Glucose 说明文档（2023 年 1 月 7 日版）

一、简介

“Glucose”是由 UCAS 2021 级计算机科学与技术专业的仲剑自主研发的一款全新开放的五子棋程序，使用 C 语言编写，兼容 Linux、Windows 操作系统且无需图形界面，具有多模式（人人对战、人机对战）、自带调试、对局记录保存等特色功能。本程序自第 1 个版本（2021 年 10 月 10 日完成）以来，在 1 年半的时间内进行了 20 余次优化，最终版本于 2023 年 1 月 7 日正式完成，并于 2023 年 1 月 9 日参加了课程“程序设计基础与实验（C 语言）”的比赛，打入了冠军 B 组，取得了 88 分的成绩。本程序的设计思想完全由作者自行摸索，甚至未参考互联网上的任何资料。

二、使用说明

1. 组成部分

所有后缀为 .c 的文件均为源文件，后缀为 .h 的文件为头文件。此外还有 2 个后缀为 .ini 的文本文件，它们是程序运行时需要的文件：config.ini 保存了程序的配置信息以及算法中的参数信息，board.ini 则保存了程序打印棋盘所需要的字符元素。

2. 编译

在 Linux 操作系统的终端里进入所有文件所在的目录，输入以下命令，可完成编译（需提前安装 gcc）：

```
gcc ./*.c -o ./Glucose -DLINUX -lm
```

在 Windows 操作系统下，可以使用 Visual Studio 软件编译，也可用 Windows Powershell 执行上述指令进行编译，同样需提前安装 gcc，但无需加 -DLINUX 选项。

注：即使加上 -Wall 选项，gcc 仍不会给出警告信息。

3. 运行

在 Linux 操作系统的终端里进入编译后的可执行文件以及 2 个 .ini 文件所在的目录，输入以下命令即可开始执行：

```
./Glucose
```

本程序支持命令行参数，使用 -h 或 --help 选项可查看用法，具体解释如下：

```
-d, --debug
        启用调试模式；
-n, --no-board
```

不显示棋盘；

`--config=<file>`

强制 <file> 替代默认的 config.ini 作为配置信息文件；

`--board=<file>`

强制 <file> 替代默认的 board.ini 作为棋盘字符文件；

`-h, --help`

显示帮助信息。

在 Windows 系统下可以直接双击运行，也可以使用命令行。但如果是在 Windows 10 等较高版本下，可能需要将控制台的属性中“使用旧版控制台”勾选才能正常显示棋盘。

运行后，程序会提示选择对战模式（人人对战或人机对战），如果选择人机对战，则会进一步提示选择是执黑棋先落子，还是执白棋后落子。选择完毕后，游戏开始，界面（人机对战模式）如下图所示：

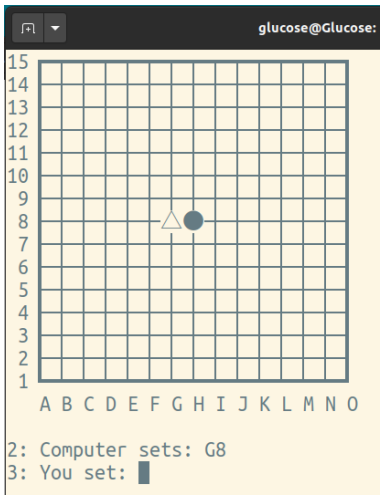

图 1 Linux 下运行界面（Ubuntu）

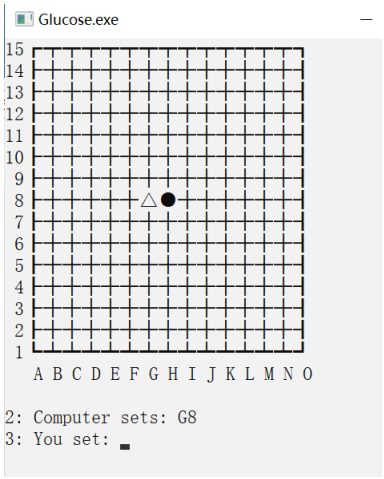

图 2 Windows 下运行界面

按照提示输入坐标进行落子即可。如果对局结束，程序会给出胜负信息，并保存对局记录到程序所在文件夹下，以对局结束的日期时间命名。

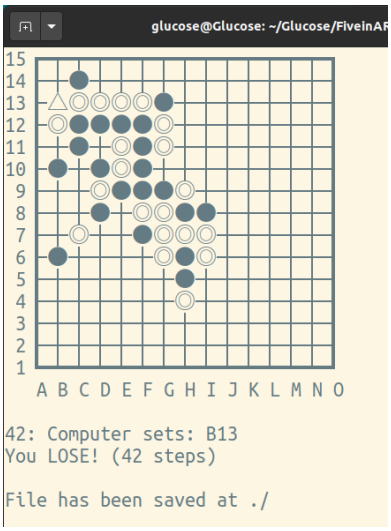

图 3 对局结束的提示

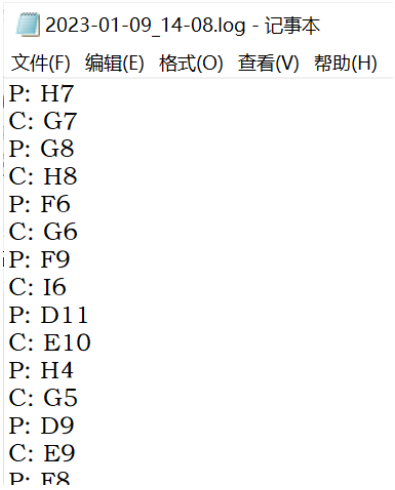

图 4 对局记录文件

4.* 调试

自带调试模式是本程序的一大特色，主要目的是用于监测自动落子的打分值，以及人工构造某些情形（例如构造出禁手测试自动落子程序能否识别）。

在命令行参数中加入 -d 或 --debug，即可进入调试模式。程序启动时会首先展示一些重要参数的值，如下图所示：

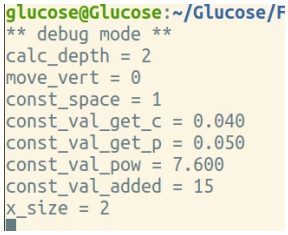

图 5 调试模式启动界面

注：move_vert 用于棋盘的竖直平移（规定了纵坐标从几开始），调试模式下为 0，非调试模式下可在配置信息文件中修改（一般设为 1）；x_size 用于棋盘的水平伸缩，规定每个横坐标占多少格，一般设为 2，如果某些终端（例如 Windows Powershell）的棋盘字符不等宽，可能需要修改为 1。

回车后进入程序，在人机对战模式下，玩家每次落子后，在计算机自动落子之前，程序会打印出按照本文档第四部分第 3 节计算出的即将落子的坐标、分值（第 1 行），以及接下来考虑的几步的分值（第 2 行为在第 1 行的坐标自动落子后，玩家一方可能的落子及分值（由浅层决策函数计算出），第 3 行为下一步自动落子的坐标及分值……）。同时出现干预提示：“Interfere: ”意为此时可人工干预程序落子，方法为：直接按回车，则在程序计算出的位置落子；输入另一个坐标，则打印该位置接下来几步的分值，并且再次出现干预提示，继续输入坐标可不断查看各点的分值，直到按下回车，则程序在最近的一个位置落子，从而实现打分监测以及落子的人工干预。如下图所示：

图 6 调试模式的使用

三、实现原理

1. 源文件

最终版本的程序包含 4 个源文件，它们的作用介绍如下：

`Glucose.c`

主文件，即 main 函数所在的源文件；

`control.c`

用于游戏控制，用于刷新、显示棋盘和控制游戏流程（人人对战、人机对战中的循环落子）；

`calc.c`

用于计算，包含程序人机对战的自动落子算法（扫描棋盘、打分决策）、胜负判断、禁手判断，是整个程序的核心；

`fileIO.c`

用于读写磁盘（额……似乎现在基本上都用 SSD 了）中的文件，包括：读取配置信息文件（默认的 config.ini）、棋盘字符文件（默认的 board.ini），写入对局记录文件。

2. 游戏控制

控制游戏的简单流程图如下：

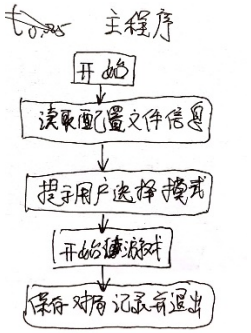

图 7 主程序流程图

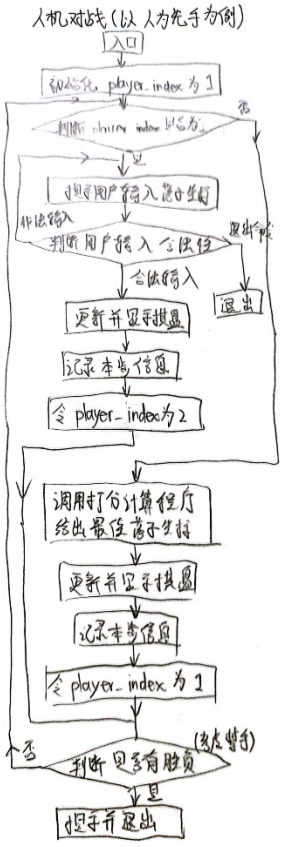

图 8 人机对战流程图（以人为先手为例）

3. 自动落子算法

本程序采用的是简单直接的打分算法，主要由源文件 calc.c 中的下面几个函数实现：

```
unsigned int value_shallow(int iCd, int x, int y, int breaker_en);
```

浅层打分：对给定的一点坐标(x,y)，扫描其 4 个方向上周围的情况，计算如果在该点落子，能给己方的连子数（进攻）以及能堵对方的连子数（防守），每连一个棋子该点的打分加 15（该值由配置信息文件中的 const_val_added 决定，可修改），4 个方向上的打分叠加；后来对该算法做了修正，由于“远水不解近渴”，当扫描到的棋子已经较远（距离达到 3 格）时，加分值折半，如果更远（距离达到 5 格）加分值再次折半。正常情况（配置信息文件中的常数值不是特离谱）下，这样计算出的分值不会特别大（远小于 900）；但如果扫描到对方已成活三、冲四，或即将形成双三、三四等“紧急情况”，或者是己方已成活三、冲四，或即将形成双三、三四等“必胜情况”，则将分值强制调整为 900 以上（按照优先级，给它们打分为 1000、1200、1400、1600 等不同层次），以确保程序在这种情况下能够及时响应；此外，如果有禁手，则将打分归零；棋盘扫描完毕后，如果该点的价值低于 900（非“紧急情况”），则为了后续计算的方便，使用一个上凸函数：

$$f(x)=900^{1-\frac{1}{k}}\cdot x^{\frac{1}{k}},\ 0\le x\le 900$$

进行变换，其中k为常数，由配置信息文件中的 const_val_pow 决定；该函数的图象如下图所示：

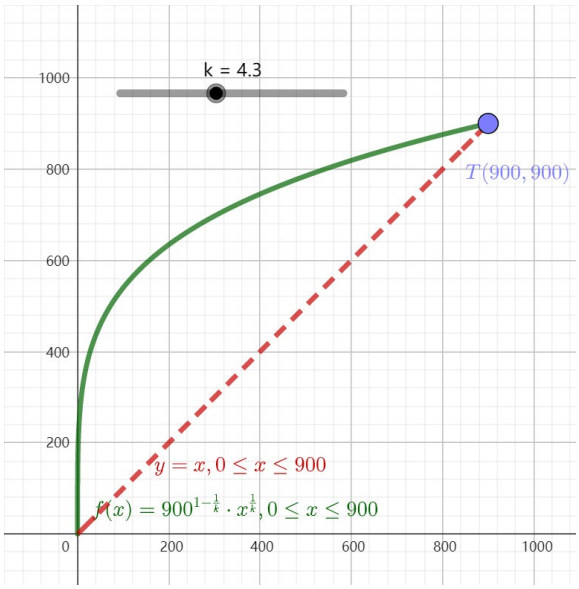

图 9 打分变换函数图象

最后为了使落子尽可能向棋盘中央靠拢，此前打的分会加上一个“距离因子”，它等于 20 减

去该点到棋盘中央的距离，这使得当程序执黑棋先落子时第 1 步总是落在中央位置。

```
unsigned int calc_shallow(int iCd, int breaker_en);
```

浅层决策：根据浅层打分函数，找到棋盘上所有空位的最大分值所对应的点，将该点的坐标连同该点的分值一同返回(您可能会问,如何用一个 unsigned int 类型返回 3 个数值? 事实上，由于x,y坐标数值一定是一位数或二位数，分值最多也不会过四位数，而 unsigned int 的最大值可达 4,294,967,295（10 位数），因此将x乘上10^6，将y乘上10^4，再与分值相加，即可作为一个 unsigned int 来返回，使用时用整除、求模运算即可提取出其中的信息)。

```
unsigned int value_deep(int iCd, int x, int y, int debug_en);
```

深层打分：将棋盘复制到一个副本（相当于草稿纸）中，模拟接下来几步的落子，并计算每一步的分值，按照一定的权重将其线性叠加后，得到某一点最终的分值（以下称为深层分值，前面的浅层打分函数给出的分值称为浅层分值）；其流程图如下所示（细节就不体现了，太复杂）：

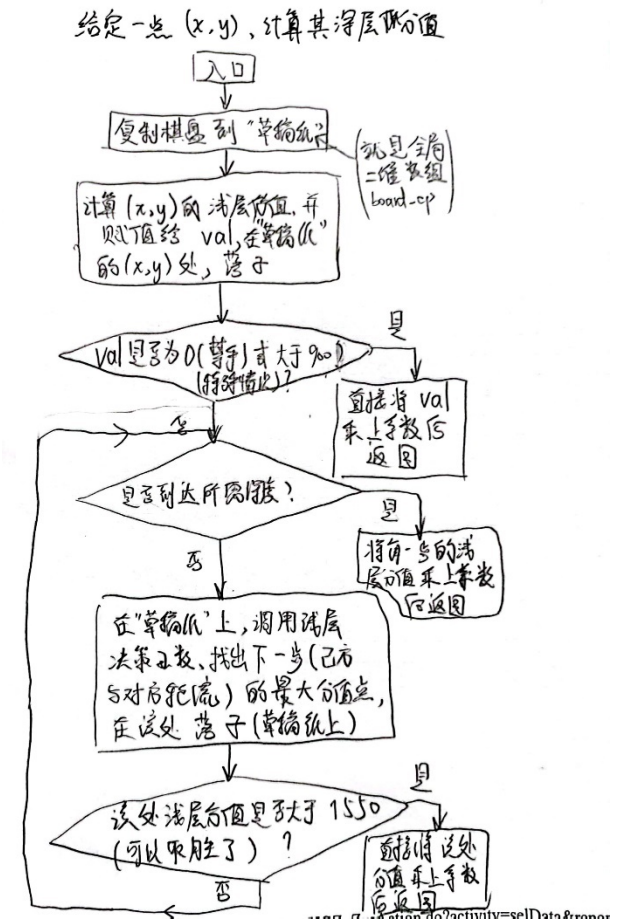

图 10 深层打分流程图

```
unsigned int calc_deep(int iCd);
```

深层决策：类似于浅层决策，根据深层打分函数，找到棋盘上所有空位的最大分值所对应的点，将该点的坐标返回。

```
unsigned int calc_breaker(int iCd, int x, int y, int Bd_cp[][BDMAX + 1]);
```

禁手判断：在早期版本中这个函数只是用来判断禁手的，不过在后期版本（2022 年 8 月 6 日的及以后）中，这个函数不仅能判断双三、双四、长连禁手，还能判断三四，也就是说它已经成为了分析某一点落子后能形成何种棋局的关键函数。实现原理很简单直接，那就是在 4 个方向上扫描，看能形成几个活三、四。

4．读写磁盘文件

首先从命令行参数 argv[0] 中提取程序所在的路径，将其存放到动态分配的内存中，以便后续使用；程序启动后，根据所在路径，打开相应的配置信息文件和棋盘字符文件，用 scanf 等函数从中读取信息，赋值给相应的全局变量，完成文件读取；人机对战过程中，每落子一步，就将该信息记录到缓冲区（全局数组）中，结束后获取系统时间，将缓冲区中的数据写入到所在路径下以时间命名的文本文件中，完成文件写入。

使用配置信息文件和棋盘字符文件的主要目的是，使程序能够在无需重新编译的条件下就可以修改配置信息中的参数以及棋盘字符的编码（GBK, UTF-8 等），避免了反复编译的开销以及修改源代码的风险。

保存对局记录的主要原因是，在程序优化、调试过程中，有时会遇到一些导致异常落子的漏洞，为了在某一局测试结束后能够重现该漏洞，故将对局记录保存到磁盘文件中，便于及时修复。

五、其它

1．源代码有很多注释都没有单独成行，故不要以行数来衡量注释量；

2．本程序的核心算法（浅层打分函数以及禁手函数）是在第一学期刚蹭课学了一点 C 语言时写下的，当时仅是听说大二的课程要写五子棋，并不知晓还要比赛，自己写程序的风格还未形成，很多本可在五子棋中用到的知识（例如链表、树等数据结构及相关算法）也完全不了解，所以那几个函数的编写水平明显比较低而且不规范，后期多次进行优化，能优化成最终版本这个样子已经非常不容易了（理论上讲，完全可以用新学的知识重新再写一遍，但实在不容易拿出这个精力了）；

3．在刚增加配置信息文件 config.ini 功能（2022 年 1 月 19 日的版本）时，其中的参数是

根据经验设置的，后来在 2022 年 7 月底至 8 月初，使用编写的一个叫“Glucose++”的程序，对这些参数进行了幼稚的、几乎无道理的自动调整（对每一个参数，在根据经验设定某个范围内按照指定步长进行扫描，然后让该参数取不同值时的程序进行对战，即“机机对战”，选取优胜的参数），最终的效果非常一般（不算很好但也不是没有）；

4. 关于本程序的命名……要追溯到作者在高一的一段时间里，体重本来就不到 50 kg 而且在那段时间持续下降，险些降到了 45 kg 以下；由于担心身体消瘦导致低血糖引起麻烦，故当时书包里随身带了一包葡萄糖；对此事印象深刻，所以在编写五子棋程序时随手将其命名为 Glucose……

5. Glucose 早期版本全览（截图是作者计算机上的代码备份文件夹）：

注：所有早期版本（2021 年 12 月以前）采用单文件放在了同一个文件夹 2021 里，后期由于分开了多个文件，每个版本单独一个文件夹。

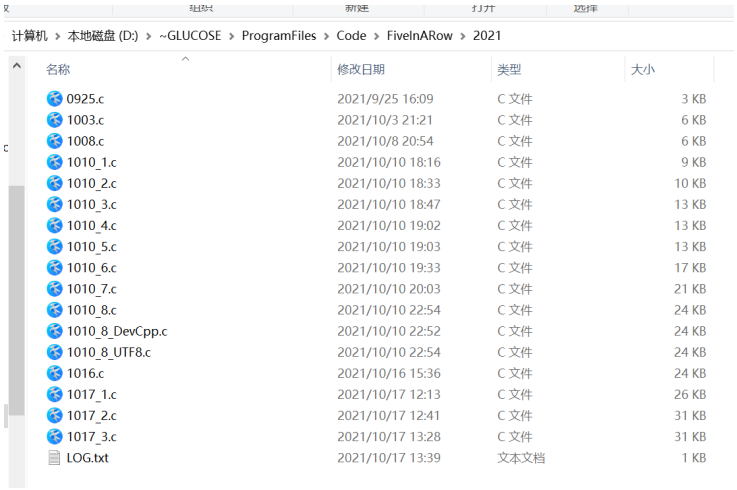

图 11 早期版本（2021 年 12 月以前）备份文件

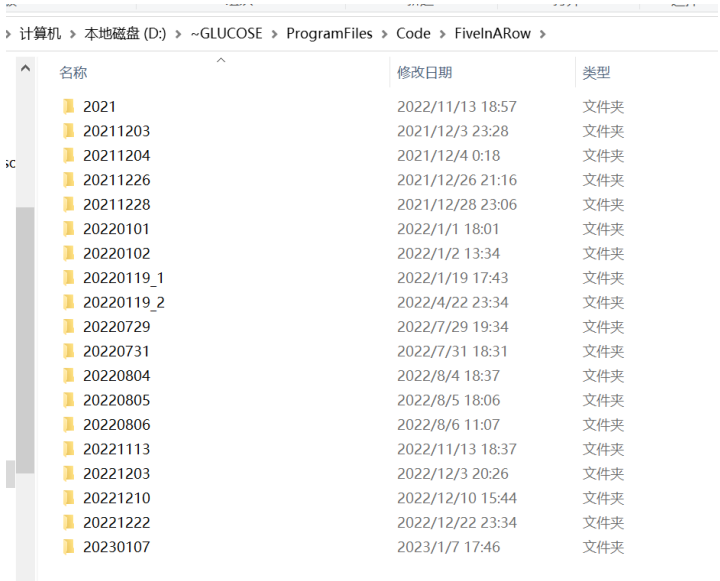

图 12 后期版本（2021 年 12 月及以后）备份文件