main{

アプリ起動

難易度選択

while(){

ゲーム処理()

表示処理()

If(ゲーム終了)break;

}

ゲーム結果表示

}

ゲーム処理()｛

If(CPUの番){

入力座標＝CPU()

}else{

入力座標＝入力処理

}

盤面の反転処理(入力座標)

｝

表示処理(){

現在の盤面や戦況を画面に表示

}

CPU(現局面の配列)｛

//木構造に前の手を表すポインタと手を表す座標を合わせた数字を挿入していく

//何手先まで読むかによって難易度調節をする

while(↓の炙り出しで出た数だけループ){

CPUが置くことが可能なマスの炙り出し

↑で炙り出したマスのうちの一つに置いた場合の局面を表す配列の作成

while(↓の炙り出しで出た数だけループ){

↑で作成した局面でのユーザーが置くことが可能なマスの炙り出し

↑で炙り出したマスのうちの一つに置いた場合の局面を表す配列の作成

　while(↓の炙り出しで出た数だけループ){

　　↑で作成した局面でのCPUが置くことが可能なマスの炙り出し

　　↑で炙り出したマスのうちの一つに置いた場合の局面を表す配列の作成

　　　　　w=局面を点数化

　　max=Math.max(max,w)

　　木構造に最終的なmaxを挿入

　}

　　　}

　}

｝

局面を点数化し木構造に記録したものを元に最善種を決定する

置くことが可能なマスの炙り出し()｛

｝

指定マスに置いた後の局面を表す配列を返す()｛

｝

局面を点数化する(局面の配列)｛

手版の色＝白

do () {

for (int i = 0; i <配列.length; i++) {

for (int j = 0; j < 配列[i].length; j++) {

if (配列 [i][j] != "□") {

continue;

}

flag =置くことが可能か判定(配列 [i][j],手版の色)

if (flag == true) {

switch (手版の色) {

case 白:

cnt++;

break;

case 黒:

cnt--;

break;

}

}

}

}

手版の色切り替え;

}while(手版の色＝黒)

return cnt;

｝