int** j = 0; j < 20; j++)
5. {
6. **if** (map[level][i][j] == 2 || map[level][i][j] == 6)
7. {
8. count = i;
9. caw = j;
10. }
11. }
12. }

如此会有以下情况

情况

│

├── 前方为空 可走

│

├── 前方为箱子

│ ├── 箱子前方为墙 不可走

│ └── 箱子前方为空 可走

│

└── 前方为墙 不可走

根据前方”路况”的信息，我们将地图的点值进行加减，如此得到地图的转换，从而实现在绘制上看起来像玩家移动一样，但其本质只是地图信息改变而打印的字符变化而已。

这里我们以向上为例，演示核心代码

1. **if** (map[level][count - 1][caw] == 0 || map[level][count - 1][caw] == 4)
2. {
3. map[level][count][caw] -= 2;
4. map[level][count - 1][caw] += 2;
5. backup();
6. }
7. **else** **if** (map[level][count - 1][caw] == 3 || map[level][count - 1][caw] == 7)
8. {
9. **if** (map[level][count - 2][caw] == 0 || map[level][count - 2][caw] == 4)
10. {
11. map[level][count][caw] -= 2;
12. map[level][count - 1][caw] -= 1;
13. map[level][count - 2][caw] += 3;
14. backup();
15. }
16. }

## 4.2 备份操作

备份操作的最终目的是为了记录操作记录，这也是为什么第三点才讲备份操作的原因。

备份操作的本质其实就是讲每一步的地图进行拷贝，这个拷贝需要连续拷贝两次。

这里可以理解成一个是拷贝的缓存区，用于每次操作的数据区域，第二次备份，才是真正的存储数据的地方。

随后当我们需要进行回退操作的时候，只需讲二备区的信息直接覆盖原地图的数据既可。

重新开始操作就更加简单了，它不会出现每次操作就需要备份一次的操作，只需要在最开始载入地图的时候将其备份一次就好了。如果在游戏过程中，需要进行重新开始，则直接将最开始备份的数据，直接覆盖原地图数据就好。比备份操作更加简单。

这里暂时展示部分代码：

1. **int** againmap[20][20] = {
2. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
3. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
4. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
5. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
6. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
7. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
8. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
9. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
10. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
11. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
12. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
13. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
14. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
15. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
16. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
17. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
18. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
19. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
20. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
21. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
22. };
24. **void** backup()
25. {
26. **for** (**int** i = 0; i < 20; i++)
27. {
28. **for** (**int** j = 0; j < 20; j++)
29. {
30. pastmap2[i][j] = pastmap[i][j];
31. }
32. }
33. **for** (**int** i = 0; i < 20; i++)
34. {
35. **for** (**int** j = 0; j < 20; j++)
36. {
37. pastmap[i][j] = map[level][i][j];
38. }
39. }
40. }
42. **void** abackup()
43. {
44. **for** (**int** i = 0; i < 20; i++)
45. {
46. **for** (**int** j = 0; j < 20; j++)
47. {
48. againmap[i][j] = map[level][i][j];
49. }
50. }
51. }
53. **void** getback()
54. {
55. **for** (**int** i = 0; i < 20; i++)
56. {
57. **for** (**int** j = 0; j < 20; j++)
58. {
59. pastmap[i][j] = pastmap2[i][j];
60. map[level][i][j] = pastmap[i][j];
61. }
62. }
63. }
65. **void** again()
66. {
67. **for** (**int** i = 0; i < 20; i++)
68. {
69. **for** (**int** j = 0; j < 20; j++)
70. {
71. map[level][i][j] = againmap[i][j];
72. }
73. }
74. }

## 4.3 地图切换与游戏结束判断

作为一个关卡类的游戏必须要一关一关的提升，地图不断的改变，才有意义。所以说地图随着游戏关卡的增加而改变的判断操作是十分重要的。

并且在最后一关结束后应该是胜利，在我们所给出的操作中，其实是有游戏的退出操作的，所以游戏结束之后的的结束界面，应该与自动退出游戏的结束界面不一致，这样在游戏性也会更加有趣。



那么如何判断是否到了某一关卡地图的结束部分呢，这里有两种方法：

1、记录最开始一共有几个箱子终点，随后在每一次操作中对于箱子终点的使用进行加减操作。当终点个数为0 时即完成本关卡

2、和第一种方法类似，但是不是在每次操作中对于终点数进行加减，而只是对每次操作结束后的地图进行判断是否还有没被占住的终点数，倘若已无，则满足晋级条件，进入下一关

经过具体详细的论证我们最终使用了方法二，因为在不考虑资源使用率的情况下，第二种方法无疑是更加方便和可读性高

于此同时我们将使用**flag**对与结束条件进行判断，倘若我们的关卡是已经是我们所有关卡的数量，则已经完成了最后一关，这样我们便可以宣告，游戏胜利。

这样做的好处，不止是关卡的判断，由于是**flag**全局变量，则在整个程序过程中都可以进行使用，所以我们也可以用来判断之前的我们遗漏的问题：如何区分自动退出游戏和胜利退出游戏。

1. **void** next()
2. {
3. **int** index = 0;
4. **for** (**int** i = 0; i < 20; i++)
5. {
6. **for** (**int** j = 0; j < 20; j++)
7. {
8. **if** (map[level][i][j] == 3)
9. index++;
10. }
11. }
12. **if** (index == 0)
13. {
14. level++;
15. flag = 0;
16. init();
17. **if** (level == max)
18. {
19. level--;
20. flag = 1;
21. }
22. }
23. }
24. 编程实现

## 5.0 mian.cpp

1. #define \_CRT\_SECURE\_NO\_WARNINGS
3. #include <windows.h>
5. #include "base.h"
6. #include "backup.h"
7. #include "map.h"
8. #include "opration.h"
10. **int** main()//主函数
11. {
12. start();
13. init();
14. **while** (actions)
15. {
16. next();
17. draws();
18. **if** (flag != 1)
19. actions = action();
20. **if** (flag == 0)
21. {
22. flag = 2;
23. }
24. **if** (flag == 1)
25. {
26. system("cls");
27. **break**;
28. }
29. system("cls");
30. }
31. **if** (flag == 1)
32. success();
33. **else**
34. end();
35. **return** 0;
36. }

## 5.1 base.h

1. #pragma once
3. **int** level = 0, max = 5, flag = 0, actions = 1;

## 5.2 map.h

1. #pragma once
3. #include "base.h"
5. // 地图
6. // 其中 0 代表着空白的地方；1 代表着墙；2 代表着人；3 代表着箱子；4 代表着箱子的中点位置。
8. **int** map[5][20][20] = {
9. {
10. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
11. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
12. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
13. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
14. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
15. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
16. {0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0},
17. {0,0,0,0,0,0,0,0,1,4,1,0,0,0,0,0,0,0,0,0},
18. {0,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,0,0,0,0},
19. {0,0,0,0,0,0,1,1,1,3,0,3,4,1,0,0,0,0,0,0}, // 10
20. {0,0,0,0,0,0,1,4,0,3,2,1,1,1,0,0,0,0,0,0},
21. {0,0,0,0,0,0,1,1,1,1,3,1,0,0,0,0,0,0,0,0},
22. {0,0,0,0,0,0,0,0,0,1,4,1,0,0,0,0,0,0,0,0},
23. {0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0},
24. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
25. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
26. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
27. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
28. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
29. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
30. },
31. {
32. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
33. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
34. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
35. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
36. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
37. {0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0},
38. {0,0,0,0,0,0,1,2,0,0,1,0,0,0,0,0,0,0,0,0},
39. {0,0,0,0,0,0,1,0,3,3,1,0,1,1,1,0,0,0,0,0},
40. {0,0,0,0,0,0,1,0,3,0,1,0,1,4,1,0,0,0,0,0},
41. {0,0,0,0,0,0,1,1,1,0,1,1,1,4,1,0,0,0,0,0}, // 10
42. {0,0,0,0,0,0,0,1,1,0,0,0,0,4,1,0,0,0,0,0},
43. {0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,0,0,0},
44. {0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,0,0,0,0,0},
45. {0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0},
46. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
47. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
48. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
49. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
50. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
51. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
52. },
53. {
54. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
55. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
56. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
57. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
58. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
59. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
60. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
61. {0,0,0,0,0,1,1,1,1,0,0,1,1,1,1,1,0,0,0,0},
62. {0,0,0,0,1,1,0,0,1,0,0,1,0,0,0,1,0,0,0,0},
63. {0,0,0,0,1,0,3,0,1,1,1,1,3,0,0,1,0,0,0,0}, // 10
64. {0,0,0,0,1,0,0,3,4,4,4,4,0,3,0,1,0,0,0,0},
65. {0,0,0,0,1,1,0,0,0,0,1,0,2,0,1,1,0,0,0,0},
66. {0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0},
67. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
68. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
69. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
70. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
71. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
72. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
73. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
74. },
75. {
76. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
77. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
78. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
79. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
80. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
81. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
82. {0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0},
83. {0,0,0,0,0,0,1,4,4,4,4,4,4,1,0,0,0,0,0,0},
84. {0,0,0,0,0,0,1,0,0,3,0,1,0,1,1,0,0,0,0,0},
85. {0,0,0,0,0,0,1,0,3,0,1,0,3,0,1,0,0,0,0,0}, // 10
86. {0,0,0,0,0,0,1,1,3,0,3,0,3,0,1,0,0,0,0,0},
87. {0,0,0,0,0,0,0,1,0,0,2,0,0,0,1,0,0,0,0,0},
88. {0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0},
89. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
90. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
91. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
92. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
93. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
94. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
95. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
96. },
97. {
98. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
99. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
100. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
101. {0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0},
102. {0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},
103. {0,0,0,1,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0},
104. {0,0,0,1,0,0,0,0,1,1,0,0,0,0,0,1,0,0,0,0},
105. {0,0,0,1,4,4,1,0,0,0,0,3,3,1,0,1,0,0,0,0},
106. {0,0,0,1,4,4,0,0,1,1,0,0,0,3,0,1,1,1,0,0},
107. {0,0,0,1,4,4,1,0,0,1,1,3,1,0,3,0,0,1,0,0}, // 10
108. {0,0,0,1,4,4,0,0,0,1,0,2,3,0,3,0,0,1,0,0},
109. {0,0,0,1,4,4,1,0,0,1,0,3,0,3,0,0,0,1,0,0},
110. {0,0,0,1,0,4,0,0,0,1,0,3,0,3,0,1,1,1,0,0},
111. {0,0,0,1,0,0,1,0,0,1,0,0,0,1,1,1,0,0,0,0},
112. {0,0,0,1,0,0,1,0,0,0,0,1,1,1,0,0,0,0,0,0},
113. {0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0},
114. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
115. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
116. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
117. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
118. }
119. };

## 5.3 opration.h

1. #pragma once
3. #include <windows.h>
4. #include <stdio.h>
5. #include <conio.h>
7. #include "base.h"
8. #include "map.h"
9. #include "backup.h"
11. **void** draws();
12. **void** init();
13. **void** data(**int** row);
14. **int** action();
15. **void** next();
16. **void** start();
17. **void** end();
18. **void** success();
20. **void** draws()
21. {
22. printf("┌────────────────────────────────────────┬────────────────────┐\n");
23. **for** (**int** i = 0; i < 20; i++)
24. {
25. printf("│");
26. **for** (**int** j = 0; j < 20; j++)
27. {
28. **switch** (map[level][i][j])
29. {
30. **case** 0:
31. printf("  ");       // 空白的地方
32. **break**;
33. **case** 1:
34. printf("■");        // 墙
35. **break**;
36. **case** 2:
37. printf("♀");        // 人
38. **break**;
39. **case** 3:
40. printf("☆");        // 箱子
41. **break**;
42. **case** 4:
43. printf("◎");        // 终点地方
44. **break**;
45. **case** 6:
46. printf("♀");        // 人在终点位置
47. **break**;
48. **case** 7:
49. printf("★");        // 箱子在终点位置
50. **break**;
51. }
52. }
53. data(i);
54. }
55. printf("└────────────────────────────────────────┴────────────────────┘\n");
56. }
58. **void** data(**int** row)
59. {
60. **switch** (row)
61. {
62. **case** 1:
63. printf("│      关卡: %3d     │\n", level + 1);
64. **break**;
65. **case** 4:
66. **if** (flag == 1)
67. printf("│      恭喜通关!     │\n");
68. **else**
69. printf("│                    │\n");
70. **break**;
71. **case** 13:
72. printf("│    ↑              │\n");
73. **break**;
74. **case** 14:
75. printf("│  ←↓→  方向键移动│\n");
76. **break**;
77. **case** 16:
78. printf("│    Q键   返回上一步│\n");
79. **break**;
80. **case** 18:
81. printf("│    Z键   重新开始  │\n");
82. **break**;
83. **default**:
84. printf("│                    │\n");
85. **break**;
86. }
87. }
89. **int** action()
90. {
91. **int** count, caw;                                                 // 行和列
92. **for** (**int** i = 0; i < 20; i++)                                    // 确定人的位置
93. {
94. **for** (**int** j = 0; j < 20; j++)
95. {
96. **if** (map[level][i][j] == 2 || map[level][i][j] == 6)
97. {
98. count = i;
99. caw = j;
100. }
101. }
102. }
103. **int** tui = \_getch();
105. **switch** (tui)
106. {
107. // 上
108. **case** 72:
109. **case** 'w':
110. **case** 'W':
111. **if** (map[level][count - 1][caw] == 0 || map[level][count - 1][caw] == 4)
112. {
113. map[level][count][caw] -= 2;
114. map[level][count - 1][caw] += 2;
115. backup();
116. }
117. **else** **if** (map[level][count - 1][caw] == 3 || map[level][count - 1][caw] == 7)
118. {
119. **if** (map[level][count - 2][caw] == 0 || map[level][count - 2][caw] == 4)
120. {
121. map[level][count][caw] -= 2;
122. map[level][count - 1][caw] -= 1;
123. map[level][count - 2][caw] += 3;
124. backup();
125. }
126. }
127. **return** 1;
128. **break**;
130. // 下
131. **case** 80:
132. **case** 's':
133. **case** 'S':
134. **if** (map[level][count + 1][caw] == 0 || map[level][count + 1][caw] == 4)
135. {
136. map[level][count][caw] -= 2;
137. map[level][count + 1][caw] += 2;
138. backup();
139. }
141. **else** **if** (map[level][count + 2][caw] == 0 || map[level][count + 2][caw] == 4)
142. {
143. **if** (map[level][count + 1][caw] == 3 || map[level][count + 1][caw] == 7)
144. {
145. map[level][count][caw] -= 2;
146. map[level][count + 1][caw] -= 1;
147. map[level][count + 2][caw] += 3;
148. backup();
149. }
150. }
151. **return** 1;
152. **break**;
154. // 左
155. **case** 75:
156. **case** 'a':
157. **case** 'A':
158. **if** (map[level][count][caw - 1] == 0 || map[level][count][caw - 1] == 4)
159. {
160. map[level][count][caw] -= 2;
161. map[level][count][caw - 1] += 2;
162. backup();
163. }
165. **else** **if** (map[level][count][caw - 2] == 0 || map[level][count][caw - 2] == 4)
166. {
167. **if** (map[level][count][caw - 1] == 3 || map[level][count][caw - 1] == 7)
168. {
169. map[level][count][caw] -= 2;
170. map[level][count][caw - 1] -= 1;
171. map[level][count][caw - 2] += 3;
172. backup();
173. }
174. }
175. **return** 1;
176. **break**;
178. // 右
179. **case** 77:
180. **case** 'd':
181. **case** 'D':
182. **if** (map[level][count][caw + 1] == 0 || map[level][count][caw + 1] == 4)
183. {
184. map[level][count][caw] -= 2;
185. map[level][count][caw + 1] += 2;
186. backup();
187. }
189. **else** **if** (map[level][count][caw + 2] == 0 || map[level][count][caw + 2] == 4)
190. {
191. **if** (map[level][count][caw + 1] == 3 || map[level][count][caw + 1] == 7)
192. {
193. map[level][count][caw] -= 2;
194. map[level][count][caw + 1] -= 1;
195. map[level][count][caw + 2] += 3;
196. backup();
197. }
198. }
199. **return** 1;
200. **break**;
202. **case** 'q':
203. **case** 'Q':
204. **if** (flag != 0)
205. {
206. getback();
207. }
208. **return** 1;
209. **break**;
211. **case** 'z':
212. **case** 'Z':
213. again();
214. **return** 1;
215. **break**;
217. **case** 'e':
218. **case** 'E':
219. **return** 0;
220. **break**;
221. }
222. }
224. **void** next()
225. {
226. **int** index = 0;
227. **for** (**int** i = 0; i < 20; i++)
228. {
229. **for** (**int** j = 0; j < 20; j++)
230. {
231. **if** (map[level][i][j] == 3)
232. index++;
233. }
234. }
235. **if** (index == 0)
236. {
237. level++;
238. flag = 0;
239. init();
240. **if** (level == max)
241. {
242. level--;
243. flag = 1;
244. }
245. }
246. }
248. **void** init()
249. {
250. abackup();
251. backup();
252. backup();
253. }
255. **void** start()
256. {
257. printf("┌─────────────────────────────────────────────────────────────┐\n");
258. printf("│                                                             │\n");
259. printf("│                                                             │\n");
260. printf("│                     \_\_        \_\_\_.                          │\n");
261. printf("│         \_\_\_\_\_\_\_\_\_\_ |  | \_\_\_\_\_\_\\\_ |\_\_ \_\_\_\_\_    \_\_\_\_          │\n");
262. printf("│        /  \_\_\_/  \_ \\|  |/ /  \_ \\| \_\_ \\\\\_\_  \\  /    \\         │\n");
263. printf("│        \\\_\_\_ (  <\_> )    <  <\_> ) \\\_\\ \\/ \_\_ \\|   |  \\        │\n");
264. printf("│       /\_\_\_\_  >\_\_\_\_/|\_\_|\_ \\\_\_\_\_/|\_\_\_  (\_\_\_\_  /\_\_\_|  /        │\n");
265. printf("│            \\/           \\/         \\/     \\/     \\/         │\n");
266. printf("│                                                             │\n");
267. printf("│                                                             │\n");
268. printf("│                       推箱子小游戏                          │\n");
269. printf("│                                                             │\n");
270. printf("│                                                             │\n");
271. printf("│                                                             │\n");
272. printf("│                                                             │\n");
273. printf("│                  >   任意键开始游戏   <                     │\n");
274. printf("│                                                             │\n");
275. printf("│                                                             │\n");
276. printf("│                                                             │\n");
277. printf("│                                                             │\n");
278. printf("└─────────────────────────────────────────────────────────────┘\n");
279. \_getch();
280. system("cls");
281. }
283. **void** end()
284. {
285. printf("┌─────────────────────────────────────────────────────────────┐\n");
286. printf("│                                                             │\n");
287. printf("│                                                             │\n");
288. printf("│                                    .\_\_\_                     │\n");
289. printf("│                   \_\_\_\_   \_\_\_\_    \_\_| \_/                     │\n");
290. printf("│                 \_/ \_\_ \\ /    \\  / \_\_ |                      │\n");
291. printf("│                 \\  \_\_\_/|   |  \\/ /\_/ |                      │\n");
292. printf("│                  \\\_\_\_  >\_\_\_|  /\\\_\_\_\_ |                      │\n");
293. printf("│                      \\/     \\/      \\/                      │\n");
294. printf("│                                                             │\n");
295. printf("│                                                             │\n");
296. printf("│                       欢迎下次游玩                          │\n");
297. printf("│                                                             │\n");
298. printf("│                                                             │\n");
299. printf("│                                                             │\n");
300. printf("│                                                             │\n");
301. printf("│                  >   任意键退出游戏   <                     │\n");
302. printf("│                                                             │\n");
303. printf("│                                                             │\n");
304. printf("│                                                             │\n");
305. printf("│                                                             │\n");
306. printf("└─────────────────────────────────────────────────────────────┘\n");
307. system("pause");
308. }
310. **void** success()
311. {
312. printf("┌─────────────────────────────────────────────────────────────┐\n");
313. printf("│                                                             │\n");
314. printf("│                                                             │\n");
315. printf("│                          \_            \_                     │\n");
316. printf("│                         (\_)          | |                    │\n");
317. printf("│                    \_ \_\_  \_  \_\_\_ \_\_\_  | |                    │\n");
318. printf("│                   | '\_ \\| |/ \_\_/ \_ \\ | |                    │\n");
319. printf("│                   | | | | | (\_|  \_\_/ |\_|                    │\n");
320. printf("│                   |\_| |\_|\_|\\\_\_\_\\\_\_\_| (\_)                    │\n");
321. printf("│                                                             │\n");
322. printf("│                                                             │\n");
323. printf("│                         游戏通关                            │\n");
324. printf("│                                                             │\n");
325. printf("│                                                             │\n");
326. printf("│                                                             │\n");
327. printf("│                                                             │\n");
328. printf("│                  >   任意键退出游戏   <                     │\n");
329. printf("│                                                             │\n");
330. printf("│                                                             │\n");
331. printf("│                                                             │\n");
332. printf("│                                                             │\n");
333. printf("└─────────────────────────────────────────────────────────────┘\n");
334. system("pause");
335. }

## 5.4 backup.h

1. #pragma once
3. #include "base.h"
4. #include "map.h"
6. **int** pastmap[20][20] = {
7. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
8. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
9. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
10. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
11. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
12. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
13. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
14. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
15. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
16. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
17. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
18. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
19. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
20. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
21. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
22. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
23. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
24. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
25. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
26. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
27. };
29. **int** pastmap2[20][20] = {
30. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
31. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
32. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
33. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
34. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
35. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
36. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
37. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
38. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
39. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
40. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
41. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
42. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
43. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
44. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
45. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
46. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
47. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
48. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
49. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
50. };
52. **int** againmap[20][20] = {
53. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
54. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
55. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
56. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
57. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
58. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
59. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
60. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
61. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
62. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
63. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
64. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
65. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
66. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
67. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
68. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
69. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
70. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
71. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
72. {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
73. };
75. **void** backup()
76. {
77. **for** (**int** i = 0; i < 20; i++)
78. {
79. **for** (**int** j = 0; j < 20; j++)
80. {
81. pastmap2[i][j] = pastmap[i][j];
82. }
83. }
84. **for** (**int** i = 0; i < 20; i++)
85. {
86. **for** (**int** j = 0; j < 20; j++)
87. {
88. pastmap[i][j] = map[level][i][j];
89. }
90. }
91. }
93. **void** abackup()
94. {
95. **for** (**int** i = 0; i < 20; i++)
96. {
97. **for** (**int** j = 0; j < 20; j++)
98. {
99. againmap[i][j] = map[level][i][j];
100. }
101. }
102. }
104. **void** getback()
105. {
106. **for** (**int** i = 0; i < 20; i++)
107. {
108. **for** (**int** j = 0; j < 20; j++)
109. {
110. pastmap[i][j] = pastmap2[i][j];
111. map[level][i][j] = pastmap[i][j];
112. }
113. }
114. }
116. **void** again()
117. {
118. **for** (**int** i = 0; i < 20; i++)
119. {
120. **for** (**int** j = 0; j < 20; j++)
121. {
122. map[level][i][j] = againmap[i][j];
123. }
124. }
125. }
126. 调试与测试

## 开始界面



## 游戏界面







## 结束界面





1. 结论与心得

蔡雯：

一周的课程设计结束了，在这次的课程设计中不仅检验了我所学习的知识，也培养.了我如何去把握一件事情，如何去做一件事情，又如何完成一件事情。在设计过程中，与同学分工设计，和同学们相互探讨，相互学习，相互监督。学会了合作，学会了运筹帷幄，学会了宽容，学会了理解，也学会了做人与处世。

同时我也提升了自己的技术，对于课程知识的更加掌握，从中发现自己平时学习的不足和薄弱环节，从而加以弥补。这是夯实的过程，也是查漏补缺的过程。相信我之后会更加得心应手的独立完成自己的程序。

吴佳柳：

我负责核心算法、人物的移动逻辑与操作列表逻辑的判断与算法设计，现在的自己还是太差了，写游戏的时候很多都需要百度，了解的太少了，主要的问题在于人物移动的判断，通过百度了解了怎么更好的判断移动，写这个游戏提高了自己的编程水平，学会了用码云进行多人开发项目，想给游戏加上背景音乐，但由于时间没来的急，现阶段的目标是熟悉数据结构，争取每天都刷力扣的题目，同时ctf也得跟进，不断地提升自己。

雷志锋：

C语言是一门计算机基础语言，通过上一学期的学习，对于它的一些特点和算法有了一定程度的了解，并能够做一些简单的题目，但并没有掌握多少。这次的课程设计可能是让我们巩固以前的知识并在此基础上能够有所突破。  
这次的课程设计，对于我来说是很困难的，因为我的C语言基础很差，还好这次游戏设计是一个团体作业，组内还有几个大佬，所以在开始的时候，主要还是大佬在操作，发现他们是真的厉害，深刻体会到大佬与小白的差距不是一般的大，有时候感觉和他们交流都有压力，哎！  
感觉C语言是一门很难的语言，我感觉学的蛮痛苦的，尤其是这次的课程设计，我更体会到了这一点，虽然我们是在同一条起跑线上学习的，可每个人的能力都是不同的，有人掌握的好，也有人掌握的不是很好。而我属于中等偏下的，因此，在做设计的时候，主要还是依靠大佬。在此过程中，我深刻体会到了知识不足交流起来都会感受到压力，同时也体会到了成功后的喜悦，也许在以后的学习生活中我会更加体会到这一点。

1. 致谢

我们要感谢的就是何铭康学长，他在我们整个项目的开发过程中全程都在帮助我们完成任务，十分细心的给我们解答问题

========== 附录 ==========

Gitee地址：<https://gitee.com/c_group/c_language_course_design>

**成绩评定表**

|  |
| --- |
| **课程设计总结**  　在课程设计之前，我们通过各个渠道查找资料后分析验证，经过多次的修改和整理，作了如上的设计思路。虽然这次设计一开始是按照设计要求去完成的，但由于在实际操作中，出现了比较大的问题，导致以上的准备资料，在实际操作中都未能派上用场。在这次的课程设计过程中，我懂得了很多，课程设计不光是让我们去“设计”，更重要的是培养我们的能力!通过本次课程设计使我对通信电子线路又有了进一步的了解，增加了对所学知识的应用。 |
| **小组自评**  蔡雯：满分  吴佳柳：满分  雷志锋：剩余分数的50%  熊海：剩余分数的50%  签字：  日期： |
| **指导教师评价**  签字：  日期： |