“五子连珠”项目说明文档

1953871 邓泉

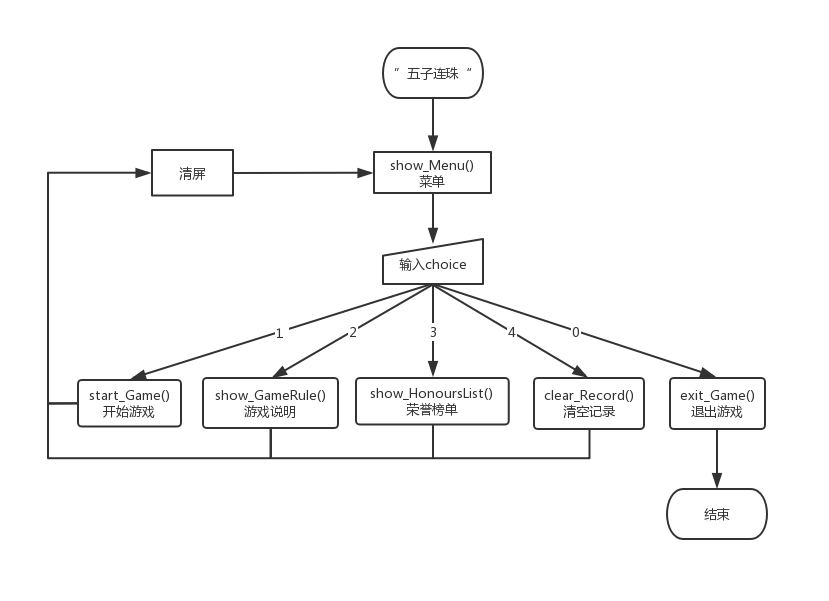
1. 整体设计思路：

为用户提供菜单界面，显示五大板块（开始游戏，游戏说明，荣誉榜单，清空记录，退出程序），用户通过输入可自行选择其中一个，完成所选板块的所有流程后按任意键回到上一级——菜单。

构建Chessboard\_Manager类，封装功能函数，成员属性设定棋子，棋盘，得分，记录，以及成员函数实现所需的临时属性。创建Chessboard\_cell类，表示棋盘上一个单元格的位置，包含在Chessboard\_Manager类的头文件中。

main函数中实例化一个Chessboard\_Manager类的对象，用switch语句根据用户输入匹配case标签，每个分支下调用成员函数实现相应功能。

总体流程图：



1. 程序功能：

显示菜单： 打印菜单界面。

* 1. 开始游戏： 完成一场新游戏，每个阶段需要给用户一个提示。
  2. 游戏说明： 查看游戏规则。
  3. 荣誉榜单： 查看历史前十名玩家和得分，每次游戏都会记录到HonoursList.txt文件中。
  4. 清空记录： 将文件中数据清空。
  5. 退出程序： 可以退出当前游戏程序。

1. 各模块的体系结构设计和描述：
   1. Chessboard\_Manager类：
      1. 构造函数 Chessboard\_Manager():

totalScore = 0 ---> init\_Chessboard() ---> load\_Record()

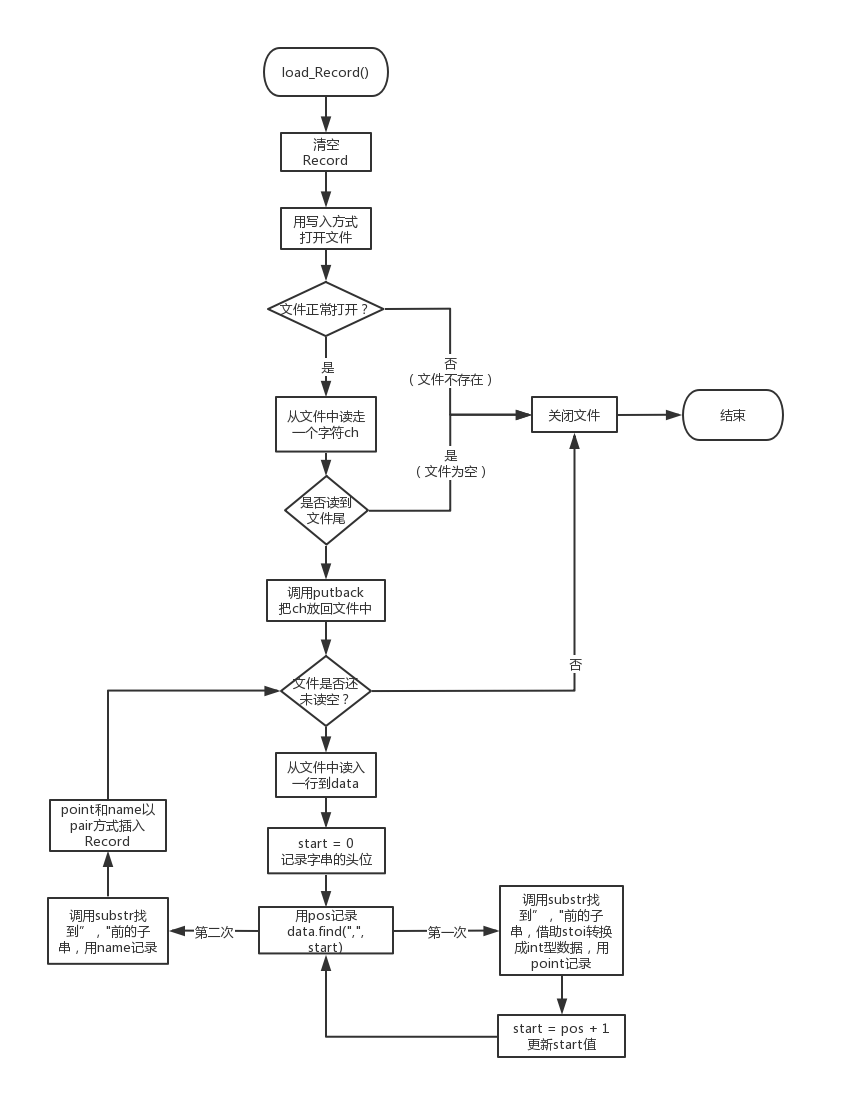
1. void init\_Chessboard():

调用成员属性：Chessboard[][],chessPieces[]。

设定随机数种子 ---> memset()把全部单元置为空格符 ---> Chessboard[i][j]（空位） = chessPieces[index]（i,j,index为随机数）

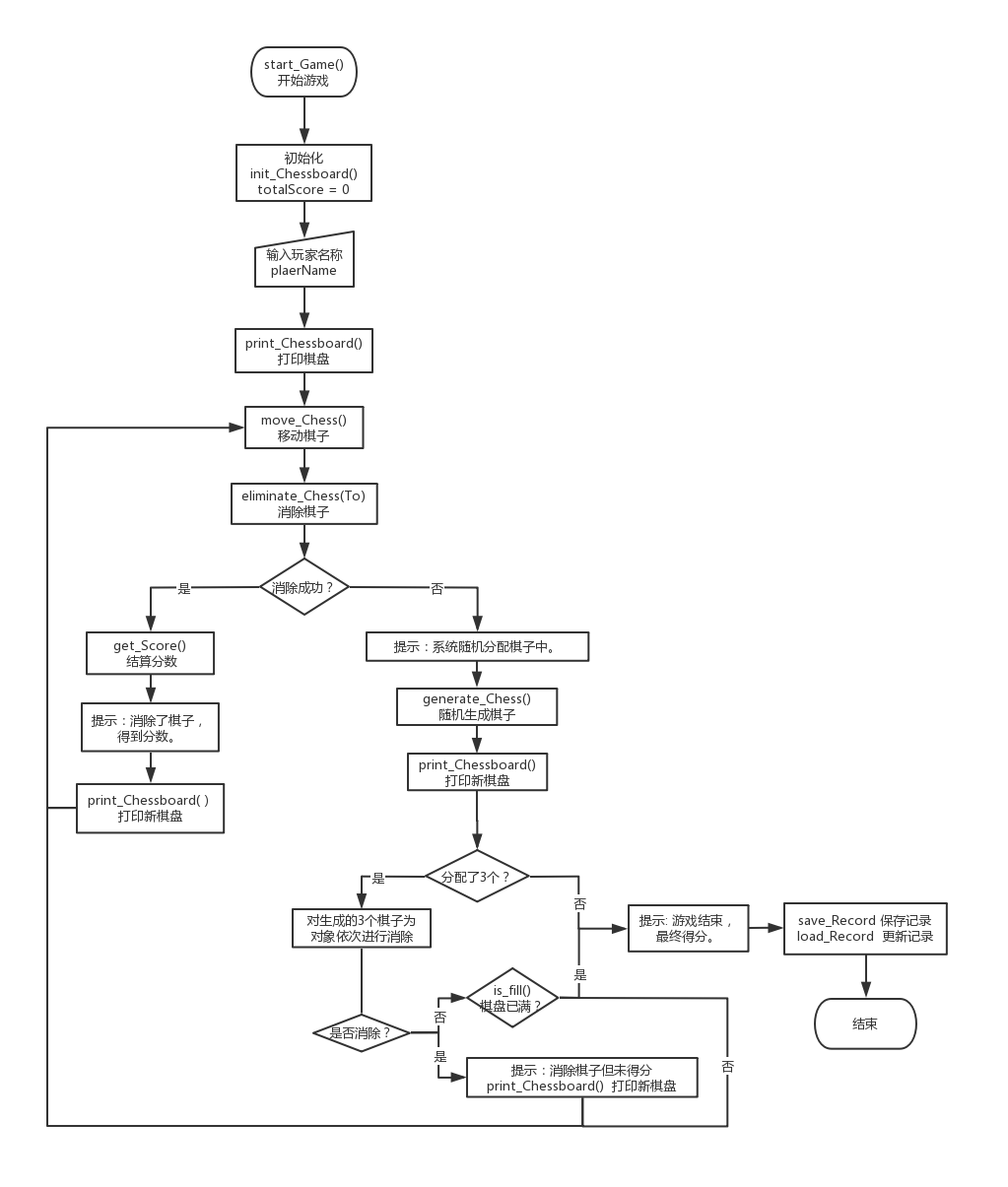
1. void load\_Record():

调用成员属性：Record。



3.1.2 菜单功能 void show\_Menu(): 输出。

3.1.3 开始游戏 void start\_Game():

Start\_Game(）函数是最重要的一大模块，其流程多而结构复杂，为增强代码可读性及易维护性，封装了多个子模块。

1. void print\_Chessboard():

调用成员属性：Chessboard[][]，Record。

清屏 ---> 打印当前棋盘 ---> 棋盘下方显示游戏记录最高分（记录为空则为0）和当前得分。

1. bool is\_fill():

调用成员属性：Chessboard[][];

棋盘已满返回true，未满返回false。

1. void save\_Record():

调用成员属性：totalScore,playerName。

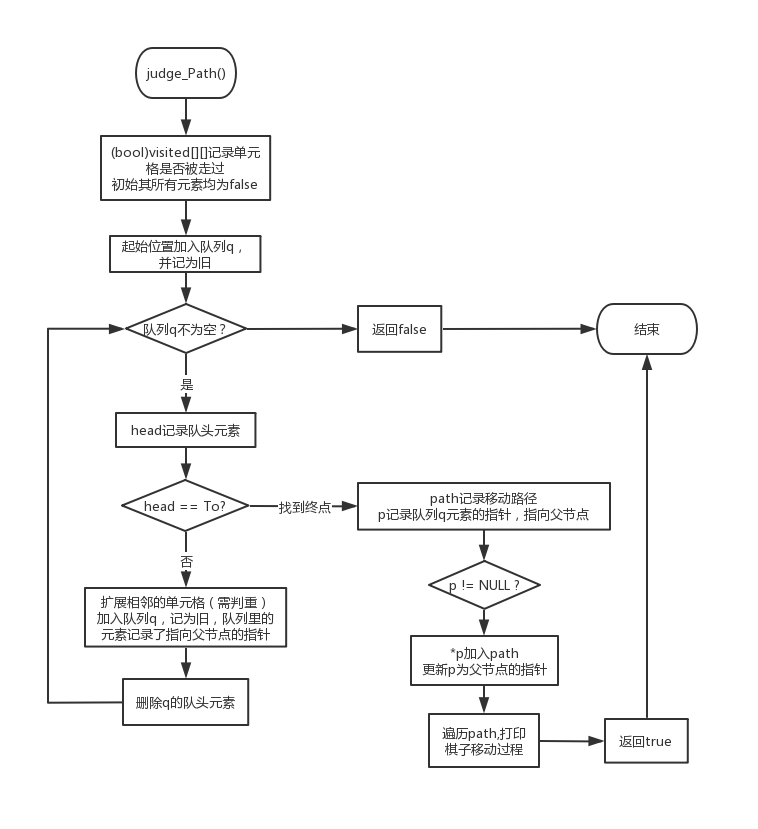
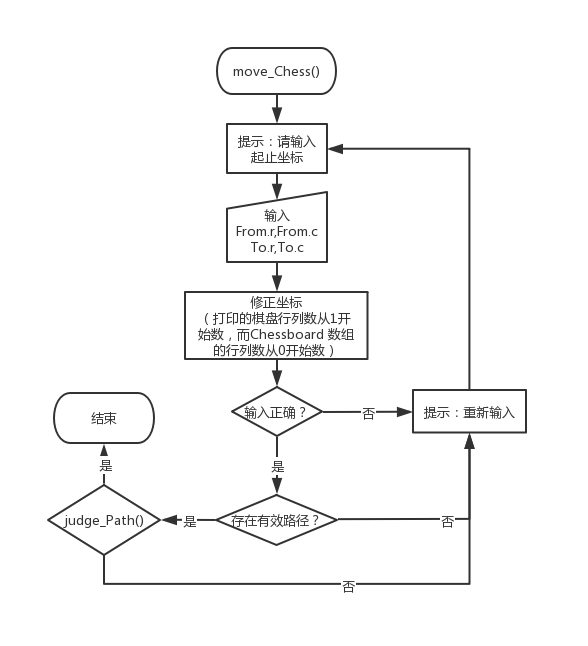
用追加的方式将总分和玩家名称写入文件中。

1. void get\_Score():

调用成员属性：directionNum,cnt,score,totalScore。

修正cnt ---> 计算score(= 2 \* cnt) ---> 累加分数到totalScore

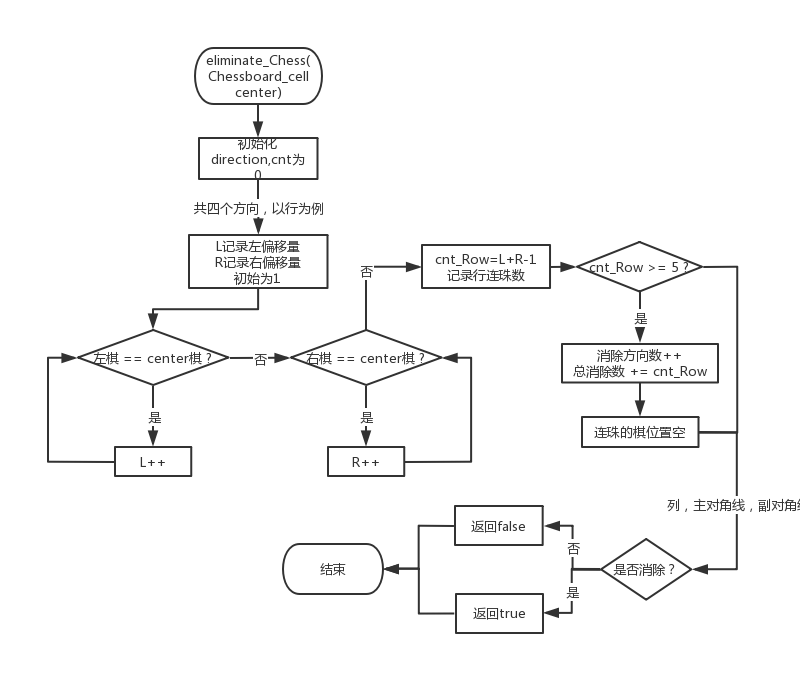
e.void move\_Chess(): 调用成员属性：From,To。 f. void judge\_Path(): 调用成员属性：From,To,Chessboard[][]。



g. bool eliminate\_Chess(Chessboard\_cell center):

调用成员属性：directionNum,cnt,Chessboard[][]。

计算各方向的连珠数，并记录连珠方向上的头尾两个邻单元的位置，达到条件就消除（并记录总消除数和消除方向数，以便最后计算得分）。



h. void generate\_Chess():

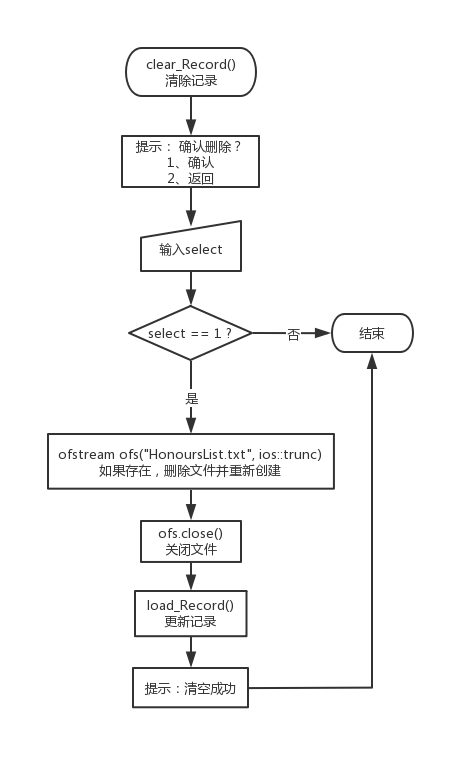
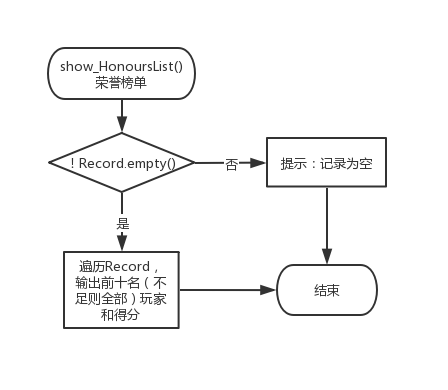
调用成员属性：randomPieces[],Chessboard[][]。

设定随机数种子 ---> 初始化randomPieces[] ---> 记录空位坐标 ---> 生成3个随机棋子置于空位（空位不足3个则置满为止）

3.1.4 游戏说明 void show\_GameRule(): 输出。

3.1.5 荣誉榜单 void HonoursList(): 3.1.6 清空记录 void clear\_Record():

调用成员属性：Record。

3.1.7 退出游戏： void exit\_Game(): 输出“欢迎下次使用！”；利用exit(0)退出程序。

3.2 Chessboard\_cell()类：

成员属性：行数r， 列数c， 指针f（在有效路径判断中用上）。

成员函数：构造函数；重载==和!=。

1. 其他事项：

4.1 在judge\_Path()函数中，采用了BFS（广度优先算法），每一个子节点元素都记录了指向父节点的指针，这样在搜索到终点后并要求得到最短路径时，可以从终点一步步往上一步推，直至起点。在记录指向父节点的指针时，注意到队列数组queue中父节点元素会被删除掉，若指针指向queue中的父节点，则在父节点被删除以后就失效了，因此需要先用new开辟父节点的新地址，然后再让指针指向这个新地址。

4.2 尝试使用DFS（深度优先算法）去实现judge\_Path()，不能保证找到最短路径，显示的移动路径总是绕远路。为使棋子在找路过程中更加智能化，应**增加“剪枝”（**可行性剪枝，最优性剪枝**），寻找过滤条件，提前减少不必要的搜索路径，以达到算法优化。**

4.3 Chessboard\_Manager类的成员属性 移动棋子的起止位置，随机分配棋子的位置 均是Chessboard\_cell类（属性：r, c, f），而Chessboard\_cell\* 的指针f对于这几个位置来说并没有用，这就导致了内存空间浪费的问题。

一种解决方案： 用pair<int, int>型数据来记录这些位置。

4.4 Record存放游戏记录，类型为multimap<int, string, greater>,int型数据记录得分，string型数据记录玩家昵称，multimap容器默认排序规则是按照key值升序排列，为达到降序的效果，加写一个仿函数greater。