

浙江大学

程序设计专题

大程序报告



大程名称：中国象棋

姓名：\_\_\_\_\_

学号：\_\_\_\_\_

电话：\_\_\_\_\_

指导老师：纪守领

2021~2022 春夏学期 2022 年 5 月 13 日

## 报告撰写注意事项

- 1) 图文并茂。文字通顺，语言流畅，无错别字。
- 2) 书写格式规范，排版良好，内容完整。
- 3) 存在拼凑、剽窃等现象一律认定为抄袭；0 分
- 4) 蓝色文字为说明，在最后提交的终稿版本，请删除这些文字。

## 目 录

<b>1</b>	<b>大程序简介 .....</b>	<b>4</b>
1.1	选题背景及意义 .....	4
1.2	目标要求 .....	4
1.3	术语说明 .....	4
<b>2</b>	<b>需求分析 .....</b>	<b>5</b>
2.1	业务需求 .....	5
2.2	功能需求 .....	6
2.3	数据需求 .....	7
2.4	性能需求 .....	8
<b>3</b>	<b>程序开发设计 .....</b>	<b>9</b>
3.1	总体架构设计 .....	9
3.2	功能模块设计 .....	9
3.3	数据结构设计 .....	10
3.4	源代码文件组织设计 .....	11
3.5	函数设计描述 .....	16
<b>4</b>	<b>部署运行和使用说明 .....</b>	<b>18</b>
4.1	编译安装 .....	18
4.2	运行测试 .....	19
4.3	使用操作 .....	21
<b>5</b>	<b>参考文献资料 .....</b>	<b>28</b>

# 中国象棋大程序设计

## 1 大程序简介

### 1.1 选题背景及意义

中国象棋起源于中国，属于二人对抗性游戏，在中国有着悠久的历史。由于用具简单，趣味性强，成为流行极为广泛的棋艺活动。中国象棋是中国棋文化，也是中华民族的文化瑰宝，它源远流长，趣味浓厚，基本规则简明易懂。

中国象棋作为中国传统文化的一部分，至今仍然受到许多人的喜爱。然而，在当今这个信息技术高度发达的时代，很少有人为了下象棋单独购买一块棋盘和一组棋子，寄生于电子设备的中国象棋程序就应运而生。而在网络信息安全保护极度匮乏的现在，一个离线中国象棋程序可以很好地帮助人们解决娱乐需求，这也是中国传统文化传承延续的一小步。为了促进中国象棋在大学生群体中进一步传播，我选择了这个题目。在电脑上即可实现两人对弈及棋谱录制，不需联网，这丰富了大学生的课余生活，并让同学们可以感受到中国传统文化的魅力。

### 1.2 目标要求

使用 libgraphics 和 simpleGUI 实现一个中国象棋游戏系统。包括以下功能：

- (1) 菜单或图标界面：包括但不限于：开始、暂停、重新开始、退出、对局录制、对局回放等。帮助：关于本软件、使用方法等。
- (2) 快捷键：菜单或操作功能快捷键，不少于 5 个。
- (3) 状态信息：对局状态的提示，例如操作倒计时、行棋提示、被吃掉的棋子等。
- (4) 功能：实现基本的中国象棋对局规则，能应对各种异常状况，例如不合规的行棋、长时间不行棋等。
- (5) 文件读取：对局的记录与回放
- (6) 多文件组织：模块化程序设计与编码规范。
- (7) 界面简洁美观、易用、执行速度快、符合代码规范。
- (8) 原创功能：悔棋，播放背景音乐，用户名登录与记录，对局胜负榜等。

### 1.3 术语说明

术语名称	说明
对弈模式	记录双方胜负情况，但不记录棋局行棋步骤。
录制模式	记录双方胜负情况，并记录棋局行棋步骤。

userID	User1 为红方，User2 为黑方
对局胜负榜	记录所有对局下的双方胜负情况
历史对局与棋谱	记录录制模式下录制的行棋步骤（游戏未结束与可以被记录）
重新开始	相同用户名重新开始一局游戏（这一局也会被记录）
再来一局	相同用户名重新开始一局游戏（这一局也会被记录）
帮助	软件使用方法的简单介绍
关于	程序设计相关内容
悔棋	每局双方拥有一次悔棋机会，不可重复悔棋

## 2 需求分析

### 2.1 业务需求

#### 1、娱乐需求

该象棋小游戏占用内存较小，简洁易懂，无需连接网络即可进行游戏，实现了游戏时间自由和游戏条件压缩。

#### 2、象棋学习需求

由于该游戏具有对局录制和回放功能，玩家可随时录制对局或者回放对局以学习象棋对弈策略。

#### 3、鲁棒性

如果处于录制模式下，中途直接退出程序，该局行棋步骤依旧会被记录。

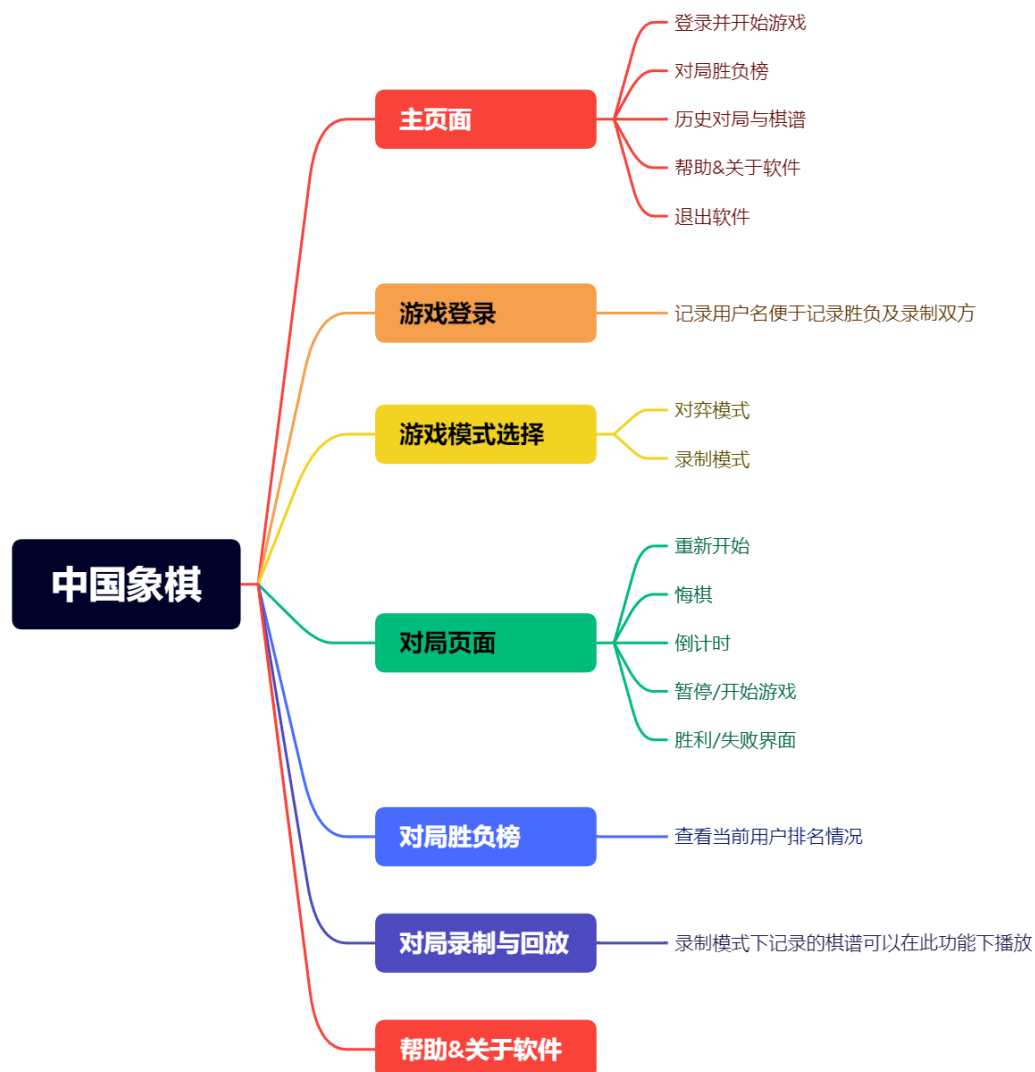
#### 4、美观需求

页面简洁美观。

#### 5、交互需求

用户可自定义用户名，选择游戏模式，暂停、开始游戏，查看当前游戏进度（操作倒计时、行棋提示、被吃掉的棋子等）。

## 2.2 功能需求



目标功能	位置	说明
开始	游戏页面	倒计时开始
暂停	游戏页面	倒计时暂停
重新开始	游戏页面	回到最初状态
退出	主页面	退出程序
对局录制	模式选择页面	对局过程录制在文件中
对局回放	对局录制与回放	文件读取记录并展示
帮助	主页面	程序相关介绍
快捷键	游戏页面	游戏页面按钮对应快捷键
倒计时	游戏页面	提醒时间，倒计时结束自动判负
行棋提示	游戏页面	提示执棋方

被吃掉的棋子	游戏页面	展示被吃掉的棋子
对局规则	游戏页面	游戏页面实现
对局记录	Writefile	文件写入部分
悔棋	游戏页面	每局可以悔棋一次
用户名记录	登录页面	记录双方用户名

## 2.3 数据需求

数据名称	数据类型	存储内容	代码实现	作用域
mode	int	当前页面编号	int mode;	全局
name1 、 name2	char*	用户名	static char name1[20],name2[20];	登录页面
map[10][9] ]	Chess	棋盘格子数据 (是否有棋子、 棋子类型等)	typedef struct { int id;//阵营 int isSelect;//是否 选中 int type;//棋子种 类 int isOn;//是否在 场 double cx,cy;//坐 标 } Chess; //棋盘格子包	游戏页面
Chess_Color	int	执棋方	static int Chess_Color;	游戏页面
mytimer	int	倒计时剩余时间	static int mytimer=time_limit;// 计时器	游戏页面
lose_r,lose _b	int	红方、黑方失子 个数	static int lose_r,lose_b;	游戏页面
loser[2][16] ]	int	记录被吃掉的 棋子	static int loser[2][16];// 记录被吃掉的棋子	游戏页面
selectx 、 selecty	int	选中棋子的坐 标	static int selectx=999,selecty=99 9;	游戏页面
retract[2][ 5]	int	上一步的移动 两个位置的参 数	static int retract[2][5];	游戏页面
kind[2][7]	const string	存储棋子名称	const string kind[2][7]= {"帅","仕","相","马"," 車","炮","兵","将","士 ","象","馬","車","砲","	游戏页面

			卒");	
save	struct	临时存储该步骤变化	struct save { int prex,prey,nowx,nowy; struct save *next; };	对局录制与回放页面
name1[20],name2[20]	static char	记录用户名和密码的数组	static char name1[20]={0},name2[20]={0};	登录页面
name1[7][20],name2[7][20]	static char	用户名	static char name1[7][20]={0},name2[7][20]={0};	对局胜负榜页面

## 2.4 性能需求

### 1、需要在切换页面时即时刷新

将页面刷新控制按钮置存在 MouseEventProcess 函数下即可即时刷新。

注：倒计时模块存在时利用 TimerEventProcess 函数刷新

文本框模块存在时利用 CharEventProcess 函数刷新

### 2、需要在读取文件展示历史记录时不重复读取

利用 firstread、first、firstdisplay 等静态全局变量判断文件是否读取完成，若已读取则刷新时不再重复读取。

### 3、程序编译时间控制在 0.2 秒以内（通常为 0.09 秒）

该程序的开发工具是 C 语言，代码运行平台为 DevC++，编译器编译时间满足要求。



## 3 程序开发设计

### 3.1 总体架构设计



### 3.2 功能模块设计

功能模块	介绍
主页面	包含 6 个按钮，对应 5 种界面和退出游戏。
游戏登录	填入红方和黑方的用户名，提交后进入模式选择界面，如不填则无法进入模式选择界面。
游戏模式选择	选择对弈模式（记录胜负）或录制模式（记录胜负并记录棋谱）。
对局页面	包含游戏页面和若干游戏进程操作按钮、提示框。
倒计时模块	为执棋方倒计时，倒计时归零直接判负。可通过暂停和开始模块暂停倒计时。
行棋提示模块	提醒现在哪一方是执棋方。
重新开始	按下此按钮会使棋盘回到初始状态，若处于录制模式下会保存当前对局记录，新游戏模式将与上一局相同。
失子提示模块	记录双方被吃掉的棋子数目和名称，便于执棋方判断形势。
悔棋模块	回到上一步落子前的状态， <b>注意：双方在一局游戏中仅共同拥有有一次悔棋机会</b>
暂停/开始模块	按下暂停按钮（  ），倒计时将停止；开始按钮（□）使倒

	计时开始。
返回按钮	每个页面（除游戏结束页面）均有返回按钮。可回到上一个页面。
游戏结束	展示胜利方，可选择再来一局或退出游戏
对局胜负榜	展示记录下的对局胜负情况，可通过翻页进行全局查看
对局录制与回放	在历史对局与棋谱中可以查看录制模式下记录的棋谱，通过游戏页面逐步展示，支持翻页查看
帮助	游戏系统功能简单介绍
关于	游戏开发概况
快捷键	在游戏界面使用，F1--暂停游戏 F2--继续游戏 F3--重新开始 F4--悔棋 ESC--返回

### 3.3 数据结构设计

数据名称	数据类型	存储内容	代码实现	作用域
mode	int	当前页面编号	int mode;	全局
name1 、 name2	char*	用户名	static char name1[20],name2[20];	login.c

游戏界面用到的静态全局变量及其作用：

```
static int Chess_Color=0; //判断执棋方，黑方为1
static int gaming=1; //判断是否游戏开始，开始为0
static int isRecord=0; //判断是否录制，录制为1
static int isGaming=0; //判断游戏进行（倒计时启动），进行为1
static int mytimer=time_limit; //计时器
static int lose_r,lose_b; //红方、黑方关子个数
static int loser[2][16]; //记录被吃掉的棋子
static int isRetract=0; //是否悔棋过
static int selecting=0; //当前是否有棋子被选中
static int selectx=999,selecty=999; //选中棋子的坐标
static const double Round=0.6; //棋子半径
const double width=12/9.0; //棋盘格子边长
typedef struct {
    int id; //阵营
    int isSelect; //是否选中
    int type; //棋子种类
    int isOn; //是否在场
    double cx,cy; //坐标
} Chess; //棋盘格子包
static Chess map[10][9]; //定义地图
static int retract[2][5]; //上一步的移动两个位置的id,type,isOn,i,j
const string kind[2][7]= {"帅","仕","相","马","車","炮","兵","将","士","象","馬","車","砲","卒"}; //存储名称
```

对局录制与回放页面用到的静态全局变量及其作用：

```
struct save {
    int prex,prey,nowx,nowy;
    struct save *next;
}; //临时存储该步骤变化
struct save *p=NULL,*head=NULL,*tail=NULL; //辅助指针
static int i=1; //辅助变量
static int first=1; //第一次进入页面
static int firstdisplay=1; //是否读取文件完成
static int page=1; //页码
static int istailed=0; //页码是否到最后一页
static char name1[7][20]= {0},name2[7][20]= {0}; //存储双方用户名
```

对局胜负榜用到的静态全局变量及其作用：

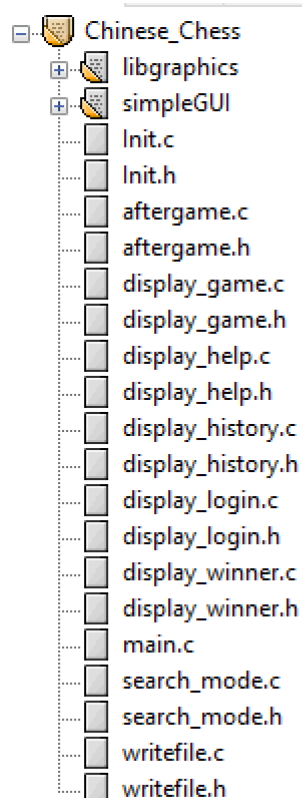
```
static int firstread=0; //是否第一次进入该页面
static int result[7]; //存储当前页面显示的胜负结果
static int i=0; //辅助变量
static char name1[7][20]= {0}, name2[7][20]= {0}; //用户名
static int total; //当前页面总共展示结果数
static int istailed=0; //是否已经展示最后一条内容
static int page=1; //页码
```

### 3.4 源代码文件组织设计

<文件目录结构>

1) 文件函数结构

文件目录一览：



.c 文件	
文件名称	函数定义
main	void Main(); //主函数 void MouseEventProcess(int x, int y, int button, int event); /// 用户的鼠标事件响应函数
init	void display0(); //主页面

display_login	void display1();//登录界面 void WriteName(char name1[],char name2[]);//将用户名写入文件 char* getname1();//传输 name1 char* getname1();//传输 name2 void CharEventProcess(char ch);// 用户的字符事件响应函数		
search_mode	void display7();//选择模式界面		
display_game	void display6();//游戏界面 void drawMap();//画棋盘 void drawChess(double px, double py, int button, int event);//棋子函数 int rule(int pi, int pj);//判断走法是否可行 void drawTimer();//倒计时模块 void record(int Record);//传入录制情况 void drawPiece();//画棋子 void movechess(int px,int py,int nx,int ny);//接收棋子移动信息 void retractit(int i,int j);//记录上一步行棋 void doretract();//悔棋 void TimerEventProcess(int timerID);// 用户的计时器时间响应函数 void KeyboardEventProcess(int key, int event);// 用户的键盘事件响应函数		
aftergame	void Gameover(int Chess_Color);//游戏结束页面 int color();//传出 Chess_Color		
display_help	void display3();//帮助 void display4();//关于		
display_history	void display5();//历史对局与棋谱界面 void openhistory();//显示对应棋局 void readdata();//读数据		
display_winner	void display2();//历史对局		
writefile	void newfile();//创建新文件 void writenew(int px,int py,int nx,int ny);//写入新步骤		
.h 文件			
文件名称	宏定义	自定义函数声明	imGUI 相关函数声明
init	#define WindowWidth 20.0 #define WindowHeight 14.0	void display0();// 主页面	void MouseEventProcess(int x, int y, int button, int event);/// 用户的鼠标 事件响应函数
display_login	#define WindowWidth 20.0 #define WindowHeight 14.0	void display1();// 登录界面 void WriteName(char name1[],char name2[]);// 将 用	void CharEventProcess(char ch);// 用户的字符事件 响应函数 void MouseEventProcess(int

		户名写入文件 char* getname1();// 传 输 name1 char* getname1();// 传 输 name2	x, int y, int button, int event);// 用户的鼠标 事件响应函数
search_mode	#define WindowWidth 20.0 #define WindowHeight 14.0	void display7();// 选择模式界面	void MouseEventProcess(int x, int y, int button, int event);// 用户的鼠标 事件响应函数
display_game	#define WindowWidth 20.0 #define WindowHeight 14.0 #define timer_interval 1000 #define time_limit 45	void display6();// 游戏界面 void drawMap();// 画 棋盘 void drawChess(double px, double py, int button, int event);// 棋子函数 int rule(int pi, int pj);//判断走法是否可行 void drawTimer();// 倒 计时模块 void record(int Record);//传入录 制情况 void drawPiece();// 画 棋子 void movechess(int px,int py,int nx,int ny);//接收 棋子移动信息 void retractit(int i,int j);//记录上一步行棋 void doretract();//	void TimerEventProcess(int timerID);// 用户的计 时器时间响应函数 void KeyboardEventProcess( int key, int event);// 用 户的键盘事件响应函 数 void MouseEventProcess(int x, int y, int button, int event);// 用户的鼠标 事件响应函数

		悔棋	
aftergame	#define WindowWidth 20.0 #define WindowHeight 14.0 #define timer_interval 1000	void Gameover(int Chess_Color);// 游戏结束页面 int color();//传出 Chess_Color	void MouseEventProcess(int x, int y, int button, int event);// 用户的鼠标 事件响应函数
display_help	#define WindowWidth 20.0 #define WindowHeight 14.0	void display3();// 帮助 void display4();// 关于	void MouseEventProcess(int x, int y, int button, int event);// 用户的鼠标 事件响应函数
display_history	#define WindowWidth 20.0 #define WindowHeight 14.0	void display5();// 历史对局与棋谱 界面 void openhistory();//显示 对应棋局 void readdata();// 读数据	void MouseEventProcess(int x, int y, int button, int event);// 用户的鼠标 事件响应函数
display_winner	#define WindowWidth 20.0 #define WindowHeight 14.0	void display2();// 历史对局	void MouseEventProcess(int x, int y, int button, int event);// 用户的鼠标 事件响应函数
writefile	#define WindowWidth 20.0 #define WindowHeight 14.0	void newfile();// 创建新文件 void writenew(int px,int py,int nx,int ny);//写入 新步骤 void close_fn(int isClose);//游戏结 束关闭记录	

## 2) 多文件构成机制

文件名	文件包含	#define保护	外部变量	外部函数
main	包含所有.h文件	.c文件无	无	无
init	在对应.h文件中 包含	#ifndef _INIT_H #define _INIT_H #endif	extern int mode;	void MouseEventProces s(int x, int y, int button, int event);//// 用户 的鼠标事件响应函

				数
display_login	在对应.h 文件中包含	#ifndef _DISPLAY_LOG IN_H #define _DISPLAY_LOG IN_H #endif	extern int mode;	void MouseEventProces s(int x, int y, int button, int event);///// 用户 的鼠标事件响应函 数
search_mode	在对应.h 文件中包含	#ifndef _SEARCH_MODE _H #define _SEARCH_MODE _H #endif	extern int mode;	void record(int Record);// 从 game.c 传入录制情 况 void MouseEventProces s(int x, int y, int button, int event);///// 用户 的鼠标事件响应函 数
display_game	在对应.h 文件中包含	#ifndef _DISPLAY_GAM E_H #define _DISPLAY_GAM E_H #endif	extern int mode;	int color();//向 aftergame传出 Chess_Color void MouseEventProces s(int x, int y, int button, int event);///// 用户 的鼠标事件响应函 数
aftergame	在对应.h 文件中包含	#ifndef _AFTERGAME_H #define _AFTERGAME_H #endif	extern int mode;	void MouseEventProces s(int x, int y, int button, int event);///// 用户 的鼠标事件响应函 数
display_help	在对应.h 文件中包含	#ifndef _DISPLAY_HEL P_H #define _DISPLAY_HEL P_H #endif	extern int mode;	void MouseEventProces s(int x, int y, int button, int event);///// 用户 的鼠标事件响应函 数
display_history	在对应.h 文件中包含	#ifndef _DISPLAY_HIS TORY_H #define	extern int mode;	void MouseEventProces s(int x, int y, int button, int

		_DISPLAY_HIS TORY_H #endif		event);//// 用户的鼠标事件响应函数
display_winner	在对应.h 文件中包含	#ifndef _WINNER_H #define _WINNER_H #endif	extern int mode;	void MouseEventProcess(int x, int y, int button, int event);//// 用户的鼠标事件响应函数
writefile	在对应.h 文件中包含	#ifndef _WRITEFILE_H #define _WRITEFILE_H #endif	extern int mode;	无

### 3.5 函数设计描述

文件名称	函数定义及功能描述	参数描述	返回值描述	重要局部变量及用途	算法描述
main	void Main();//主函数	无	无	无	初始化窗口和外部接口函数
	void MouseEventProcess(int x, int y, int button, int event);//// 用户的鼠标事件响应函数	x,y 鼠标位置坐标 event 发生事件	无	无	游戏界面鼠标移动刷新
init	void display0();//主页面	无	无	无	绘制主页面
display_login	void display1();//登录界面	无	无	无	绘制登录页面，接收用户名
	void WriteName(char name1[],char name2[]);//将用户名写入文件	两个用户名指针	无	无	将用户名写入文件
	char* getname1();//传输 name1	无	指向用户名的指针	无	得到 name1
	char* getname1();//传输 name2	无	指向用户名的指针	无	得到 name2
	void CharEventProcess(ch	输入字符	无	无	根据输入字符写入 name




	ar ch);// 用户的字符 事件响应函数				数组
search_m ode	void display7();// 选 择模式界面	无	无	无	选择录制或 不录制
display_g ame	void display6();// 游 戏界面	无	无	无	绘制游戏页 面，实现用 户交互
	void drawMap();// 画 棋盘	无	无	无	绘制地图
	void drawChess(double px, double py, int button, int event);//棋 子函数	棋子绝对 坐标、鼠 标事件	无	无	与用户进行 交互，移动、 选择棋子等
	int rule(int pi, int pj);// 判断走法是否 可行	想将棋子 移动到的 位置坐标	走法是否可 行，可行为 1	无	判断走法是 否可行
	void drawTimer();// 倒计时模块	无	无	无	绘制倒计时 模块
	void doretract();// 悔 棋	无	无	无	悔棋操作
	void record(int Record);// 传入录制 情况	录制模式 或对弈模 式的选择	无	无	获得是否录 制的指令
	void drawPiece();//画 棋子	无	无	无	绘制棋子
	void movechess(int px,int py,int nx,int ny);//接收棋子移动 信息	棋子上一步和这一步的坐标	无	无	接收棋子移 动信息
	void retractit(int i,int j);//记录上一步行棋	上一步棋 子坐标	无	无	记录上一步 行棋
	void TimerEventProcess(in t timerID);// 用户的 计时器时间响应函 数	时间 ID	无	无	(1) 计时器 根据游戏是 否在进行进 行倒计时——timerID1 (2) 除游戏 界面外其他 界面 每 1/1000 秒刷 新一次
	void KeyboardEventProce	按下按键 名称和动	无	无	实现快捷键

	ss(int key, int event);// 用户的键盘事件响应函数	作			
aftergame	void Gameover(int Chess_Color);// 游戏结束页面	胜利方	无	无	绘制结束页面，用户选择下一步
	int color();// 传出 Chess_Color	无	Chess_Color 值	无	传 出 Chess_Color
display_help	void display3();// 帮助	无	无	无	绘制帮助界面
	void display4();// 关于	无	无	无	绘制关于界面
display_history	void display5();// 历史对局与棋谱界面	无	无	无	绘制历史对局选择界面
	void openhistory();// 显示对应棋局	无	无	无	绘制历史棋局展示页面
	void readdata();// 读取数据	无	无	文件指针，读取步骤	读取数据
display_winner	void display2();// 历史对局	无	无	文件指针，读取用户名	绘制历史胜负页面
writefile	void newfile();// 创建新文件	无	无	文件指针，创建/写入文件	创建新文件
	void writenew(int px,int py,int nx,int ny);// 写入新步骤	上一步和这一步的坐标	无	文件指针，写入文件	写入新步骤
	void close_fn(int isClose);// 游戏结束关闭记录	是否游戏结束	无	无	关闭文件指针

## 4 部署运行和使用说明

### 4.1 编译安装

📁 工程文件

 3210106034\_王伟杰\_大程序设计

- 1、下载压缩文件并解压
- 2、打开工程文件

exe_and_file	2022/5/13 16:50	文件夹
libgraphics	2022/5/5 16:19	文件夹
obj	2022/5/13 16:50	文件夹
simpleGUI	2022/4/17 20:12	文件夹
src	2022/5/8 21:40	文件夹
Chinese_Chess	2022/5/10 22:59	Dev-C++ Project File
Chinese_Chess.layout	2022/5/13 15:38	LAYOUT 文件
Makefile.win	2022/5/13 16:50	WIN 文件
程序设计专题大程报告	2022/5/13 16:47	DOC 文档
录屏演示	2022/5/11 21:57	媒体文件(.mp4)

### 3、按 F11 编译并运行



## 4.2 运行测试

在测试程序的过程中遇到了一些问题，在这里列举其中部分发现的问题及我的解决方案。

1、查看历史记录时对于只有一步的记录无法查看  
解决方案：

将判断文件结束的操作放在链表建立之后即可避免。

```

fp=fopen(concat(ch, ".txt"), "r");
// InitConsole();
while(1) {
    p=(struct save*)malloc(sizeof(struct save));
    fscanf(fp, "%d %d %d %d\n", &p->prex, &p->prey, &p->nowx, &p->nowy);
    // printf("%d %d %d %d\n", p->prex, p->prey, p->nowx, p->nowy);
    p->next=NULL;
    if(head==NULL) {
        head=p;
    } else {
        tail->next=p;
    }
    tail=p;
    if(feof(fp))break;
}

```

2、历史记录和胜负记录只能展示七条

解决方案:

新增一个 page 和 istailed 变量, 判断是否到页码末尾和判断页数。

**static int istailed=0;** //是否已经展示最后一条内容

**static int page=1;** //页码

3、登录窗口用户名输入时不能及时刷新

解决方案:

问题产生的原因是我将刷新操作放在了 MouseEventProcess 函数里, 在输入字符时鼠标未进行操作。在 CharEventProcess 函数中新增刷新操作即可解决。

```

void CharEventProcess(char ch) {
    uiGetChar(ch); // GUI 字符输入
    display1();
}

```

4、飞将操作未写入程序

新增飞将判断。

```

if(!map[pi][pj].type&&selecty==pj&&map[pi][pj].isOn) {
    int k,cntt=0;
    for(k=selectx+1; k<pj; k++)if(map[k][pj].isOn)cntt++;
    for(k=pj+1; k<selectx; k++)if(map[k][pj].isOn)cntt++;
    if(!cntt)return 1;
} // 飞将

```

5、悔棋操作进行后上一步被吃掉的棋子仍在失子队列中

在 doretract 函数里新增上一步是否吃棋子的判断, 若吃掉棋子则失子数量减一。

```

if(retract[1][2]) {
    if(Chess_Color&&lose_b>0)lose_b--;
    else if(!Chess_Color&&lose_r>0)lose_r--;
}

```

6、将刷新操作写在了鼠标函数中, 导致鼠标移动页面才会刷新。

新增 timerID0, 将刷新操作放在时间函数里。

```

if(!timerID){
    switch(mode) {
    case 0:
        display0();
        break;//初始界面
    case 1:
        display1();
        break;//登录
    case 2:
        display2();
        break;//对局记录
    case 3:
        display3();
        break;//帮助
    case 4:
        display4();
        break;//关于
    case 5:
        display5();
        break;//历史对局与棋谱
    case 7:
        // ...
    }
}

```

7、在游戏界面按下空格键会回到登录界面。

原因：CharEventProcess 函数在游戏界面仍会继续运行。

```

void CharEventProcess(char ch) {
    uiGetChar(ch); // GUI 字符输入
    if(mode==1)
        display1();
}

```

解决方案：

加上判断语句是否现在处于登录界面。

8、悔棋操作不会被计入文本。

原因：悔棋函数中没有记录函数写入。

解决方案：写入记录函数 writenew()。

9、游戏未结束不能返回查看对局步骤。

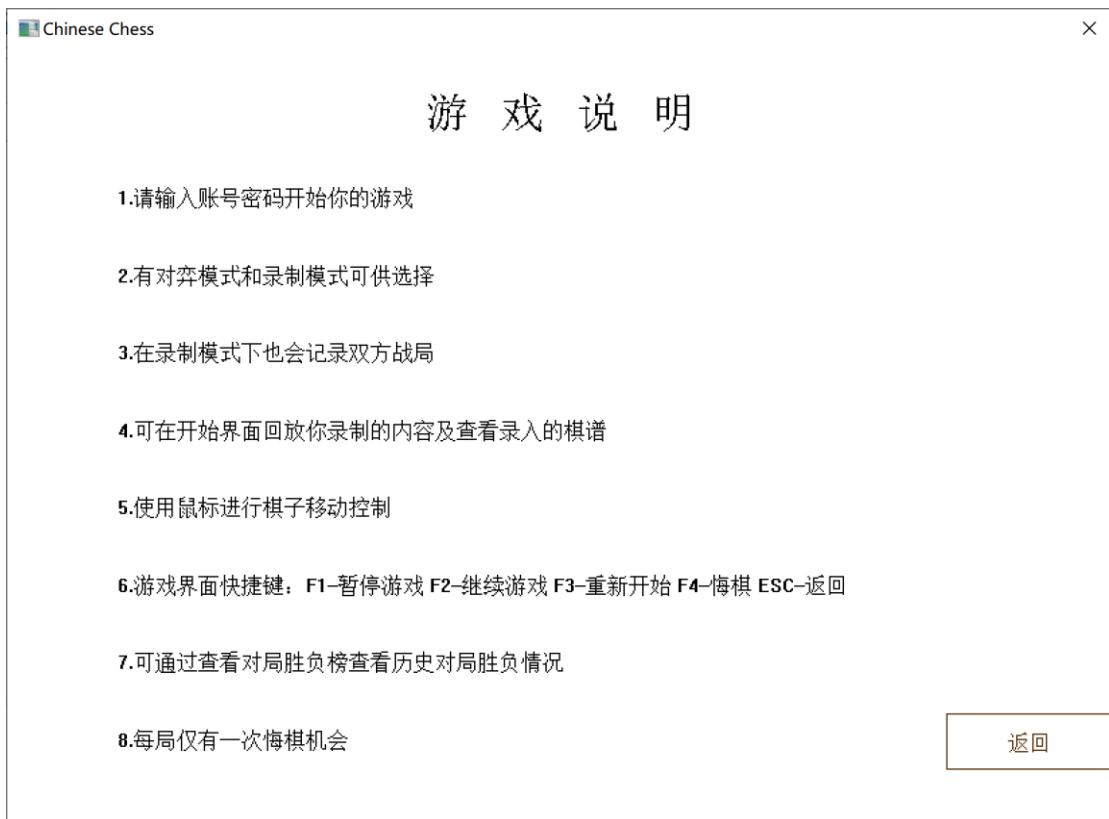
原因：未关闭文件指针。

解决方案：在 Gameover 函数中写入判断游戏结束的函数 close\_fn()，使其关闭文件指针。

## 4.3 使用操作

1、打开程序

2、帮助



查看游戏相关内容，点击返回按钮返回主页面。

### 3、关于



查看游戏相关内容，点击返回按钮返回主页面。

#### 4、对局胜负榜

展示历史对局胜负情况。上一页/下一页按钮用来翻页，若处于首页上一页按钮不会起作用，末尾页下一页按钮不起作用。点击返回按钮返回主页面。

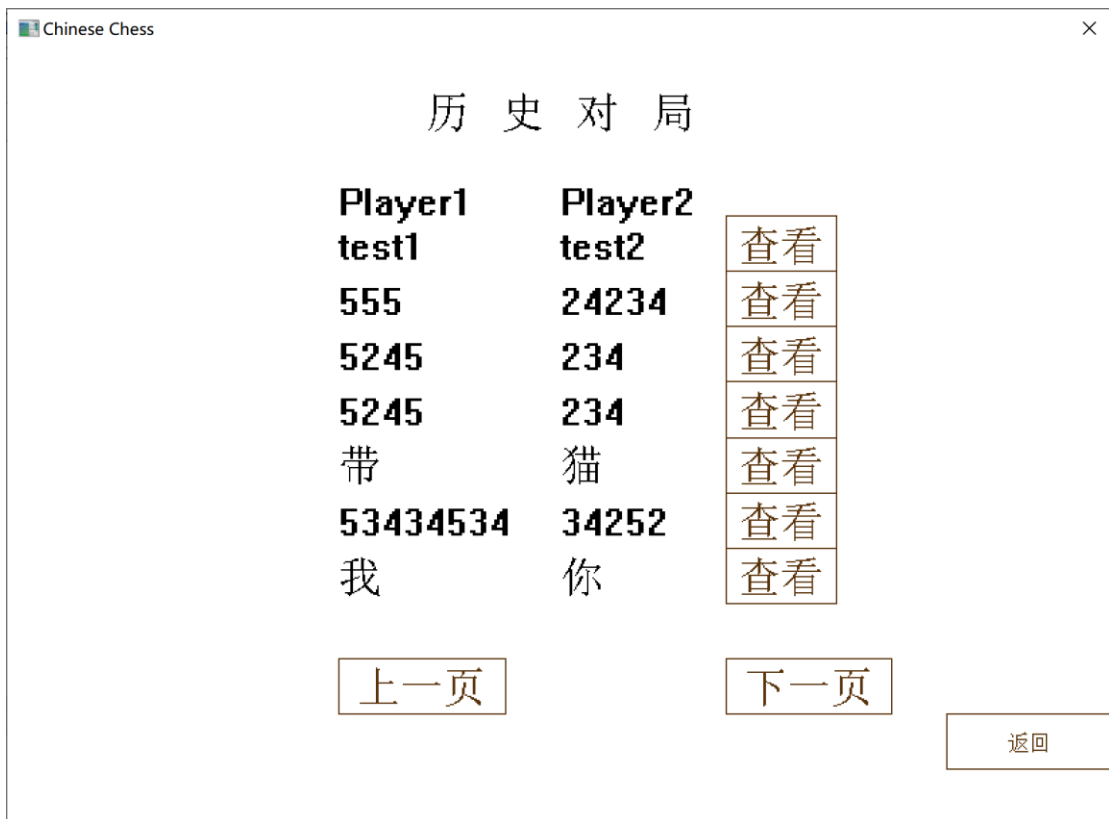




## 5、历史对局与棋谱

展示录制下的历史对局。中央是双方用户名，点击相应用户名后面的查看按钮查看该局情况。上一页/下一页按钮用来翻页，若处于首页上一页按钮不会起作用，末尾页下一页按钮不起作用。点击返回按钮返回主页面。





进入查看页面。展示初始棋盘。点击下一步，棋盘会复原下一步棋子位置，直到录制内容播放完毕。点击返回按钮返回历史对局与棋谱页面。

## 6、登录并开始游戏

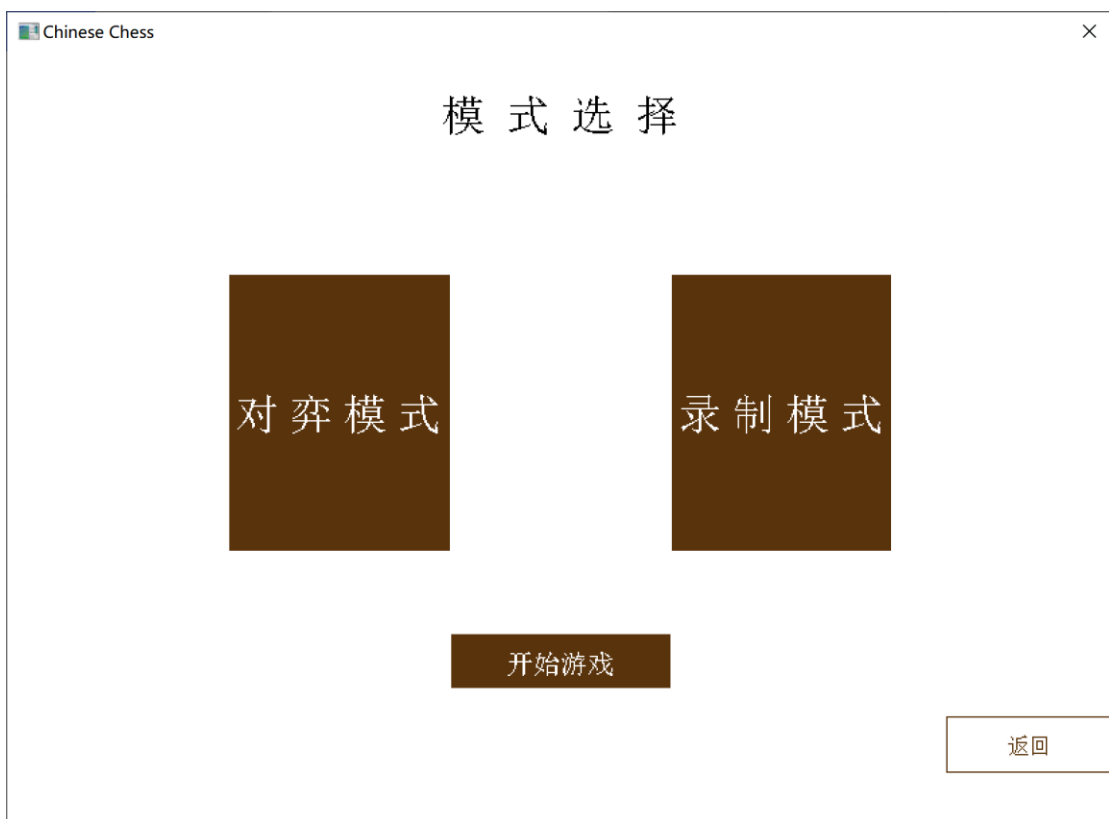
### (1) 登录页面



The image shows a window titled "Chinese Chess" with a close button (X) in the top right corner. The main title "游戏登录" (Game Login) is centered at the top. Below it, there are two input fields: "user1 ID:" followed by a text box containing a blue cursor, and "user2 ID:" followed by an empty text box. Below these fields is a "提交" (Submit) button. In the bottom right corner, there is a "返回" (Return) button.

填写双方用户名，未填写不能进入下一页面。点击返回按钮返回主页面。

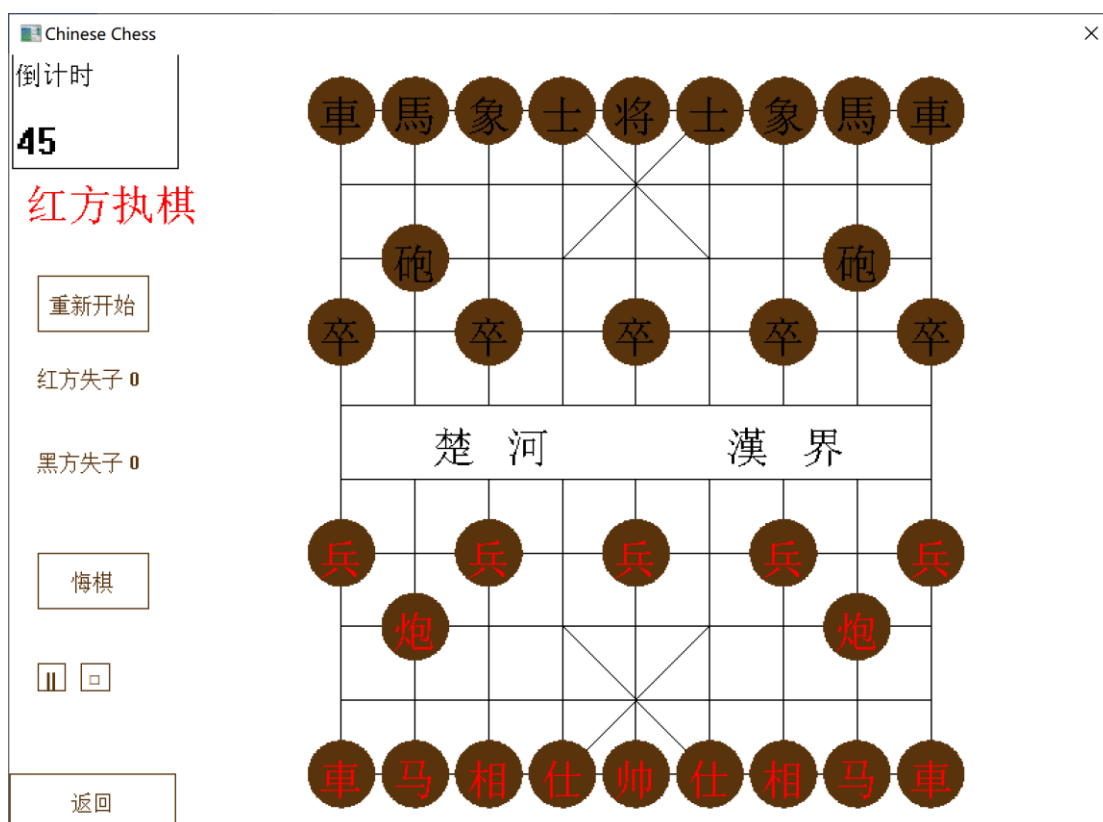
(2) 模式选择页面



The image shows a window titled "Chinese Chess" with a close button (X) in the top right corner. The main title "模式选择" (Mode Selection) is centered at the top. Below it, there are two large brown rectangular buttons: "对弈模式" (Play Mode) on the left and "录制模式" (Record Mode) on the right. Below these two buttons is a "开始游戏" (Start Game) button. In the bottom right corner, there is a "返回" (Return) button.

选择是否录制，录制模式下将录制对局并记录胜负情况，对弈模式下仅记录胜负情况。未选择不会开始游戏。可点击返回按钮返回上一步修改用户名。

### (3) 游戏页面



左上方为倒计时模块，倒计时结束后执棋方判负，红方第一次选择棋子后倒计时开始计时。

下面是行棋提示模块，提醒执棋方。

下面是失子模块，分别记录红方和黑方失子个数及名称。

悔棋模块一局仅可使用一次。

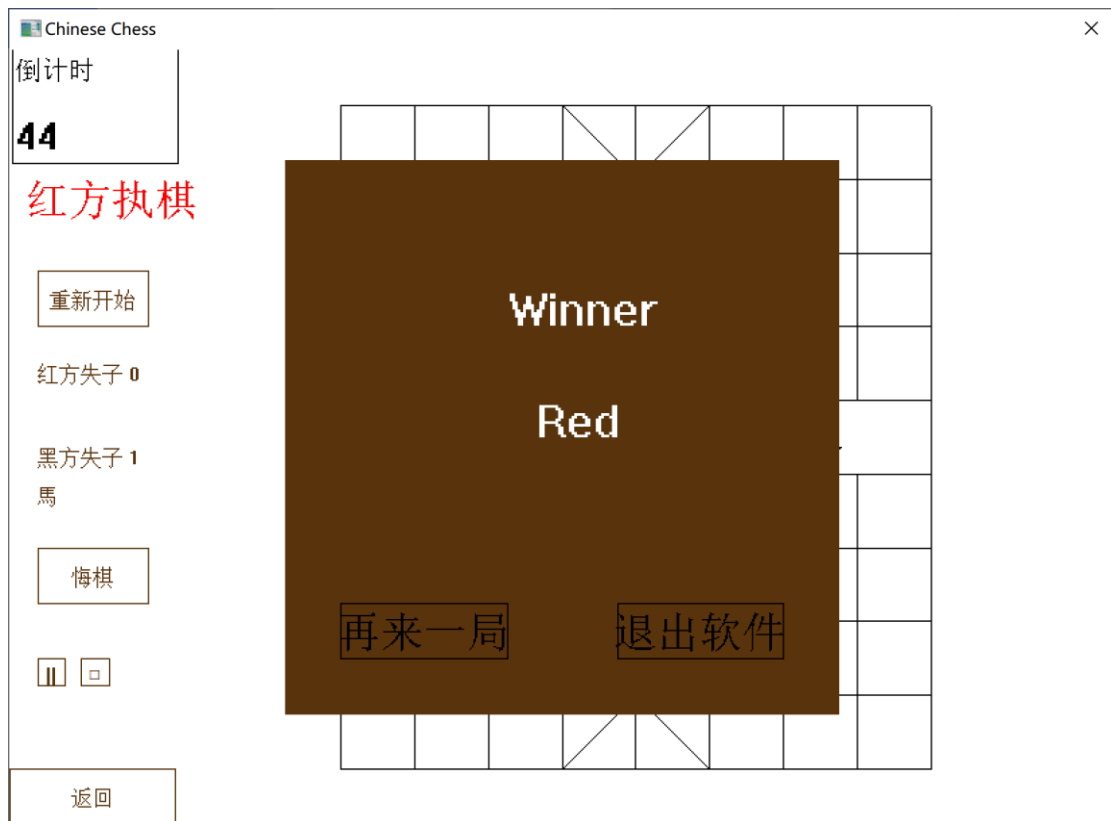
暂停/开始模块包括两个按钮，按下暂停按钮游戏暂停，按下开始按钮游戏开始。

左下角为返回按钮，将返回上一步模式选择。

右侧为棋盘，双方在自己执棋时通过先点击要移动的棋子，后点击移动到的位置实现棋子移动和吃掉对方棋子。

当倒计时结束或将/帅被吃掉游戏结束。

### (4) 游戏结束页面



游戏结束展示胜利方，用户可有两个选择：退出或再来一局。若选择再来一局，会以当前用户名和模式再次进行游戏。

## 5 参考文献资料

- [1] C 语言程序设计（第 4 版）高等教育出版社
- [2] 程序设计专题课件