**C语言项目报告**

**项目名称：德州扑克**

**项目成员：魏宇栋、杨凯越**

**填写日期：2021/7/6**

[一、摘要（Abstract）](#摘要)

[二、问题描述（Problem Statement）](#问题描述)

[三、组内分工（Group Division）](#组内分工)

[四、分析（Analysis）](#分析)

[五、设计（Design）](#设计)

[主程序](#主程序)

[洗\发牌](#洗发牌)

[牌型判断](#牌型判断)

[双人对战](#双人对战)

[人机对战](#人机对战)

[模块流程图](#模块流程图)

[六、实施（Implementation）](#实施)

[主程序核心代码](#主程序核心代码)

[洗\发牌核心代码](#洗发牌核心代码)

[牌型判断核心代码](#牌型判断核心代码)

[排序](#排序)

[牌组牌型判断](#牌组牌型判断)

[双人对战核心代码](#双人对战核心代码)

[人机对战核心代码](#人机对战核心代码)

[电脑选择](#电脑选择)

[记牌器](#记牌器)

[七、测试（Test）](#测试)

[主程序测试](#主程序测试)

[洗\发牌测试](#洗发牌测试)

[牌型判断测试](#牌型判断测试)

[双人对战测试](#双人对战测试)

[人机对战测试](#人机对战测试)

**一、摘要（Abstract）**

本项目旨在开发一款具有洗、发牌，牌型判断，双人对战以及人机对战等功能的软件以模拟德州扑克游戏。本项目从全新的角度对牌型判断以及人机对战程序进行了设计。

**二、问题描述（Problem Statement）**

项目背景: 前几年德州扑克在国内受到打击，众多德州扑克的游戏软件都被迫下架。故希望开发一款具有洗、发牌，牌型判断，双人对战以及人机对战等功能的软件以模拟德州扑克。

拟解决的问题:1.设计主程序（欢迎用户、模式选择、开始游戏）

2.设计洗\发牌程序（生成一副牌、按需发牌）

3.设计牌型判断程序（对高牌、一对等共计10种牌型进行判断，并赋予相应的牌型点数）

4.设计双人对战程序（进行四轮游戏，记录筹码，判断胜负等）

5.设计人机对战程序（在游戏中使电脑自动做出选择，并保持一定胜率）

6.测试并完善整个程序

**三、组内分工（Group Division）**

组长：魏宇栋

组员：杨凯越

具体分工：

魏宇栋：设计主程序，设计洗\发牌程序，设计牌型判断程序，设计双人对战程序，设计人机对战程序（部分），测试及完善程序（部分）。(负责工作量约80%)

杨凯越：设计人机对战程序（部分），测试及完善程序（部分）。（负责工作量约20%）

**四、分析（Analysis）**

主体框架分为五个模块：主程序、洗\发牌程序、牌型判断程序、双人对战程序、人机对战程序。

设计的运行逻辑为：首先进入主程序，当用户选择游戏模式（双人对战or人机对战）后，进入相应的对战程序。接着运行洗、发牌程序，构建本局游戏所需的牌组。然后继续运行对战程序，并在需要的时候运行牌型判断程序。当一局游戏运行结束后，自动开始新的一局游戏。

**五、设计（Design）**

**主程序：**

(1)欢迎用户进入软件

(2)请用户选择游戏模式

(3)当一场游戏结束后重新回到主程序，并开始新的游戏

**洗\发牌：**

(1)生成一副牌共52张(无大小王)

(2)依次生成若干随机数，每个随机数对应一张牌，并写入相应的牌组(如公共牌、玩家牌等)

(3)按需要发牌，输出牌的点数和花色

**牌型判断：**

(1)将相应牌组中的牌进行点数从大到小的排序(以下所提及的牌组均为排序后的牌组)

(2)依次判断牌组的牌型是否为高牌、一对、两对、三条、顺子、同花、满堂彩、四条、同花顺，并赋相应的牌型点数

**双人对战：**

(1)请求用户输入初始筹码数，然后依次展示双方底牌

(2)依次进行第一、二、三、决胜轮，每轮包括发牌、下注\弃牌、跟注\弃牌、加注\不加注等

(3)四轮全部结束或有一方筹码不足则进入开牌阶段，展示双方底牌以及牌河的全部牌，比较双方牌型大小，判断获胜方，然后进入筹码检查阶段(若有一方弃牌，则直接进入筹码检查阶段)

(4)筹码检查阶段：清算筹码，然后判断是否有一方筹码不足100，若有则判断筹码充足一方获得最终胜利，并重新运行主程序。否则，进入下一局游戏

**人机对战：**

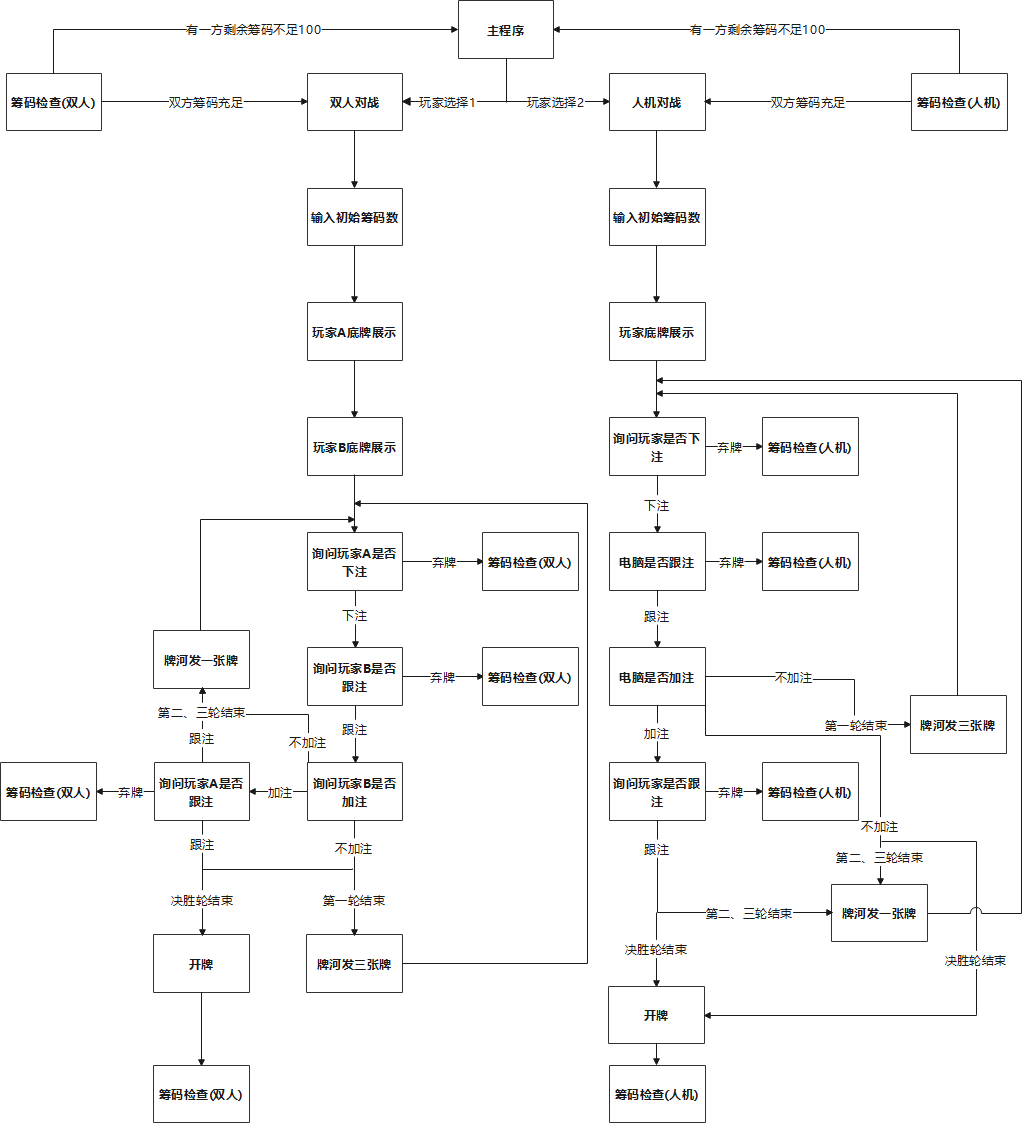
(1)请求用户输入初始筹码数，然后依次展示玩家底牌(玩家的底牌将写入记牌器)

(2)依次进行第一、二、三、决胜轮，每轮包括发牌(牌河的牌将写入记牌器)、下注\弃牌、跟注\弃牌、加注\不加注等，并根据电脑的牌来决定其选择

(3)四轮全部结束或有一方筹码不足则进入开牌阶段，展示双方底牌以及牌河的全部牌，比较双方牌型大小，判断获胜方，然后进入筹码检查阶段(若有一方弃牌，则直接进入筹码检查阶段)

(4)筹码检查阶段：清算筹码，然后判断是否有一方筹码不足100，若有则判断筹码充足一方获得最终胜利，并重新运行主程序。否则，进入下一局游戏

**模块流程图：**



**六、实施（Implementation）**

**主程序核心代码：**

int main()

{

int gamemode = -1;//初始化游戏模式

games = 1;//初始化游戏局数

cardtype = 0;//初始化卡牌点数

cardtypeA = 0;//初始化玩家A的卡牌点数

cardtypeB = 0;//初始化玩家B（电脑）的卡牌点数

pointplus = 0;//初始化加注数

choose = -1;//初始化选择

chooseplus = -1;//初始化加注选择

printf("欢迎来到德州扑克的世界!(本程序仅供娱乐,请勿沉迷赌博!!!)\n\n");

printf("请选择游戏模式:\n1.双人对战\n2.人机对战\n\n请输入1或2,并以回车结束\n");

scanf\_s("%d", &gamemode);

for(; (gamemode != 1) && (gamemode != 2);)

{

getchar();

printf("\n输入有误,请输入1或2,并以回车结束\n");

scanf\_s("%d", &gamemode);

}

if (gamemode == 1)//双人对战模式

{

printf("\n\n您已经选择了双人对战模式,下面将介绍下注规则\n");

system("pause>nul");

printf("\n单注为100,下注无上限,每局自动下底注100\n");

system("pause>nul");

printf("\n现在请决定谁为玩家A (玩家A将在接下去的游戏过程中首先下注)\n");

system("pause>nul");

printf("\n现在请输入双方初始筹码数量(应为100的倍数),并以回车键结束(建议不低于2000)\n");

scanf\_s("%d", &betA);

for (; (betA % 100 != 0) || (betA <= 100);)

{

getchar();

printf("输入的值有误,请输入100的倍数,并以回车结束\n");

scanf\_s("%d", &betA);

}

betB = betA;

for (; (betA >= 100) && (betB >= 100);)

{

PVP();

}

}

if (gamemode == 2)//人机对战模式

{

printf("\n\n您已经选择了人机对战模式,下面将介绍规则\n");

system("pause>nul");

printf("\n单注为100,下注无上限,每局自动下底注100\n您为庄(每轮您将先下注)\n");

system("pause>nul");

printf("\n现在请输入双方初始筹码数量(应为100的倍数),并以回车键结束(建议不低于2000)\n");

scanf\_s("%d", &betA);

for (; (betA % 100 != 0) || (betA <= 100);)

{

getchar();

printf("输入的值有误,请输入100的倍数,并以回车结束\n");

scanf\_s("%d", &betA);

}

betB = betA;

for (; (betA >= 100) && (betB >= 100);)

{

PVE();

}

}

}

**洗\发牌核心代码：**

int i,k;

struct cards card[53];//定义52张牌[1]~[52]

struct cards\* tr;

card[1].point = 1;//点数1为A

card[1].colour = 1;//花色1为方块

card[2].point = 1;

card[2].colour = 2;//花色2为梅花

card[3].point = 1;

card[3].colour = 3;//花色3为红桃

card[4].point = 1;

card[4].colour = 4;//花色4为黑桃

tr = &card[4];

for (i = 2; i <= 13; i++)//点数11为J,点数12为Q,点数13为K

{

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

{

tr = tr + 1;

tr->point = i;

tr->colour = k;

}

} //一副牌52张已定义

struct cards cardA1;

cardA1.point = 20;//给玩家A的牌1的点数定义为空

cardA1.colour = 20;//给玩家A的牌1的花色定义为空

struct cards cardA2;

cardA2.point = 20;//给玩家A的牌2的点数定义为空

cardA2.colour = 20;//给玩家A的牌2的花色定义为空

struct cards cardB1;

cardB1.point = 20;//给玩家B的牌1的点数定义为空

cardB1.colour = 20;//给玩家B的牌1的花色定义为空

struct cards cardB2;

cardB2.point = 20;//给玩家B的牌2的点数定义为空

cardB2.colour = 20;//给玩家B的牌2的花色定义为空

srand((unsigned)time(NULL));//取一个随机数种子

int randnum1,randnum2,randnum3,randnum4,randnum5,randnum6,randnum7,randnum8,randnum9;//定义九个随机数

randnum1= rand();//给randnum1随机取一个数

int\* ptr = &randnum1;

ptr=randnumber(ptr);//使用randnumber函数将randnum1控制在1~52中

cardA1 = card[randnum1];//给玩家A的第一张牌随机赋点数和花色

randnum2 = rand();

ptr = &randnum2;

ptr = randnumber(ptr);

for (; randnum2 == randnum1;)//确保randnum2!=randnum1

{

ptr = randnumber(ptr);

}

cardA2 = card[randnum2];//给玩家A的第二张牌随机赋点数和花色

randnum3 = rand();

ptr = &randnum3;

ptr = randnumber(ptr);

for (; randnum3 == randnum1 || randnum3 == randnum2;)

{

ptr = randnumber(ptr);

}

cardB1 = card[randnum3];//给玩家B的第一张牌随机赋点数和花色

randnum4 = rand();

ptr = &randnum4;

ptr = randnumber(ptr);

for (; randnum4 == randnum1 || randnum4 == randnum2 || randnum4 == randnum3;)

{

ptr = randnumber(ptr);

}

cardB2 = card[randnum4];//给玩家B的第二张牌随机赋点数和花色

struct cards cardall[6];//定义庄家的牌[1]~[5]

cardall[1] .point = 20;//给庄家牌中的第一张牌的点数定义为空

cardall[1].colour = 20;//给庄家牌中的第一张牌的花色定义为空

cardall[2].point = 20;//给庄家牌中的第二张牌的点数定义为空

cardall[2].colour = 20;//给庄家牌中的第二张牌的花色定义为空

cardall[3].point = 20;//给庄家牌中的第三张牌的点数定义为空

cardall[3].colour = 20;//给庄家牌中的第三张牌的花色定义为空

cardall[4].point = 20;//给庄家牌中的第四张牌的点数定义为空

cardall[4].colour = 20;//给庄家牌中的第四张牌的花色定义为空

cardall[5].point = 20;//给庄家牌中的第五张牌的点数定义为空

cardall[5].colour = 20;//给庄家牌中的第五张牌的花色定义为空

randnum5 = rand();

ptr = &randnum5;

ptr = randnumber(ptr);

for (; randnum5 == randnum1 || randnum5 == randnum2 || randnum5 == randnum3||randnum5 == randnum4;)

{

ptr = randnumber(ptr);

}

cardall[1] = card[randnum5];

randnum6 = rand();

ptr = &randnum6;

ptr = randnumber(ptr);

for (; randnum6 == randnum1 || randnum6 == randnum2 || randnum6 == randnum3 || randnum6 == randnum4||randnum6==randnum5;)

{

ptr = randnumber(ptr);

}

cardall[2] = card[randnum6];

randnum7 = rand();

ptr = &randnum7;

ptr = randnumber(ptr);

for (; randnum7 == randnum1 || randnum7 == randnum2 || randnum7 == randnum3 || randnum7 == randnum4 || randnum7 == randnum5 || randnum7==randnum6;)

{

ptr = randnumber(ptr);

}

cardall[3] = card[randnum7];

randnum8 = rand();

ptr = &randnum8;

ptr = randnumber(ptr);

for (; randnum8 == randnum1 || randnum8 == randnum2 || randnum8 == randnum3 || randnum8 == randnum4 || randnum8 == randnum5 || randnum8 == randnum6 || randnum8 == randnum7;)

{

ptr = randnumber(ptr);

}

cardall[4] = card[randnum8];

randnum9 = rand();

ptr = &randnum9;

ptr = randnumber(ptr);

for (; randnum9 == randnum1 || randnum9 == randnum2 || randnum9 == randnum3 || randnum9 == randnum4 || randnum9 == randnum5 || randnum9 == randnum6 || randnum9 == randnum7 || randnum9 == randnum8;)

{

ptr = randnumber(ptr);

}

cardall[5] = card[randnum9];//庄家的五张牌全部随机生成完成

struct cards A[7];//定义玩家A的两张牌和庄家的五张牌组成的牌组

A[0] = cardA1;

A[1] = cardA2;

A[2] = cardall[1];

A[3] = cardall[2];

A[4] = cardall[3];

A[5] = cardall[4];

A[6] = cardall[5];

struct cards B[7];//定义玩家B的两张牌和庄家的五张牌组成的牌组

B[0] = cardB1;

B[1] = cardB2;

B[2] = cardall[1];

B[3] = cardall[2];

B[4] = cardall[3];

B[5] = cardall[4];

B[6] = cardall[5];

**牌型判断核心代码：**

**排序：**

void PVPsorting(struct cards\* tr,int len)//将牌的点数从大到小排序的函数

{

struct cards zero;

zero.colour = 0;

zero.point = 0;

struct cards\* tmp=NULL;

tmp = &zero;

int i, j;

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

{

for (j = i; j <=6; j++)

{

if (((tr+i-1)->point)<((tr+j)->point))

{

tmp->colour = (tr + i-1)->colour;

(tr + i-1)->colour = (tr + j)->colour;

(tr + j)->colour = tmp->colour;

tmp->point = (tr + i-1)->point;

(tr + i-1)->point = (tr + j)->point;

(tr + j)->point = tmp->point;

}

}

}

**牌组牌型判断（以高牌为例）：**

void Highcard(struct cards\* tr, int len)//判断是否为高牌的函数(1<=cardtype<=24)

{

int i = 0;

if (((tr + len - 1)->point) == 1)//A的高牌

{

if (((tr + len - 1)->colour) == 4)

{

cardtype = 24;//黑桃A

return 0;

}

if (((tr + len - 1)->colour) == 3)

{

cardtype = 23;//红桃A

return 0;

}

if (((tr + len - 1)->colour) == 2)

{

cardtype = 22;//梅花A

return 0;

}

if (((tr + len - 1)->colour) == 1)

{

cardtype = 21;//方块A

return 0;

}

}

for (i = 1; i<=5; i++)//K到9的高牌

{

if ((tr->point) == 14-i)

{

if ((tr->colour) == 4)

{

cardtype = (24 - 4 \* i);

return 0;

}

if ((tr->colour) == 3)

{

cardtype = (23 - 4 \* i);

return 0;

}

if ((tr->colour) == 2)

{

cardtype = (22 - 4 \* i);

return 0;

}

if ((tr->colour) == 1)

{

cardtype = (21 - 4 \* i);

return 0;

}

}

}

}

**双人对战核心代码（以第一轮为例）：**

printf("\n\n当前是第%d局\n", games);

printf("双方自动下底注100\n\n");

betA = betA - 100;

betB = betB - 100;

betall = 200;

printf("玩家A剩余的筹码为:%d 玩家B剩余的筹码为:%d\n\n", betA, betB);

system("pause>nul");

printf("下面将展示玩家A的底牌,请玩家B暂时回避\n");

system("pause>nul");

printf("请玩家A记住您的底牌,完成后按回车即可\n\n");

colour(&cardA1);

point(&cardA1);

colour(&cardA2);

point(&cardA2);

system("pause>nul");

printf("\r");

printf(" \n");

printf("下面将展示玩家B的底牌,请玩家A暂时回避\n");

system("pause>nul");

printf("请玩家B记住您的底牌,完成后按回车即可\n\n");

colour(&cardB1);

point(&cardB1);

colour(&cardB2);

point(&cardB2);

system("pause>nul");

printf("\r");

printf(" \n");

if (betA < 100)//玩家A的筹码已不足下一注

{

printf("\n玩家A的筹码已不足100,下面将直接发五张牌并开牌\n");

system("pause>nul");

printf("\n下面将进入开牌阶段\n");

printf("牌河为:");

colour(&cardall[1]);

point(&cardall[1]);

colour(&cardall[2]);

point(&cardall[2]);

colour(&cardall[3]);

point(&cardall[3]);

colour(&cardall[4]);

point(&cardall[4]);

colour(&cardall[5]);

point(&cardall[5]);

printf("\n\n");

printf("玩家A的底牌为:");

colour(&cardA1);

point(&cardA1);

colour(&cardA2);

point(&cardA2);

printf("\n\n玩家B的底牌为:");

colour(&cardB1);

point(&cardB1);

colour(&cardB2);

point(&cardB2);

typeA(&A[0], 7);//判断A的牌型

typeB(&B[0], 7);//判断B的牌型

system("pause>nul");

printf("\n\n玩家A的牌型为:");

typeprint(cardtypeA);

printf("\n\n玩家B的牌型为:");

typeprint(cardtypeB);

system("pause>nul");

if (cardtypeA > cardtypeB)

{

printf("\n\n玩家A的牌型大,恭喜玩家A赢得这局游戏!!");

betA = betA + betall;

if (betB < 100)//B的筹码不足,判定A获胜

{

printf("\n玩家B的筹码已不足100,恭喜玩家A获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

if (cardtypeB > cardtypeA)

{

printf("\n\n玩家B的牌型大,恭喜玩家B赢得这局游戏!!");

betB = betB + betall;

if (betA < 100)//A的筹码不足,判定B获胜

{

printf("\n玩家A的筹码已不足100,恭喜玩家B获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

if (cardtypeA == cardtypeB)

{

if (cardtypeA <= 24)//高牌比大小

{

Highcardhole(&cardA1, &cardA2, &cardB1, &cardB2);

if (Highcardtype == 1)

{

printf("\n\n玩家A的牌型大,恭喜玩家A赢得这局游戏!!");

betA = betA + betall;

if (betB < 100)

{

printf("\n玩家B的筹码已不足100,恭喜玩家A获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

if (Highcardtype == 2)

{

printf("\n\n玩家B的牌型大,恭喜玩家B赢得这局游戏!!");

betB = betB + betall;

if (betA < 100)//A的筹码不足,判定B获胜

{

printf("\n玩家A的筹码已不足100,恭喜玩家B获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

}

printf("\n\n本局双方平局");

betA = betA + betall / 2;

betB = betB + betall / 2;

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

}

if (betA >= 100)//A的筹码足够下至少一注

{

printf("下面进入第一轮,请玩家A选择下注或弃牌\n");//第一轮

printf("1.下注 0.弃牌\n");

int choose = -1;

scanf\_s("%d", &choose);

betBround = 0;

betAround = 0;

chooseplus = -1;

pointplus = -1;

for (; (choose != 1) && (choose != 0);)//输入有误则重新输入

{

getchar();

printf("\n输入有误,请输入1或0,并以回车结束\n");

scanf\_s("%d", &choose);

}

if (choose == 0)

{

printf("\n玩家A选择弃牌,这局游戏B获胜\n");

betB = betB + betall;

printf("玩家A剩余的筹码为:%d 玩家B剩余的筹码为:%d\n\n", betA, betB);

games++;//局数加1

if (betA < 100)

{

printf("\n玩家A的筹码已不足100,恭喜玩家B获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

if (betB < 100)

{

printf("\n玩家B的筹码已不足100,恭喜玩家A获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

if (choose == 1)

{

printf("您剩余的筹码为%d\n",betA);

printf("请输入您要下注的数量(100的倍数):\n");

scanf\_s("%d", &betAround);

for (; ((betAround%100)!=0)||(betAround<100)||(betAround>betA);)//输入有误则重新输入

{

if(betAround>betA)

{

getchar();

printf("\n您的筹码不足,请重新输入您要下注的数量,并以回车结束\n");

scanf\_s("%d", &betAround);

}

if(((betAround % 100) != 0) || (betAround < 100))

{

getchar();

printf("\n输入有误,请输入100的倍数,并以回车结束\n");

scanf\_s("%d", &betAround);

}

}

betA = betA - betAround;

betall = betall + betAround;

}

printf("\n\n玩家A下注数量为%d", betAround);//玩家A的第一轮结束

printf("\n\n下面请玩家B选择跟注或弃牌\n");//玩家B的第一轮开始

printf("1.跟注 0.弃牌\n");

choose = -1;

scanf\_s("%d", &choose);

for (; (choose != 1) && (choose != 0);)//输入有误则重新输入

{

getchar();

printf("\n输入有误,请输入1或0,并以回车结束\n");

scanf\_s("%d", &choose);

}

if (choose == 0)//B选择弃牌

{

printf("\n玩家B选择弃牌,这局游戏A获胜\n");

betA = betA + betall;

printf("玩家A剩余的筹码为:%d 玩家B剩余的筹码为:%d\n\n", betA, betB);

games++;//局数加1

if (betA < 100)

{

printf("\n玩家A的筹码已不足100,恭喜玩家B获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("\npause>nul");

main();

}

if (betB < 100)

{

printf("\n玩家B的筹码已不足100,恭喜玩家A获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

if (choose == 1)

{

if (betB <= betAround)//B的筹码不足,自动全部跟注

{

betBround = betB;

betB = 0;

betall = betBround + betall;

printf("\n您剩余的筹码已不足玩家A下的注,已自动为您全部跟注\n\n");

printf("下面将进入开牌阶段\n");

system("pause>nul");

printf("牌河为:");

colour(&cardall[1]);

point(&cardall[1]);

colour(&cardall[2]);

point(&cardall[2]);

colour(&cardall[3]);

point(&cardall[3]);

colour(&cardall[4]);

point(&cardall[4]);

colour(&cardall[5]);

point(&cardall[5]);

printf("\n\n");

printf("玩家A的底牌为:");

colour(&cardA1);

point(&cardA1);

colour(&cardA2);

point(&cardA2);

printf("\n\n玩家B的底牌为:");

colour(&cardB1);

point(&cardB1);

colour(&cardB2);

point(&cardB2);

typeA(&A[0], 7);//判断A的牌型

typeB(&B[0], 7);//判断B的牌型

system("pause>nul");

printf("\n\n玩家A的牌型为:");

typeprint(cardtypeA);

printf("\n\n玩家B的牌型为:");

typeprint(cardtypeB);

system("pause>nul");

if (cardtypeA > cardtypeB)

{

printf("\n\n玩家A的牌型大,恭喜玩家A赢得这局游戏!!");

betA = betA + betall ;

if (betB < 100)

{

printf("\n玩家B的筹码已不足100,恭喜玩家A获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

if (cardtypeB > cardtypeA)

{

printf("\n\n玩家B的牌型大,恭喜玩家B赢得这局游戏!!");

betB = betB + betall;

if (betA < 100)//A的筹码不足,判定B获胜

{

printf("\n玩家A的筹码已不足100,恭喜玩家B获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

if(cardtypeA==cardtypeB)//平局

{

if (cardtypeA <= 24)//高牌比大小

{

Highcardhole(&cardA1, &cardA2, &cardB1, &cardB2);

if (Highcardtype == 1)

{

printf("\n\n玩家A的牌型大,恭喜玩家A赢得这局游戏!!");

betA = betA + betall;

printf("\n玩家B的筹码已不足100,恭喜玩家A获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

if (Highcardtype == 2)

{

printf("\n\n玩家B的牌型大,恭喜玩家B赢得这局游戏!!");

betB = betB + betall;

if (betA < 100)//A的筹码不足,判定B获胜

{

printf("\n玩家A的筹码已不足100,恭喜玩家B获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

}

}

printf("\n\n本局双方平局");

betA = betA + betall / 2;

betB = betB + betall / 2;

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

}

else //B的筹码充足

{

betBround = betAround;

betB = betB - betBround;

betall = betall + betBround;

chooseplus = -1;

if (betB >= 100)

{

printf("\n已为您自动跟注,请您选择是否加注:1.加注 0.不加注\n");

scanf\_s("%d", &chooseplus);

for (; (chooseplus != 1) && (chooseplus != 0);)

{

getchar();

printf("\n输入有误,请输入1或0,并以回车结束\n");

scanf\_s("%d", &chooseplus);

}

}

if (chooseplus == 1)//B选择加注

{

printf("您剩余的筹码为%d\n", betB);

printf("请输入您要加注的数量(100的倍数):\n");

scanf\_s("%d", &pointplus);

for (; ((pointplus % 100) != 0) || (pointplus < 100) || (pointplus > betB);)//输入有误则重新输入

{

if (pointplus > betB)

{

getchar();

printf("\n您的筹码不足,请重新输入您要下注的数量,并以回车结束\n");

scanf\_s("%d", &pointplus);

}

if (((pointplus % 100) != 0) || (pointplus < 100))

{

getchar();

printf("\n输入有误,请输入100的倍数,并以回车结束\n");

scanf\_s("%d", &pointplus);

}

}

betB = betB - pointplus;

betBround = betBround + pointplus;

betall = pointplus + betall;

}

}

}//B下注完成

if (chooseplus == 1)//B加注后询问A是否跟注

{

printf("\n玩家B选择加注%d,请玩家A选择跟注或弃牌\n", pointplus);

printf("1.跟注 0.弃牌\n");

int choose = -1;

scanf\_s("%d", &choose);

for (; (choose != 1) && (choose != 0);)//输入有误则重新输入

{

getchar();

printf("\n输入有误,请输入1或0,并以回车结束\n");

scanf\_s("%d", &choose);

}

if (choose == 0)//玩家A选择在B加注后弃牌

{

printf("\n玩家A选择弃牌,这局游戏B获胜\n");

betB = betB + betall;

printf("玩家A剩余的筹码为:%d 玩家B剩余的筹码为:%d\n\n", betA, betB);

games++;//局数加1

if (betA < 100)

{

printf("\n玩家A的筹码已不足100,恭喜玩家B获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

if (betB < 100)

{

printf("\n玩家B的筹码已不足100,恭喜玩家A获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

if (choose == 1)//玩家A选择在B加注后跟注

{

if (betA <= pointplus)//A的筹码不足,自动全部跟注

{

betall = betA + betall;

betAround = betA + betAround;

betA = 0;

printf("\n您剩余的筹码已不足玩家B加的注,已自动为您全部跟注\n\n");

printf("下面将进入开牌阶段\n");

printf("牌河为:");

colour(&cardall[1]);

point(&cardall[1]);

colour(&cardall[2]);

point(&cardall[2]);

colour(&cardall[3]);

point(&cardall[3]);

colour(&cardall[4]);

point(&cardall[4]);

colour(&cardall[5]);

point(&cardall[5]);

printf("\n\n");

printf("玩家A的底牌为:");

colour(&cardA1);

point(&cardA1);

colour(&cardA2);

point(&cardA2);

printf("\n\n玩家B的底牌为:");

colour(&cardB1);

point(&cardB1);

colour(&cardB2);

point(&cardB2);

typeA(&A[0], 7);//判断A的牌型

typeB(&B[0], 7);//判断B的牌型

system("pause>nul");

printf("\n\n玩家A的牌型为:");

typeprint(cardtypeA);

printf("\n\n玩家B的牌型为:");

typeprint(cardtypeB);

system("pause>nul");

if (cardtypeA > cardtypeB)

{

printf("\n\n玩家A的牌型大,恭喜玩家A赢得这局游戏!!");

betA = betA + betall;

if (betB < 100)//B的筹码不足,判定A获胜

{

printf("\n玩家B的筹码已不足100,恭喜玩家A获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

if (cardtypeB > cardtypeA)

{

printf("\n\n玩家B的牌型大,恭喜玩家B赢得这局游戏!!");

betB = betB + betall;

if (betA < 100)//A的筹码不足,判定B获胜

{

printf("\n玩家A的筹码已不足100,恭喜玩家B获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

if (cardtypeA == cardtypeB)

{

if (cardtypeA <= 24)//高牌比大小

{

Highcardhole(&cardA1, &cardA2, &cardB1, &cardB2);

if (Highcardtype == 1)

{

printf("\n\n玩家A的牌型大,恭喜玩家A赢得这局游戏!!");

betA = betA + betall;

if (betB < 100)//B的筹码不足,判定A获胜

{

printf("\n玩家B的筹码已不足100,恭喜玩家A获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

if (Highcardtype == 2)

{

printf("\n\n玩家B的牌型大,恭喜玩家B赢得这局游戏!!");

betB = betB + betall;

if (betA < 100)//A的筹码不足,判定B获胜

{

printf("\n玩家A的筹码已不足100,恭喜玩家B获得胜利!!!\n\n");

system("pause>nul");

printf("按任意键重新开始游戏\n");

system("pause>nul");

main();

}

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

}

printf("\n\n本局双方平局");

betA = betA + betall / 2;

betB = betB + betall / 2;

games++;

printf("\n\n第%d局已游戏结束,按任意键进入下一局游戏\n", games - 1);

system("pause>nul");

PVP();

}

}

betA = betA - pointplus;//玩家A的筹码足够跟注

betall = betall + pointplus;;

betAround = betAround + pointplus;

printf("\n玩家A选择跟注,第一轮共下注%d\n", betAround+betBround);

system("pause>nul");

}

}

}

if ((chooseplus == 0)||(chooseplus==-1))

{

printf("\n玩家B选择不加注,第一轮共下注%d\n", betAround+betBround);

system("pause>nul");

}//第一轮结束

**人机对战核心代码（只展示电脑选择的代码和记牌器的代码，其余与双人对战类似）：**

**电脑选择（以第一轮为例）：**

rander1 = rand();

rander2 = rand();

choose = -1;

//电脑进行判断是否跟注和加注

typeA(&A[0], 7);

typeB(&B[0], 7);

if (cardtypeB <= 24)//电脑为高牌

{

if ((rander1 % 4) == 0)//电脑四分之一概率跟注

{

choose = 1;

}

if (((rander1 % 4) == 1)|| ((rander1 % 4) == 2)|| ((rander1 % 4) == 3))//电脑四分之三概率弃牌

{

choose = 0;

}

}

if ((cardtypeB >= 25) && (cardtypeB <= 102))//电脑为一对

{

rander1 = rand();//重新获得一个随机数

if ((rander1 % 2) == 0)//电脑二分之一概率跟注

{

choose = 1;

}

if ((rander1 % 2) == 1)//电脑二分之一概率弃牌

{

choose = 0;

}

}

if ((cardtypeB >= 103) && (cardtypeB <= 2910))//电脑为两对

{

if (((rander1 % 4) == 0) || ((rander1 % 4) == 1) || ((rander1 % 4) == 2))//电脑四分之三概率跟注

{

choose = 1;

}

if (((rander1 % 4) == 3))//电脑四分之一概率弃牌

{

choose = 0;

}

}

if (cardtypeB >= 2911)//电脑牌型大于两对

{

choose = 1;//电脑跟注

}

if (betB <= 200)//如果电脑筹码已不多,自动跟注

{

choose = 1;//电脑跟注

}

if (betB > betAround)//电脑跟注且筹码充足

{

betB = betB - betAround;

betBround = betAround;

betall = betall + betBround;

if (betB < 100)//电脑筹码不足100,不加注

{

chooseplus = 0;

}

if (betB >= 100)

{

if (cardtype <= 102)//电脑牌型为高牌,电脑不加注

{

chooseplus = 0;

}

if ((cardtypeB >= 103) && (cardtypeB <= 2910))//电脑牌型为两对

{

rander1 = rand();

if (rander1 % 5 == 0)//电脑五分之一概率加注100

{

if (betB <= 500)

{

chooseplus = 1;

pointplus = 100;//电脑加注100

betBround = betBround + pointplus;

betB = betB - pointplus;

betall = betall + pointplus;

}

}

if (rander1 % 5 == 1)//电脑五分之一概率加注300

{

if (betB > 500)

{

chooseplus = 1;

pointplus = 300;//电脑加注300

betBround = betBround + pointplus;

betB = betB - pointplus;

betall = betall + pointplus;

}

else

{

chooseplus = 1;

pointplus = 100;

betBround = betBround + pointplus;

betB = betB - pointplus;

betall = betall + pointplus;

}

}

if ((rander1 % 5 == 2) || (rander1 % 5 == 3) || (rander1 % 5 == 4))//电脑五分之三概率不加注

{

chooseplus = 0;

}

}

if (cardtype >= 2911)//电脑牌型大于两对

{

if (betB < 300)

{

chooseplus = 1;

pointplus = 100;

betBround = betBround + pointplus;

betB = betB - pointplus;

betall = betall + pointplus;

}

if ((betB < 1000) && (betB >= 300))

{

chooseplus = 1;

pointplus = 300;

betBround = betBround + pointplus;

betB = betB - pointplus;

betall = betall + pointplus;

}

if (betB >= 1000)

{

chooseplus = 1;

pointplus = 1000;

betBround = betBround + pointplus;

betB = betB - pointplus;

betall = betall + pointplus;

}

}

}

}

**记牌器（以第一轮为例）：**

system("taskkill/im notepad.exe>nul");

out = fopen(filename, "w");

again = fopen(filename, "at");

fputs("", out);

fputs("您的底牌为: ", again);

putcard(&cardA1);

fputs(" ", again);

putcard(&cardA2);

fclose(again);

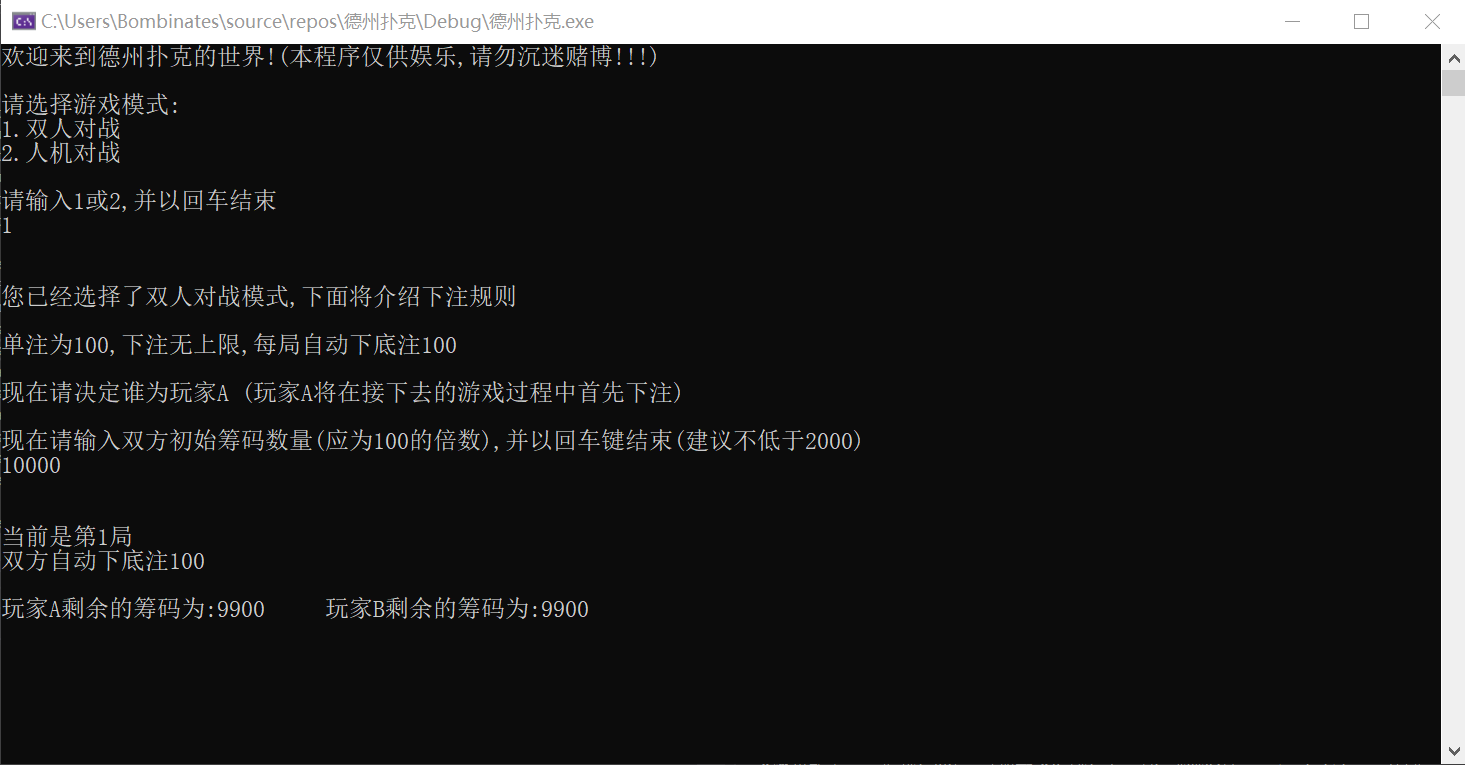
fclose(out);

system("start/b memory.txt");

**七、测试（Test）**

**主程序测试（成功）：**

模式选择未出现问题，相关提示文字均正确。



**洗\发牌测试（成功）：**

经多次测试，牌组中未出现重复的牌且发牌次序准确，详见双人对战测试（或人机对战测试）中牌的花色和点数。

**牌型判断测试（成功）：**

经多次测试，牌型判断,牌型大小比较以及胜负判断均正确。



**双人对战测试（成功）：**

经多次测试，双方下注，跟注，加注，弃牌等均无异常。筹码统计正常。相关提示文字均正确。

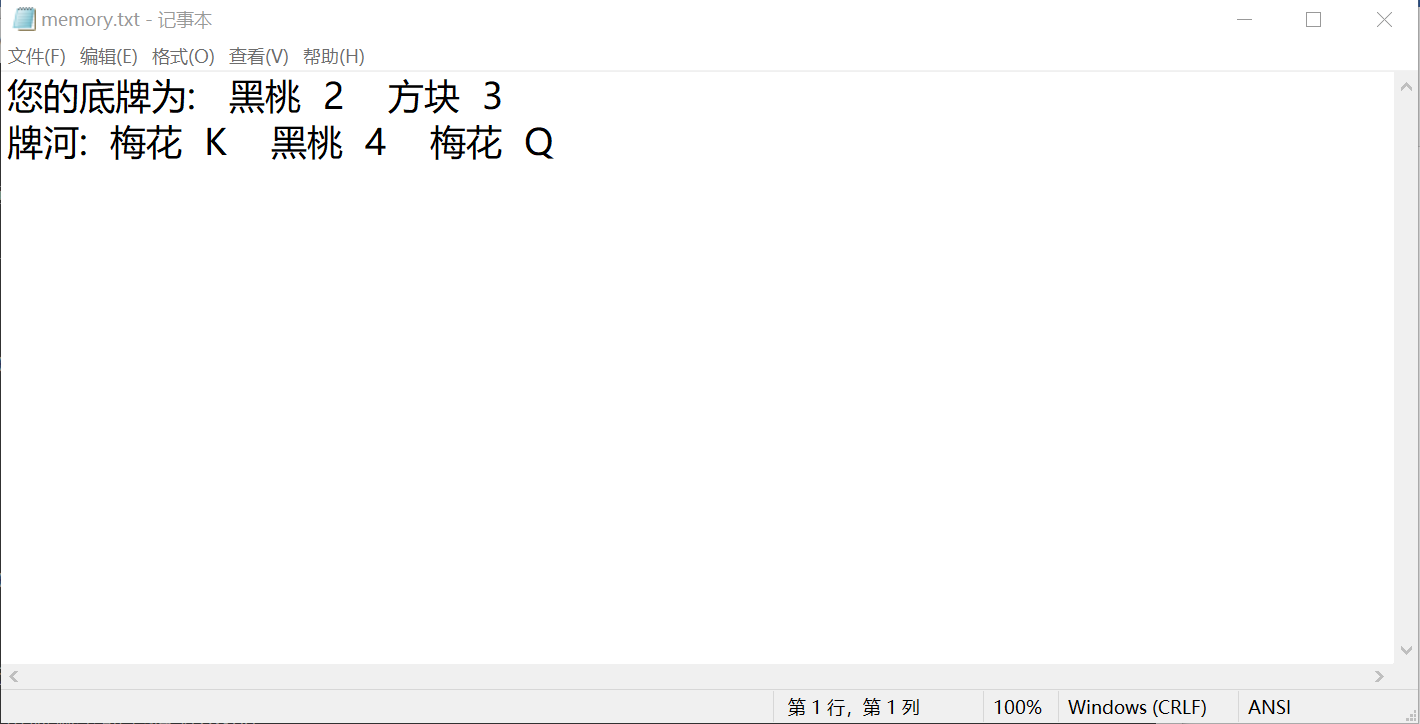




**人机对战测试（成功）：**

经多次测试，玩家下注，跟注，加注，弃牌等均无异常。电脑下注，跟注，加注，弃牌等均无异常且电脑胜率保持在50%左右。筹码统计正常。相关提示文字均正确。记牌器打开，关闭，写入均无异常。

****

****