オセロゲームの簡単な実装のソースコード

2019/11/14
C言語で書いたものは見つからなかったんだけど、C#に単純移植したものは残ってたので記録。

実行形式ダウンロード

simpleoth20191115.zip
- .NET Core 3.0、.NET Framework 4.5.2、.NET Framework 4.7.2 用バイナリを収めたzipファイル

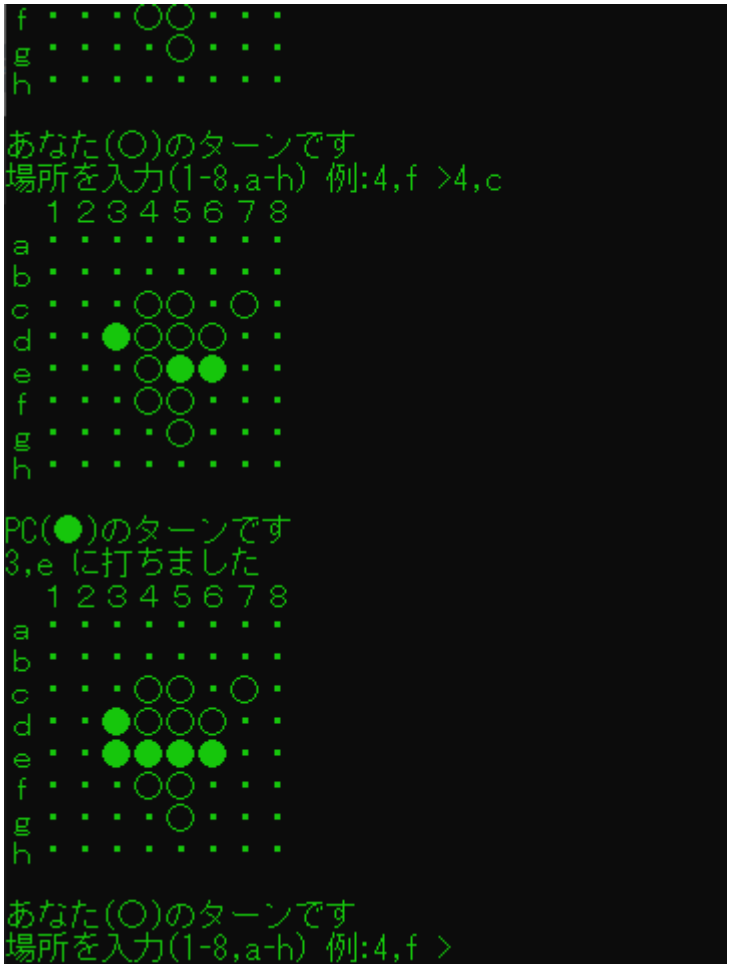
.NET Core なら MacでもLinuxでも動くと聞いたけどよくわかってません。

コンパイル方法

make.batとProgram.csを同じフォルダにおいて、make.batを実行、plum002.exe が生成されるのでこれを実行してください。コンソール用のゲームです。

make.batの環境変数 DNETPATH で使う .NET の版を指定します。

実行すると、こんな懐かしい感じでゲームができます。



make.bat

make.bat

```
rem SET DNETPATH="C:\Windows\Microsoft.NET\Framework\v2.0.50727"
rem SET DNETPATH="C:\Windows\Microsoft.NET\Framework\v3.5"
SET DNETPATH="C:\Windows\Microsoft.NET\Framework\v4.0.30319"

%DNETPATH%\Csc.exe /noconfig /errorreport:prompt /warn:4 /define:TRACE
/filealign:512 /optimize+ /out:plum002.exe /target:exe Program.cs

PAUSE
```

Program.cs

Program.cs

```
using System;
using System.Collections.Generic;
using System.Collections;
//using System.Linq;
using System.Text;

namespace peroapp0002
{
    enum chips
    {
             blank
            , nonBoard
            , BlackChip
            , WhiteChip
            , BlueChip
    }

    struct point
    {
        public int posx;
        public int posy;

        public point(int x, int y)
        {
            posx = x;
            posy = y;
        }
    };
```

```
    struct vector
    {
        public int vecx;
        public int vecy;

        public vector(int vx, int vy)
        {
            vecx = vx;
            vecy = vy;
        }
    };

    /// <summary>
    /// ボードのクラス
    /// </summary>
    class playBoard
    {
        private chips[,] playbd;
        private point[][] thinkData = {
                                        new point[]{  new point(1,1) ,
new point(8,8) ,  new point(1,8) ,  new point(8,1) },
                                        new point[]{  new point(3,4) ,
new point(4,6) ,  new point(6,5) ,  new point(5,3)
                                                    , new point(3,5) ,
new point(6,4) ,  new point(5,6) ,  new point(4,3) },
                                        new point[]{  new point(1,4) ,
new point(8,5) ,  new point(5,1) ,  new point(4,8)
                                                    , new point(5,8) ,
new point(8,4) ,  new point(4,1) ,  new point(1,5) },
                                        new point[]{  new point(8,3) ,
new point(3,8) ,  new point(6,1) ,  new point(1,6)
                                                    , new point(6,8) ,
new point(8,6) ,  new point(3,1) ,  new point(1,3) },
                                        new point[]{  new point(4,2) ,
new point(5,2) ,  new point(2,4) ,  new point(2,5)
                                                    , new point(4,7) ,
new point(5,7) ,  new point(7,5) ,  new point(7,4) },
                                        new point[]{  new point(1,2) ,
new point(1,7) ,  new point(2,8) ,  new point(7,8)
                                                    , new point(8,7) ,
new point(8,2) ,  new point(7,1) ,  new point(2,1)
                                                    , new point(2,6) ,
new point(3,7) ,  new point(6,7) ,  new point(7,6)
                                                    , new point(7,3) ,
new point(6,1) ,  new point(3,1) ,  new point(2,3) },
                                        new point[]{  new point(2,2) ,
new point(6,6) ,  new point(6,3) ,  new point(2,7)
                                                    , new point(7,7) ,
new point(3,3) ,  new point(7,2) ,  new point(3,6) }
                                      };
        private vector[] searchVecter = {
```

```
                                                new vector( 1, 0), new
vector( 1, 1), new vector( 0, 1), new vector(-1, 1)
                                              , new vector(-1, 0), new
vector(-1,-1), new vector( 0,-1), new vector( 1,-1)
                                            };

        /// <summary>
        ///コンストラクタ
        /// </summary>
        public playBoard()
        {
            playbd = new chips[10, 10];

            ////ブランクで埋め尽くす
            for (int x = 0; x < 10; x++)
            {
                for (int y = 0; y < 10; y++)
                {
                    playbd[x, y] = chips.blank;
                }
            }

            ////外枠の設定
            for (int xy = 0; xy < 10; xy++)
            {
                playbd[xy, 0] = chips.nonBoard;
                playbd[xy, 9] = chips.nonBoard;
                playbd[0, xy] = chips.nonBoard;
                playbd[9, xy] = chips.nonBoard;
            }

            ////中央の４つのコマ
            playbd[4, 4] = chips.BlackChip;
            playbd[5, 5] = chips.BlackChip;
            playbd[4, 5] = chips.WhiteChip;
            playbd[5, 4] = chips.WhiteChip;
        }

        /// <summary>
        ///指定の場所に駒を置いた場合いくつ取れるのか確認する
        /// </summary>
        /// <returns>ひっくり返せる駒の数</returns>
        private int searchBoard(chips myChip, int posx, int posy)
        {
            int count;
            chips enemyChip;

            if ((1 > posx) || (8 < posx) || (1 > posy) || (8 < posy))
return 0;

            if (playbd[posx, posy] != chips.blank) return 0;
```

```
            count = 0;
            switch (myChip)
            {
                case chips.WhiteChip:
                    enemyChip = chips.BlackChip;
                    break;

                case chips.BlackChip:
                    enemyChip = chips.WhiteChip;
                    break;

                default:
                    enemyChip = chips.blank;
                    break;
            }

            //// ８方向に検索をかけてみる
            for (int i = 0; i < searchVecter.Length; i++)
            {
                int bx = posx + searchVecter[i].vecx;
                int by = posy + searchVecter[i].vecy;
                int vecCount = 0;

                while (playbd[bx, by] == enemyChip)
                {
                    vecCount++;
                    bx += searchVecter[i].vecx;
                    by += searchVecter[i].vecy;
                }

                switch (playbd[bx, by])
                {
                    case chips.blank:
                    case chips.nonBoard:
                        vecCount = 0;
                        break;
                }

                count += vecCount;
            }

            return count;
        }

        /// <summary>
        /// 指定の場所に駒を置いて相手の駒をひっくり返す
        /// </summary>
        /// <returns>ひっくり返した数</returns>
        public int reverseChip(chips myChip, int posx, int posy)
        {
```

```
            int count;
            chips enemyChip;

            if ((1 > posx) || (8 < posx) || (1 > posy) || (8 < posy))
return 0;

            if (playbd[posx, posy] != chips.blank) return 0;

            count = 0;
            switch (myChip)
            {
                case chips.WhiteChip:
                    enemyChip = chips.BlackChip;
                    break;

                case chips.BlackChip:
                    enemyChip = chips.WhiteChip;
                    break;

                default:
                    enemyChip = chips.blank;
                    break;
            }

            //// 8方向に検索をかけてみる
            for (int i = 0; i < searchVecter.Length; i++)
            {
                int bx = posx + searchVecter[i].vecx;
                int by = posy + searchVecter[i].vecy;
                int vecCount = 0;

                while (playbd[bx, by] == enemyChip)
                {
                    vecCount++;
                    bx += searchVecter[i].vecx;
                    by += searchVecter[i].vecy;
                }

                switch (playbd[bx, by])
                {
                    case chips.blank:
                    case chips.nonBoard:
                        vecCount = 0;
                        break;

                    //// 自分の駒にたどり着いたなら戻りつつひっくり返す
                    default:
                        bx -= searchVecter[i].vecx;
                        by -= searchVecter[i].vecy;
                        while (playbd[bx, by] == enemyChip)
                        {
```

```
                            playbd[bx, by] = myChip;
                            bx -= searchVecter[i].vecx;
                            by -= searchVecter[i].vecy;
                        }
                        break;
                }

                count += vecCount;
            }

            if (count != 0)
            {
                playbd[posx, posy] = myChip;
            }

            return count;
        }

        /// <summary>
        ///駒をおけるのか確認する
        /// </summary>
        /// <returns>置けるなら true</returns>
        public bool isPutting(chips myChip)
        {
            for (int lvl = 0; lvl < thinkData.Length; lvl++)
            {
                for (int sel = 0; sel < thinkData[lvl].Length; sel++)
                {
                    if (0 < searchBoard(myChip,
thinkData[lvl][sel].posx, thinkData[lvl][sel].posy))
                    {
                        return true;
                    }
                }
            }
            return false;
        }

        /// <summary>
        ///お勧めの場所を教えてもらう
        /// </summary>
        /// <returns>無ければ(0,0),あればその場所のpoint構造体が返る</returns>
        public point searchBestPoint(chips myChip)
        {
            int count = 0;
            point bestPoint = new point(0, 0);

            for (int lvl = 0; lvl < thinkData.Length; lvl++)
            {
                int selCount = 0;
                for (int sel = 0; sel < thinkData[lvl].Length; sel++)
```

```
                {
                    selCount = searchBoard(myChip,
thinkData[lvl][sel].posx, thinkData[lvl][sel].posy);
                    if (count < selCount)
                    {
                        count = selCount;
                        bestPoint.posx = thinkData[lvl][sel].posx;
                        bestPoint.posy = thinkData[lvl][sel].posy;
                    }
                }
                if (count != 0) break;
            }

            return bestPoint;
        }

        /// <summary>
        /// ボードのひょうじ
        /// </summary>
        public void display()
        {
            string sy = "□□□□□□□□□";

            Console.WriteLine("□□□□□□□□□");

            for (int y = 1; y < 9; y++)
            {
                Console.Write(sy.Substring(y - 1, 1));

                for (int x = 1; x < 9; x++)
                {
                    switch(playbd[x,y])
                    {
                        case chips.BlackChip:
                            Console.Write("○");
                            break;

                        case chips.WhiteChip:
                            Console.Write("●");
                            break;

                        case chips.blank:
                            Console.Write("□");
                            break;
                    }
                }

                Console.WriteLine("");
            }

            Console.WriteLine("");
```

```
        }

        /// <summary>
        ///入力
        /// </summary>
        public point input(chips myChip)
        {
            point pos = new point(0, 0);

            while (1 == 1)
            {
                Console.Write("場所を入力(1-8,a-h) 例:4,f >");

                string inline = Console.ReadLine();
                if (inline.Length > 2)
                {
                    pos.posx =
"12345678".LastIndexOf(inline.Substring(0, 1)) + 1;
                    pos.posy =
"abcdefgh".LastIndexOf(inline.Substring(2, 1)) + 1;

                    if ((0 < pos.posx) && (pos.posx < 9) && (0 <
pos.posy) && (pos.posy < 9))
                    {
                        if (0 < searchBoard(myChip, pos.posx, pos.posy)
) break;
                    }
                }
            }
            return pos;
        }

    }


    /// <summary>
    ///主処理
    /// </summary>
    class Program
    {

        static void Main(string[] args)
        {
            playBoard pbd = new playBoard();
            chips turn = chips.BlackChip;

            if (args.Length > 1)
            {
                switch (args[0])
                {
                    case "black":
```

```
                        turn = chips.BlackChip;
                        break;

                    case "white":
                        turn = chips.WhiteChip;
                        break;

                    default:
                        turn = chips.BlackChip;
                        break;
                }
            }
            else
            {
                turn = chips.BlackChip;
            }

            Console.WriteLine("PCと楽しむおせろもどき");
            int count_m = 2;
            int count_e = 2;

            while (1 == 1)
            {
                pbd.display();

                if ((false == pbd.isPutting(chips.BlackChip)) && (false
== pbd.isPutting(chips.WhiteChip)))
                {
                    Console.WriteLine("ゲーム終了");
                    Console.WriteLine("あなた: {0},  PC: {1} ", count_m,
count_e);

                    if (count_m > count_e)
                    {
                        Console.WriteLine("あなた の勝ちっす");
                    }
                    if (count_m < count_e)
                    {
                        Console.WriteLine("PC の勝ちっす");
                    }
                    if (count_m == count_e)
                    {
                        Console.WriteLine("だっせぇ、同点っす");
                    }
                    Console.Write("<キーを押して終了>");
                    Console.ReadLine();
                    break;
                }


                if (turn == chips.BlackChip)
```

```
                {
                    Console.WriteLine("あなた(○)のターンです");

                    if (true == pbd.isPutting(chips.BlackChip))
                    {
                        point p = pbd.input(chips.BlackChip);
                        count_m+=pbd.reverseChip(chips.BlackChip,
p.posx, p.posy);

                        pbd.display();
                    }
                    else
                    {
                        Console.WriteLine("打つ場所がないのでパスします。");
                    }

                    turn = chips.WhiteChip;
                }

                //Console.Write("<Enterキーを押してねー>");
                //Console.ReadLine();
                //Console.Clear();


                if (turn == chips.WhiteChip)
                {
                    Console.WriteLine("PC(●)のターンです");

                    if (true == pbd.isPutting(chips.WhiteChip))
                    {
                        point p = pbd.searchBestPoint(chips.WhiteChip);
                        count_e += pbd.reverseChip(chips.WhiteChip,
p.posx, p.posy);
                        Console.WriteLine("{0},{1} に打ちました", p.posx,
"abcdefgh".Substring(p.posy - 1, 1));
                    }
                    else
                    {
                        Console.WriteLine("打つ場所がないのでパスします。");
                    }

                    turn = chips.BlackChip;
                }

            }

            return ;

        }
    }
```

```
}
```

Windows, oth, c#