**程序设计文档**

**国际象棋**

**目录**

**1. 基本信息 ……2**

**2. 需求分析 ……2**

**3. 功能实现 ……2**

**3.1 行棋规则 ……2**

**3.1.1 简易行棋规则 ……2**

**3.1.2 特殊行棋规则 ……5**

**3.2 鼠标点击操控 ……11**

**3.3 可行域显示 ……12**

**3.4 和棋和认输 ……13**

**3.4.1 和棋 ……13**

**3.4.2 认输 ……14**

**3.5 时间 ……14**

**3.5.1 游戏时间 ……14**

**3.5.2 每回合倒计时 ……14**

**4. 程序流程图 ……16**

**5. 函数和变量说明 ……17**

**1. 基本信息**

本程序主要实现国际象棋有关功能。

本程序的编程语言为C语言。

本程序的编译环境为Visual Studio 2022。

为更好地进行图形编程，本程序使用了图形编程工具EasyX, 版本为EasyX\_20220116。

**2. 需求分析**

为使得玩家能够轻松进行游戏相关的操作，本程序需要具有完整的图形界面。

本程序要实现国际象棋的有关规则。包括简易行棋规则、特殊行棋规则。简易行棋规则为各个棋子的移动，特殊行棋规则则有吃过路兵、兵升变、王车易位。

为了更加方便玩家的操作，本程序需要支持鼠标点击操控。还需要支持可行域显示，即支持单击选中棋子并显示可移动的棋盘格位置。

为了增加玩家的可选择性，本程序需要支持和棋和认输。即一方点击和棋图标并确认和棋，另一方同意。游戏结束，平局。一方点击认输图标并确认认输。游戏结束，认输一方负。

为了方便玩家控制时间，本程序需要支持游戏时间显示，以及每回合倒计时的显示。

本程序还需支持判断并显示游戏结果。

**3. 功能实现**

**3.1 行棋规则**

**3.1.1 简易行棋规则**

本程序的简易行棋规则包括：

1. 王（K）：横、直、斜都可以走，但每次限走一步。

2. 后（Q）：横、直、斜都可以走，移动步数不受限制，但不能转向或越过其他棋子。

3. 车（R）：横、竖均可以走，步数不受限制，不能斜走。除王车易位外不能越子。

4. 象（B）：只能斜走。格数不限，不能越子。因此白格象只能在白格走动，黑格象只能在黑格走动。

5. 马（N）：每步棋先横走或直走一格，然后再往外斜走一格；或者先斜走一格，最后再往外横走或竖走一格（即走“日”字）。可以越子，没有中国象棋中的“蹩马腿”限制。

6. 兵（P）：只能向前直走，从第二步开始每次只能走一格。但走第一步时，可以走一格或两格。兵的吃子方法与行棋方向不一样，它是直走斜吃，即如果兵的斜进一格内有对方棋子，就可以吃掉它而占据该格。

实现这些基本规则的总体思路是

1. 判断鼠标点击的是哪个棋子；

2. 根据这个棋子的性质，找出这个棋子所有可移动的棋盘格位置；

3. 判断鼠标点击的棋盘格是否是可移动的棋盘格，若是则移动，若不是则重新开始。

下面以较为复杂的“后”为例进一步说明。



白方的一个回合

int wQueen(int a[2])

{

if (x[0] == a[0] && y[0] == a[1]) //判断鼠标单击位置是否为白方王后，若是则执行

{

for (i = 1; i <= 7; i++) //后每个坐标最多移动7，这里从近（1）到远（7）

if (a[0] - i >= 1 && a[0] - i <= 8) //考虑后向左移动，移动坐标不得超过棋盘范围（1-8）

{

temp[0] = a[0] - i; //将假设移动后的横纵坐标写入二维数组temp

temp[1] = a[1];

wBlock(temp); //调用wBlock函数，判断temp处是否有白棋，有则mark=1

if (mark == 0)

{

D[n][0] = temp[0]; //没有阻碍则将temp的横纵坐标写入可行域D

D[n][1] = temp[1];

n++;

}

if (mark == 1) //mark=1说明有白棋，无法再向远移动，因而退出循环

break;

bBlock(temp); //调用bBlock函数，判断temp处是否有黑棋，有则mark=1

if (mark == 1) //mark=1说明有黑棋，无法再向远移动，因而退出循环

break;

}

……

for (n = 0; n <= 63; n++)

{

//在可移动的棋盘格上画绿色圆圈

if (D[n][0] >= 1 && D[n][0] <= 8 && D[n][1] >= 1 && D[n][1] <= 8)

ellipse(200 + (D[n][0] - 1) \* le, 50 + (D[n][1] - 1) \* wi, 200 + D[n][0] \* le, 50 + D[n][1] \* wi);

}

click(); //获取第二次点击

x[1] = xm;

y[1] = ym;

for (n = 0; n <= 63; n++)

{

if (x[1] == D[n][0] && y[1] == D[n][1]) //若第二次点击是可移动的棋盘格

{

a[0] = D[n][0]; //横坐标移动

a[1] = D[n][1]; //纵坐标移动

sign = 1; //移动成功标志

break; //移动成功退出循环

}

}

wEat(a);

}

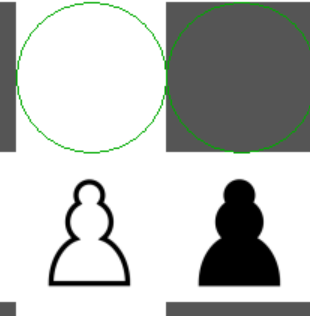
return 0;

}

**3.1.2 特殊行棋规则**

本程序的特殊行棋规则包括：

**1. 吃过路兵：**如果对方的兵第一次行棋且直进两格，刚好形成本方有兵与其横向紧贴并列，则本方的兵可以立即斜进，把对方的兵吃掉，并视为一步棋。这个动作必须立刻进行，缓着后无效。

****

本程序的吃过路兵实现：先定义了一些变量

//用于判断吃过路兵是否可行，1表示可以，0表示不可以

int wEn\_passant = 0, wEn\_passantx = 0, wEn\_passanty = 0;

int bEn\_passant = 0, bEn\_passantx = 0, bEn\_passanty = 0;

以白方吃黑方的过路兵为例。黑方任何一兵前进两格后，该兵横纵坐标分别记为wEn\_passantx、wEn\_passanty，且在下一个白方回合wEn\_passant = 1，白方士兵如果左或右的横纵坐标为wEn\_passantx、wEn\_passanty，则可吃过路兵。具体说明如下：

if (wEn\_passant == 1) //如果黑方士兵上一回合前进两步

{

if (a[0] - 1 == wEn\_passantx && a[1] == wEn\_passanty) //判断左边是否为该士兵

{

D[n][0] = a[0] - 1; //若是则记入可行域D

D[n][1] = a[1] - 1;

}

if (a[0] + 1 == wEn\_passantx && a[1] == wEn\_passanty) //判断左边是否为该士兵

{

D[n][0] = a[0] + 1; //若是则记入可行域D

D[n][1] = a[1] - 1;

}

}

……

if (x[1] == wEn\_passantx && y[1] == wEn\_passanty - 1)

{

for (p = 0; p < 8; p++)

{

if (wEn\_passantx == bP[p][0] && wEn\_passanty == bP[p][1])

{

bP[p][0] = -1; //黑方过路兵被吃掉

bP[p][1] = -1;

wEn\_passant = 2;

}

}

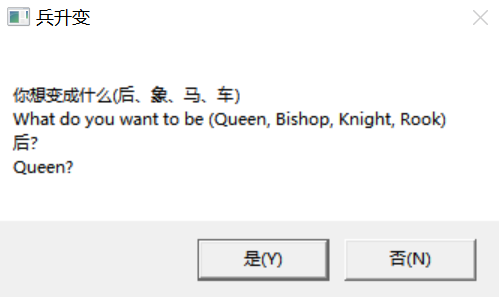
}

……

**2. 兵升变：**

本方任何一个兵直进达到对方底线时，即可升变为除“王”和“兵”以外的任何一种棋子，可升变为“后”、“车”、“马”、“象”，不能不变。这被视为一步棋。

本程序通过对话框询问操作者，实现了兵可以升变为“后”、“车”、“马”、“象”中的任何一个。



if (a[1] == 1)

{

yn3 = MessageBox(NULL, "你想变成什么(后、象、马、车）\nWhat do you want to be (Queen, Bishop, Knight, Rook)\n后？\nQueen?", "兵升变", MB\_YESNO | MB\_SYSTEMMODAL);

if (yn3 == 6)

{

wQp[wn][0] = a[0]; //白棋兵升变后坐标

wQp[wn][1] = a[1];

wn++;

a[0] = -1; a[1] = -1; //原兵被舍弃

}

else

{

yn3 = MessageBox(NULL, "你想变成什么(后、象、马、车）\nWhat do you want to be (Queen, Bishop, Knight, Rook)\n象？\nBishop?", "兵升变", MB\_YESNO | MB\_SYSTEMMODAL);

if (yn3 == 6)

{

wBp[wn][0] = a[0];

wBp[wn][1] = a[1];

wn++;

a[0] = -1; a[1] = -1;

}

else

{

yn3 = MessageBox(NULL, "你想变成什么(后、象、马、车）\nWhat do you want to be (Queen, Bishop, Knight, Rook)\n马？\nKnight?", "兵升变", MB\_YESNO | MB\_SYSTEMMODAL);

if (yn3 == 6)

{

wNp[wn][0] = a[0];

wNp[wn][1] = a[1];

wn++;

a[0] = -1; a[1] = -1;

}

else

{

MessageBox(NULL, "你想变成什么(后、象、马、车）\nWhat do you want to be (Queen, Bishop, Knight, Rook)\n车？\nRook?", "兵升变", MB\_OK | MB\_SYSTEMMODAL);

wRp[wn][0] = a[0];

wRp[wn][1] = a[1];

wn++;

a[0] = -1; a[1] = -1;

}

}

}

}



**3. 王车易位**

每局棋中，双方各有一次机会，让王朝车的方向移动两格，然后车越过王，放在与王紧邻的一格上，作为王执行的一步棋。

本程序考虑了王车之前是否移动过（王或车移动后无法王车易位）





本程序的王车易位实现：先定义了一些变量

//用于判断王车易位是否可行，1表示可以，0表示不可以

int wCastling = 1, wCastling1 = 1, wCastling2 = 1;

int bCastling = 1, bCastling1 = 1, bCastling2 = 1;

以白方王车易位为例，如果王移动，wCastling = 0，车1移动，wCastling1 = 0, 车2移动，wCastling2 = 0。最后只有同时满足wCastling = 1和wCastling1或wCastling2 = 1，才能实现王与车1或车2的王车易位。具体说明如下：

if (wCastling == 1) //王未移动过

{

if (wCastling1 == 1) //车1未移动过

{

for (i = 1; i < 4; i++)

{

temp[0] = a[0] - i;

temp[1] = a[1];

wBlock(temp);

if (mark == 1)

break;

bBlock(temp);

if (mark == 1)

break;

}

if (mark == 0) //王和车之间没有棋子

{

D[n][0] = a[0] - 2; //记入王的可行域

D[n][1] = a[1];

n++;

}

}

……



**3.2 鼠标点击操控**

本程序的“鼠标点击操控”具体是指鼠标左键单击进行操控，该功能通过本程序的click函数的一部分实现。具体说明如下：

if (peekmessage(&m, EM\_MOUSE | EM\_KEY))

{

if (m.message == WM\_LBUTTONDOWN)

{

mx = m.x;

my = m.y;

//获取离鼠标最近的点的坐标信息

for (i = 1; i <= 8; i++)

{

for (j = 1; j <= 8; j++)

{

if (abs(m.x + le / 2 - i \* le - 200) < 50 && abs(m.y + wi / 2 - j \* wi - 50) < 50)

{

xm = i;

ym = j;

}

}

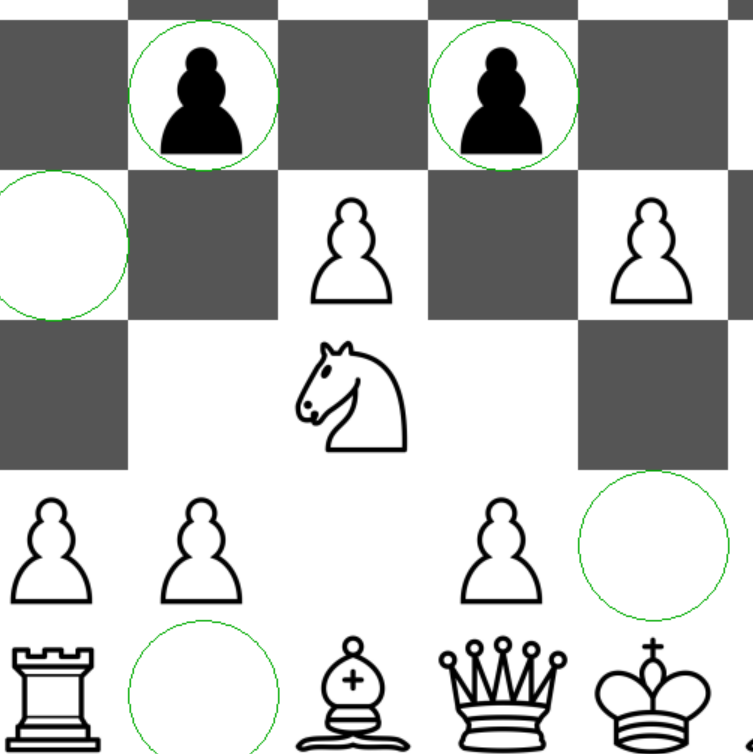
}

}

}

其中，m为消息结构体，peekmessage函数从m消息结构体中获取信息，有信息则返回1。如果该消息为鼠标左键单击，则将左键单击的位置化为坐标，储存为xm，ym，退出click。

**3.3 可行域显示**

****

在实现行棋规则后，可行域D的所有坐标都已知道，执行以下代码即可做到可行域的显示。

for (n = 0; n <= 63; n++)

{

if (D[n][0] >= 1 && D[n][0] <= 8 && D[n][1] >= 1 && D[n][1] <= 8)

ellipse(200 + (D[n][0] - 1) \* le, 50 + (D[n][1] - 1) \* wi, 200 + D[n][0] \* le, 50 + D[n][1] \* wi);

}

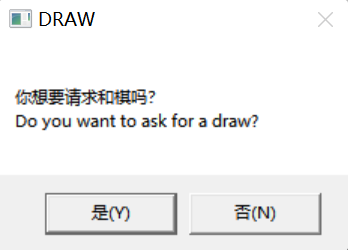
其中，ellipse是easyx.h的一个函数，表示画一个不填充的圆。

这样，就做到了可行域显示。

**3.4 和棋和认输**

****

**3.4.1 和棋**



和棋功能通过对话框实现，具体说明如下：

yn1 = MessageBox(NULL, "你想要请求和棋吗？\nDo you want to ask for a draw?", "DRAW", MB\_YESNO | MB\_SYSTEMMODAL);

if (yn1 == 6)

{

yn1 = MessageBox(NULL, "你接受和棋吗？\nDo you accept draw?", "DRAW", MB\_YESNO | MB\_SYSTEMMODAL);

if (yn1 == 6)

{

final = 3;

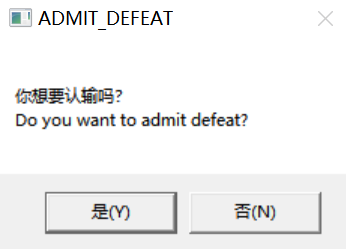
break;

}

}

其中MessageBox为对话框函数，点击“是Yes”返回数值6，MB\_YESNO为对话框的形式，MB\_SYSTEMMODAL是为使之出现在最前方。

**3.4.2 认输**

****

认输功能通过对话框实现，具体说明如下：

yn2 = MessageBox(NULL, "你想要认输吗？\nDo you want to admit defeat?", "ADMIT\_DEFEAT", MB\_YESNO | MB\_SYSTEMMODAL);

if (yn2 == 6)

{

final = 2;

break;

}

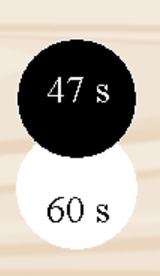
其中MessageBox为对话框函数，点击“是Yes”返回数值6，MB\_YESNO为对话框的形式，MB\_SYSTEMMODAL是为使之出现在最前方。

**3.5 时间**

**3.5.1 游戏时间**

****

**3.5.2 每回合倒计时**

****

游戏启动，将时刻记为start。白方回合开始，将时刻记为wt。黑方回合开始，将时刻记为bt。现在的时间记为now。

两个时间的具体说明如下：

{

now = clock();

duration = int(now - start) / 1000; //游戏总时长，化为秒

minute = duration / 60; //化为分

second = duration % 60; //化为秒

wduration = 60 - int(now - wt) / 1000; //白方倒计时

bduration = 60 - int(now - bt) / 1000; //黑方倒计时

sprintf\_s(s1, "Time: %d min %d s", minute, second); //时间：xx分xx秒

sprintf\_s(ws, "%d s", wduration); //白方倒计时字符串

sprintf\_s(bs, "%d s", bduration); //黑方倒计时字符串

setbkcolor(WHITE);

settextcolor(BLACK);

settextstyle(30, 10, "Times New Roman");

setlinecolor(WHITE);

setfillcolor(WHITE);

fillcircle(200 + le \* 4, -300, 340);

outtextxy(120 + le \* 4, 0, s1);

if (wb == 1 && wduration >= 0) //若是白方回合

{

setfillcolor(WHITE);

fillcircle(60, 4 \* wi + 80, 40);

setlinecolor(GREEN);

outtextxy(40, 80 + le \* 4, ws); //打印白方倒计时字符串

setbkcolor(BLACK);

settextcolor(WHITE);

outtextxy(40, le \* 4, "60 s"); //打印60s

}

if (wb == -1 && bduration >= 0) //若是黑方回合

{

setfillcolor(BLACK);

fillcircle(60, 4 \* wi + 20, 40);

setlinecolor(GREEN);

outtextxy(40, 80 + le \* 4, "60 s");//打印60s

setbkcolor(BLACK);

settextcolor(WHITE);

outtextxy(40, le \* 4, bs); //打印黑方倒计时字符串

}

FlushBatchDraw();

}

**4. 程序流程图**



**5. 函数和变量说明**

**函数说明**

void initial(); //开始

void info(); //信息

void chessboard(); //棋盘

void click(); //鼠标点击

void round(); //一个回合

int wKing(int a[2]); //白方国王行棋规则

int wQueen(int a[2]); //白方王后行棋规则

int wBishop(int a[2]); //白方两象行棋规则

int wKnight(int a[2]); //白方两马行棋规则

int wRook(int a[2]); //白方两车行棋规则

int wPawn(int a[2]); //白方士兵行棋规则

int bKing(int a[2]); //黑方国王行棋规则

int bQueen(int a[2]); //黑方王后行棋规则

int bBishop(int a[2]); //黑方两象行棋规则

int bKnight(int a[2]); //黑方两马行棋规则

int bRook(int a[2]); //黑方两车行棋规则

int bPawn(int a[2]); //黑方士兵行棋规则

int wEat(int a[2]); //判断白棋能否吃黑棋，若能则执行

int bEat(int a[2]); //判断黑棋能否吃白棋，若能则执行

int wBlock(int a[2]); //判断该位置有无白棋，从而判断能否落子

int bBlock(int a[2]); //判断该位置有无黑棋，从而判断能否落子

void judge(); //判断比赛是否决出胜负，以及谁胜谁负

void putw(); //依据坐标绘制黑棋图像

void putb(); //摆放黑棋

void display(); //更新棋盘

**变量说明**

//定义图像变量：p01~p06,p11~p16

IMAGE p01, p02, p03, p04, p05, p06; //白棋王、后、象、马、车、兵

IMAGE p11, p12, p13, p14, p15, p16; //黑棋王、后、象、马、车、兵

int a = 360, b = 600, c = 40; //a：开始界面长度；b：开始界面宽度；c：图标边长

int le = 100, wi = 100; //le：棋子、格子长度；wi：棋子、格子长度

//i，j：计数；k：同种棋子序号；n：可行格子序号；p：暂时性同种棋子序号，用于put，block，eat等

int i = 0, j = 0, k = 0, n = 0, p = 0, bn = 0, wn = 0;

//yn1, yn2, yn3:用于记录对话框选择的结果

int yn1 = 0, yn2 = 0, yn3 = 0;

int sign = 0; //sign：用于判断单次操作是否完成

int mark = 0; //mark: 用于判断棋子是否能落在某处

int wb = 0; //判断是白方还是黑方正在下棋

//用于判断王车易位是否可行，1表示可以，0表示不可以

int wCastling = 1,wCastling1 = 1,wCastling2 = 1;

int bCastling = 1,bCastling1 = 1,bCastling2 = 1;

//用于判断吃过路兵是否可行，1表示可以，0表示不可以

int wEn\_passant = 0, wEn\_passantx = 0, wEn\_passanty = 0;

int bEn\_passant = 0, bEn\_passantx = 0, bEn\_passanty = 0;

int wCheck = 0;

int bCheck = 0;

int final = 0; //final：用于判断游戏是否结束

time\_t start, now, wt, bt; //start：游戏开始时刻；now：现在时刻；wt：白方回合开始时刻；bt：黑方回合开始时刻

char s1[100], ws[100], bs[100]; //s1：游戏时间字符串；ws：白方倒计时；bs：黑方倒计时

int wK[2] = { 5,8 }; //wK：白棋国王坐标

int wQ[2] = { 4,8 }; //wQ：白棋皇后坐标

int wB[2][2] = { {3,8},{6,8} }; //wB：白棋两象坐标

int wN[2][2] = { {2,8},{7,8} }; //wN：白棋两马坐标

int wR[2][2] = { {1,8},{8,8} }; //wR：白棋两车坐标

int wP[8][2] = { {1,7},{2,7},{3,7},{4,7},{5,7},{6,7},{7,7},{8,7} }; //wP：白棋士兵坐标

int wQp[8][2]; //wQp：白棋兵升变后坐标

int wBp[8][2]; //wBp：白棋兵升变象坐标

int wNp[8][2]; //wNp：白棋兵升变马坐标

int wRp[8][2]; //wRp：白棋兵升变车坐标

int bK[2] = { 5,1 }; //bK：黑棋国王坐标

int bQ[2] = { 4,1 }; //bQ：黑棋王后坐标

int bB[2][2] = { {3,1},{6,1} }; //bB：黑棋两象坐标

int bN[2][2] = { {2,1},{7,1} }; //bN：黑棋两马坐标

int bR[2][2] = { {1,1},{8,1} }; //bR：黑棋两车坐标

int bP[8][2] = { {1,2},{2,2},{3,2},{4,2},{5,2},{6,2},{7,2},{8,2} }; //bP：黑棋士兵坐标

int bQp[8][2]; //bQp：黑棋兵升变后坐标

int bBp[8][2]; //bBp：黑棋兵升变象坐标

int bNp[8][2]; //bNp：黑棋兵升变马坐标

int bRp[8][2]; //bRp：黑棋兵升变车坐标

int D[65][2] = { '\0' }; //可行域

int mx = -1, my = -1; //鼠标位置

int xm = -1, ym = -1; //鼠标坐标

int x[2] = { -1,-1 }, y[2] = { -1,-1 }; //坐标

int temp[2] = { -1,-1 }; //暂时性坐标

——END——