

《高级语言程序设计》实验大作业反思报告

大作业题目	重力四子棋及设计			类型	游戏
班 号	20L0220		学 号	120L022004	
所在院系	计算机与电子通信	学 期	2020 年秋季学期	任课教师	袁永峰
实验类型	综合设计型				

实验目的:

- 掌握程序设计的基本算法和简单数据结构基础，能够综合运用基本控制语句、算法和数据结构，以及自顶向下、逐步求精的模块化设计方法，能够设计具有一定规模的系统级C语言程序，提高系统编程能力；
- 针对计算相关的复杂工程问题，能够使用恰当的算法和数据结构，完成计算、统计、排序、检索、匹配等相关的软件系统的构造、测试与实现；
- 掌握常用的程序调试和测试方法。

实验要求:

- 采用自顶向下、逐步求精的模块化设计思想设计一个小型信息库管理系统，或者闯关式游戏程序。
- 要求解释说明采用了什么数据结构和算法，为什么选择这种数据结构或算法，系统实现过程中遇到了哪些问题，这些问题是如何解决的，还有什么问题尚未解决，今后打算从哪几个方面进行改进，本设计的亮点和难点在哪里，实验结果如何，有哪些收获和学习体会；
- 编写程序完成以下实验大作业内容并完成实验大作业反思报告。

实验内容:

设计一个四子棋游戏。

平台: Windows 10 + Visual Studio Code + Windows Terminal

工具: MinGW-W64 + CMake + git + Powershell 7

四子棋游戏规则:

四子棋的棋盘是垂直摆放。两名玩者轮流每次把一只棋子放进棋盘任何未全满的一行中，棋子会占据一行中最底未被占据的位置。两名玩者任何一方先以四只棋子在横，竖或斜方向联成一条直线，便可获胜，游戏亦结束。假如棋盘已完全被棋子填满，但仍未有任何一方成功把四只棋子成一直线，则成为和局。

功能要求

- (1) 自定义棋盘大小，进行键盘操作的四人四子棋游戏。
- (2) 实现流畅美观的命令行图形显示。
- (3) 历史记录功能，可以保存对局，并随时回顾之前的对局。
- (4) 回顾对局的时候，可以任意前进后退。

详细功能划分

- (1) 编写一个命令行图形函数集，用于命令行输出界面
- (2) 设计一个统一的菜单和用户输入跳转系统
- (3) 设计一个统一的、带有目录结构的文件读写函数集
- (4) 设计一个链表数据结构存储多条历史记录
- (5) 设计一套棋盘系统处理落子、判断胜负平局
- (6) 设计一套历史记录系统用于记录对局的步数、胜负。

程序运行后显示菜单，共三个选项

- (1) 双人游戏
- (2) 查看历史纪录
- (3) 退出

双人游戏界面包括自定义棋盘大小界面和游戏主界面。

查看历史记录界面包括

- (1) 历史记录条目选择界面
- (2) 单条历史记录查看主界面

实验环境：

操作系统：Windows 10

集成开发环境：Visual Studio Code + CMake + MinGW-W64 + git + Powershell 7

外部库：本程序为单独一份可执行文件。但在编译过程中运用了静态库链接。如果您想详细了解，请阅读文件夹下的「评分前请读！.pdf」文件

输入输出设计：

程序能够处理所有的非法输入数据和文件打开失败。

程序能够处理内存分配失败。

通过 `CreateResult` 结构体的设计，函数入口处的数据全部可以保证为有效数据。

部分输入采用阻塞式即时响应方法。用户体验和健壮性良好。

输入数据：

用户键盘选择菜单选项。设计为字符/字符串类型。包括：

初始界面菜单选择。接受输入：键盘按键 0、1、2

游戏主界面。接受输入：j、k、b 控制游戏进行。

历史记录查看界面。接受输入：j、k、r、b 控制选择菜单的上/下一项，确定和返回。

用户选择棋盘大小。设计为整形输入。

读取两个范围在 5~22 之内的有符号整数，作为棋盘大小。

文件输入读取历史记录。设计为整形/字符串结合。包括：

游戏日期（整形），棋盘大小（整形），用户名（字符型）等。

输出数据：

命令行输出：

游戏选择界面（若干）和游戏主界面。运用英文、汉字、各种制表符和特殊字符组成。

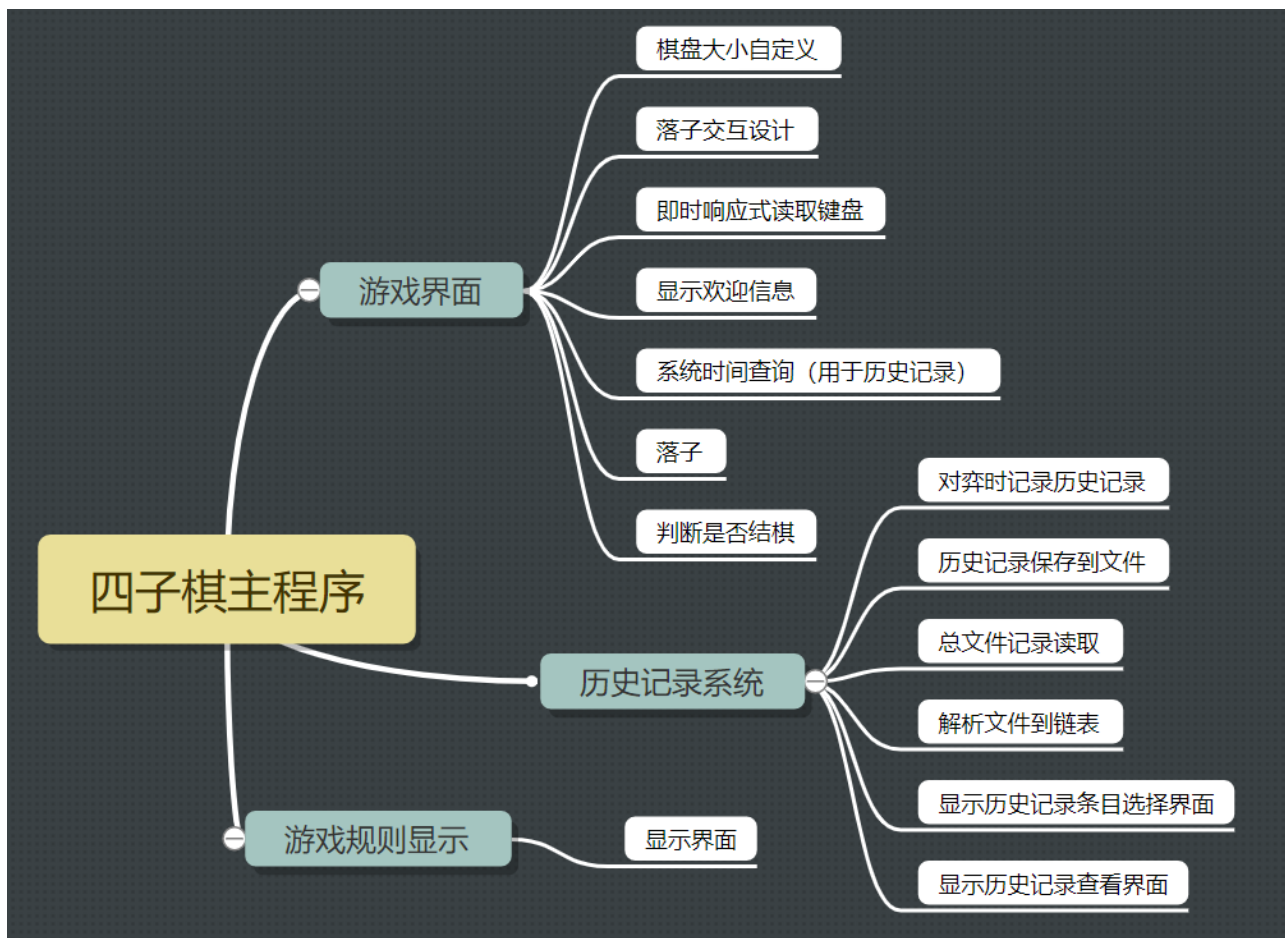
文件输出：

输出历史记录。设计为若干整形/字符串结合。和输入方面相同

系统设计与实现:

1. 系统功能模块划分

对系统进行自顶向下的模块分解，画出系统各个功能模块之间的结构图如下：



2. 函数功能和外部接口设计

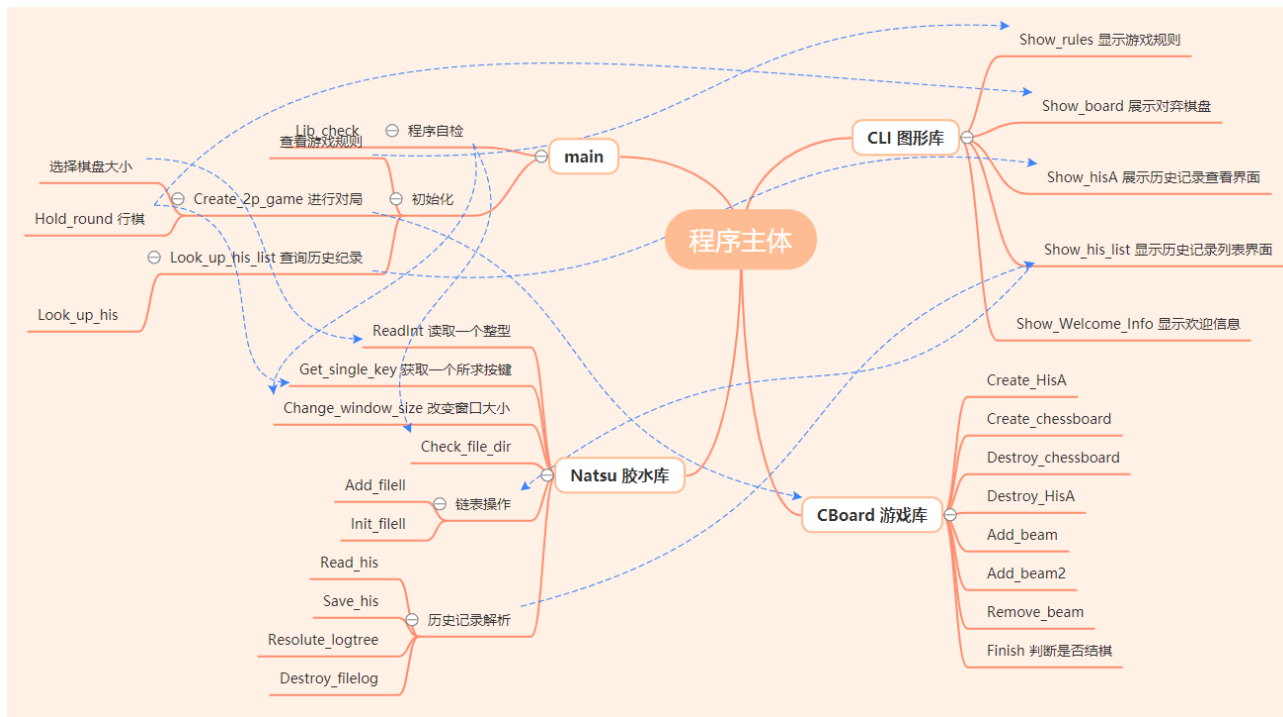
本系统总计设计了29个函数，每个函数的功能和接口设计如下表所示：

序号	函数名	函数功能	函数参数	函数返回值
1	ReadInt	手写的读 int	int *x	void
2	Get_single_key_input	阻塞式等待键盘输入字符，即时响应，并匹配 req 中的字符。匹配成功则返回其在 req 中的下标，从 0 开始。如果传入 _ 代表一个通配符，匹配除其余指明字符外的所有字符。	char *req	int
3	Change_window_size	改变命令行的窗口大小。	int height, int width	void
4	Check_file_dir	检查历史记录的目录结构。	void	void

5	Save_his	保存历史记录到文件。	HisA* his	void
6	Add_filell	链表的添加操作	FileLL *last, FILE *addfp	FileLL*
7	Init_filell	链表的初始化操作	void	FileLL*
8	Resolute_logtree	解析文件树到链表	FileLL* head	FileLL*
9	Destroy_filelog	销毁链表和其内存	FileLL* head	void
10	Read_his	从文件读历史记录到 HisA 结构体。	FileLL *logid, HisA *his	void
11	Show_rules	展示游戏规则	void	void
12	Show_board	展示对弈棋盘	Board *bd, int cursorpos, int goer	void
13	Show_hisA	展示历史记录查看界面	Board* bd, HisA *his, int cnt_pos	void
14	Show_his_list	显示历史记录列表界面	int sel	int
15	Show_Welcome_Info	显示欢迎信息。	void	void
16	Create_HisA	创建一条历史, [1] 写入当前时间 / [0] 时间初始为 NULL	int write_cur_time	CreateHisA*
17	Create_chessboard	创建一个棋盘 h by w	int height, int width, char *usrA, char *usrB, int idA, int idB	CreateResult*
18	Destroy_chessboard	删除一个棋盘并释放内存	CreateResult *result	void
19	Destroy_HisA	删除一条历史记录并释放内存	HisA *his	void
20	Add_beam	添加一个子 [1 成功 0 满格-失败]	Board *bd, int xpos, int color	int
21	Remove_beam	移除一个子	Board *bd, int xpos, int ypos	void
22	Add_beam2	HisView 专用 - 添加一个子	Board *bd, int xpos, int ypos, int color	void
23	Finish	判断是否结棋	Board *bd	int
24	Hold_round	行一步棋。	Board *bd, int cursorpos, int goer	int
25	Create_2p_game	创建一个双人游戏	void	void
26	Look_up_his	查询一条历史记录	FileLL* logid	void
27	Look_up_his_list	在历史记录菜单中选择	void	void
28	Welcome	初始的欢迎页面	void	void

29	Lib_check	[测试代码] 检查各个库的链接情况	void	void
----	-----------	-------------------	------	------

各个函数之间的调用关系如下所示：



3. 数据结构

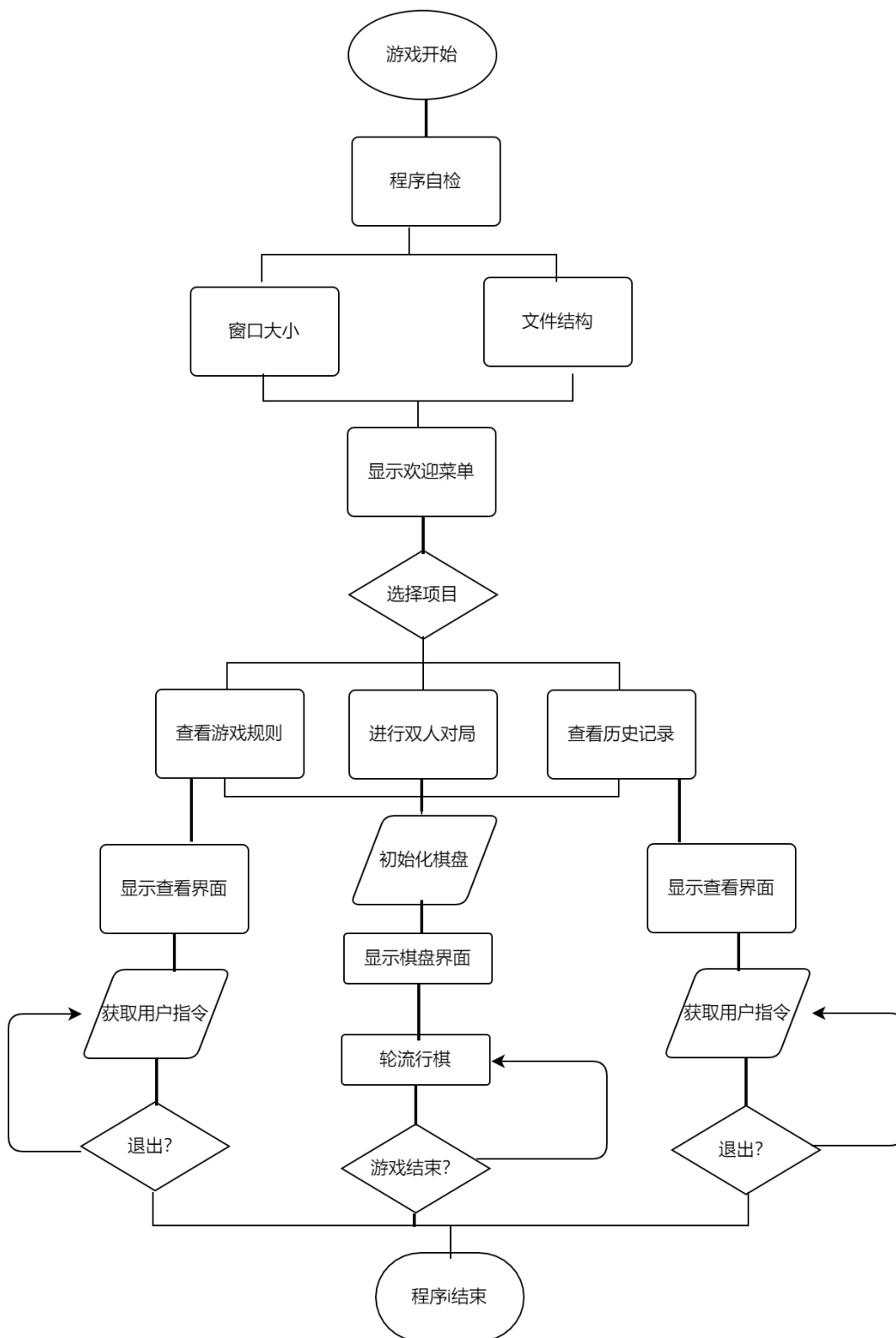
结构体数组 + 指针（结构体指针/指针数组/函数指针/动态数组等）
链表等动态数据结构

4. 算法

枚举/递推/迭代/分类统计
排序/查找
模糊匹配
文件操作
递归

5. 程序流程图

系统总体流程图如下：



实验过程中遇到的问题及解决方法与思路:

问题 1:

原因: Natsu 作为胶水库会被多方调用, 目录关系还是同层的, CMake 构建不通过。

把 natsu 单独分离编译行, 结果 ld 一直报错 undefined reference to 'Get_single_key_input'。

解决方法:

通过查阅 CMake 官方文档, 获知 CMakeLists.txt 需要添加:

```
add_library(natsu STATIC IMPORTED)
```

```
set_property(TARGET natsu PROPERTY IMPORTED_LOCATION ./src/natsu/libnatsu.a)
```

```
target_link_libraries(chess4 natsu)
```

此后问题解决。

问题 2:

复杂的库依赖导致编译器报错 redefinition of struct XXX。

解决方法:

搜索阅读 GCC 编译器文档后在 natsu.h 内添加一行 #pragma once 确保该文件只被包含一次。

问题 3:

程序测试未通过。查看运行信息发现内存异常退出

解决方法:

跟踪源代码, 发现 strcpy 时, 操作对象 (char*)CBoard->usrA 未分配空间。

遂修正代码。

问题 4:

cursorpos 变量每次都会初始化, 游戏体验极差

解决方法:

通过重构模块设计使得函数回传 cursorpos 变量的结束值。

问题 5:

测试游玩时发现没有判断平局的代码。

解决方法:

修改代码, 添加。

问题 6:

历史记录系统要求移除棋子, CBoard 库缺少对应功能。

解决方法:

编写 Remove_beam 函数, 修改历史记录结构体使其支持记录每步行棋位置。

测试用例和系统测试结果:

测试用例 1:

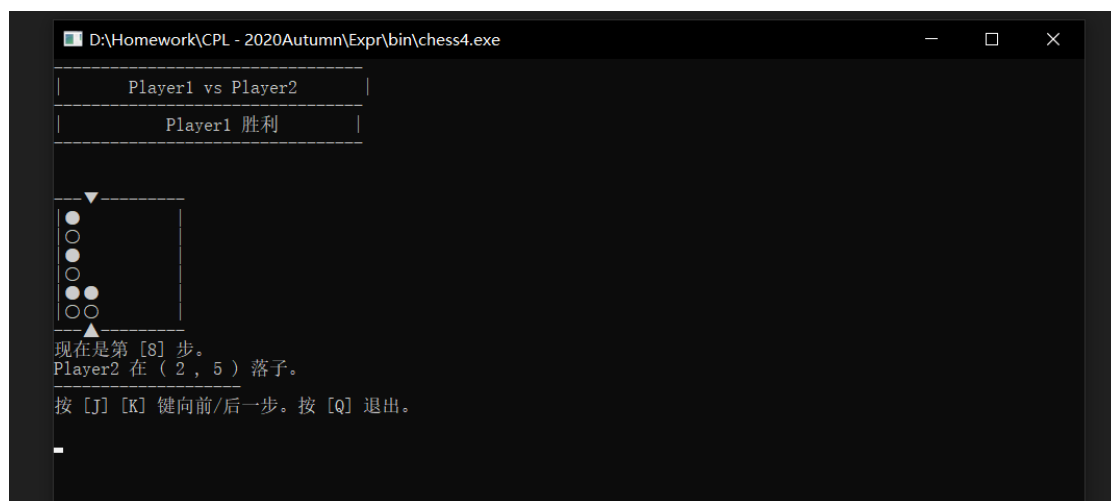
测试 1 测试历史记录功能。测试用例 1 附加在文件夹中，是名为 Histories 的文件夹。

测试目的:

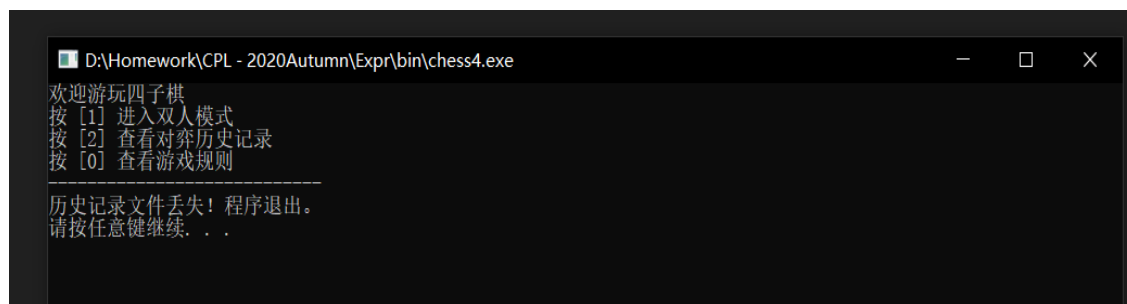
测试历史记录功能是否运行正常，对错误文件结构的处理能力。

测试方法:

1. 复制此文件夹到程序同级目录，后运行程序，使用查看历史记录功能查看历史记录。



2. 删除该文件夹的全部或一部分，测试程序是否有 bug。



3. 多次按 J、K 键，测试选择到边界时是否越界

结果：未出现越界。

测试成功，程序未出现异常结束情况。

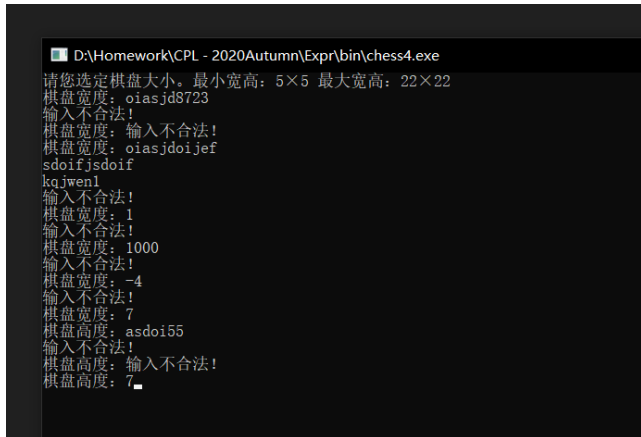
测试用例 2:

测试方法:

选择双人模式进行一局游戏。

测试目的:

测试程序是否能正常游戏。



```
D:\Homework\CPL - 2020Autumn\Expr\bin\chess4.exe
请您选定棋盘大小。最小宽高: 5×5 最大宽高: 22×22
棋盘宽度: oiasjd8723
输入不合法!
棋盘宽度: 输入不合法!
棋盘宽度: oiasjdoiief
sdoifjsdoif
kqjwenl
输入不合法!
棋盘宽度: 1
输入不合法!
棋盘宽度: 1000
输入不合法!
棋盘宽度: -4
输入不合法!
棋盘宽度: 7
棋盘高度: asdoi55
输入不合法!
棋盘高度: 输入不合法!
棋盘高度: 7
```



测试结果: 正常。

程序的全部源代码：

```
001 #include <stdio.h>
002 #include <string.h>
003 #include <math.h>
004 #include <windows.h>
005 #include <conio.h>
006 #include <time.h>
007
008 typedef struct tm __Sys_time;
009 // 棋盘类 struct 声明
010 // 历史记录
011 struct _HistoryA
012 {
013     int winner;
014     int tot_steps, height, width;
015     int *posx, *posy;
016     char *userA, *userB;
017     __Sys_time *game_time;
018     time_t std_fmt_time;
019 };
020 typedef struct _HistoryA HisA;
021 // 盘面
022 struct _Board
023 {
024     int **mat;
025     int height, width;
026     // 对弈
027     char *userA, *userB; // 用户名
028     int idA, idB; // 用户 Hash
029     int winner; // .. 0 / 1 / 2 / -1[平局]
030     // 历史
031     HisA *his;
032 };
033 typedef struct _Board Board;
034 // 创建 CBoard 的结果
035 struct _CreateResult
036 {
037     // 创建成功返回 1
038     // 内存空间不足返回 -1
039     int result;
040     Board bd;
041 };
042 typedef struct _CreateResult CreateResult;
043 // 创建历史记录的结果
044 struct _CreateHisA
045 {
046     int result;
047     HisA his;
048 };
```

```
049 typedef struct _CreateHisA CreateHisA;
050 // 文件链表
051 struct _FileLL
052 {
053     int id;
054     FILE *fp;
055     struct _FileLL *prev, *next;
056 };
057 typedef struct _FileLL FileLL;
058
059 // 函数如下
060
061 void HelloNatsu(void);
062
063 // 手写的读 int
064 void ReadInt(int *x);
065
066 // 阻塞式等待键盘输入字符，即时响应
067 // 并匹配 req 中的字符
068 // 匹配成功则返回其在 req 中的下标，从 0 开始
069 // 如果传入 _ 代表一个通配符，匹配除其余指明字符外的所有字符。
070 int Get_single_key_input(char *req);
071
072 // 改变命令行的窗口大小。
073 void Change_window_size(int height, int width);
074
075 // 检查历史记录的目录结构。
076 void Check_file_dir(void);
077
078 // 保存历史记录到文件。
079 void Save_his(HisA* his);
080
081 // 链表的添加操作
082 FileLL* Add_filell(FileLL *last, FILE *addfp);
083
084 // 链表的初始化操作
085 FileLL* Init_filell(void);
086
087 // 解析文件树到链表
088 FileLL* Resolute_logtree(FileLL* head);
089
090 // 销毁链表和其内存
091 void Destroy_filelog(FileLL *head);
092
093 // 从文件读历史记录到 HisA 结构体。
094 void Read_his(FileLL *logid, HisA *his);
095
096 void HelloCli(void);
097
```

```
098 // 展示游戏规则
099 void Show_rules(void);
100
101 // 展示对弈棋盘
102 void Show_board(Board *bd, int cursorpos, int goer);
103
104 // 展示历史记录查看界面
105 void Show_hisA(Board* bd, HisA *his, int cnt_pos);
106
107 // 显示历史记录列表界面
108 int Show_his_list(int sel);
109
110 // 显示欢迎信息。
111 void Show_Welcome_Info(void);
112
113 void HelloCBoard(void);
114
115 // 创建一条历史, [1] 写入当前时间 / [0] 时间初始为 NULL
116 CreateHisA* Create_HisA(int write_cur_time);
117
118 // 创建一个棋盘 h by w
119 CreateResult* Create_chessboard(int height, int width, char *usrA, char *usrB, int idA, int idB);
120
121 // 删除一个棋盘并释放内存
122 void Destroy_chessboard(CreateResult *result);
123
124 // 删除一条历史记录并释放内存
125 void Destroy_HisA(HisA *his);
126
127 // 添加一个子 [1 成功 || 0 满格-失败]
128 int Add_beam(Board *bd, int xpos, int color);
129
130 // 移除一个子
131 void Remove_beam(Board *bd, int xpos, int ypos);
132
133 // HisView 专用 - 添加一个子
134 void Add_beam2(Board *bd, int xpos, int ypos, int color);
135
136 // 判断是否结棋
137 int Finish(Board *bd);
138
139 void ReadInt(int *x)
140 {
141     (*x) = 0; int k = 1; char ch = 0;
142     while (ch < '0' || ch > '9')
143     {
144         ch = getchar();
145         if (ch == '-') k = -1;
146     }
```

```
147     while (ch >= '0' && ch <= '9')
148     {
149         (*x) = (*x) * 10 + ch - '0';
150         ch = getchar();
151     }
152     (*x) *= k;
153 }
154
155 int Get_single_key_input(char *req)
156 {
157     int reqlen = strlen(req);
158     int i, pos_default = -1;
159     char c;
160
161     for (i = 0; i < reqlen; i++)
162         if (req[i] == '_')
163             pos_default = i;
164
165     while (c = getch())
166     {
167         for (i = 0; i < reqlen; i++)
168             if ((c == req[i]) && (i != pos_default))
169                 return i;
170         if (pos_default != -1)
171             return pos_default;
172     }
173     return -1;
174 }
175
176 // 改变窗口大小 h by w
177 void Change_window_size(int height, int width)
178 {
179     char commands[30];
180     sprintf(commands, "mode con cols=%d lines=%d", height, width);
181     system(commands);
182     return;
183 }
184
185 // 输出 Hello Natsu 测试
186 void HelloNatsu(void)
187 {
188     puts("Hello Natsu!");
189     system("pause");
190     return;
191 }
192
193 void Check_file_dir(void)
194 {
195     system("md Histories 2>nul >nul");
```

```
196     system("if not exist Histories\\HList.txt echo a 2>Histories\\HList.txt");
197     system("md Histories\\Logs 2>nul >nul");
198     system("cls");
199     return;
200 }
201
202 FileLL* Add_filell(FileLL *last, FILE* addfp)
203 {
204     FileLL* add = (FileLL*)calloc(1, sizeof(FileLL));
205     if (add == NULL)
206     {
207         puts("程序内存不足！即将退出。");
208         system("pause");
209         exit(3);
210     }
211
212     add->prev = last;
213     last->next = add;
214     add->next = NULL;
215     add->fp = addfp;
216     add->id = last->id + 1;
217     return add;
218 }
219
220 void Save_his(HisA* his)
221 {
222     FILE *fplist = fopen("Histories/HList.txt", "a");
223     if (fplist == NULL)
224     {
225         puts("文件不存在！程序退出。");
226         system("pause"); exit(4);
227     }
228     fprintf(fplist, "log_%lld.c4log\n", his->std_fmt_time);
229     fclose(fplist);
230
231     char fname[30];
232     sprintf(fname, "Histories/Logs/log_%lld.c4log", his->std_fmt_time);
233
234     FILE *fphis = fopen(fname, "w");
235     if (fphis == NULL)
236     {
237         puts("文件不存在！程序退出。");
238         system("pause");
239         exit(4);
240     }
241
242     fprintf(fphis, "%lld\n", his->std_fmt_time);
243     fprintf(fphis, "%d %d %d\n",
244             his->game_time->tm_year + 1900, his->game_time->tm_mon + 1,
```

```
245         his->game_time->tm_mday);
246     fprintf(fphis, "%d %d %d\n",
247         his->game_time->tm_hour, his->game_time->tm_min,
248         his->game_time->tm_sec);
249     fprintf(fphis, "%s %s\n", his->userA, his->userB);
250     fprintf(fphis, "%d %d\n", his->height, his->width);
251     fprintf(fphis, "%d %d\n", his->winner, his->tot_steps);
252     int i, j = his->tot_steps;
253     for (i = 1; i <= j; i++)
254         fprintf(fphis, "%d %d\n", his->posx[i], his->posy[i]);
255     fclose(fphis);
256     return;
257 }
258
259 FileLL* Init_filell(void)
260 {
261     FileLL* ret = (FileLL*)calloc(1, sizeof(FileLL));
262     if (ret == NULL)
263     {
264         puts("程序内存不足! 即将退出。");
265         system("pause");
266         exit(3);
267     }
268
269     ret->next = ret->prev = NULL;
270     ret->fp = NULL;
271     ret->id = 1;
272     return ret;
273 }
274
275 void Read_his(FileLL *logid, HisA *his)
276 {
277     int i, j, a1[6];
278     fscanf(logid->fp, "%lld%d%d%d%d%d",
279         &(his->std_fmt_time), &a1[0], &a1[1], &a1[2],
280         &a1[3], &a1[4], &a1[5]);
281     his->game_time = gmtime(&(his->std_fmt_time));
282     his->game_time->tm_year += 1900;
283     his->game_time->tm_mon += 1;
284     fscanf(logid->fp, "%s%s", his->userA, his->userB);
285     fscanf(logid->fp, "%d%d%d%d",
286         &(his->height), &(his->width), &(his->winner), &(his->tot_steps));
287     j = his->tot_steps;
288     for (i = 1; i <= j; i++)
289         fscanf(logid->fp, "%d%d", &(his->posx[i]), &(his->posy[i]));
290     return;
291 }
292
293 FileLL* Resolute_logtree(FileLL *head)
```

```
294 {
295     char str[50],fname[35];
296     int i = 0;
297     FILE *fplog = NULL;
298     FileLL *tail = head;
299
300     FILE *fplist = fopen("Histories/HList.txt","r");
301     if (fplist == NULL)
302     {
303         puts("历史记录目录文件丢失，无法解析历史记录！程序退出。");
304         system("pause"); exit(4);
305     }
306
307     while (fgets(str, 50, fplist) != NULL)
308     {
309         i++;
310
311         strcpy(fname,"Histories/Logs/");
312         sscanf(str, " %s\n", fname + 15);
313
314         fplog = fopen(fname,"r");
315         if (fplog == NULL)
316         {
317             puts("历史记录文件丢失！程序退出。");
318             system("pause"); exit(11);
319         }
320         tail = Add_filell(tail, fplog);
321     }
322     return head;
323 }
324
325 void Destroy_filelog(FileLL *head)
326 {
327     if (head->next != NULL) Destroy_filelog(head->next);
328     fclose(head->fp);
329     free(head);
330     return;
331 }
332
333 void HelloCli()
334 {
335     HelloNatsu();
336     puts("Hello Cli!");
337     system("pause");
338     return;
339 }
340
341 void Show_rules()
342 {
```



```
343     system("cls");
344     puts("四子棋游戏规则：");
345     puts("四子棋的棋盘是垂直摆放。");
346     puts("两名玩者轮流每次把一只棋子放进棋盘任何未全满的一行中，棋子会占据一行中最底未被占据的位置。");
347     puts("两名玩者任何一方先以四只棋子在横，竖或斜方向联成一条直线，便可获胜，游戏亦结束。");
348     puts("假如棋盘已完全被棋子填满，但仍未有任何一方成功把四只棋子成一直线，则成为和局。");
349     puts("按 [1] 返回上个页面。");
350
351     Get_single_key_input("1");
352     return;
353 }
354
355 void Show_board(Board *bd, int cursorpos, int goer) // 0●
356 {
357     system("cls");
358     int i, j, ii, jj;
359
360     printf("-----\n");
361     printf("|    %10s vs %-10s    |\n", bd->userA, bd->userB);
362     printf("-----\n");
363     printf("\n\n");
364
365     putchar(' ');
366     for (i = 1; i <= bd->width; i++)
367     {
368         if (i != cursorpos) printf(" ");
369         else printf("↓");
370     }
371     printf("\n\n");
372
373     putchar('-');
374     for (i = 1; i <= bd->width; i++)
375     {
376         if (i == cursorpos)
377             printf("▼");
378         else
379             printf("--");
380     }
381     printf("-\n");
382     for (i = 1; i <= bd->height; i++)
383     {
384         putchar('|');
385         for (j = 1; j <= bd->width; j++)
386         {
387             switch (bd->mat[i][j])
388             {
389                 case 0:
390                     printf(" ");
```

```
391         break;
392     case 1:
393         printf("o");
394         break;
395     case 2:
396         printf("●");
397         break;
398     }
399 }
400 putchar('|'); putchar('\n');
401 }
402 putchar('-');
403 for (i = 1; i <= bd->width; i++)
404 {
405     if (i == cursorpos)
406         printf("▲");
407     else
408         printf("--");
409 }
410 printf("-\n");
411
412 printf("-----\n");
413 puts("按 [J] [K] 键切换落子位置向左/向右, 按 [B] 键落子。");
414 printf("现在是 %s 落子.\n", (goer == 1) ? bd->userA : bd->userB);
415
416 return;
417 }
418
419
420 void Show_hisA(Board* bd, HisA *his, int cnt_pos)
421 {
422     system("cls");
423     int i, j, ii, jj, cursorpos;
424
425     printf("-----\n");
426     printf("|    %10s vs %10s    |\n", his->userA, his->userB);
427     printf("-----\n");
428     if (his->winner == -1)
429         printf("|                平局                |\n");
430     else
431         printf("|    %10s 胜利    |\n",
432             (his->winner == 1) ? his->userA : his->userB);
433     printf("-----\n");
434     printf("\n\n");
435
436     cursorpos = his->posx[cnt_pos];
437
438     putchar('-');
439     for (i = 1; i <= bd->width; i++)
```

```
440     {
441         if (i == cursorpos)
442             printf("▼");
443         else
444             printf("--");
445     }
446     printf("-\n");
447     for (i = 1; i <= bd->height; i++)
448     {
449         putchar('|');
450         for (j = 1; j <= bd->width; j++)
451         {
452             switch (bd->mat[i][j])
453             {
454                 case 0:
455                     printf(" ");
456                     break;
457                 case 1:
458                     printf("o");
459                     break;
460                 case 2:
461                     printf("●");
462                     break;
463             }
464         }
465         putchar('|'); putchar('\n');
466     }
467     putchar('-');
468     for (i = 1; i <= bd->width; i++)
469     {
470         if (i == cursorpos)
471             printf("▲");
472         else
473             printf("--");
474     }
475     printf("-\n");
476
477     printf("现在是第 [%d] 步。 \n", cnt_pos);
478     printf("%s 在 ( %d , %d ) 落子。 \n",
479           (cnt_pos % 2 == 1) ? his->userA : his->userB,
480           his->posx[cnt_pos], his->posy[cnt_pos]);
481
482     printf("-----\n");
483     puts("按 [J] [K] 键向前/后一步。按 [Q] 退出。 \n");
484     return;
485 }
486
487 int Show_his_list(int sel)
488 {
```

```
489     system("cls");
490
491     char str[50];
492     int i = 0;
493     __Sys_time *systm;
494     time_t std_fmt_systm;
495
496     FILE *fplist = fopen("Histories/HList.txt","r");
497     if (fplist == NULL)
498     {
499         puts("文件不存在！程序退出。");
500         system("pause"); exit(4);
501     }
502
503     printf("-----\n");
504     printf("|      选择历史记录:      |\n");
505     printf("-----\n");
506
507     while (fgets(str, 50, fplist) != NULL)
508     {
509         i++;
510         sscanf(str, "log_%lld.c4log", &std_fmt_systm);
511         systm = gmtime(&std_fmt_systm);
512
513         printf("[A 类记录] 游戏时间: %d/%d/%d %02d:%02d:%02d (英国 0 时区)",
514             systm->tm_year + 1900, systm->tm_mon + 1, systm->tm_mday,
515             systm->tm_hour, systm->tm_min, systm->tm_sec);
516
517         if (i == sel)
518             printf("          [当前选中]\n");
519         else
520             printf("\n");
521         printf("-----\n");
522     }
523
524     printf("\n\n 按 [J] [K] 上/下选择。按 [B] 确定。按 [Q] 退出。 \n");
525     printf("-----\n");
526     return i;
527 }
528
529 void Show_Welcome_Info(void)
530 {
531     system("cls");
532     puts("欢迎游玩四子棋");
533     puts("按 [1] 进入双人模式");
534     puts("按 [2] 查看对弈历史记录");
535     puts("按 [0] 查看游戏规则");
536     puts("-----");
537     return;
```

```
538 }
539
540 void HelloCBoard()
541 {
542     HelloNatsu();
543     printf("Hello CBoard!\n");
544     system("pause");
545     return;
546 }
547
548 CreateResult* Create_chessboard(int height, int width, char *usrA, char *usrB, int idA, int idB)
549 {
550     int i, j, ii, jj;
551     CreateResult *result = (CreateResult*)calloc(1, sizeof(CreateResult));
552
553     if (result == NULL)
554     {
555         printf("无法获取最低所需的内存进行游戏! \n 程序退出。 \n");
556         system("pause");
557         exit(2);
558     }
559
560     CreateHisA *chis = Create_HisA(1);
561     if (chis->result != 1)
562     {
563         result->result = -1;
564         return result;
565     }
566     result->bd.his = &(chis->his);
567
568     result->bd.height = height;
569     result->bd.width = width;
570     result->bd.his->height = height;
571     result->bd.his->width = width;
572     result->bd.mat = (int**)calloc(height + 2, sizeof(int *));
573
574     if (result->bd.mat == NULL)
575     {
576         result->result = -1;
577         return result;
578     }
579
580     for (i = 0; i <= height + 1; i++)
581     {
582         result->bd.mat[i] = (int*)calloc(width + 2, sizeof(int));
583         if (result->bd.mat[i] == NULL)
584         {
585             result->result = -1;
586             return result;
```

```
587     }
588 }
589
590 result->bd.userA = (char*)calloc(10, sizeof(char));
591 result->bd.userB = (char*)calloc(10, sizeof(char));
592 if (result->bd.userA == NULL || result->bd.userB == NULL)
593 {
594     result->result = -1;
595     return result;
596 }
597
598 for (i = 0; i <= height; i++)
599     for (j = 0; j <= width; j++)
600         result->bd.mat[i][j] = 0;
601
602 strcpy(result->bd.userA, usrA);
603 strcpy(result->bd.his->userA, usrA);
604 strcpy(result->bd.userB, usrB);
605 strcpy(result->bd.his->userB, usrB);
606 result->bd.idA = idA;
607 result->bd.idB = idB;
608 result->bd.winner = 0;
609
610 result->result = 1;
611 return result;
612 }
613
614 int Finish(Board *bd)
615 {
616     int i, j, h, w, ii, jj;
617     HisA *his = bd->his;
618     int **ma = bd->mat;
619     h = bd->height;
620     w = bd->width;
621
622     // ----
623     for (i = 1; i <= h; i++)
624         for (j = 1; j <= w - 3; j++)
625             if (ma[i][j] != 0)
626                 {
627                     int ok = 1;
628                     for (ii = 1; ii <= 3; ii++)
629                         if (ma[i][j + ii] != ma[i][j])
630                             ok = 0;
631                     if (ok == 1)
632                     {
633                         his->winner = ma[i][j];
634                         return ma[i][j];
635                     }
636                 }
```

```
636     }
637     // ||||
638     for (i = 1; i <= w; i++)
639         for (j = 1; j <= h - 3; j++)
640             if (ma[j][i] != 0)
641             {
642                 int ok = 1;
643                 for (ii = 1; ii <= 3; ii++)
644                     if (ma[j + ii][i] != ma[j][i])
645                         ok = 0;
646                 if (ok == 1)
647                 {
648                     his->winner = ma[j][i];
649                     return ma[j][i];
650                 }
651             }
652     // ////
653     for (i = 1; i <= h - 3; i++)
654         for (j = 4; j <= w; j++)
655             if (ma[i][j] != 0)
656             {
657                 int ok = 1;
658                 for (ii = 1; ii <= 3; ii++)
659                     if (ma[i + ii][j - ii] != ma[i][j])
660                         ok = 0;
661                 if (ok == 1)
662                 {
663                     his->winner = ma[i][j];
664                     return ma[i][j];
665                 }
666             }
667     // \\\ /
668     for (i = 1; i <= h - 3; i++)
669         for (j = 1; j <= w - 3; j++)
670             if (ma[i][j] != 0)
671             {
672                 int ok = 1;
673                 for (ii = 1; ii <= 3; ii++)
674                     if (ma[i + ii][j + ii] != ma[i][j])
675                         ok = 0;
676                 if (ok == 1)
677                 {
678                     his->winner = ma[i][j];
679                     return ma[i][j];
680                 }
681             }
682
683     return 0;
684 }
```

```
685
686 int Add_beam(Board *bd, int xpos, int color)
687 {
688     HisA *his = bd->his;
689     if (bd->mat[1][xpos] != 0)
690         return 0;
691
692     int ypos = bd->height;
693     while (bd->mat[ypos][xpos] != 0)
694         ypos--;
695
696     bd->mat[ypos][xpos] = color;
697
698     his->tot_steps++;
699     int tot = his->tot_steps;
700     his->posx[tot] = xpos;
701     his->posy[tot] = ypos;
702     return 1;
703 }
704
705 void Destroy_chessboard(CreateResult *result)
706 {
707     Destroy_HisA(result->bd.his);
708     int i, j;
709     free(result->bd.userA);
710     free(result->bd.userB);
711     for (i = 0; i <= result->bd.height + 1; i++)
712         free(result->bd.mat[i]);
713     free(result->bd.mat);
714     free(result);
715     return;
716 }
717
718 CreateHisA* Create_HisA(int write_cur_time)
719 {
720     CreateHisA* ret = (CreateHisA*)calloc(1, sizeof(CreateHisA));
721
722     if (ret == NULL)
723     {
724         printf("无法获取最低所需的内存进行游戏! \n 程序退出。 \n");
725         system("pause");
726         exit(2);
727     }
728
729     if (write_cur_time == 0)
730         ret->his.game_time = NULL;
731     else
732     {
733         time_t std_format_time;
```



```
734     time(&std_format_time);
735     ret->his.game_time = gmtime(&std_format_time);
736     ret->his.std_fmt_time = std_format_time;
737 }
738
739 ret->his.userA = (char*)calloc(10, sizeof(char));
740 ret->his.userB = (char*)calloc(10, sizeof(char));
741 if (ret->his.userA == NULL || ret->his.userB == NULL)
742 {
743     ret->result = -1;
744     return ret;
745 }
746
747 ret->his.posx = (int*)calloc(500, sizeof(int));
748 ret->his.posy = (int*)calloc(500, sizeof(int));
749
750 if (ret->his.posx == NULL || ret->his.posy == NULL)
751 {
752     ret->result = -1;
753     return ret;
754 }
755
756 ret->his.tot_steps = 0;
757 ret->his.winner = 0;
758
759 ret->result = 1;
760 return ret;
761 }
762
763 void Destroy_HisA(HisA *his)
764 {
765     free(his->posx);
766     free(his->posy);
767     free(his->userB);
768     free(his->userA);
769     return;
770 }
771
772 void Remove_beam(Board *bd, int xpos, int ypos)
773 {
774     bd->mat[ypos][xpos] = 0;
775     return;
776 }
777
778 void Add_beam2(Board *bd, int xpos, int ypos, int color)
779 {
780     bd->mat[ypos][xpos] = color;
781     return;
782 }
```

```
783
784 // 行一步棋。
785 int Hold_round(Board *bd, int cursorpos, int goer);
786
787 // 创建一个双人游戏
788 void Create_2p_game(void);
789
790 // 查询一条历史记录
791 void Look_up_his(FileLL* logid);
792
793 // 在历史记录菜单中选择
794 void Look_up_his_list(void);
795
796 // 初始的欢迎页面
797 void Welcome(void);
798
799 // [测试代码] 检查各个库的链接情况
800 void Lib_check(void);
801
802 int main()
803 {
804     Change_window_size(110,40);
805     Check_file_dir();
806     Welcome();
807
808     return 0;
809 }
810
811
812 int Hold_round(Board *bd, int cursorpos, int goer)
813 {
814     int call_back = 0;
815     while (1)
816     {
817         Show_board(bd, cursorpos, goer);
818         call_back = Get_single_key_input("jkb");
819
820         switch (call_back)
821         {
822             case 0:
823                 if (cursorpos > 1)
824                     cursorpos--;
825                 break;
826             case 1:
827                 if (cursorpos < bd->width)
828                     cursorpos++;
829                 break;
830             case 2:
831                 if (Add_beam(bd, cursorpos, goer) == 1)
```

```
832         return cursorpos;
833     break;
834 }
835 }
836 }
837
838 void Create_2p_game(void)
839 {
840     int w, h, i, j;
841     int cursorpos, goer;
842     int win_status;
843     int call_back;
844
845     game_2p_start:
846
847     w = h = -1;
848     cursorpos = 1; goer = 0;
849     win_status = 0;
850
851     system("cls");
852     puts("请您选定棋盘大小。最小宽高: 5×5 最大宽高: 22×22");
853
854     printf("棋盘宽度: ");
855     scanf("%d", &w);
856     while (w < 5 || w > 22)
857     {
858         printf("输入不合法! \n 棋盘宽度: ");
859         ReadInt(&w);
860     }
861     printf("棋盘高度: ");
862     scanf("%d", &h);
863     while (h < 5 || h > 22)
864     {
865         printf("输入不合法! \n 棋盘高度: ");
866         ReadInt(&h);
867     }
868
869     CreateResult *res;
870     res = Create_chessboard(h, w, "Player1", "Player2", 1001, 1002);
871
872     if (res->result != 1)
873     {
874         puts("创建棋盘时遇到错误, 程序退出!");
875         system("pause");
876         exit(1);
877     }
878
879     puts("对局开始!");
880     system("pause");
```

```
881
882     while (win_status == 0)
883     {
884         goer ++;
885         if (goer == 3) goer = 1;
886
887         cursorpos = Hold_round(&(res->bd), cursorpos, goer);
888
889         win_status = Finish(&(res->bd));
890         if (res->bd.his->tot_steps == res->bd.height * res->bd.width)
891         {
892             win_status = -1;
893             break;
894         }
895     }
896
897     Show_board(&(res->bd), cursorpos, goer);
898     if (win_status == -1)
899         printf("游戏结束! 平局。\\n");
900     else
901         printf("游戏结束! 胜者是 %s", (goer == 1) ? res->bd.userA : res->bd.userB);
902
903     res->bd.his->winner = win_status;
904
905     puts("您要保存本次对局的历史记录吗? 按 [Y] 保存, 按 [N] 不保存。");
906     call_back = Get_single_key_input("yn");
907     if (call_back == 0)
908         Save_his(res->bd.his);
909
910     Destroy_chessboard(res);
911
912     puts("按 [1] 退出。按 [4] 再来一局。");
913     call_back = Get_single_key_input("14");
914     if (call_back == 0) return;
915     else goto game_2p_start;
916 }
917
918 void Look_up_his(FileLL* logid)
919 {
920     system("cls");
921     CreateHisA *reshis = Create_HisA(0);
922
923     if (reshis->result != 1)
924     {
925         puts("创建历史记录对象时遇到错误, 程序退出! ");
926         system("pause");
927         exit(6);
928     }
929 }
```

```
930     Read_his(logid, &(reshis->his));
931     HisA *hhis = &(reshis->his);
932
933     CreateResult *res = Create_chessboard(hhis->height, hhis->width,
934         hhis->userA, hhis->userB, 101, 102);
935
936     if (res->result != 1)
937     {
938         puts("创建棋盘时遇到错误, 程序退出!");
939         system("pause");
940         exit(1);
941     }
942
943     int call_back, cnt_pos = 1;
944     Add_beam2(&(res->bd), hhis->posx[1], hhis->posy[1], 1);
945     while (1)
946     {
947         Show_hisA(&(res->bd), hhis, cnt_pos);
948         call_back = Get_single_key_input("jkq");
949
950         switch (call_back)
951         {
952             case 0:
953                 if (cnt_pos > 1)
954                 {
955                     Remove_beam(&(res->bd), hhis->posx[cnt_pos], hhis->posy[cnt_pos]);
956                     cnt_pos--;
957                 }
958                 break;
959             case 1:
960                 if (cnt_pos < hhis->tot_steps)
961                 {
962                     cnt_pos++;
963                     Add_beam2(&(res->bd), hhis->posx[cnt_pos],
964                         hhis->posy[cnt_pos], (cnt_pos % 2) ? 1 : 2);
965                 }
966                 break;
967             case 2:
968                 goto Look_up_his_end;
969         }
970     }
971     Look_up_his_end:
972     return;
973 }
974
975 void Look_up_his_list(void)
976 {
977     FileLL *head = Init_filell();
978     Resolute_logtree(head);
```

```
979     head = head->next;
980
981     char str[100];
982
983     int call_back, sel = 1, log_num, i;
984     while (1)
985     {
986         log_num = Show_his_list(sel);
987
988         call_back = Get_single_key_input("jkbq");
989         switch (call_back)
990         {
991             case 0:
992                 if (sel > 1)
993                 {
994                     sel--;
995                     head = head->prev;
996                 }
997                 break;
998             case 1:
999                 if (sel < log_num)
1000                 {
1001                     sel++;
1002                     head = head->next;
1003                 }
1004                 break;
1005             case 2:
1006                 if (head == NULL)
1007                 {
1008                     printf("历史记录解析错误！程序退出。\\n");
1009                     system("pause"); exit(12);
1010                 }
1011                 Look_up_his(head);
1012                 goto lklist_sel_end;
1013             case 3:
1014                 goto lklist_sel_end;
1015         }
1016     }
1017
1018     lklist_sel_end:
1019     while (head->prev != NULL)
1020         head = head->prev;
1021     Destroy_filelog(head);
1022     return;
1023 }
1024
1025 void Welcome(void)
1026 {
1027     fnt:
```

```
1028     Show_Welcome_Info();
1029
1030     int keyid = Get_single_key_input("012");
1031     switch (keyid)
1032     {
1033         case 0:
1034             Show_rules();
1035             goto fnt; // 我偏要用 goto。
1036         case 1:
1037             Create_2p_game();
1038             goto fnt;
1039         case 2:
1040             Look_up_his_list();
1041             goto fnt;
1042     }
1043 }
1044
1045 void Lib_check(void)
1046 {
1047     printf("-----Library Check Starts-----\n");
1048     HelloNatsu();
1049     HelloCBoard();
1050     HelloCli();
1051     system("cls");
1052 }
```

分析总结、收获和体会:

优点:

函数分离度高,可复用性强。

交互设计科学、直观、方便。

程序功能直观,设计人性化。

使用了成熟的工具链完成作业:

Visual Studio Code+git+CMake+MinGW-W64

提高效率。

有统一的代码风格和命名规范。

创新之处:

设计了模糊匹配输入的函数,并支持即时响应的输入。提高游戏体验。

使用多页面设计,方便交互和用户理解。

错误处理完整,运行流畅

不足之处:

程序完成度欠缺,起初设计了账号系统和人机对弈,碍于时间有限无法实现。

需要改进的地方:

缺少 GUI 图形界面和音效,游戏体验有待提升。

收获与学习体会:

学会了 CMake 和 git 的基本使用方法。
提高了功能的系统设计能力和模块分离结构能力。
提高了大规模代码编写和测试的能力。
对 C 语言及其编码技巧有了更深的理解。

自我评价:	是	否
程序运行是否无 bug?	√	
是否在撰写报告之前观看了 spoc 里的代码规范视频?	√	
程序代码是否符合代码规范(对齐与缩进, 有必要的注释)?	√	
是否按模块化要求进行了程序设计, 系统功能是否完善?	√	
是否是独立完成?	√	

自我评语:
作业完成情况良好, 态度认真, 时间投入多, 收获颇丰!

报告完成日期: 2020 年 12 月 31 日